Méthode de détection et de diagnostic des attaques de blocage dans les systèmes manufacturiers flexibles incertains

Amaury Beaudet

N°d'ordre NNT : 2024ISAL0048

THESE de DOCTORAT DE L'INSA LYON, membre de l'Université de Lyon

Ecole Doctorale N°ED 160
Electronique, Electrotechnique, Automatique

Spécialité/ discipline de doctorat :
Automatique

Soutenue publiquement le 07/06/2024, par :
Amaury Beaudet

Méthode de détection et de diagnostic des attaques de blocage dans les systèmes manufacturiers flexibles incertains

Devant le jury composé de :

BERRUET, Pascal, Professeur des universités, Université de Bretagne Sud Président

LEFEBVRE, Dimitri, Professeur des universités, Université de Normandie	Rapporteur
ESPES, David, Prof. des universités, Université de Bretagne Occidentale	Rapporteur
BERRUET, Pascal, Professeur des universités, Université de Bretagne Sud	Examinateur
MARANGÉ, Pascale, Maître de conférences, Université de Lorraine	Examinatrice
HENRY, Sébastien, Maître de conférences, IUT Lyon 1	Examinateur
LE CHEVILLER, François-Régis, DGA	Invité
ZAMAÏ, Éric, Professeur des universités, INSA-LYON	Directeur de thèse

Méthode de détection et de diagnostic des attaques de blocage dans les systèmes manufacturiers flexibles incertains

Amaury Beaudet [1]
Directeur de thèse : M. Éric Zamaï
Encadrement DGA : M. Stéphane de Fleuriau et M. Francois-Regis Le Cheviller

20 mai 2024

[1] Convention de Subvention DGA : N° 1981 0057
Email : amaury.beaudet@insa-lyon.fr

Remerciements

Ce manuscrit de thèse conclut quatre années de travail au sein d'un milieu professionnel que je ne pensais jamais fouler alors étudiant, celui de la recherche académique. Il y a quatre ans, cette thèse représentait un défi face à mes doutes, tant bien sur mes compétences scientifiques que sur mes capacités personnelles de rigueur et d'autonomie. J'avais envie de me plonger sur un sujet qui me passionnait et d'en devenir un expert. Je pense avoir atteint cet objectif, ce manuscrit l'attestant, après de nombreux mois troublés d'incertitudes face aux blocages rencontrés et emplis de motivation dès l'obtention d'avancées positives. Néanmoins, ces quatre années de travail ont été entourées de nombreuses personnes qui m'ont soutenu, accompagné et aidé à respirer aux moments opportuns. Je souhaite en premier lieu remercier toutes ces personnes.

Tout d'abord, je remercie profondément mon directeur de thèse, Pr. Éric Zamaï, qui a fait preuve d'une patience indéfectible à toutes les étapes de mon doctorat. Il a gardé confiance en mon travail malgré le peu de transparence qui accompagnait parfois mes présentations et mes rapports. Son regard critique et pragmatique sur mon travail a été essentiel pour me sortir de mes égarements réguliers dans le doute et l'idéalisme. Ayant été initialement le tuteur de mon stage de fin d'étude, il fut l'une de mes motivations principales à candidater à cette thèse.

Sous la direction d'Éric, toute l'équipe de recherche m'a accompagné lors de mes quatre années de thèse. Je souhaite donc remercier Emil Dumitrescu et Cédric Escudero pour le partage de leurs compétences techniques et théoriques face aux problématiques de recherche que j'ai pu rencontrer. Leur présence systématique et leurs retours constructifs lors de réunions d'équipe ont permis de recadrer, d'une part, et d'apporter de nouvelles idées, d'autre part, à mes travaux de recherche. Plus généralement, j'aimerais remercier l'ensemble de mes collègues du laboratoire Ampère avec qui j'ai pu partager de nombreux moments sportifs, festifs, et de débats d'idées. Ils ont été une vraie source de soutien et un échappatoire après mes journées intenses de recherche au laboratoire.

Ma thèse a été financée par la Direction Générale de l'Armement (DGA) et j'aimerais par conséquent remercier M. De Fleuriau pour le suivi de mon travail, son enthousiasme face à mes résultats et ses propositions de recherche pertinentes. Je remercie aussi M. Le Chevillier pour la continuité de l'accompagnement après le départ de M. De Fleuriau.

Je souhaite dédier ce dernier paragraphe aux personnes de mon cercle personnel qui m'ont accompagné pendant ma thèse. Tout d'abord, je remercie profondément Clémentine, ma compagne, pour son soutien indéfectible, pour son écoute attentive face à mes exposés confus et passionnés, et pour sa relecture complète de ce manuscrit. Je souhaite aussi remercier Poisson, notre chat, de m'avoir tenu compagnie pendant mes longues heures de rédaction en télé-travail.

Par ailleurs, j'ai une pensée affectueuse pour ma belle-famille qui a porté un intérêt continu sur ma thèse lorsqu'ils m'ont accueilli chaleureusement chez eux. Enfin, je souhaite remercier mes amis, de Lyon ou d'ailleurs, pour leur présence tout au long de ma thèse ; en particulier, je remercie Paul pour le partage de son expérience d'ancien doctorant et pour nos échanges critiques sur mes recherches.

Table des matières

Table des figures

Liste des tableaux

Nomenclature

Acronymes

$\mathcal{L}_a(N, M, \mathcal{M}_c)\|_{min}$	Langage minimal des attaques de blocage, page 71
$A^\star\mathcal{P}$	Algorithme de recherche $A^\star$ pour le calcul des profils d'attaquant, page 106
$A^\star BT$	Recherche hybride $A^\star$ et retour sur trace (Backtracking), page 100
$CTRL_k$	Contrôleur de la ressource R_k, page 10
$R_k - CTRL_k$	Boucle de contrôle commande entre R_k et son contrôleur local $CTRL_k$, page 10
API	Automate Programmable Industriel, page 10
CFC	Composante Fortement Connexe, page 148
CIM	Computer Integrated Manufacturing, page 9
DCS	Distributed Control System, page 10
DMS	Dedicated Manufacturing System, page 7
DZ	Dead Zone ou Zone morte du modèle S^{3}PR d'un FMS, page 22
ERP	Enterprise Resource Planning, page 9
FBM	First Bad Marking, page 97
FMS	Flexible Manufacturing Systems, page 6
ICS	Industrial Control System, page 9
IT	Information Technology, page 9
KPI	Key Performance Indicator, page 10
LZ	live-Zone ou Zone vivace du modèle S^{3}PR d'un FMS, page 22
MES	Manufacturing Execution System, page 10
MOCN	Machine Outil à Commande Numérique, page 8
OT	Operational Technology, page 9
RdP	Réseau de Petri, page 6
RMS	Reconfigurable Manufacturing System, page 7
RTU	Remote Terminal Unit, page 10
S^{2}P	Simple Sequential Process, page 13
S^{2}PR	Simple Sequential Process with resources, page 14
S^{3}PR	System of Simple Sequential Processes with Resources, page 12
SAP	Système Automatisé de Production, page 6
SC-net	Modèle temporisé d'un S^{3}PR pour le calcul de l'ordonnancement, page 230
SCADA	Supervisory Control And Data Acquisition, page 10
SED	Système à Événements Discrets, page 12

SMS Strict Minimal Siphon ou Siphon Strict Minimal, page 23

TRS Taux de Rendement Synthétique, page 10

Notations

$\mathrm{N}_\alpha = (P_\alpha, T_\alpha, F_\alpha, E_\alpha, l_\alpha, Ct_\alpha, D_\alpha)$ Modèle réduit des attaques de blocage, page 89

$(M, \Delta) \in x_d$ Paire d'un état x_d d'un diagnostiqueur associant un ensemble de label Δ à un marquage M, page 124

(M, R) État d'un RdP P-temporisé, page 86

(M_0, R_0) État initial d'un RdP P-temporisé, page 87

(M_{init}, R_{init}) Marquage initial d'un algorithme de recherche d'ordonnancement, page 101

(M_{obj}, R_{obj}) État objectif d'un modèle SC-net, page 100

$(p, t), (t, p) \in F$ Arcs orientés d'un RdP, page 225

(P_c^{NB}, F_c^{NB}) Places de contrôle temporaires construites à partir de N_{NB}^C, page 154

(V, F) graphe orienté avec V et F les ensembles de nœuds et d'arcs orientés du graphe, page 227

2^Q Ensemble de tous les sous-ensembles d'un ensemble Q, page 72

$[N]$ Matrice d'incidence de N, page 226

$\delta \in \Delta$ Labels d'un diagnostiqueur, page 124

$\delta_{\mathcal{R}_j}^M$ Label du $mode_{\mathcal{R}_j}$ pour un état $M \in \mathcal{R}(N_t, M_0^t)$, page 171

$\delta_f \in \Delta_f$ Labels de faute d'un diagnostiqueur, page 124

$\delta_{\mathcal{P}_i}$ Label d'un profil d'attaquant $\mathcal{P}_i$, page 131

$\delta_{\mathcal{P}_i}^{M_k}$ Label du profil d'attaquant $\mathcal{P}_i$ débuté depuis un état M_k du FMS, page 133

$\delta_{\mathcal{P}}^{M_k, \mathcal{R}_i}$ Label d'un profil d'attaquant depuis un état M_k et un mode $Mode_{\mathcal{R}_i}$, page 179

δ_{A1}, δ_{A2} Labels associés aux attaques de blocage, page 129

$\Gamma_{(M,R)}$ Temps écoulé depuis une état temporisé (M, R), page 86

λ Séquence d'événements, page 47

λ_a Séquence d'événements de E_a pouvant contenir des événements attaqués, page 48

$\mathcal{L}(N_L, M)$ Langage d'un RdP labellisé généré à partir de M, page 48

$\mathcal{L} \setminus s$ Ensemble de tous les suffixes de s dans $\mathcal{L}$, page 48

$\mathcal{L}_a(N, M)|_{sr}$ Langage des attaques sournoises depuis M dans un RdP labellisé N, page 50

$\mathcal{L}_a(N, M)$ Langage de l'attaquant depuis M dans un RdP labellisé N, page 50

$\mathcal{L}_o$ Projections d'un langage $\mathcal{L}$ sur E_o, page 48

$\mathcal{L}_{\leq n}$ Langage $\mathcal{L}$ restreint aux séquences d'événements λ bornées par $|\lambda| \leq n$, page 48

$\mathcal{L}_a(N, M, \mathcal{M}_c)$ Langage des attaques de blocage ciblant un état $M \in \mathcal{M}_c$ depuis M dans N, page 71

$\mathcal{M}_e$ Ensemble des marquages atteignables depuis les paires d'un état $x_d \in X_d$ après l'occurrence de e, page 125

$\mathcal{M}_{DZ}$ Ensemble des marquages de la DZ d'un RdP (N, M_0), page 22

$\mathcal{M}_{LZ}$ Ensemble des marquages de la LZ d'un RdP (N, M_0), page 22

$\mathcal{P}$ Profil d'attaquant, page 72

$\mathcal{RG}(N, M)$ graphe des marquages accessibles de (N, M), page 227

$\mathcal{RM}(mode_{\mathcal{R}_i})$ Modes atteignables depuis $mode_{\mathcal{R}_i}$, page 171

$\mathcal{R}(N, M)$	Ensemble des marquages accessibles dans N depuis M, page 226
$\mathcal{R}_k$	Ensemble de ressources du FMS, page 19
$\mathcal{R}_{ind}$	Ensemble des ressources pouvant devenir indisponibles dans N, page 20
Ω_N	Fonction de franchissement d'une transition dans un RdP P-temporisé, page 86
$\overline{Ct_a}$	Moyenne de l'ensemble des coûts unitaires attribués par la fonction Ct, page 83
$\parallel I \parallel$	Support d'un P-invariant I, page 23
Π	Ensemble de tous les siphons d'un RdP, page 23
ρ	Politique de contrôle, page 48
ρ'	Politique de contrôle interdisant le franchissement des transitions de reprise dans $N_{\mathcal{R}_j}$, page 152
σ	séquence de transitions dans $T^\star$, page 226
$\sigma(M, mode_{\mathcal{R}_j})$	Ordonnancement complet calculé depuis M lors d'un changement de mode vers $mode_{\mathcal{R}_j}$), page 155
σ_f	Ordonnancement calculé à partir d'un RdP, page 24
σ_{Ct_a}	Écart-type de l'ensemble des coûts unitaires attribués par la fonction Ct, page 83
τ_a	Fonction d'association d'une transition attaquée $t_a \in T_a$ à la transition non attaquée $t \in T$, page 80
Υ	Fonction donnant l'indice d'un ordonnancement σ_f depuis un état $(M, R) \in \mathcal{R}(N_\alpha, (M_{init}, R_{init})$, page 233
$\|Q\|$	Nombre d'éléments d'un ensemble Q, page 225
A, B	Recettes du FMS, page 7
C	Circuit de N, page 227
C_{NB}	Ensemble des circuits de places activité et d'entrée de N_{NB}, page 154
Cd_m	Condition de changement de mode, page 176
$CLOSED$	Liste des états déjà explorés d'un algorithme de recherche $A^\star$, page 106
Cn_d	Condition de ré-ordonnancement du FMS, page 122
Csc	Ensemble des conséquences de cascades d'indisponibilités pour un changement de modes $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$, page 152
Ct, CT	Fonctions coût unitaire d'un événement et coût global d'une attaque, page 82
$Ctr \in CTR$	Contraintes spécifiques à un profil pour la recherche $A^\star\mathcal{P}$, page 233
D	Fonction de temps de délai des places d'un RdP, page 85
$Diag(N_l, M_{init}) = (X_d, E_d, f_d, x_0^d)$	Diagnostiqueur construit pour N_l à partir de M_{init}, page 124
$Diag_a(N_\alpha, M_{init}^\alpha) = (X_d^a, E_d^a, f_d^a, x_0^a)$	Diagnostiqueur des attaques de blocage, page 129
$Diag_c, Diag_c^i$	Diagnostiqueur commun des attaques de blocage et des profils d'attaquant, page 132
$Diag_m(N_t, x_0^m) = (X_m, E_m, f_m, x_0^m)$	Diagnostiqueur des changements de mode depuis un état x_0^m, page 169
$Diag_{\mathcal{P}}(N_\alpha, M_{init}^\alpha) = (X_d^{\mathcal{P}}, E_d^{\mathcal{P}}, f_d^{\mathcal{P}}, x_o^{\mathcal{P}})$	Diagnostiqueur des profils d'attaquant, page 131
$Diag_{cu}^i(N_\alpha^{\mathcal{R}_{ind}}, x_0^{cu_i}) = (X_{cu}^i, E_{cu}^i, f_{cu}^i, x_0^{cu_i})$	Diagnostiqueur commun des attaques de blocage dans un contexte incertain, page 175
$e \in E$	Ensemble des événements d'un RdP labellisé, page 47
$E^\star$	Ensemble des séquences d'événements de E, page 47

$e^{ind}_{DEF}(r_k)+, e^{dis}_{DEF}(r_k)+$	Événements d'attaque contes les événements observables de défaillance et de reprise d'une ressource, page 163
E_+, E_-	Ensembles des événements insérés et supprimés de E, page 49
E_-, E_+	Ensembles des événements des attaques de suppression et d'insertion, page 80
E_a	Ensemble de événements dans un RdP attaqué, page 49
E_c, E_{ic}, E_o, E_{io}	Événements contrôlables, incontrôlables, observables et inobservables de E, page 47
E_d	Ensemble des événements labellisant les arcs d'un diagnostiqueur, page 125
e_{δ_f}	Événement d'occurrence de la faute labellisée par δ_f, page 124
F	Ensemble des arcs orientés d'un RdP, page 225
$f(M, R)$	Fonction d'estimation du coût total d'un ordonnancement de (M_{init}, R_{init}) à (M_{obj}, R_{obj}) traversant (M, R), page 102
$f^\star(M, R)$	Coût total réel d'un ordonnancement de (M_{init}, R_{init}) à (M_{obj}, R_{obj}) traversant (M, R), page 102
f_α	Fonction de conversion des marquages de N_a vers N_α, page 92
f_a	Fonction de projection d'une séquence attaquée $\lambda_a \in E_a^\star$ sur G (f_a^1) et sur S (f_a^2), page 49
F_c	Arcs reliant les places moniteurs $p_c \in P_c$ aux transitions du modèle S^{3}PR qu'elles contrôlent, page 79
$f_d(x_d, e_d)$	Fonction transition d'un diagnostiqueur, page 125
G	Le système physique, page 50
$g(M, R)$	Coût réel observé entre l'état initial (M_{init}, R_{init}) et (M, R) , page 102
$h(M, R)$	Fonction heuristique d'estimation de coût restant depuis l'état (M, R) jusqu'à l'état (M_{obj}, R_{obj}), page 101
$h^star(M, R)$	Coût réel restant depuis l'état (M, R) jusqu'à l'état (M_{obj}, R_{obj}), page 101
H_{r_k}	Ensemble des places opérations requérant r_k, page 20
I	Vecteur P-invariant de dimensions $1 \times \lvert P\rvert$, page 23
K_{cn_d}	Constantes paramétrables de la condition de ré-ordonnancement, page 123
$KLL(S)$	Marquage maximal d'un siphon S lorsqu'un état de blocage partiel est atteint, page 113
l	Fonction de labellisation des transitions d'un RdP, page 47
$L(N, M)$	Ensemble de toutes les séquences de transitions franchissables depuis M dans N, page 227
$L(N, M)_{\leq n}$	Ensemble $L(N, M)$ restreint aux séquences de transitions σ bornées par $\lvert\sigma\rvert \leq n$, page 227
M	Marquage des places d'un RdP, page 225
$M(Q)$	Marquage total des places de $Q \subset P$, page 225
$M[t > M'$	M' atteignable depuis M en franchissant la transition t, page 226
$M\lvert_Q$	Marquage M projeté aux places de $Q \subset P$, page 225
M_0	Marquage initial d'un RdP noté (N, M_0), page 225
$M_c \in \mathcal{M}_c$	Ensemble des marquages critiques ciblés par un attaquant, page 48
M_{init}	Marquage du SC-net utilisé pour le calcul de l'ordonnancement, page 122

M_{rc} Marquage racine d'un CFC de $N_{\mathcal{R}_k}$, page 247

M_{rl} Marquage réellement atteint par le modèle SC-net du FMS incluant l'arrivée de nouveau produit, page 122

$Mode_{\mathcal{R}_i,\mathcal{R}_j}$ Changement de mode de $Mode_{\mathcal{R}_i}$ à $Mode_{\mathcal{R}_j}$, page 19

$Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$ Changement de mode d'un mode $Mode_{\mathcal{R}_i}$ à un mode $Mode_{\mathcal{R}_j}$ depuis un état M, page 143

$Mode_{\mathcal{R}_k}$ Mode de fonctionnement où les ressources de $\mathcal{R}_k$ sont indisponibles, page 19

$MSK(R)$ Matrice des temps requis entre deux places activités d'un S^{3}PR, page 114

$Mx(A)$ Nombre de produits maximum pouvant être fabriqués simultanément pour une recette A, page 122

N Un Réseau de Petri, page 225

$N|_\rho, L|_\rho, \mathcal{R}|_\rho, \mathcal{M}_\rho$ RdP labellisé, son langage, ses marquages accessibles et ses états interdits par la politique de contrôle ρ, page 48

$N_a = (P_{GS}, T_a, F_a, E_a, l_a)$ Modèle des attaques de blocage, page 80

$N_B = (P_B, T_B, F_B)$ Un sous-RdP de N contenant ses places bloquées par des ressources indisponibles, page 144

$N_G = (P_G, T_G, F_G, E_G, l_G)$ Modèle de l'état réel de G, page 78

N_L RdP labellisé, page 47

$N_S = (P_S, T_S, F_S, E_S, l_S)$ Modèle de l'état de G observé par S, page 78

$N_t = (P_t = P_A \cup P_R \cup P_E \cup P_S, T, F_t, D)$ Modèle SC-net d'un modèle S^{3}PR N, page 100

$N_{\mathcal{R}_{ind}}$ RdP de reprise de N selon $\mathcal{R}_{ind}$, page 20

$N^\alpha_{\mathcal{R}_{ind}} = (P^\alpha_{\mathcal{R}_{ind}}, T^\alpha_{\mathcal{R}_{ind}}, F^\alpha_{\mathcal{R}_{ind}}, E^\alpha_{\mathcal{R}_{ind}}, l^\alpha_{\mathcal{R}_{ind}}, Ct^\alpha_{\mathcal{R}_{ind}}, D^\alpha_{\mathcal{R}_{ind}})$ Modèle des attaques de blocage dans un contexte incertain, page 164

$N^{GS}_{\mathcal{R}_{ind}}$ Modèle RdP fusionné incluant les réseaux de reprises pour G et S, page 163

$N_{GS} = (P_{GS}, T, F_{GS}, E, l)$ Modèle fusionné de G et S, page 78

$N_{NB} = (P_{NB}, T_{NB}, F_{NB})$ Un sous-RdP de N contenant ses places non-bloquées par des ressources indisponibles, page 144

N^C_{NB} Modèle S^{3}PR conservant uniquement les circuits C_{NB} de N_{NB} et les ressources requises, page 154

N^{C-t}_{NB} Modèle SC-net construit à partie de N^C_{NB}, page 154

O Label d'un état optimal du FMS, page 129

O^A_i Opération i de la recette A, page 7

$O_{\mathcal{P}}$ Label d'un état du FMS associé à aucun profil d'attaquant, page 131

$OPEN$ Liste des états à explorer d'un algorithme de recherche $A^\star$, page 106

$p \in P$ Places d'un RdP, page 225

P'_{blck} Ensemble des places bloquées indépendamment des ressources indisponibles, page 152

P'_{DZ} Ensemble des places bloquées par un blocage partiel apparu suite à un changement de mode, page 152

$p^0 \in P^0$ Places d'entrée du FMS, page 13

$P^0_{r_k}$ Ensemble des places d'entrée bloquées en cas d'indisponibilité de la ressource r_k, page 144

P^B_{ptg}	Ensemble des places activités de N_B requérant une ressource partagée avec une place activité de N_{NB}, page 145
p^{ind}	Place d'indisponibilité de la ressource associée à $p \in P_R \cup P_A$, page 20
$P^{post}_{r_k}$	Places activité de N_{NB} appartement à des circuits bloqués de N, page 145
$P^{pre}_{r_k}$	Ensemble des places activité de N dont les produits qu'elles opèrent requièrent systématiquement la ressource r_k pour être terminés, page 144
$p_a \in P_A$	Places activité du FMS, page 13
$p_c \in P_c$	Places moniteurs d'un modèle S^{3}PR, page 26
$p_e \in P_E$	Places d'entrée d'un modèle SC-net, page 100
P_o	Fonction de projection d'un événement ou d'une séquence d'événements sur E_o, page 47
$p_r \in P_r$	Places ressource du FMS, p_{r_k} est associée à la ressource notée R_k ou r_k, page 14
$p_s \in P_S$	Places de sortie d'un modèle $SC - net$, page 100
$P_{sk}(S)$	Ensemble des places puits d'un siphon S, page 114
$Post$	Matrice d'incidence avant, page 226
Pre	Matrice d'incidence arrière, page 226
R	Matrice des temps de délais restants d'un RdP P-temporisé, page 86
R_k, r_k	Ressource k du FMS, page 7
S	Le superviseur, page 50
S	Un siphon d'un RdP, page 23
$s < s'$	s suffixe de s', page 48
$t \in T$	Transitions d'un Réseau de Petri, page 225
$T^\star$	Ensemble de toutes les séquences de transitions, $\sigma \in T^\star$, page 227
$t^{ind}_{p+}, t^{dis}_{p+}$	Transitions d'indisponibilité et de reprise associées à la place p, page 163
T_-, T_+	Ensembles des transitions des attaques de suppression et d'insertion, page 80
t^{dis}_p	Transition de reprise de la ressource associée à $p \in P_R \cup P_A$, page 20
t^{ind}_p	Transition de mise à l'arrêt de la ressource associée à $p \in P_R \cup P_A$, page 20
T_{fr}	Ensemble des transitions d'attaque franchissables pour le calcul d'un profil d'attaquant par la recherche $A^\star\mathcal{P}$, page 107
TRI	Consignes de tri spécifiques à un profil pour la recherche $A^\star\mathcal{P}$, page 233
$U(MSK(R), M, S)$	Ensemble des combinaisons temps nécessaires pour vider un siphon S jusqu'à avoir $M(S) - KLL(S)$ jetons, page 115
U_{max}	Nombre maximum de ressources détenues au sein des états de blocage d'un FMS, page 110
$UR(\mathcal{M})$	Ensemble des marquages atteignables depuis $M \in \mathcal{M}$ après l'occurrence d'une séquence non-observable $\lambda_{io} \in E^\star_{io}$, page 125
W	Fonction de pondération des arcs de F, page 225
WRT	Matrice du temps d'opération minimum requis par ressource pour chaque place activité d'un modèle SC-net, page 102
$x^\bullet, {}^\bullet x$	Preset et postset d'un élément $x \in P \cup T$, page 225
$x_1 x_2 ... x_{k+1}$	Chemin dans un RdP N ($x_i \in P \cup T$) ou dans un graphe orienté (V, E) ($x_i \in V$), page 227

X_d	Ensemble des états d'un diagnostiqueur, page 124

Variables physiques

C^i_{max}	Borne supérieur du coût d'un profil $\mathcal{P}_i$, page 73
K_{max}	Taille maximum de l'espace d'états exploré lors d'une recherche $A^\star$ local dans l'algorithme $A^\star BT$, page 100

Résumé

Les systèmes manufacturiers flexibles (FMSs en anglais) sont conçus avec l'objectif de pouvoir réaliser différentes recettes en parallèle en utilisant conjointement des ressources flexibles et un superviseur allouant ces ressources aux différentes recettes en cours. Par conception, un FMS évolue dans un environnement critique, en présence d'états d'allocation des ressources bloquants pour la réalisation des recettes, et incertain, en raison d'événements imprévus conduisant à l'indisponibilité temporaire des ressources. Au sein des FMSs modernes, l'utilisation pour l'allocation des ressources et l'amélioration de la productivité de composants de contrôle hautement interconnectés entre eux et avec des réseaux internet ouverts a rendu ces systèmes vulnérables aux cyber-attaques. Aux origines de ces cyber-attaques, différents profils d'attaquant peuvent être distingués selon leurs objectifs, leurs origines, leurs compétences et leurs moyens.

Ainsi, bien qu'un FMS soit initialement construit pour faire face aux états de blocage et aux indisponibilités de ressources, un profil d'attaquant expert est capable de manipuler les décisions d'allocation et les disponibilités des ressources pour conduire ce FMS dans un état de blocage ciblé. A partir de cette observation, la problématique de recherche suivante émerge : au sein du contexte incertain des FMSs, comment diagnostiquer correctement l'origine, naturelle ou malveillante, d'un état de blocage détecté et identifier le profil d'attaquant à l'origine de l'attaque ?

En réponse à cette problématique, trois contributions principales sont proposées. Premièrement, les attaques de blocage et les profils d'attaquant sont définis et modélisés dans un contexte certain afin de structurer le développement d'un module de diagnostic de ces derniers. Puis, ce module est étendu au contexte incertain des FMS au sein duquel l'indisponibilité des ressources est prise en compte et peut être manipulée par un attaquant. Enfin, ce module de diagnostic est implémenté sur une plateforme manufacturière expérimentale afin d'être évalué.

Ces propositions de recherche sont fondées sur la théorie des systèmes à événements discrets (SEDs), et sur les outils d'ordonnancement et de prévention des états de blocage au sein des modèles S^3PR, sous-classe des Réseaux de Petri (RdP) représentant le fonctionnement de l'allocation des ressources au sein d'un FMS. Puis, à partir d'un modèle RdP des attaques de blocage, les profils d'attaquant sont évalués à l'aide de la recherche A*, cette dernière permettant d'obtenir une séquence d'événements attaquée optimale selon les objectifs et caractéristiques des différents profils. Enfin, le module de diagnostic est fondé sur la théorie des diagnostiqueurs SEDs au sein desquels les labels symbolisent le diagnostic de la malveillance d'un état de blocage d'une part et les profils d'attaquant d'autre part. Une extension des S3PRs et des méthodes de prévention et d'ordonnancement au contexte incertain est requise pour rendre le module de diagnostic robuste aux indisponibilités des ressources.

Flexible Manufacturing Systems (FMS) aim to achieve different processes in parallel. This production strategy is made concrete with the conjoint use of flexible resources and a supervisor allocating these resources to the running processes. By design, FMSs operate in a critical environment, due to blocking allocation states or deadlock states, and uncertain, as resources can become temporarily unavailable following unexpected production events. In modern FMSs, the deployment for resources allocation and productivity enhancement of control components highly inter-connected and connected to the internet has made FMS vulnerable to cyberattacks. The origins of these cyberattacks are diverse and different attacker profiles can be defined based on their objectives, background, skills, tools and financing.

Hence, although FMSs are initially built to deal with deadlock states and resources unavailability, an expert attacker profile can be able to reach deadlock states by manipulating resources allocation decisions and resources availability. From this statement, the following research problematic arises : in FMSs uncertain environment, how can one diagnose the origin, natural or malicious, of a deadlock state and identify the attacker profile responsible for the attack ?

In answer to this problematic, three main contributions are developed. First, deadlock attacks and attacker profiles are defined and modelled in a certain environment. A deadlock attack diagnosis module is then structured from these models. Second, this module is extended to FMS uncertain environment where resource availability is considered and can be manipulated by an attacker. Third, the diagnosis module is implemented on a manufacturing platform to assess its experimental results.

These reasearch propositions rely on the theory of Discrete Event Systems (DESs), and on the scheduling and deadlock prevention methods within S^3PR, a subclass of Petri Net (PN) modelling FMS resources allocation policy. Then, from a PN model of deadlock attacks, attacker profiles are generated by an A* algorithm. It allows to obtain an optimal deadlock attack sequence optimized in accordance with the profile objectives and characteristics. Finally, the diagnosis module is built on the DES diagnoser theory. In the diagnoser, the labels symbolise the malevolence of a deadlock state on one hand, and the associated attackers profiles on the other hand. In FMS uncertain environment, S^3PR, attack models, prevention and scheduling methods are extended to make the diagnosis module robust to resources unavailability.

Introduction

Au sein de l'industrie manufacturière, les systèmes manufacturiers flexibles (FMSs) garantissent un maintien des performances, une réduction des coûts et une flexibilité de production face à des produits de plus en plus diversifiés, aux courts cycles de vie et manufacturés en faible quantité. Cette flexibilité des FMSs est fondée sur le partage de ses ressources de production entre les différentes recettes à produire. Par conception, un FMS évolue dans un environnement critique, du fait d'états d'allocation de ses ressources pouvant bloquer la réalisation des recettes, et incertain, en raison d'événements conduisant à l'indisponibilité temporaire des ressources (e.g. casses, pannes, maintenances préventives). Afin de remédier aux problématiques de cet environnement, les ressources d'un FMS sont connectées à une architecture de contrôle-commande industrielle. Cette architecture pilote l'allocation des ressources via un superviseur dans le but d'éviter les blocages et surveille les états des ressources pour prédire et prévenir leurs indisponibilités. Cette architecture de contrôle-commande est développée à partir de composants numériques de contrôle (Régulateurs, API, RTU), de supervision (IHM, SCADA) et de pilotage (MES) inter-connectés entre eux par un réseau de communication industriel (Ethernet TCP/IP). Cependant, le déploiement de ces composants numériques et l'ouverture aux réseaux de communication ont introduit une nouvelle problématique à l'environnement des FMSs héritée du domaine des technologies de l'information : les cyber-attaques.

Depuis les années 2000, différentes cyber-attaques ciblant les systèmes de contrôle commande industriels dont font partie les FMSs ont été enregistrées. L'attaque Stuxnet en 2010, orchestrée par les États-Unis contre une centrale d'enrichissement d'uranium iranienne, ou plus récemment, au cœur du conflit entre le Russie et l'Ukraine, les cyber-attaques multiples menées par des hackers indépendants contre les infrastructures industrielles russes et, de manière antagoniste, les cyber-attaques russes réalisées contre les réseaux électriques et de télécommunication ukrainiens, peuvent être prises en exemple. Ainsi, aux origines de ces cyber-attaques, différents profils d'attaquant peuvent être définis selon leurs objectifs, leurs origines, leurs compétences et leurs moyens.

Dans ce contexte, bien qu'un FMS soit initialement développé pour faire face aux états de blocage et aux indisponibilités de ressources, certains profils d'attaquant sont capables de manipuler les décisions d'allocation et les disponibilités des ressources pour conduire le FMS dans un état de blocage ciblé. Par exemple, un profil d'attaquant étatique sans limite de moyens (financiers, matériels, humains) est capable de compromettre les composants de contrôle des ressources (régulateurs, API, RTU) afin de manipuler toutes les décisions d'allocation et les événements d'indisponibilité échangés entre les ressources et le superviseur du FMS (SCADA/MES). Ainsi, les méthodes de gestion des états de blocage implémentées au sein du superviseur deviennent inefficaces en présence d'un attaquant. Face à ce type d'attaques, les

méthodes de détection des états de blocage peuvent être envisagées en raison de leur capacité à diagnostiquer la présence d'un blocage malveillant à partir de données extraites depuis différents composants de l'architecture de contrôle-commande du FMS. Cependant, la capacité d'un tel profil d'attaquant à manipuler la disponibilité des ressources lui permet, d'une part, de dissimuler son attaque au sein des événements d'indisponibilité naturelle (panne, défaillance) et, d'autre part, d'atteindre des états de blocage non référencés par la méthode de détection en forçant l'indisponibilité d'une ou plusieurs ressources. A partir de ces observations, la problématique de recherche suivante émerge : au sein du contexte incertain des FMSs, comment détecter un état de blocage, diagnostiquer son origine, naturelle ou malveillante, et identifier le profil d'attaquant à l'origine de l'attaque ? Cette problématique n'a, à ce jour, pas été considérée par les travaux de la littérature.

Ce manuscrit est organisé selon cinq grandes parties.

La première partie présente notre système d'étude, les FMSs. Après avoir introduit le fonctionnement général et l'architecture de pilotage d'un FMS, un modèle réseau de Petri (RdP) de l'allocation des ressources est introduit. Puis, l'environnement des FMSs est détaillé à travers les objectifs de fonctionnement d'un FMS, l'indisponibilité de ses ressources et la définition de ses états critiques bloquants. En conclusion de cette partie, les différentes solutions intégrées au FMS lui permettant d'être résilient vis-à-vis de son environnement sont exposées au regard de la littérature scientifique. Les méthodes d'ordonnancement de l'allocation des ressources et les méthodes de gestion des états de blocage au sein des FMSs sont référencées dans les contextes sans ou avec indisponibilités.

La deuxième partie s'intéresse à la cyber-malveillance à l'encontre des FMSs. Les vulnérabilités face aux attaques et les solutions de cyber-sécurité existantes pour y faire face sont exposées dans le contexte général des systèmes de contrôle-commande industriels. A partir de cette présentation du contexte général et du travail introductif sur les FMSs réalisé dans la première partie, les vulnérabilités d'un FMS, de sa boucle de contrôle et de son architecture sont répertoriées, et les objectifs et les méthodes des attaques pouvant cibler les FMSs sont définis. Parmi ces objectifs, les attaques de blocage des FMSs réalisées par un attaquant expert sont retenues dans le cadre de nos travaux. Afin d'identifier la problématique et les verrous de notre étude, trois états de l'art sur les attaques de blocage, sur les méthodes de détection d'attaque dans les systèmes à événements discrets et sur les méthodes de diagnostic conjoint des attaques et des indisponibilités sont menés.

La troisième partie propose une méthode de diagnostic des attaques de blocage et des profils d'attaquant à l'origine de ces attaques dans un contexte sans indisponibilité. Dans un premier temps, les différents profils d'attaquant considérés sont définis et un modèle RdP des attaques de blocage est développé. Puis, au regard de l'intégration de notre méthode au pilotage du FMS, deux méthodes de gestion des états de blocage et d'ordonnancement de la littérature sont appliquées. A partir du choix de ces méthodes, un algorithme original de calcul des profils d'attaquant est présenté. Enfin, grâce au modèle d'attaque et aux profils ainsi calculés, un module de diagnostic des attaques de blocage et des profils d'attaquant dans un contexte sans indisponibilité est proposé.

La quatrième partie étend le module de diagnostic au contexte incertain en présence d'indisponibilités des ressources. Dans ce nouvel environnement incertain, les conséquences de l'indisponibilité d'une ressource sont identifiées et conduisent à l'adaptation des méthodes de

gestion des blocages et d'ordonnancement à ces indisponibilités. Pour leur part, le modèle des attaques de blocage et l'algorithme de calcul des profils d'attaquant intègrent les attaques contre la disponibilité des ressources afin de permettre le développement d'un module de diagnostic capable de distinguer les attaques de ces indisponibilités naturelles.

La cinquième partie applique le module de diagnostic des attaques de blocage sur une plateforme manufacturière expérimentale. Dans un premier temps, notre programmation du fonctionnement FMS de la plateforme et des attaques de blocage est détaillée. Dans un second temps, trois scénarios d'attaque de blocage sont implémentés en ligne sur la plateforme et permettent de tester et de valider, hors-ligne, les mécanismes de diagnostic.

Chapitre 1

Les systèmes manufacturiers flexibles : contexte et fonctionnement

Introduction

Dans ce premier chapitre, les systèmes de contrôle commande industriels étudiés dans nos travaux, les systèmes manufacturiers flexibles (FMSs), sont introduits au regard de leurs caractéristiques propres puis selon celles de l'environnement auquel ces systèmes doivent faire face. Tout d'abord, dans la partie 1.1, une présentation des FMSs est proposée. Le fonctionnement des FMSs pour l'allocation de ressources aux recettes à produire, l'architecture de contrôle commande dans laquelle ils évoluent et la modélisation de leur fonctionnement par les réseaux de Petri font l'objet respectivement des sous-parties 1.1.1, 1.1.2, 1.1.3. Dans une deuxième partie 1.2, l'environnement des FMSs est présenté. Ce dernier correspond à un environnement contraint par des objectifs de fonctionnement relatifs aux systèmes industriels manufacturiers, incertain en raison de l'indisponibilité des ressources de production et critique en présence d'états d'allocation des ressources à ne pas atteindre : les états de blocage. Ces trois attributs de l'environnement des FMSs sont détaillés dans les sous-parties 1.2.1, 1.2.2, 1.2.3. En conclusion de ce chapitre, les différentes solutions intégrées au FMS lui permettant d'être résilient vis-à-vis de son environnement sont exposées au regard de la littérature scientifique. Les méthodes d'ordonnancement de l'allocation des ressources et les méthodes de gestion des états de blocage au sein des FMSs sont référencées dans les sous-parties 1.3.1 et 1.3.2. Dans la sous-partie 1.3.3, les méthodes d'ordonnancement et de gestion des états de blocage sont finalement analysées selon leur robustesse face à l'indisponibilité incertaine des ressources.

1.1 Les systèmes manufacturiers flexibles (FMSs)

La première section 1.1 de ce manuscrit a une vocation introductive et présente les systèmes manufacturiers flexibles ou FMSs (Flexible Manufacturing Systems). Dans un premier temps (sous-partie 1.1.1), le principe de fonctionnement des FMSs pour l'allocation de ressources et ses origines industrielles seront détaillés. Dans un second temps (sous-partie 1.1.2), l'architecture de contrôle commande des FMSs considérée sera introduite et le positionnement de notre travail dans cette architecture sera mis en exergue. Finalement (sous-partie 1.1.3), en conclusion de cette partie, la modélisation de l'allocation des ressources au sein d'un FMS sera définie à l'aide de l'outil des Réseaux de Petri (RdPs).

1.1.1 Principe de fonctionnement et objectifs d'un FMS

Historiquement, les FMSs ont été développés à partir des années 1960 dans l'objectif de réduire les coûts de production et d'apporter de la flexibilité de production aux systèmes manufacturiers face à des produits de plus en plus diversifiés, aux courts cycles de vie et manufacturés par lots de petites tailles [1].

Dans nos travaux, un FMS est défini comme un système automatisé de production (SAP) capable de piloter la réalisation de différentes recettes, à savoir des séquences d'opérations que doit suivre un produit depuis son état initial pour atteindre son état final, à l'aide d'un ensemble partagé de ressources matérielles[1]. Par définition d'un SAP, un FMS est composé d'un système physique (le flux de production)(sous-partie 1.1.1), sur lequel agit une partie opérative (les ressources)(sous-partie 1.1.1) pilotée et supervisée par une architecture de contrôle commande industrielle (sous-partie 1.1.2). Au sein des SAPs, les FMSs sont une alternative aux

systèmes de production dédiés (DMS) et aux systèmes manufacturiers reconfigurables (RMS)[2], [3]. Contrairement à un DMS, un FMS est dit flexible par sa capacité à allouer dynamiquement les ressources qu'il contrôle aux différentes recettes en cours et non d'avoir pour chaque ressource une recette fixe qui lui est désignée. Ainsi, le déploiement d'un FMS permet de réduire le nombre de ressources nécessaires à la réalisation des recettes et d'optimiser le taux d'utilisation d'une ressource par l'allocation de cette dernière à plusieurs recettes. Toutefois, les FMSs ne possèdent pas la reconfigurabilité multiple des RMSs puisque les recettes, la configuration spatiale des ressources ou l'architecture et les programmes de contrôle-commande sont immuables.

Les ressources pilotées par un FMS peuvent être des ressources de production, responsables de la transformation des produits (e.g. assemblage, extrusion, polissage, soudage), ou des ressources de transitique, responsables du déplacement des produits entre deux ressources tiers (e.g. bras de robot, convoyeur, véhicule guidé automatisé). Une ressource peut être dédiée à une unique recette ou être partagée entre plusieurs recettes. Chaque ressource possède une capacité propre qui correspond au nombre de produits qu'elle peut opérer simultanément. Une action de production ou de transitique réalisée par une ressource est appelée une opération. Cette dernière possède un début, une fin et par conséquent une durée et pour être réalisée, requiert une ou plusieurs ressources. Une séquence planifiée d'opérations réalisées par les différentes ressources du FMS est appelée une recette. Au sein d'une recette, les parallélismes d'opérations sont autorisés, à savoir qu'une sous-séquence d'opérations inclue dans la recette peut être réalisée par deux combinaisons distinctes de ressources. Du point de vue d'un produit assigné à cette recette, cela signifie qu'il peut emprunter deux chemins différents à travers les ressources du FMS pour la réalisation d'une même sous-séquence d'opérations.

A partir de la définition d'une ressource, d'une opération et d'une recette, la définition suivante du fonctionnement d'un FMS pour l'allocation des ressources est choisie. Soit un produit en entrée du FMS à son état initial devant suivre une recette A pour atteindre son état final désiré. Ce produit est disposé dans un espace d'attente, un stock tampon par exemple, avant d'être transformé et transporté par les ressources du FMS jusqu'à son état final. Le lancement de la recette A sur le produit a lieu lorsque la première opération O_1^A de la recette A sur le produit est lancée. Cette opération requiert un ensemble de ressources $R_{k_1}, R_{k_2}, ..., R_{k_f}$ pour être effectuée et le rôle du FMS est d'allouer ces ressources à la réalisation d'O_1^A. L'allocation ne peut avoir lieu uniquement si les ressources requises ont une capacité disponible équivalente à la capacité requise par l'opération. Une fois l'allocation terminée, l'opération débute et le produit détient alors les ressources $R_{k_1}, R_{k_2}, ..., R_{k_f}$ en partie ou en totalité selon les capacités respectives de chaque ressource et les capacité requises par l'opération O_1^A. Les capacités détenues de chaque ressource deviennent non disponibles. Lorsque l'opération est terminée, le produit requiert un nouvel ensemble de ressources $R_{l_1}, R_{l_2}, ..., R_{l_f}$ pour réaliser l'opération O_2^A. Si ces ressources sont disponibles selon les capacités requises par O_i^A, l'allocation peut avoir lieu. Au moment où l'allocation est validée par l'ordonnanceur du FMS, l'opération O_2^A débute, les capacités détenues des ressources $R_{k_1}, R_{k_2}, ..., R_{k_f}$ sont libérées et celles du nouvel ensemble $R_{l_1}, R_{l_2}, ..., R_{l_f}$ deviennent détenues par le produit. Pour chaque opération de la recette A, la succession des étapes - ressources requises, ressources allouées, ressources détenues et ressources libérées - est identique. Lorsque la dernière opération de la recette A est terminée sur le produit, aucune ressource n'est requise car le produit a atteint son état final. Ce dernier est évacué hors du FMS, dans un stock de produits finis par exemple, et les capacités des ressources qu'il détenait sont alors libérées. Dans la recette A, un parallélisme d'opération signifie que l'opération O_i^A

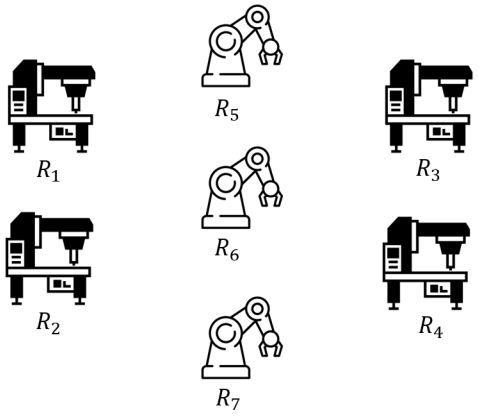


FIGURE 1.1 – Organisation spatiale d'un FMS

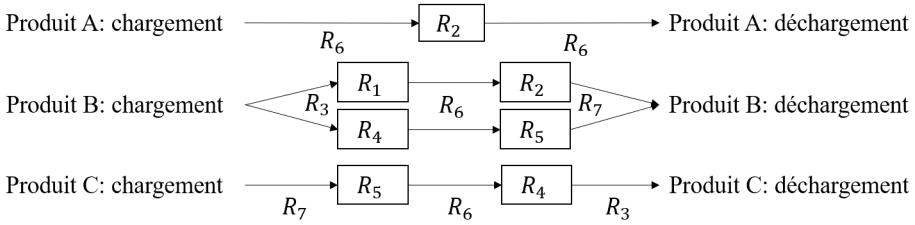


FIGURE 1.2 – Recettes d'un FMS

précédent ce parallélisme est succédée par plusieurs opérations possibles $O^A_{i+1}, O^A_{i+2}, ...O^A_{i+n}$ requérant différents ensembles de ressources. Chacune de ces opérations parallèles débute un circuit parallèle d'opérations de la recette A. Tous les circuits parallèles sont équivalents du point de vue du respect de la recette et se rejoignent et convergent au cours de la recette A à travers des opérations communes, au plus tard lors de l'évacuation du produit hors du FMS. En présence d'un parallélisme d'opération, le rôle du FMS est alors de prendre une décision d'allocation pour le produit en sortie de l'opération O^A_i, à savoir décider quelle opération débuter parmi l'ensemble $O^A_{i+1}, O^A_{i+2}, ...O^A_{i+n}$.

Nous proposons d'illustrer la définition d'un FMS introduite préalablement par un exemple. Soit un FMS composé de 7 ressources $R_1, R_2, R_3, R_4, R_5, R_6, R_7$ avec R_1, R_2, R_3, R_4 des ressources de production de type machine outil à commande numérique (MOCN) et R_5, R_6, R_7 des ressources de transitique apparentées à des bras de robots chargés du transport des produits entre deux MOCNs. L'organisation spatiale des ressources du FMS est schématisée dans la figure 1.1. Ce FMS est programmé pour la réalisation de trois recettes A, B, C dont les séquences d'opérations sont représentées dans la figure 1.2. Pour chaque opération, la ressource nécessaire à sa réalisation est indiquée au niveau d'une flèche pour les robots et dans un rectangle pour les MOCNs. Notons que la recette B présente un parallélisme d'opérations. Dans cet exemple, la flexibilité d'un FMS est illustrée par le partage de ressources entre les recettes, en particulier avec la recette B qui partage les ressources de l'une de ses branches avec la recette A et celles de sa seconde branche avec la recette C.

Le pilotage des ressources par le FMS pour la réalisation des recettes repose sur une architecture de contrôle commande complexe connectant ressources, système d'allocation des

ressources et planification globale de la production. Cette architecture est présentée dans la suite de ce manuscrit.

1.1.2 Architecture de contrôle commande d'un FMS

Le pilotage automatisé et la supervision de l'allocation des ressource au sein d'un FMS est supporté par un système de contrôle commande industriel ou ICS (Industrial Control System). Dans ces travaux, l'architecture de contrôle commande des FMSs choisie est définie selon le modèle de la pyramide CIM (Computer Integrated Manufacturing) [4]. L'architecture CIM ou modèle Purdue est une architecture pyramidale par strates horizontales décrivant le fonctionnement hiérarchique au sein d'une entreprise (fig.1.3). L'objectif de cette décomposition est de faciliter la conception et l'implémentation d'un système de contrôle commande industriel en séparant distinctement au sein de la production les différents pôles, cellules et procédés. L'architecture CIM est composée de six couches horizontales. Elles sont hiérarchisées les unes par rapport aux autres selon une autorité verticale descendante. Les niveaux 3 et 4 de l'architecture CIM représentent la partie IT (Information Technology) de l'entreprise tandis que les niveaux 2, 1, 0 et le système physique constituent les couches OT (Operational Technology) et l'ICS en tant que tel. Tous les niveaux de la pyramide CIM sont présentés dans la suite de ce manuscrit.

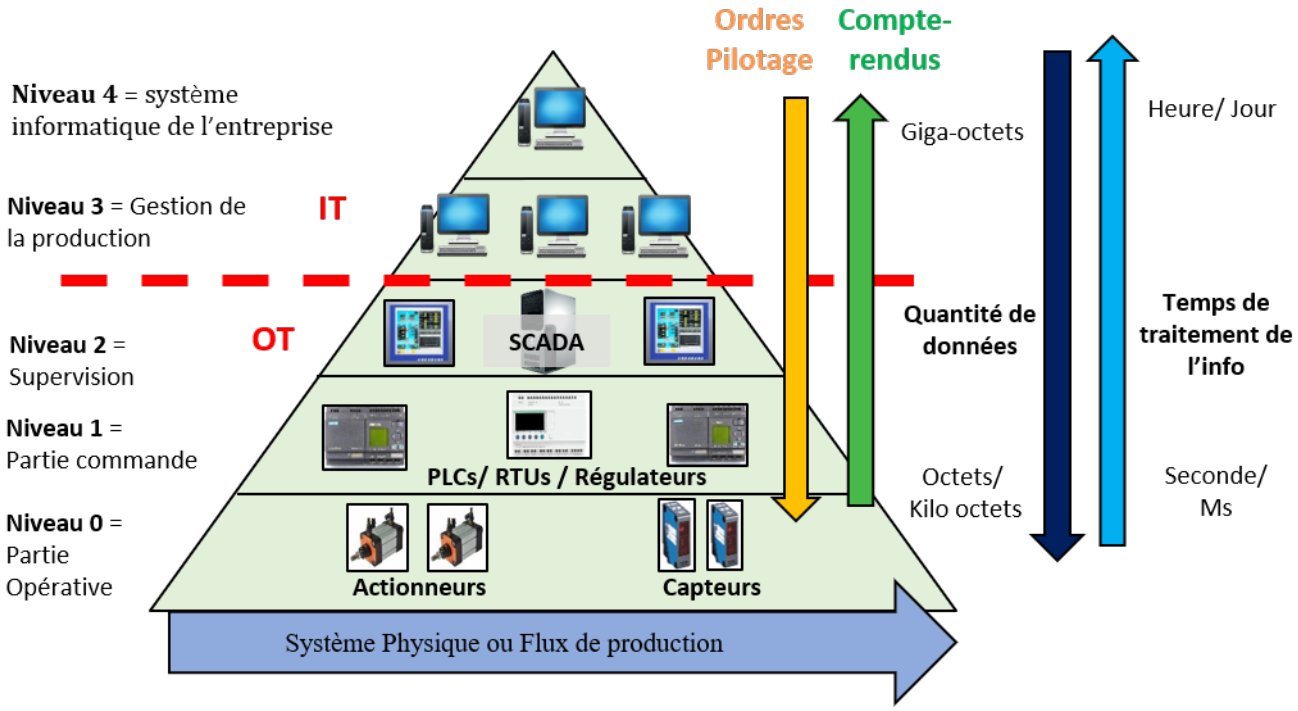


FIGURE 1.3 – Architecture CIM d'un système de contrôle commande industriel

Niveau 4 : ce niveau est le système d'information de l'entreprise. Il correspond à la tâche de la gestion globale de l'entreprise, des ressources humaines au pôle achat. On associe souvent ce niveau aux technologies ERP (Enterprise Resource Planning) déployées dans l'entreprise.

Niveau 3 : ce niveau représente la gestion de la production par l'entreprise. Il s'apparente aux tâches de planification et de suivi de la production ainsi que du contrôle qualité et de

l'analyse d'indicateurs de performance (TRS, KPI). Il est associé aux outils de gestion tels que le MES (Manufacturing Execution System).

Niveau 2 : ce niveau correspond à la supervision de la production. Cette couche a pour mission le pilotage, la surveillance et l'ordonnancement temps-réel du procédé industriel. Le niveau 2 est associé au niveau SCADA (Supervisory Control And Data Acquisition) ou aux DCSs (Distributed Control System). Le SCADA stocke l'ensemble des données provenant des niveaux inférieurs afin de produire une représentation temps-réel du système, compréhensible et interactive pour les opérateurs humains.

Niveau 1 : le niveau 1 de la pyramide est associé à la partie commande de l'ICS. Ce niveau inclut les différentes lois de commande structurant le contrôle automatisé de la partie opérative. Chaque loi de commande contrôle un procédé local spécifique à travers l'échange de messages avec ce procédé, schématisé sous la forme d'une boucle de rétrocontrôle. L'implémentation de la loi de commande est faite sur des systèmes embarqués de type API (Automate Programmable industriel), ou RTU (Remote Terminal Unit) pour le contrôle de variables discrètes et sur des régulateurs pour le contrôle de variables continues.

Niveau 0 : le niveau 0 de la pyramide CIM représente la partie opérative de l'ICS. La partie opérative est composée d'une chaîne d'action et d'une chaîne d'acquisition. La chaîne d'action opère et modifie le système physique selon les commandes envoyées depuis le niveau 1 et est construite à partir d'actionneurs tels que les vérins et les moteurs. La chaîne d'acquisition a pour objectif de surveiller la bonne réalisation du procédé physique grâce à un jeu de capteurs. Les informations collectées sont ensuite communiquées sous la forme de compte-rendus au niveau 1 pour être analysées et permettre la mise à jour en temps réel de la loi de commande.

Système Physique ou Flux de production : le système physique représente l'objet matériel sur lequel souhaite agir l'ICS. Il peut être de différentes natures selon le domaine d'application de l'ICS. Dans le domaine manufacturier, le système physique est désigné par le flux de production, ce dernier mobilisant tous les composants, matières et pièces qui sont transformés lors de la production.

Les communications entre les niveaux : au sein de la pyramide CIM, les niveaux communiquent entre eux de deux manières différentes (voir figure 1.3). Les communications descendantes entre les niveaux sont des ordres de pilotage et les communications ascendantes entre les niveaux sont des rapports et des compte-rendus relatifs à la bonne exécution ou non des ordres. Le tableau ci-dessous (fig.1.1) présente plus précisément les communications entre les différents niveaux. Afin de définir le format des trames des messages et la manière dont les composants communiquent entre eux, une grande variété de protocoles sont déployés dans les ICSs. Le réseau de communication entre les niveaux 1 et 2 est appelé réseau industriel et celui entre les niveaux 0 et 1, réseau de terrain.

Au sein de l'architecture CIM, les FMSs se positionnent dans les niveaux opérationnels, à savoir les niveaux 0-1-2. Les différentes ressources du FMS appartiennent à la partie opérative (niveau 0), et sont commandées par des APIs, RTUs et régulateurs, composants numériques associés au niveau contrôle-commande de l'architecture (niveau 1). On considère que chaque ressource R_k est contrôlée par un unique contrôleur $CTRL_k$ au sein d'une boucle de contrôle commande locale notée $R_k - CTRL_k$. L'ensemble des ressources et de leurs boucles de contrôle est piloté pour la réalisation des recettes par un module de supervision (niveau 2). Ce dernier est

	Descendant	Ascendant
Niveau 4 - Niveau 3	Stratégie de l'entreprise, commandes clients	État global de la production, respect de la stratégie de l'entreprise, besoins pour la production
Niveau 3 - Niveau 2	Planification, recettes produits, objectifs de production	Réalisation des tâches planifiées, état global de l'ICS
Niveau 2 - Niveau 1	Ordres de fabrication, d'arrêt	Informations de fonctionnement, indicateurs de performance, alarmes
Niveau 1- Niveau 0	Ordre/ Commande	Signaux capteurs

TABLEAU 1.1 – Nature des messages descendant et ascendant échangés entre les différents niveaux de la pyramide CIM

chargé de prendre des décisions d'allocation à partir des informations sur l'état des ressources et des opérations reçues depuis les contrôleurs (niveau 1) et de leur transmettre ces décisions sous la forme d'ordres de fabrication et de début d'opération. Ainsi, le module de supervision pilote la réalisation des recettes en communiquant avec tous les contrôleurs au travers d'une boucle de contrôle commande dédiée à l'allocation des ressources et supportée par le réseau industriel. Cette architecture de pilotage est qualifiée de centralisée [5] car un organe central, ici le module de supervision, pilote et supervise seul toutes les boucles de contrôle locales. L'architecture de contrôle commande centralisée d'un FMS est illustrée par la figure 1.4. Les FMSs peuvent aussi être pilotés de manière décentralisée [6]-[9], le module de supervision pour l'allocation des ressources étant dupliqué et déporté au sein de chaque contrôleur local.

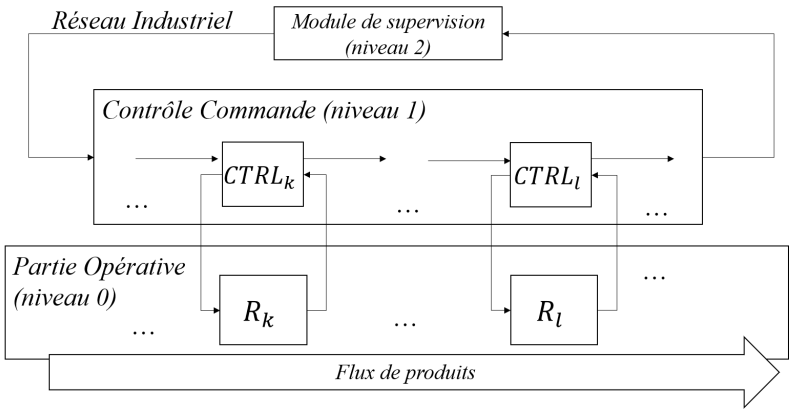


FIGURE 1.4 – Architecture de contrôle commande d'un FMS

Les travaux présentés dans ce manuscrit se focalisent sur l'étude de la boucle de contrôle commande du FMS pour l'allocation des ressources. Notre positionnement au sein de l'architecture CIM se situe donc entre le niveau 1, le réseau industriel et le niveau 2, dans un contexte de pilotage centralisé de plusieurs contrôleurs distincts. Dans la prochaine section, une modélisation des FMSs est introduite afin de proposer une représentation mathématique concrète du procédé d'allocation des ressources.

1.1.3 Modélisation des FMSs : modèle S^{3}PR

Le procédé d'allocation des ressources d'un FMS possède une dynamique assimilable à un système à événements discrets (SED). Par définition, un SED est un système dont la dynamique peut être représentée par un ensemble d'états discrets et de transitions entre ces états déclenchées par l'occurence d'événements [10]. Par exemple, un voyant lumineux peut être modélisé par un SED avec deux états disjoints, voyant allumé et voyant éteint, où la transition de l'état éteint à l'état allumé est déclenchée par l'événement On tandis que la transition inverse est déclenchée par l'événement Off. Dans le cas de la modélisation d'un FMS, un état discret représente un état d'allocation des ressources pour la réalisation des recettes. D'après le fonctionnement des FMSs présenté préalablement, un état d'allocation doit modéliser (i) l'état en entrée du FMS (produits à fabriquer), (ii) l'état des recettes en cours (opérations en cours), (iii) les ressources détenues par les opérations en cours et (iv) les capacités disponibles de chaque ressource. Les évènements déclenchant la transition entre deux états d'allocation sont les décisions d'allocation prises par le module de supervision et ayant pour conséquence l'allocation et la libération de ressources. Dans la littérature, la modélisation SED des FMSs est majoritairement réalisée à l'aide des outils mathématiques discrets des automates [11]-[13], des digraphes [14]-[16] et des réseaux de Petri (RdPs) [1], [16].

Dans nos travaux les RdPs ont été choisis comme outils de modélisation des FMSs pour les raisons suivantes. Premièrement, un RdP est un outil permettant de modéliser de manière synthétique et visuelle un procédé industriel ou une recette en représentant chaque opération par une place et chaque fin/début d'opération par des transitions. Une opération en cours est modélisée dans un RdP par la présence d'un ou plusieurs jetons dans la place qui lui correspond. De la même manière, une ressource peut-être modélisée par une place dont les jetons symbolisent la capacité disponible de la ressource. Lors du franchissement d'une transition (début/fin d'une opération), des jetons sont échangés entre places ressource et places opération pour symboliser l'allocation et la libération de ressources. L'état discret global du FMS est alors interprétable visuellement par la répartition des jetons dans les différentes places du RdP. Deuxièmement, un RdP permet la modélisation des caractéristiques structurelles des recettes des FMSs, telles que la séquentialité, la concurrence, la synchronisation, ou encore le parallélisme [1], une hypothèse prise lors de la présentation du fonctionnement des FMSs. Enfin, un RdP présente un ensemble de propriétés structurelles et mathématiques que nous utiliserons dans la suite de nos travaux. En particulier, la propriété de vivacité d'un RdP est fondamentale pour l'étude des états de blocage. Dans la littérature, les RdPs sont majoritairement préférés pour le représentation des FMSs pour ces mêmes raisons [1].

L'outil de modélisation des réseaux de Petri est défini intégralement dans l'annexe 6.1 de ce manuscrit. Subséquemment, différentes sous-classes de RdPs utilisées pour la modélisation des FMSs sont présentées. Parmi ces sous-classes, les modèles S^{3}PR (System of Simple Sequential Processes with Resources) ont été choisis pour nos travaux et sont détaillés et illustrés par l'exemple pour conclure cette partie 1.1.

Classes de modélisations des FMSs

La modélisation des FMSs s'appuie sur une diversité de sous-classes des RdPs choisissant chacune des caractéristiques différentes du FMS comme hypothèses de modélisation. La classification des différentes sous-classes peut ainsi être réalisée à partir des caractéristiques du FMS

choisies par chaque sous-classe. Ces caractéristiques sont (i) le type de circuits d'opération, (ii) la capacité d'une ressource requise par une opération, (iii) les capacités propres des ressources, (iv) le nombre de ressources requises par une opération et (v) le nombre d'opérations successives pouvant détenir une même ressource. Ces caractéristiques sont présentées dans le prochain paragraphe.

(i) Un circuit est définit par une séquence d'opérations reliant l'état initial d'une recette à son état final. Un circuit est par conséquent propre à une recette et chaque recette possède au moins un circuit. On distingue les FMSs où chaque recette possède un circuit unique, ceux où les recettes peuvent avoir plusieurs circuits parallèles possibles pour la réalisation d'un produit (parallélisme d'opérations) et enfin les FMSs possédant des recettes dont l'exécution jointe de circuits concurrents permet la réalisation d'un même produit. Ces circuits à exécution parallèle sont la conséquence d'opérations d'assemblage ou de désassemblage présentes dans la recette. (ii) Une opération peut requérir une ressource pour une capacité de 1 ou pour une capacité supérieure ou égale à 1. (iii) La capacité d'une ressource peut être bornée à une opération ou une ressource est capable de réaliser plusieurs opérations simultanément grâce à une capacité supérieure à 1. (iv) Une opération peut requérir une unique ressource ou plusieurs ressources différentes sans considération des capacités requises. (v) Une ressource peut être détenue par plusieurs opérations successives avant d'être libérée. Dans la littérature, une sous-classe est définie selon les caractéristiques choisies ci-dessus. Une plus large présentation de ces sous-classes peut-être trouvée au sein des références suivantes : [1], [17], [18].

Dans nos travaux, les caractéristiques des FMSs ont été restreintes afin de pouvoir étudier un système simple pour la résolution de notre problématique de recherche. Ainsi, le FMS que nous considérons est défini par des recettes à circuits uniques ou parallèles (i), des ressources à capacité multiple (iii) et ne pouvant pas être détenues par plusieurs opérations successives (v), et des opérations ne requérant qu'une unique ressource (iv) pour une capacité de 1 (ii). Un tel FMS peut être modélisé à l'aide de la sous-classe des "System of Sequential Simple Processes with Resources" ou S^3PR. Cette dernière est présentée subséquemment.

Les modèles S^3PR

Un "System of Simple Sequential Processes with Resources" ou S^3PR est une sous-classe des RdPs utilisée pour modéliser les systèmes manufacturiers flexibles. Dans cette sous-partie, les modèles S^3PR vont être introduits par étapes, en définissant tout d'abord deux sous-classes des RdPs sur lesquelles s'appuie la théorie des S^3PRs.

Un "Simple Sequential Process" (S^2P), traduit par *Recette Séquentielle Simple*, est un RdP $N = (P_A \cup p^0, T, F)$ où P_A est l'ensemble des places opérations ou activités du FMS, $p^0 \notin P_A$ la place d'entrée du FMS, T un ensemble de transitions et F l'ensemble des arcs orientés. N est une machine à état fortement connexe, à savoir $\forall M, M' \in \mathcal{R}(N, M_0), \exists \sigma \in T^\star | M[\sigma > M'$ et tous les circuit de N contiennent p_0. Dans l'annexe 6.1, $\mathcal{R}(N, M_0)$ est l'ensemble des états accessibles par le RdP N depuis M_0 ; $M[\sigma > M'$ signifie que le marquage M' est atteignable depuis M en franchissant la séquence de transition $\sigma \in T^\star$; et un circuit est une séquence de places et de transitions dont le premier et le dernier élément sont identiques. Un modèle S^2P représente donc une recette du FMS et les opérations successives qui la composent. Un exemple simple de S^2P est proposé dans la figure 1.5a. Dans la prochaine sous-classe, les ressources

utilisées par chaque opération sont définies.

Un S^{2}P avec ressources (S^{2}PR) est un RdP $N = (P_A \cup P^0 \cup P_R, T, F)$, où P_A est l'ensemble des places opérations du FMS, $p^0 \notin P_A$ la place d'entrée du FMS. P_R est l'ensemble des places ressources avec $P_R \neq \varnothing$ et $P_R \cap (P_A \cup P^0) = \varnothing$. Un S^{2}PR a les propriétés suivantes :

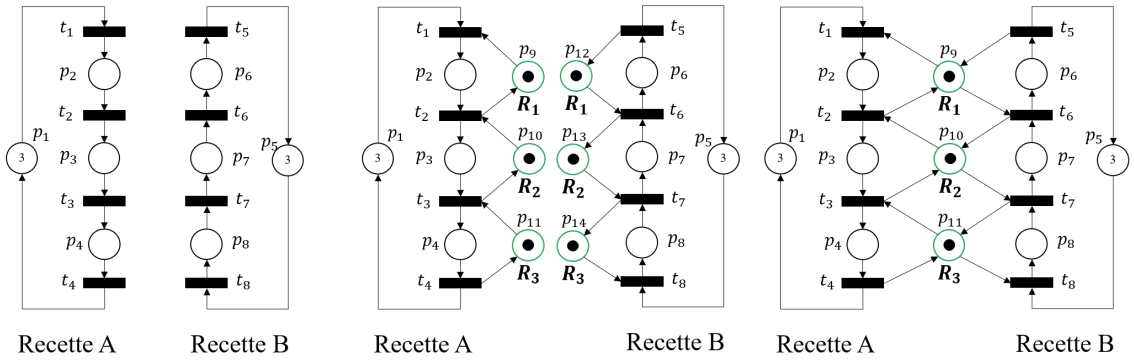
1. Le sous-réseau généré à partir de $X = P_A \cup P^0 \cup T$ est un S^{2}P ;
2. $\forall p \in P_A, \forall t \in {}^\bullet p, \forall t' \in p^\bullet, \exists\, r_p \in P_R$ tel que ${}^\bullet p \cap P_R = p^\bullet \cap P_R = r_p$;
3. $\forall r \in P_R, {}^{\bullet\bullet} r \cap P_A = r^{\bullet\bullet} \cap P_A \neq \varnothing$;
4. $\forall r \in P_R, {}^\bullet r \cap r^\bullet = \varnothing$;
5. ${}^{\bullet\bullet}(p^0) \cap P_R = (p^0)^{\bullet\bullet} \cap P_R = \varnothing$ (Aucune ressource n'est affectée à la place d'entrée) ;

Le marquage d'une place ressource $p_r \in P_r$ représente la disponibilité de la ressource. Cette dernière est disponible si sa place est marquée, non disponible si elle est vide.

Un modèle S^{2}PR représente une séquence de production en liant chaque opération d'une recette S^{2}P avec la ressource que requiert cette opération. Chaque place opération est dorénavant connectée dans le RdP à au moins une place ressource. Dans un S^{2}PR $N = (P_A \cup P^0 \cup P_R, T, F)$, un marquage initial est qualifié d'acceptable si et seulement si :

1. $M_0(p^0) \geqslant 1$; Une recette peut-être lancée au moins une fois ou plusieurs fois simultanément.
2. $M_0(p) = 0, \forall p \in P_A$; Aucune opération n'est en cours initialement.
3. $M_r(p) \geqslant 1, \forall r \in P_R$; Chaque ressource est disponible initialement et a une capacité égale ou supérieure à 1.

Un exemple simple de S^{2}PR est proposé dans la figure 1.5b ci-dessous.



(A) Exemple d'un modèle S^{2}P (B) Exemple d'un modèle S^{2}PR (C) Exemple d'un modèle S^{3}PR

Un S^{3}PR désigne un système de S^{2}PRs et peut ainsi être construit récursivement à partir d'un S^{2}PR ou plus. Un S^{3}PR est défini par les propriétés suivantes :

1. Un S^{2}PR est un S^{3}PR ;
2. Soit $N_i = (P_{A_i} \cup P_i^0 \cup P_R, T_i, F_i)$, $i \in \{1, 2\}$, deux S^{3}PRs respectant $P_{R_1} \cap P_{R_2} \neq \varnothing$, $(P_{A_1} \cup p_1^0) \cap (P_{A_2} \cup p_2^0) = \varnothing$, $T_1 \cap T_2 = \varnothing$. N_1 et N_2 peuvent être composés en $N = (P_A \cap p^0, T, F)$ à travers leurs ressources communes $P_{R_1} \cap P_{R_2}$, et N et toujours un S^{3}PR défini par $P_A = P_{A_1} \cup P_{A_2}$, $p^0 = p_1^0 \cup p_2^0$, $T = T_1 \cup T_2$, $F = F_1 \cup F_2$.

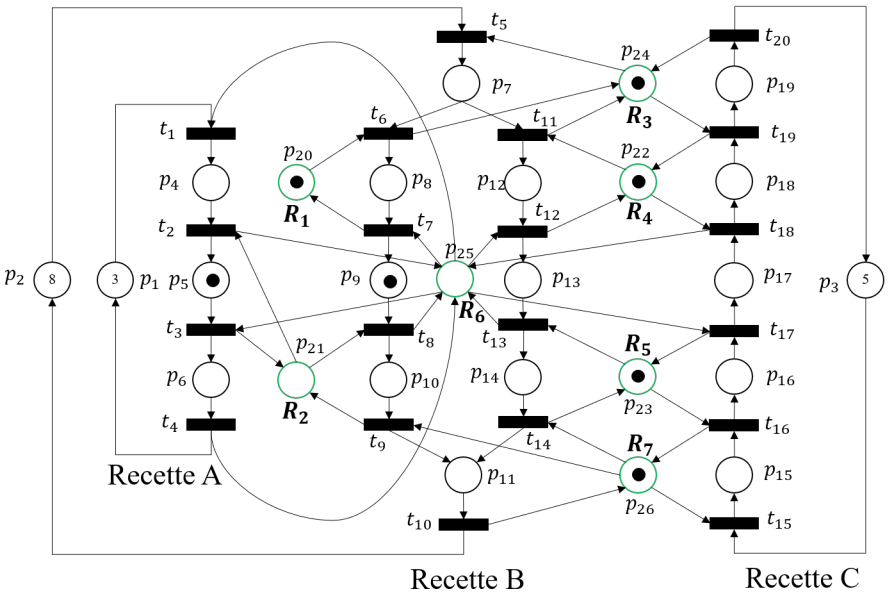


FIGURE 1.6 – Modélisation S3PR de l'exemple des figures 1.1 et 1.2

Cette dernière propriété signifie que deux S^{3}PRs distincts sont composables s'ils partagent un ensemble de ressources communes. Le S^{3}PR résultant est alors une composition des deux modèles originaux. Dans un S^{3}PR $N = (P_A \cup P^0 \cup P_R, T, F)$, un marquage initial M_0 est qualifié d'acceptable si et seulement si :

1. (N, M_0) est un marquage acceptable selon les conditions propres aux S^{2}PRs
2. $N = N_1 \circ N_2$, tel que (N_i, M_{i_0}) est un marquage acceptable du S^{3}PR et :
   (a) $\forall i \in \{1,2\}, \forall p \in P_i \cup P_i^0,\ m_0(p) = m_{0_i}(p)$
   (b) $\forall i \in \{1,2\}, \forall r \in P_{R_i} \setminus (P_{R_1} \cap P_{R_2}),\ m_0(r) = m_{0_i}(r)$
   (c) $\forall r \in (P_{R_1} \cap P_{R_2}),\ m_0(r) = max\{m_{0_1}(r), m_{0_2}(r)\}$

Un exemple simple de S^{3}PR est proposé dans la figure 1.5c.

Exemple

L'exemple présenté dans la partie 1.1.1 est modélisé par un S^{3}PR dans la figure 1.6. Les trois recettes A, B et C sont caractérisés par les places d'entrée p_1, p_2 et p_3. Les places opérations sont les places p_{4-19} et les places ressources les places p_{20-26} selon la répartition $(R_1, p_{20}), (R_2, p_{21}), (R_3, p_{22}), (R_4, p_{23}), (R_5, p_{24}), (R_6, p_{25}), (R_7, p_{26})$. Le marquage initial de cet exemple est $M_0(p_1) = 3$, $M_0(p_2) = 8$, $M_0(p_3) = 5$ et $M_0(p_{20}) = M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = M_0(p_{24}) = M_0(p_{25}) = M_0(p_{26}) = 1$. Cet exemple a 1650 marquages accessibles et 24 états de blocage.

Dans cette partie 1.1.1, une introduction sur le fonctionnement des FMSs, leur architecture de contrôle commande et leur modélisation mathématique a été proposée. Dans la suite de ce manuscrit, l'environnement de fonctionnement des FMSs va être étudié plus en détail à travers ses différents objectifs de commande, ses états critiques et l'indisponibilité de ses ressources.

1.2 L'environnement des FMSs

L'allocation de ressources par le FMS pour la réalisation de recettes n'est pas un comportement déterministe, robuste et sûr. A contrario, le FMS évolue dans un environnement contraint, incertain et critique. Les contraintes sont relatives aux objectifs de fonctionnement qu'un FMS doit atteindre, respecter et optimiser. Les incertitudes sont héritées de la non fiabilité des ressources, pouvant se traduire par leur indisponibilité en raison d'une défaillance ou d'une opération de maintenance. Enfin, les états critiques des FMSs sont relatifs aux états atteignables que le comportement d'allocation des ressources doit éviter pendant son fonctionnement : les états de blocage. Ces trois points font l'objet des trois prochaines sous-parties 1.2.1, 1.2.2 et 1.2.3.

1.2.1 Les objectifs de fonctionnement des FMSs

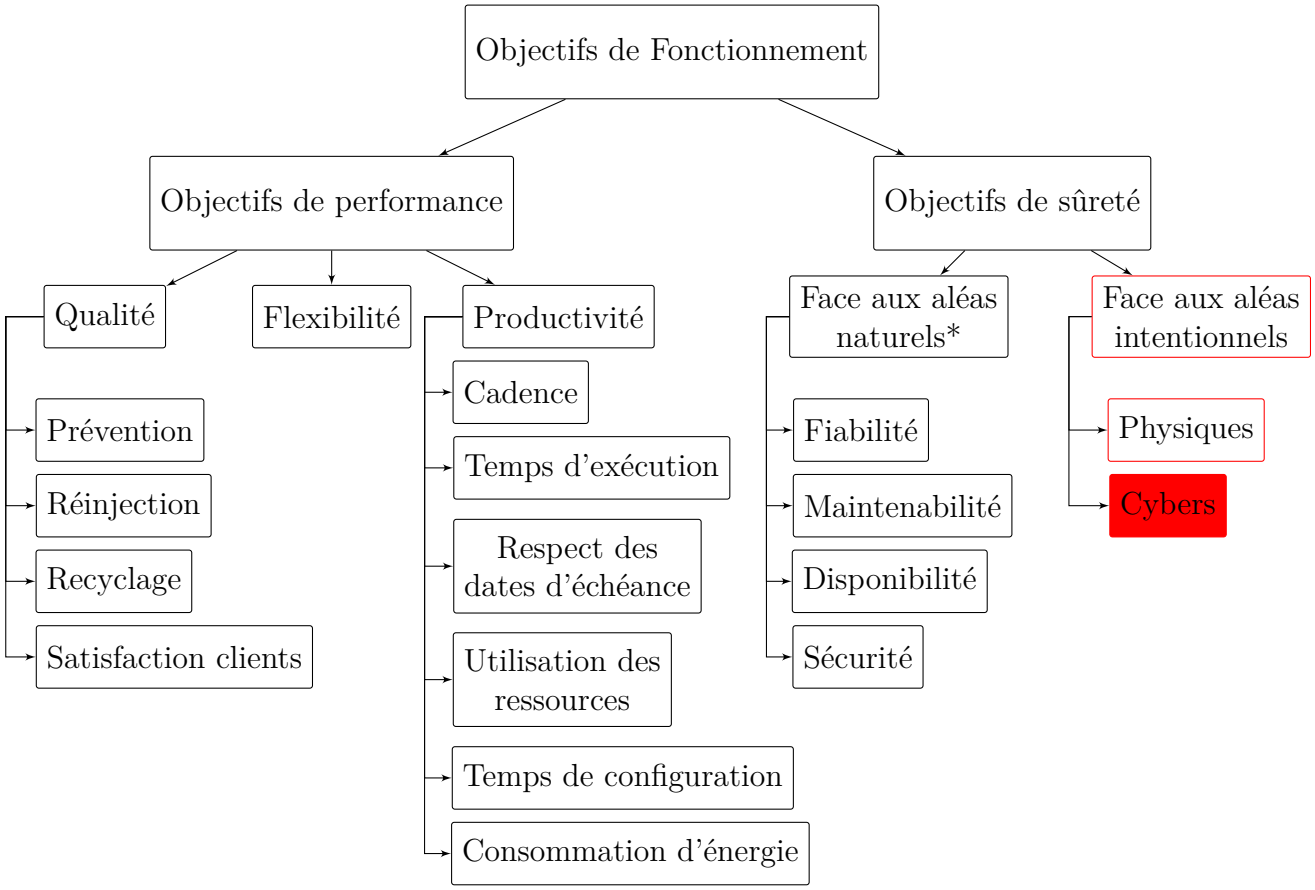
Lors de sa conception et pendant son fonctionnement, un FMS suit et priorise différents objectifs de fonctionnement. Ces objectifs ont pour mission principale le maintien et l'amélioration de la qualité de la production, i.e. la capacité de l'entreprise à assurer la livraison de ses commandes en temps, en quantité et en qualité désirés par le client et en minimisant l'utilisation de ses ressources [19]. Selon l'auteur, ces objectifs de fonctionnement sont relatifs à la qualité, la planification et la maintenance du système manufacturier. Dans le domaine des FMSs, les objectifs de qualité et de planification sont complétés par l'objectif de flexibilité et la planification s'apparente à l'objectif de productivité [20]-[22]. Ces trois objectifs (qualité, flexibilité et productivité) sont désignés dans nos travaux comme les **objectifs de performance** [20]. L'objectif de maintenance est défini au sein de l'**objectif de sûreté** du FMS qui différencie l'origine de l'aléa auquel il fait face, entre aléa de fonctionnement [23], [24] et aléa malveillant [24]-[27]. L'ensemble des objectifs et sous-objectifs de fonctionnement des FMSs sont illustrés dans le graphique de la figure 1.7 et sont introduits dans la suite de cette sous-partie.

Objectifs de performance

La **Qualité** d'un produit peut être définie comme le degré d'excellence des produits finis par rapport aux besoins initiaux des clients [20]. La qualité est donc le respect des standards et caractéristiques du produit fixés à un intervalle de tolérance donné par l'entreprise et le client [28]. L'objectif de qualité se définit par la réduction des coûts liés à la prévention de défauts sur la ligne de production [20], [28], [29], aux produits défectueux [20] qui sont réinjectés dans la ligne de production [22], [29], ou rebutés et recyclés si non ré-injectables [22], [29] et enfin à la non satisfaction des spécifications clients (e.g. rappel de produits) [22], [28], [29].

La **Flexibilité** d'un FMS se définie comme la capacité du système à s'adapter face aux évolutions de son environnement et des demandes clients. On distingue ainsi la flexibilité des ressources (capacité d'une ressource a réaliser plusieurs opérations), la flexibilité des produits (capacité de changer le circuit d'un produit), la flexibilité des recettes (capacité du FMS à réaliser plusieurs recettes) et enfin la flexibilité de volume [20], [30]-[32].

La **productivité** d'un FMS se définit comme le rapport entre la valeur de la production finale en sortie du système et l'ensemble des ressources mis en œuvre pour l'obtenir [33]. Elle peut être simplifiée par le rapport entre les "outputs" du système (sa production) et ses "inputs" (ses coûts de production) [20]. Au sein des FMSs, la productivité peut être rapportée aux critères



*synonyme d'aléas de fonctionnement : casse actionneur, panne capteur...[23]

FIGURE 1.7 – Objectifs de commande des FMSs

que l'on cherche à optimiser [34], [35] pour faire croître la capacité de production de ce dernier, à savoir la réalisation des produits présents en entrée du FMS. Ainsi, les coûts liés aux achats, à la gestion des stocks et à la main d'œuvre [20], [22] ne sont pas considérés. Les sous-objectifs de productivité d'un FMS sont la cadence, le temps d'exécution des produits ou des lots, le respect des dates d'échéances, l'utilisation des ressources, la réduction de la consommation d'énergie et enfin la réduction des temps de configuration des ressources [18], [34]-[36].

Objectifs de sûreté

La **sûreté face aux aléas de fonctionnement**, ou sûreté de fonctionnement, des FMSs est "l'aptitude d'un système à remplir une ou plusieurs fonctions requises dans des conditions données" et s'apparente à la science des défaillances et des pannes [23]. Au sein des FMSs, la sûreté de fonctionnement englobe la fiabilité des ressources [32], [37], [38], leur maintenabilité [39], leur disponibilité [40], [41] et la sécurité des équipements, des individus et de l'environnement [42], [43].

Cette définition de la sûreté de fonctionnement est antérieure à l'apparition et à l'étude des aléas malveillants synonymes d'attaques contre les FMSs. Elle est toutefois transposable aux aléas malveillants puisque les "conditions données" dans lesquelles le système doit être

capable de remplir ses fonctions peuvent être étendues - au delà des défaillance et des pannes - à des conditions d'attaques. Dans le domaine de la sûreté de fonctionnement, les méthodes et algorithmes utilisés sont généralement intégrés sous la forme de lois de commande au sein même des composants de contrôle-commande du FMS (contrôleur, API, API de sécurité, SCADA, etc...). Néanmoins, comme nous le montrerons dans la suite de ce manuscrit, ces composants sont vulnérables aux attaques, et les mécanismes de sûreté telle que la surveillance, le diagnostic ou la reconfiguration ne peuvent pas être considérés comme robustes aux aléas provoqués intentionnellement [24]. Par conséquent, de nouvelles méthodes doivent être développées pour faire face à ces aléas intentionnels en s'appuyant sur des composants et algorithmes non vulnérables aux attaques. Ces méthodes justifient la création d'un nouveau sous-objectif de sûreté, celui de la sûreté face aux aléas intentionnels, encadré en rouge dans la figure 1.7 et détaillé dans le prochain paragraphe.

La **sûreté face aux aléas intentionnels** caractérise la capacité du FMS à faire face à des attaques d'origine physique [27], [44], lorsque l'attaquant vient physiquement dégrader le fonctionnement du FMS, ou d'origine cyber [25]-[27], lorsque l'attaquant modifie le fonctionnement du FMS à travers la manipulation des composants numériques de celui-ci (contrôleurs, APIs, SCADA etc.). L'objectif de sûreté face aux aléas intentionnels désigne ainsi la science de la résilience du FMS face aux attaques ciblant ses outils de production et de contrôle commande. **La cyber-malveillance contre les FMSs est l'objet de nos travaux et sera approfondie dans le chapitre 2.**

Remarque 1.2.1. Il est à noté que les objectifs introduits précédemment sont interdépendants et s'influent mutuellement [19], [28]. Par exemple, la sûreté de fonctionnement des ressources influe sur les performances, une ressource non-disponible pouvant dégrader la flexibilité et la productivité du FMS. Dans la prochaine partie, l'objectif de disponibilité sera étudié à travers la définition et la modélisation de l'indisponibilité d'une ressource d'un FMS dans un contexte incertain. ⌟

1.2.2 L'indisponibilité des ressources

En fonctionnement normal d'un FMS, une ressource peut devenir indisponible en raison d'une défaillance ou d'une opération de maintenance préventive [41]. Dans le premier cas, la ressource est victime d'une panne ou d'une défaillance (e.g. casse d'un outils, actionneur non alimenté, actionneur non contrôlable, perte d'observabilité, pénurie de matière première) et elle entre dans un état dégradé ou indisponible via une décision de son contrôleur local transmise au module de supervision. Dans notre travail, nous ne considérons pas le fonctionnement dégradé d'une ressource. Ainsi, une défaillance entraine systématiquement l'indisponibilité de la ressource pour son allocation par le FMS. Dans le second cas, l'indisponibilité de la ressource est programmée par les opérateurs pour la réalisation d'une opération de maintenance préventive. Cette indisponibilité est pilotée directement depuis le module de supervision qui force le passage de la ressource dans un état indisponible via un ordre de mise à l'arrêt. Le schéma de communication de l'information sur l'indisponibilité d'une ressource est illustré dans la figure 1.8 et met en avant le caractère ascendant en rouge, en cas de défaillance, ou descendant en vert, en cas d'opération de maintenance, de cette information.

Lorsqu'une ressource devient indisponible, le module de supervision doit adapter sa loi de pilotage pour la réalisation tout de même des recettes en n'allouant uniquement que les ressources

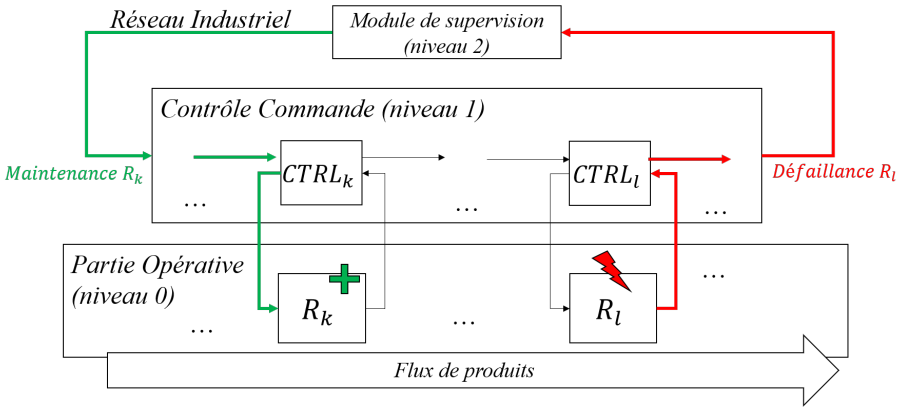


FIGURE 1.8 – Communication de l'information d'indisponibilité au sein du FMS

disponibles. On définit ainsi les modes de fonctionnement du FMS selon les ressources disponibles et indisponibles au sein du mode. Un mode noté $Mode_{\mathcal{R}_k}$ désigne un état de fonctionnement du FMS où les ressources $\mathcal{R}_k = \{R_{k_1}, R_{k_2}, ..., R_{k_3}\}$ sont indisponibles. Le $Mode_0$ désigne le mode où toutes les ressources sont disponibles. Un changement de mode est défini comme le passage d'un $Mode_{\mathcal{R}_i}$ à un second $Mode_{\mathcal{R}_j}$ où $\mathcal{R}_i, \mathcal{R}_j$ sont deux ensembles distincts de ressources indisponibles. Un changement de mode est la conséquence de la mise à l'arrêt et/ou de la reprise de ressources du FMS et est noté $Mode_{\mathcal{R}_i,\mathcal{R}_j}$.

Afin de pouvoir maintenir sa fonction de pilotage en cas de changement de mode, le FMS doit s'appuyer sur un modèle de l'allocation des ressources prenant en compte l'indisponibilité des ressources [12], [41], [45], [46]. Soit $N = (P_A \cup P^0 \cup P_R, T, F)$ le modèle S^{3}PR du FMS. Soit R_k une ressource modélisée par $p_{r_k} \in P_r$ et $H_{r_k} = p_r^{\bullet\bullet} \cap P_a$ l'ensemble des places opérations requérant R_k. La ressource R_k peut devenir indisponible dans deux situations : (i) lorsqu'elle est inactive et en attente d'être allouée et (ii) lorsqu'une opération la détenant est en cours [41]. Ces deux situations sont modélisées dans N à travers des sous-réseaux appelés réseaux de reprise (depuis une indisponibilité) [45].

Pour (i), soit R_k, modélisée par $p_i \in P_R$, une ressource du FMS pouvant devenir indisponible. On définit son réseau de reprise par le RdP $(N_{r_k}, M_{r_k0}) = (\{p_i, p_i^{ind}\}, \{t_{p_i}^{ind}, t_{p_i}^{dis}\}, F_{r_k}, M_{r_k})$ avec p_i^{ind} la place modélisant l'indisponibilité de la ressource lorsque $M(p_i^{ind}) = 1$, $t_{p_i}^{ind}$ et $t_{p_i}^{dis}$ les transitions modélisant par leur franchissement la mise à l'arrêt et la reprise de R_k, $F_{r_k} = \{(p_i, t_{p_i}^{ind}), (t_{p_i}^{ind}, p_i^{ind}), (p_i^{ind}, t_{p_i}^{dis}), (t_{p_i}^{dis}, p_i)\}$ les arcs reliant les éléments de N_{r_k} et $M_{r_k0}(p_i) = 1$ et $M_{r_k0}(p_i^{ind}) = 0$.

Pour le cas (ii), on définit un sous-réseau de reprise pour tout $p_j \in H_{r_k}$ sous la forme $(N_{p_j}, M_{p_j0}) = (\{p_j, p_j^{ind}\}, \{t_{p_j}^{ind}, t_{p_j}^{dis}\}, F_{p_j}, M_{p_j})$ avec $F_{p_j} = \{(p_j, t_{p_j}^{ind}), (t_{p_j}^{ind}, p_j^{ind}), (p_j^{ind}, t_{p_j}^{dis}), (t_{p_j}^{dis}, p_j)\}$ et $M_{p_j0}(p_j) = M_{p_j0}(p_j^{ind}) = 0$. Dans ce sous-RdP, le franchissement de $t_{p_j}^{ind}$ représente la mise à l'arrêt de la ressource lorsque l'opération O_j modélisée par p_j est en cours et celui de $t_{p_j}^{dis}$ la reprise de l'opération O_j lorsque la ressource R_k est de nouveau disponible.

Définition 1.2.1 (RdP de reprise $N_{R_{ind}}$)**.**
Soit $\mathcal{R}_{ind}$ l'ensemble des ressources pouvant devenir indisponibles dans N et $P_{\mathcal{R}_{ind}} \subset P_R$ l'ensemble des places ressources associées à $\mathcal{R}_{ind}$. On définit par $N_{\mathcal{R}_{ind}}$ le RdP incluant tous les sous-réseaux de reprises pour chaque ressource $r_k \in \mathcal{R}_{ind}$ et pour les cas (i) et (ii). ◇

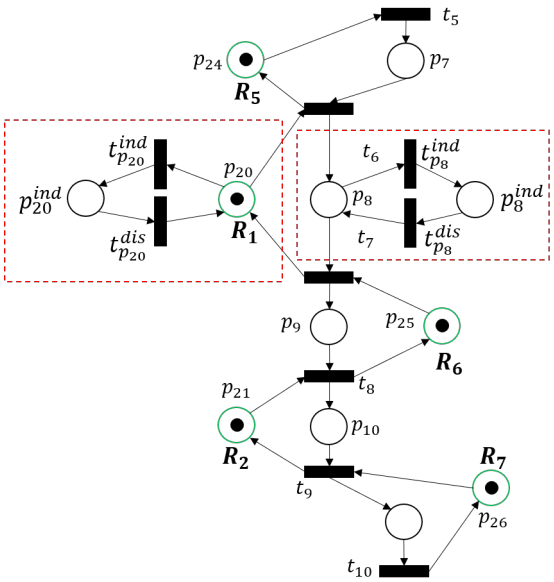


FIGURE 1.9 – Exemple de modélisation des indisponibilités dans le modèle S^{3}PR

Les sous-réseaux de reprise sont illustrés dans la figure 1.9. Dans cet exemple, présenté dans la section 1.1.3, on modélise l'indisponibilité de la ressource $R_1(p_{20})$ par deux rectangles en pointillés rouges. Dans cet exemple, le cas (i) est modélisé par $p_{20}, p^{ind}_{20}, t^{ind}_{p_{20}}, t^{dis}_{p_{20}}$ et les arcs qui les relient. Quant au cas (ii), il est représenté par $p_8, p^{ind}_8, t^{ind}_{p_8}, t^{dis}_{p_8}$ et les arcs qui les relient. Nous supposons dans nos travaux que toutes les ressources peuvent devenir indisponibles suite à une défaillance ou à une opération de maintenance et selon les cas (i) et (ii), i.e. $P_{\mathcal{R}_{ind}} = P_R$.

L'environnement incertain du FMS se traduit par l'incapacité de ce dernier à prévoir un changement de modes et l'indisponibilité nouvelle d'une ressource. Lorsqu'un changement de mode a lieu, l'allocation des ressources est impactée par les changements suivants. Premièrement, les décisions d'allocation possibles changent, avec pour r_k indisponible, tout $t \in p^\bullet_{r_k}$ devenant infranchissable car $M(p_{r_k} \cup H_{r_k}) = 0$. Deuxièmement, les transitions non franchissables entrainent une évolution de l'ensemble des états accessibles $\mathcal{R}(N, M_{ind})$ avec M_{ind} l'état du FMS lorsque le changement de mode a lieu et $M_{ind}(p_{r_k} \cup H_{r_k}) = 0$. Troisièmement, un état du FMS peut devenir critique lors d'un changement de mode lorsqu'il devient un état de blocage, $M \in \mathcal{R}(N, M_{ind}) | \nexists t \in T, M' \in \mathcal{R}(N, M_{ind}), M[t > M'$, si les transitions préalablement franchissables depuis M ne le sont plus. Les états critiques de blocage sont présentés dans la prochaine section 1.2.3.

Dans nos travaux, le contexte incertain dans lequel évolue les FMSs sera pris en compte par notre méthode de sûreté face aux aléas intentionnels. De fait, un attaquant peut exploiter les indisponibilités des ressources pour faciliter son attaque ou rester discret sous couvert de changements de modes. Le chapitre 4 traitera de ce sujet.

1.2.3 États critiques des FMSs : les états de blocage

Pendant son fonctionnement, l'arrêt de la production d'un FMS est un état dégradant pour son objectif de productivité, la faisant de fait radicalement chuter. Au sein d'un FMS,

cet état d'arrêt peut être provoqué par des états de blocage ou de "deadlock". Les états de blocage, désignés comme états critiques pour l'objectif de productivité, sont inhérents aux caractéristiques de conception des FMSs [1]. Ainsi, 4 conditions nécessaires à l'existence d'états de blocage au sein d'un FMS peuvent être identifiées [1] :

- Condition d'exclusion mutuelle : une capacité d'une ressource ne peut pas être utilisée par plus d'une recette à la fois ;
- Condition de détention/attente : les recettes détenant des ressources peuvent en requérir de nouvelles ;
- Condition de non préemption : aucune ressource ne peut être retirée d'une recette qui la détient et l'utilise, et une ressource peut être libérée uniquement par une action explicite liée à la recette.
- Condition d'attente circulaire : deux recettes ou plus forment une chaine circulaire où chaque produit attend une ressource que le produit suivant au sein de la chaîne détient.

Dans les FMSs, les 3 premières conditions sont propres aux caractéristiques de conception du système et à ses ressources. Elles sont systématiquement vraies. La dernière condition définit donc un état de blocage du FMS lorsque cette dernière est vraie. Une mauvaise séquence de décisions d'allocation des ressources par le module de supervision peut conduire le FMS dans un état d'attente circulaire, donc de blocage, à partir duquel aucune reprise n'est possible sans modifier les circuits des recettes ou sans intervenir physiquement sur les ressources et sur les produits en cours de fabrication.

Remarque 1.2.2. Dans nos travaux, les états critiques du FMS relatifs aux interactions physiques entre les ressources et/ou les produits (e.g. collision entre deux bras de robots, collision entre deux produits) ne sont pas considérés et l'on suppose que de telles interactions ont été sécurisées lors de la phase de conception du FMS. ⌟

Dans un modèle S^3PR, trois types de blocages sont identifiés : les états de blocage, les états de pré-blocage et les états de blocage partiel.

Définition 1.2.2 (État de blocage)**.** Un état de blocage du FMS est un marquage bloquant (6.1.13) de son modèle S^3PR (N, M_0), à savoir un marquage $M \in \mathcal{R}(N, M_0) | \nexists t \in T, M' \in \mathcal{R}(N, M_0), M' \neq M$ tel que $M[t > M'$. ◇

Définition 1.2.3 (État de pré-blocage)**.** Un état de pre-blocage est défini par un marquages $M \in \mathcal{R}(N, M_0)$ à partir duquel le système ne peut atteindre que des états de blocage. Autrement dit, $\forall M' \in \mathcal{R}(N, M_0)$ tel que $\exists \sigma \in L(N, M)$ et $M[\sigma > M'$, on a que M' est un état de blocage ou de pré-blocage. ◇

Définition 1.2.4 (État de blocage partiel)**.** Un marquage M est un blocage partiel si M n'est pas un état de blocage, $M_0 \notin \mathcal{R}(N, M)$ et $\forall M' \in \mathcal{R}(N, M)$, M' n'est pas un marquage bloquant [47]. ◇

Un blocage partiel représente pour un FMS un ensemble de marquages connectés, à savoir qu'il existe une séquence de transitions reliant n'importe quelle paire de marquages de cet ensemble, depuis lesquels l'état initial n'est pas accessible (non réversibles). Dans un système manufacturier, l'état initial représente un état connu et stable dans lequel le système souhaite

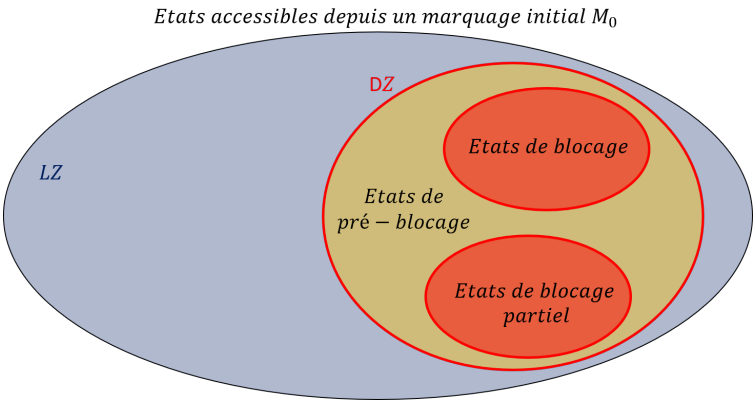


FIGURE 1.10 – Représentation de la LZ et de La DZ

être arrêté et démarré [48], ne plus pouvoir l'atteindre représente donc une situation critique pour la productivité du FMS. Au sein du modèle S^{3}PR, un blocage partiel correspond à un état du FMS où une partie des recettes est bloquée à cause d'une condition d'attente circulaire entre leurs produits tandis que le reste des recettes est encore libre.

Ces trois types d'états, blocage, pre-blocage et blocage partiel définissent la zone "morte" du FMS ou deadzone (DZ) [49], [50] et représentent l'ensemble des états critiques que doit éviter le système lors de l'allocation des ressources.

Définition 1.2.5 (Zone morte et Zone vivace)**.** La DZ est définie par l'ensemble d'états $\mathcal{M}_{DZ} = \{M \in \mathcal{R}(N, M_0) | M_0 \notin \mathcal{R}(N, M)\}$, à savoir tous les marquages non réversibles de (N, M_0).

Dans $\mathcal{R}(N, M_0)$, l'ensemble des marquages n'appartenant pas à $\mathcal{M}_{DZ}$ font partie de la zone vivace ou live-zone (LZ) avec $\mathcal{M}_{LZ} = \{M \in \mathcal{R}(N, M_0) | M_0 \in \mathcal{R}(N, M)\}$. ◇

La séparation de l'ensemble des marquages accessibles par (N, M_0) entre DZ et LZ est illustré dans la figure 1.10. Dans l'exemple de la figure 1.6, l'ensemble des marquages accessibles (1650 marquages) se décompose en 998 marquages réversibles au sein de la LZ, et 652 marquages non réversibles de la DZ dont 24 marquages bloquants, 468 de pré-blocage et 160 de blocage partiel.

Outre la définition des états de la DZ introduite à travers l'ensemble des marquages accessibles de (N, M_0), la classe des S^{3}PRs possède un ensemble de propriétés sur les états de blocage relatif à un élément structurel des RdPs, les siphons [17], [51], [52].

Définition 1.2.6 (P-invariant)**.** Dans un Rdp N, un $|P|$-vecteur non nul $I : P \to \mathbb{Z}$ est un $P - invariant$ si $I \geq 0$ (tous les éléments de I sont positifs) et $I^T \times [N] = 0^T$. Le support d'un $P - invariant$ I est définit par $\| I \| = \{p \in P | I(p) \neq 0\}$. En d'autres termes, le support d'un $P - invariant$ est un ensemble de places dont le marquage $M(\| I \|)$ est constant. ◇

Définition 1.2.7 (Siphon)**.** Un siphon est une ensemble non vide de places $S \in P$ tel que ${}^\bullet S \subset S^\bullet$. L'ensemble de tous les siphons de N est noté Π. Un siphon S est dit minimal si $\nexists S' \in \Pi | S' \subset S$. Un siphon est dit strict si il n'est le support d'aucun $P - invariant$. Un siphon strict minimal est noté SMS pour "Strict Minimal Siphon". ◇

Théorème 1.2.1. *Soit (N, M_0) un RdP ordinaire et Π l'ensemble de ses siphons. Le RdP est*

sans-blocage si $\forall S \in \Pi, \forall M \in \mathcal{R}(N, M_0), M(S) > 0$.

Théorème 1.2.2. *Soit* (N, M_0) *un RdP ordinaire et* M *est un état de blocage de* N*. Alors,* $S = \{p \in P | M(p) = 0\}$ *est un siphon.*

Corollaire 1. *Un* S^3PR *dans un état de blocage contient au moins un siphon vide.*

Les preuves de ces différents théorèmes sont détaillées dans les travaux de [51]. Si l'on reprend l'exemple précédent de la figure 1.6, on peut identifier le siphon strict minimal suivant $S = p_1 1 + p_1 6 + p_2 3 + p_2 6$. Lorsque le FMS atteint le marquage $M = p_1 + 3p_2 + 4p_3 + p_4 + p_5 + p_7 + p_8 + p_1 2 + p_1 4 + p_1 5$, S est vide ($M(S) = 0$) et le système est dans un état de blocage.

Dans nos travaux, les états critiques de blocage constituent les états ciblés par les aléas intentionnels que nous considérons et seront par conséquent définis dans le prochain chapitre.

Dans cette partie, l'environnement des FMSs a été présenté et a permis de mettre en exergue les différents éléments influençant le processus d'allocation des ressources, à savoir les objectifs de commande, la disponibilité des ressources et les états de la zone de blocage. Face à ces éléments, le module de supervision est équipé d'un ordonnanceur et d'un superviseur dédiés. Ces deux blocs constitutifs du module de supervision sont présentés dans la prochaine partie de ce manuscrit au regard de la littérature récente et de leur déploiement dans un contexte incertain.

1.3 Ordonnancement et supervision des FMSs

Un FMS réalise sa mission première d'allocation de ressources au sein d'un environnement contraint, incertain et critique. Dans la partie suivante, les différentes méthodes existantes pour assurer cette mission au cœur du contexte présenté dans la partie 1.2 sont répertoriées et exposées. Premièrement, pour répondre aux objectifs de productivité, les méthodes d'ordonnancement des FMSs sont présentées. Deuxièmement, le pilotage du FMS hors des états critiques est rendu possible par les méthodes de prévention, d'évitement et de détection des états de blocage. Ces différentes méthodes, appelées méthodes de supervision, sont exhibées dans la sous-partie 1.3.2. Enfin, les méthodes d'ordonnancement et de supervision prenant comme hypothèse la non-disponibilité plausible des ressources du FMS sont discutées en conclusion de cette partie.

La connaissance étendue des différentes méthodes de supervision et d'ordonnancement introduites dans cette partie permettra de justifier le positionnement et les caractéristiques des méthodes choisies et développées dans la suite de ce manuscrit pour faire face aux aléas malveillants considérés.

1.3.1 Méthodes d'ordonnancement

Les méthodes d'ordonnancement d'un FMS sont développées pour le calcul d'une ou plusieurs séquences de décisions d'allocation optimales. Selon les méthodes, la taille des séquences peut être unitaire, partielle - lorsque la séquence n'est pas unitaire mais ne permet pas de terminer tous les produits en cours et en entrée du FMS - ou totale - lorsque la séquence permet

la fabrication de tous ces produits -. L'optimalité de la séquence dépend d'une part des critères à optimiser et d'autre part, du degré d'optimalité recherché par la méthode, à savoir si cette dernière cible la meilleure séquence existante ou une séquence acceptable parmi les séquences possibles. Parmi les critères optimisables, on peut citer le temps d'exécution de tous les produits, le temps d'exécution moyen d'un produit, le temps de fabrication de tous les produits en cours, la cadence de production, le respect des dates d'échéances, la minimisation du retard, l'utilisation des ressources ou encore la réduction des temps de configuration des ressources [18], [53]-[55].

Dans cette sous-partie, les différentes méthodes seront classifiées selon le type de méthode utilisée (à base de règles, algorithmes de recherche, meta-heuristiques, apprentissage [53]) puis comparées selon la taille de la séquence calculable, l'optimalité de la méthode, la taille de l'espace de solutions parcouru et enfin le déterminisme du calcul, défini comme la propriété de calculer une solution d'ordonnancement identique pour deux appels distincts à la méthode avec les mêmes paramètres d'entrée. Par ailleurs, nous restreignons cette étude bibliographique aux méthodes d'ordonnancement fondées sur les RdPs afin de garder une cohérence et une capacité de comparaison entre les modèles proposés précédemment et les séquences d'ordonnancement calculées par ces méthodes. Par conséquent, une solution d'ordonnancement sera notée σ_f en tant que séquence de transitions à franchir dans un RdP.

Les méthodes à bases de règles sont appelées méthodes de dispatching et reposent sur des règles de répartition des ressources disponibles aux produits en cours. D'un point de vue du RdP, la méthode connait l'état actuel du FMS, les transitions franchissables, les produits en cours et les places ressources disponibles et décide parmi les transitions franchissables laquelle franchir selon les règles de dispatching qu'elle suit. Les intérêts principaux des méthodes de dispatching sont leur adaptabilité et leur réactivité face aux événements incertains du FMS [53]. Certains auteurs se sont intéressés plus précisément à ces incertitudes en utilisant des modèles de RdP stochastiques et prenant en compte les indisponibilités de ressource [56]-[58]. Implémentée en ligne, une méthode de dispatching offre aussi un faible temps de calcul car la séquence calculée est unitaire et ne demande que l'application unique des règles lorsque le FMS entre dans un nouvel état. Cependant, ces méthodes calculent des solutions non-optimales et parfois non-déterministes lorsque des règles stochastiques sont utilisées pour résoudre les conflits entre transitions [59].

Les méthodes de recherche s'appuient sur différents algorithmes hérités de la théorie des graphes pour trouver un chemin acceptable au sein du graphe des marquages accessibles $\mathcal{RG}(N, M_0)$. A partir d'un état initial, ces méthodes s'appuient sur un ou plusieurs objectifs pour faire converger leur recherche. Les objectifs principaux sont : un état objectif du FMS à atteindre (dernier état du chemin), la taille du chemin souhaitée (ou taille de la séquence) et l'optimalité minimale souhaitée de la solution. Ces méthodes s'appuient sur des fonctions heuristiques pour réduire l'espace de recherche de chemins optimaux. Ces heuristiques classent les chemins partiels découverts lors de la recherche afin de sélectionner et d'explorer les plus prometteurs. La construction et le choix d'un heuristique dépend des critères à optimiser mais aussi de l'optimalité souhaitée de la solution. En effet, un heuristique est qualifié d'admissible s'il permet d'obtenir un chemin optimal en fin de recherche [60]. Ces algorithmes de recherche sont légions et on distingue pour l'ordonnancement des FMSs les algorithmes de recherche par faisceau [61], [62], les algorithmes de recherche A* [63], [64], les algorithmes de Séparation et évaluation [65], [66], ou encore les algorithmes de recherche à arbre de Monte-Carlo [67]. Les méthodes de recherche peuvent être hybridées entre elles afin d'obtenir de nouvelles méthodes

plus rapides, plus optimales ou utilisant moins d'espace mémoire. Par exemple, les algorithmes de recherche A* peuvent être combinés avec des méthodes de parcours en profondeur, de retour sur trace ou encore de fenêtres dynamiques [53], [64], [68].

Les méthodes des méta-heuristiques sont des méthodes de recherche utilisant en entrée une ou plusieurs solutions possibles au problème d'ordonnancement et améliorant ces solutions jusqu'à atteindre un ordonnancement acceptable [53], [69], [70]. Les méthodes de méta-heuristiques sont utilisées pour résoudre des problèmes d'ordonnancement complexes, par exemple lorsque la taille du graphe des marquages accessibles ou le nombre de produits à fabriquer sont élevés, en un temps réduit. Elles s'appuient majoritairement sur des algorithmes itératifs stochastiques et elles ne permettent donc pas l'optimalité de la ou les solutions finales calculées. Parmi les méthodes méta-heuristiques utilisant les RdPs pour l'ordonnancement de FMSs, on distingue les méthodes de Scatter-Search [71], [72], de Tabu-search [73], de Simulated Annealing [74] ou encore les méthodes évolutionnistes telles que le Genetic Algorithm (GA) [75]-[79], l'Ant Colony (AC) [80], [81] ou le Particle Swarm Optimization (PS0) [82], [83].

Enfin, des méthodes d'ordonnancement fondées sur des **algorithmes d'apprentissage** ont été proposées récemment [84]-[88]. Ces méthodes permettent de trouver des solutions non-optimales et non-déterministes pour des problèmes d'ordonnancement complexes au sein des RdPs. L'ensemble des méthodes présentées précédemment sont rassemblées et comparées dans le tableau 1.2. Dans un futur chapitre, ce tableau comparatif nous permettra de justifier à l'aune de nos hypothèses de travail et de notre problématique de recherche le choix et le développement d'une méthode d'ordonnancement implémentée au sein du module de supervision du FMS et responsable de son pilotage.

Tableau 1.2 – Tableau de comparaison des familles de méthodes d'ordonnancement

	Règles de Dispatching		**Méthode de recherche**		**Méta-heuristique**	**Apprentissage**
	Règles stochastiques	**Règles déterministes**	**Admissible**	**Non Admissible**		
Taille	Unitaire	Unitaire	Totale	Totale	Partielle/Totale	Partielle/Totale
Optimalité	Non-optimale	Non-optimale	Optimale	Sous-optimale	Sous-optimale	Sous-optimale
Espace de solutions	Nul	Nul	Élevé	Moyen	Moyen	Moyen
Déterminisme	Stochastique	Déterministe	Déterministe	Déterministe	Stochastique	Stochastique

1.3.2 Méthodes de supervision

Les méthodes de supervision des FMSs ont pour objectif le pilotage robuste et le maintient du fonctionnement du FMS en présence d'états de blocage. Il existe trois types de méthodes : les méthodes de prévention, les méthodes d'évitement et les méthodes de détection puis reprise.

Méthodes de prévention

Les méthodes de **prévention** des états de blocage reposent sur une loi de commande construite hors-ligne, rendant les états de blocage inaccessibles en contraignant le processus d'allocation des ressources. Dans les modèles S^3PR, cette loi de commande se traduit par des places p_c appelées "moniteurs" qui autorisent ou non le franchissement des transitions auxquelles

ils sont connectés ($t \in T \cap p_c^\bullet$) selon les jetons qu'ils possèdent. Les places moniteurs gagnent en retour des jetons lors du franchissement de transitions $t \in T \cap {}^\bullet p_c$. Par conséquent, ces moniteurs interdisent les décisions d'allocation dangereuses pouvant conduire à un état bloquant. Dans la littérature, les méthodes de prévention sont classifiées selon la méthode utilisée pour construire ces moniteurs [1], [49].

D'une part, il existe les méthodes de prévention construite à partir du graphe des marquages accessibles $\mathcal{RG}(N, M_0)$, dans lequel les états de la DZ sont identifiés, interdits et à partir desquels des moniteurs sont dérivés grâce à différentes théories (Theorie des régions [49], First Bad Marking et invariants de place [50], [89]). Ces méthodes sont totalement permissives car elles ne contraignent que les états de la DZ et aucun état de la LZ. Cependant, elles souffrent d'un temps de calcul très élevé, conséquence de l'exploration complète de $\mathcal{RG}(N, M_0)$, et ne sont pas adaptées aux systèmes S^3PR ne possédant pas, par conception, de superviseur optimal [1].

D'autre part, les moniteurs peuvent être calculés en s'appuyant sur l'analyse structurelle du modèle S^{3}PR et de ses siphons. Ces méthodes s'appuient sur le théorème 1.2.1 et ont pour objectif de construire des moniteurs contrôlant les siphons de Π à ne jamais être vides, i.e. $\forall M \in \mathcal{N}, \mathcal{M}_0, \forall S \in \Pi, M(S) > 0$ [51]. Ces méthodes sont évaluées selon leur complexité de calcul, leur complexité structurelle (nombre de moniteurs) et leur permissivité, à savoir leur capacité à contraindre le minimum d'états de la LZ possible. Afin de diminuer leurs complexités et d'améliorer leur permissivité, qui n'est jamais totale, les méthodes d'analyse structurelle s'appuient sur différents types de siphons (siphons élémentaires et redondants [90], base de siphons contrôlable [91]), différents algorithmes pour le calcul des siphons (énumération partielle des siphons [92], ressource-transition circuits [93], [94], GPMSE [95], [96]), en se fondant sur des structures particulières au sein des S^{3}PRs (S^{3}PR sans ξ ressource [97]) ou encore en améliorant leurs algorithmes de calcul (MIP [98], [99]). Des états de l'art de ces méthodes peuvent être trouvés dans les travaux de [1], [17], [52].

Méthodes d'évitement

Les méthodes **d'évitement** s'appuient sur le modèle S^{3}PR du FMS pour calculer en ligne les séquences de décisions pouvant être prises par le module de supervision à partir de l'état actuel du FMS, puis pour interdire parmi ces séquences celles menant à un état de la DZ. Les méthodes d'évitement peuvent être implémentées sur des systèmes de grande taille car ces dernières n'explorent que partiellement le $\mathcal{RG}$ du modèle S^{3}PR. En effet, les comportements d'allocation du FMS sont calculés sur une séquence de marquages de taille réduite et bornée. Néanmoins, tous les états de la DZ ne peuvent être totalement éliminés par une méthode d'évitement en raison de la taille réduite de la fenêtre d'exploration des séquences d'états accessibles et la non connaissance a priori de tous les états de la DZ [1]. Par exemple, un état de pré-blocage ne peut être caractérisé par une méthode d'évitement si tous les états de blocage le succédant n'ont pas été identifiés par cette même méthode. Certaines méthodes proposent de faire face à cette limite en proposant un calcul robuste de la taille minimum nécessaire de la fenêtre d'exploration pour découvrir systématiquement tous les états de blocage du système [100]-[105]. Ces méthodes sont qualifiées d'optimales car elle n'interdisent que les séquences menant aux états de la DZ et assurent de toutes les éviter. Cependant, elles s'appuient sur des propriétés structurelles particulières des S^{3}PR pour éliminer les états de pré-blocage [100]-[103] ou sur une connaissance a priori des états de blocage et de pré-blocage [104], [105]. Parmi les autres méthodes d'évitement existantes, nous pouvons citer les travaux fondés sur l'algorithme

de Banker de [106], [107], ceux de [108] sur les RdPs commandés ou encore ceux de [109] ayant recours à une nouvelle structure des RdPs appelée structure de circuits d'événements.

Méthodes de détection et reprise

Les méthodes de **détection et reprise** sont composées de deux étapes successives, la détection d'un état de la DZ dans lequel se trouve le FMS et la mise en place d'une procédure de reprise conduisant le système vers un état de la LZ. L'étape de reprise différencie les méthodes les une des autres et on distingue les méthodes requérant pour la reprise des stocks de régularisation [110], [111] de celles requérant la relaxation d'opérations via des transitions de reprise [112]-[115]. Dans la littérature, les méthodes de détection et reprise sont peu développées en raison du passage nécessaire du système dans un état dangereux de la DZ.

D'après l'étude bibliographique réalisée dans cette partie, une méthode de supervision des états de la DZ d'un FMS peut donc être caractérisée selon les critères de complexité de calcul, de complexité structurelle, de permissivité, d'optimalité, d'existence ainsi que selon les propriétés structurelles du S^3PR nécessaires à son application. Dans ce manuscrit, une méthode de supervision sera ultérieurement développée et implémentée au sein du module de supervision à partir de ces critères et selon la problématique et les hypothèses de travail prises dans les prochains chapitres.

1.3.3 Ordonnancement et supervision dans un contexte incertain

Les méthodes d'ordonnancement et de supervision d'un FMS sont intégrées au sein du module de supervision présenté dans la section 1.1.2 de ce chapitre. Elles évoluent donc dans l'environnement incertain des FMSs introduit dans la sous-partie 1.2.2 et doivent être capables de s'adapter aux indisponibilités des ressources. Dans cette partie, les approches référencées se fondent sur les RdPs ou plus généralement sur les outils de modélisation des SEDs.

Méthodes d'ordonnancement dans un contexte incertain

Les méthodes d'ordonnancement fondées sur les RdPs et prenant en compte l'indisponibilité des ressources font partie des méthodes d'ordonnancement dans un contexte incertain [116]-[119] où l'indisponibilité d'une ressource est une source stochastique d'incertitude. Dans la littérature, les méthodes d'ordonnancement dans un contexte incertain sont classifiées selon leur manière de faire face à un événement incertain, à savoir de manière préventive, réactive ou une hybridation des deux, en résultent les méthodes préventives-réactives et pro-actives-réactives [119].

Les **méthodes préventives** cherchent à calculer hors-ligne une solution d'ordonnancement robuste face aux indisponibilités des ressources non fiables. Par exemple, les travaux de [120] proposent une méthode de calcul d'ordonnancements robustes dans les RdPs stochastiques temporisés ainsi qu'une fonction d'évaluation du risque de déviation non contrôlable (lors de la défaillance d'une ressource par exemple) pour une séquence d'ordonnancement non robuste donnée. En reprenant le principe d'évaluation du risque de déviation, [121] présentent un algorithme d'ordonnancement fondé sur la recherche par faisceau locale au sein du $\mathcal{RG}$ où l'heuristique prend en compte le risque de déviation et la sévérité de la déviation dans son estimation du temps restant jusqu'à l'état de référence. Cette méthode est étendue par

[122] en intégrant à l'heuristique le risque de déviation des états de l'ordonnancement déjà explorés. L'algorithme de recherche par faisceau est aussi utilisé par [123] sur un FMS non fiable conjointement à une méthode de prévention des blocages à l'aide d'arcs inhibiteurs à intervalles. À partir d'automates temporisés stochastiques, [124], [125] ont développé une méthode générique d'évaluation de la robustesse d'un ordonnancement selon sa probabilité d'avoir un temps d'exécution inférieur à une certaine durée d'échéance en présence d'incertitudes. Enfin, les auteurs [74] utilisent des RdPs stochastiques généralisés pour évaluer les performances de leur méthode d'ordonnancement dans un contexte incertain. Leur méthode est fondée sur l'algorithme de recuit simulé appliqué à la sélection de règles de priorité pour chaque atelier de fabrication du système manufacturier considéré. La robustesse de leur solution est calculée par simulation en comparant ses performances avec celles de méthodes à base de règles de la littérature.

Les **méthodes réactives** réalisent l'ordonnancement en ligne à chaque changement d'état du FMS selon les événements incertains ayant lieu. La majorité de ces méthodes se fondent sur des règles de priorité appliquées aux recettes en cours pour prendre une décision d'allocation sans considération de l'ordonnancement futur du FMS [56], [58], [126], [127]. Parmi ces méthodes, [56], [58] utilisent respectivement les RdPs stochastiques temporisés et les RdPs colorés stochastiques temporisés pour évaluer les performances de leurs méthodes en présence d'incertitudes. Plus récemment, des travaux s'intéressent à l'ordonnancement réactif d'un FMS à partir des RdPs en calculant à chaque nouveau marquage atteint une séquence de transitions pouvant conduire le système de son état M à un marquage objectif M_{ref}. Cette séquence de transition est l'ordonnancement à proprement parlé et est calculée par des algorithmes de recherche par faisceau [41], [122], ou par la commande prédictive appliquée aux SEDs [128].

Les méthodes **préventives réactives** s'appuient sur un ensemble de solutions d'ordonnancement calculées hors-ligne selon différents scénarios de fonctionnement (avec ou sans indisponibilités) et échangées entre elles par l'ordonnanceur lorsque des événements perturbateurs ont lieu en temps réel. A notre connaissance, aucune méthode préventive-réactive d'ordonnancement n'a recours aux RdPs dans son développement.

Enfin, les **méthodes pro-actives réactives** suivent deux phases de fonctionnement, une phase hors-ligne pendant laquelle un ordonnancement complet est proposé, et une phase en ligne pendant laquelle l'ordonnancement est actualisé dès qu'un événement incertain a lieu. Dans leurs travaux, [129] ont recours à la recherche A* pour le calcul de l'ordonnancement hors-ligne et à une recherche robuste dynamique sur une fenêtre de profondeur d'ordonnancement donnée pour le ré-ordonnancement en ligne. L'heuristique utilisé dans cette méthode oriente les produits requérant ultérieurement une ressource défaillante vers les "ressources dépendantes de la défaillance les plus proches" afin de ne pas bloquer les autres produits ne requérant pas la ressource défaillante. Une seconde méthode pro-active réactive a été proposée par [130] et utilise un $\mathcal{RG}$ temporisé étendu conjointement à un algorithme de Djikstra pour le calcul de l'ordonnancement hors-ligne et la reconfiguration en ligne en cas d'indisponibilité d'une ressource en mettant à jour le $\mathcal{RG}$ temporisé étendu.

Méthodes de supervision dans un contexte incertain

A l'instar des méthodes d'ordonnancement, de nombreuses méthodes de supervision ont été développées en prenant en considération l'indisponibilité des ressources du FMS.

Parmi ces travaux, ceux fondés sur la **prévention** des états de la DZ, calculés à partir d'un modèle intégrant l'indisponibilité des ressources, ne sont pas adaptés aux modèles S3PR où la capacité des ressources est unitaire [131]. En effet, ces méthodes, à partir de l'étude du $\mathcal{RG}$ [47], [123] ou de l'analyse structurelle des siphons [8], [45], [132]-[141] et des circuits du RdP [131], [142], viennent interdire le lancement d'une opération requérant une ressource non fiable (pouvant devenir indisponible) depuis un état M si l'indisponibilité de la ressource ayant lieu à l'état M entraine un état de blocage partiel ou total du FMS. Si la capacité de la ressource est de 1, sa défaillance entraine systématiquement le blocage de toutes les recettes l'utilisant et pour contraindre ces états de blocage, les méthodes de prévention interdisent l'allocation de cette ressource à toute opération. Par ailleurs, ces méthodes sont peu permissives et inadaptées en présence de nombreuses ressources non fiables car elle contraignent les marquages non robustes aux indisponibilités pour assurer la vivacité du FMS en cas, par exemple, de défaillance, bien que ces marquages appartiennent initialement à la LZ du système sans indisponibilité. Néanmoins, certaines méthodes de prévention peuvent être adaptées aux modèles S^3PR aux capacités unitaires en étant capables de faire face aux indisponibilités en reconfigurant leurs moniteurs uniquement lorsqu'une indisponibilité a lieu [46], [143].

Les méthodes d'**évitement** des états de la DZ sont plus adaptées aux modèles S^3PR où la capacité des ressources est unitaire et où de nombreuses ressources sont non fiables car elles sont dynamiques et adaptent leur politique de contrôle au dernier état discret du FMS, en présence ou non d'indisponibilités de ressources. Preuve de leur capacité d'adaptation, la majorité des méthodes d'évitement prenant en compte les indisponibilités des ressources construisent leur politique de contrôle avec l'objectif de maintenir, lorsque des ressources deviennent indisponibles, la réalisation des circuits de production du RdP ne requérant pas ces ressources [7], [8], [138], [144]-[153]. En effet, l'indisponibilité d'une ressource R_{k_1} entraine le blocage des produits requérant ultérieurement cette ressource, et en détenant d'autre ressources $R_{k_2}, ..., R_{k_f}$ ces produits bloquent à leur tour les recettes ne requérant pas R_{k_1} mais requérant certaines ressources parmi $R_{k_2}, ..., R_{k_f}$. Nonobstant, certaines méthodes de prévention considèrent aussi cet objectif lors de la conception de leur moniteurs [131], [134], [142].

Remarque 1.3.1. À notre connaissance, il n'existe aucune méthode de détection et reprise prenant comme hypothèse l'indisponibilité des ressources. ⌟

La sélection, le développement et l'implémentation d'une méthode de supervision et d'une méthode d'ordonnancement au sein du module de supervision du FMS devra prendre en considération les incertitudes de fonctionnement liées à l'indisponibilité des ressources. Ainsi, les familles de méthodes robustes à ces indisponibilités présentées dans cette partie serviront de fondement à la construction d'un module de supervision capable de piloter un FMS dans un environnement incertain et répondant à la problématique de recherche introduite dans le prochain chapitre : **la sûreté face aux blocages intentionnels et malveillants d'un FMS**.

Conclusion

Dans ce premier chapitre, les FMSs ont été introduits en trois temps ; au regard de leur fonctionnement standard, de leur environnement et de leur résilience face à un environnement incertain. Ainsi, dans la partie 1.1, le fonctionnement des FMSs pour l'allocation des ressources aux recettes à produire, l'architecture de contrôle commande pilotant le FMS et une sous-classe des RdPs, les modèle S^3PRs, capables de modéliser le fonctionnement des FMSs, ont été présentés et approfondis. Néanmoins, le fonctionnement des FMSs reste influencé par l'environnement dans lequel il évolue, un environnement contraint, incertain et critique. Dans la deuxième partie 1.2, l'environnement des FMSs a été décrit. Celui-ci a été décomposé en un ensemble d'objectifs de commande que le FMS doit respecter et améliorer, en un ensemble d'incertitudes relatives aux indisponibilités des ressources et enfin en un ensemble d'états critiques que l'allocation des ressources ne doit pas atteindre : les états de blocage. Ces différents aspects environnementaux sont prépondérants dans la conception et le perfectionnement du FMS, pour lequel de nombreuses solutions sont développées afin de le rendre plus performant et résilient face à son environnement. Dans la dernière partie 1.3 de ce chapitre, une introduction bibliographique aux méthodes d'ordonnancement et aux méthodes de gestion des états de blocage des FMSs reposant sur la théorie des RdPs a premièrement été proposée. Dans un second temps, les méthodes d'ordonnancement et de supervision ont été analysées au regard de leur robustesse face aux indisponibilités des ressources. Dans la suite de ce manuscrit, un nouvel objectif de commande des FMSs, source d'incertitudes peu considérée dans ce domaine, est étudié : la sécurité des FMSs face aux cyber-attaques.

Chapitre 2

La cybermalveillance dans le domaine du pilotage des FMSs

Introduction

Dans ce deuxième chapitre, la problématique de la cyber-malveillance à l'encontre des FMSs est introduite. Dans une première partie (2.1), le contexte global de la cyber-malveillance ciblant les systèmes de contrôle commande industriels (ICSs) est illustré par les attaques réelles récentes et les vulnérabilités propres à ces systèmes face aux attaques. A partir de ce contexte général, une définition des attaques contre les ICSs est proposée et un horizon bibliographique des approches de cyber-sécurité existantes est introduit au regard des différentes menaces identifiées préalablement. La deuxième partie de ce chapitre (2.2) s'appuie sur le travail introductif réalisé dans le chapitre 1 pour répertorier les vulnérabilités d'un FMS, de sa boucle de contrôle et de son architecture, et définir les objectifs et les méthodes des attaques pouvant cibler les FMSs. Parmi ces objectifs, les attaques de blocage des FMSs réalisées par un attaquant expert sont retenues dans le cadre de nos travaux. Enfin, trois états de l'art de la littérature récente des systèmes à événements discrets (SEDs) sont élaborés dans la dernière partie de ce chapitre (2.3). Ces états de l'art, portant sur les attaques de blocage, la détection d'attaques expertes et le diagnostic commun de fautes et d'attaques dans les SEDs, permettront de mettre en lumière une problématique de recherche et des verrous scientifiques auxquels ce manuscrit est dédié.

2.1 Cyber-malveillance et cyber-sécurité des ICSs

Depuis le début des années 2000, les systèmes de contrôle commande industriels (ICSs) ont mis l'accent sur l'intégration de moyens de communication plus transparents et standardisés, en particulier au travers de protocoles tels que TCP-IP, permettant ainsi d'améliorer l'interopérabilité et la communication au sein de l'ICS, la réactivité face aux aléas de fonctionnement et donc, in fine, la productivité. Cette révolution a permis de connecter les différents systèmes entre eux, de les contrôler à distance depuis des terminaux isolés et de pouvoir disposer de la puissance d'analyse de serveurs informatiques modernes. Cependant, cette intégration des nouvelles technologies de l'information et de la communication ainsi que l'ouverture des technologies de l'OT aux réseaux IT accessibles hors de l'entreprise ont exposé les ICSs à de nouvelles vulnérabilités

allant de la non maîtrise de cette révolution rapide à celles de la malveillance [154].

Dans cette partie, la cyber-malveillance contre les ICSs est introduite au regard du contexte actuel - entre attaques réelles et vulnérabilités des composants de l'OT - de la définition d'une attaque et d'un attaquant et enfin selon les grandes familles de solutions de cyber-sécurité développées pour y faire face.

2.1.1 Cyber-malveillance contre l'OT : vulnérabilités et attaques réelles

L'existence de la cyber-malveillance ciblant spécifiquement les ICSs et les systèmes physiques qu'ils contrôlent est démontrée par les attaques réelles ayant eu lieu. Ainsi, de nombreux actes de malveillance portés spécifiquement contre les ICSs sont à déplorer ; STUXNET [155], en 2010, a été l'attaque qui a permis la prise de conscience internationale de cette nouvelle problématique et des dangers que représentent les attaques ciblant les systèmes physiques et leur contrôle. L'attaque STUXNET avait pour cible une centrale d'enrichissement d'uranium Iranienne - site clé pour le développement de l'arme nucléaire en Iran - et pour objectif le ralentissement des centrifugeuses responsables de l'enrichissement en cascade de l'uranium. Pour y parvenir, les APIs Siemens contrôlant les centrifugeuses étaient spécifiquement ciblées par un ver afin de modifier subtilement les consignes de vitesse, ce vecteur d'attaque étant révélateur de la maitrise de l'OT par les attaquants. Parmi les attaques ciblant les ICSs [156]-[159], la mise à l'arrêt malveillante d'un fourneau de fonte de l'acier en Allemagne (2014), les malwares Black-Energy (2015) et Industroyer (2016) ciblant le réseau électrique Ukrainien et sa coupure, le malware Triton (2017) provoquant le passage en mode d'arrêt d'urgence du système de sûreté d'une usine de pétrochimie Saoudienne, ou plus récemment (2021) la tentative malveillante d'empoisonnement de l'eau dans une station d'épuration en Floride à travers son SCADA [160], [161], sont autant d'exemples récents révélateurs de cette cyber-malveillance.

Dans le domaine du manufacturier, de nombreuses entreprises ont récemment été ciblées par des attaques affectant leurs outils de production et leurs performances. En 2018, le réseau informatique de l'entreprise Saint-Gobain a été contaminé par la cyber-attaque Not Peyta [159], [162], et des pertes résultantes de plus de 220 millions d'euros ont été estimées. En 2019, l'entreprise Norsk Hydro [159], [163], puis, en 2020, l'entreprise Honda [159], [164] ont toutes deux été contraintes de ralentir pendant plusieurs jours leur production sur certains de leurs sites à la suite de cyber-attaques. En 2021, a l'instar d'Honda et de Norsk Hydro, les quatre entreprises, WestRock [165], Beneteau [166], Bombardier [167] et Molson Coor [168], ont déclaré avoir été victimes de cyber-attaques ayant pour conséquence une dégradation voire l'arrêt de leurs moyens de production. Enfin, en 2022 [169], [170], les entreprises KP Snacks [171], Nordex [172], un fournisseur de Toyota [173], AGCO [174], Foxconn [175], JBS [176] puis Dole [177] en 2023 ont été victimes d'attaques affectant leur production. La sécurité des systèmes manufacturiers a par ailleurs été fragilisée en 2022 par deux attaques globales, la première exploitant les vulnérabilités d'une solution de supervision de réseau IT, Solarwinds [178], [179], et la seconde les vulnérabilités de l'outil d'historisation Log4Shell [180], [181] présents dans certains SCADA. Ces différents exemples d'attaques et de vulnérabilités mettent ainsi en exergue la conclusion suivante : les ICSs manufacturiers sont devenus des cibles privilégiées et vulnérables face aux cyber-attaques.

En outre, de nombreux rapports d'instituts professionnels spécialisés dans la sécurité des

technologies opérationnelles confirment et attestent cette tendance. Ainsi, selon un rapport de la société Claroty [182], [183], le domaine du manufacturier a été en 2020 le secteur d'activité le plus propice à la découverte de vulnérabilités au sein de ses composants de contrôle et de supervision. Plus récemment, 1 434 et 1 435 vulnérabilités ont été diagnostiquées en 2021 et 2022 selon Claroty dont 74% portant sur les composants OT, 20% avec une sévérité critique et selon Nozomi [184] 181 des 300 vulnérabilités critiques concernent le secteur manufacturier. Par ailleurs, parmi les vulnérabilités critiques découvertes lors du second semestre de 2022, 35% n'avaient pas encore de correctif début 2023 [185]. La vulnérabilité élevée des ICSs manufacturiers est de nouveau soulignée par un rapport de SANS [186] dans lequel un sondage réalisé auprès de 332 experts révèle que 23% d'entre eux estiment que le secteur manufacturier est l'un des domaines avec la plus grande probabilité de compromission et de dégradation de leur système OT.
Au delà des vulnérabilités de leurs composants, les couches OT présentent aussi des vulnérabilités liées aux pratiques de leurs opérateurs. Ainsi, l'entreprise Nozomi [184] a découvert plus de 32 000 adresses IPs de composants OTs être accessibles en utilisant des identifiants n'ayant pas été modifiés et sécurisés par les intégrateurs humains (e.g. "nprocnproc","adminadmin"). Soulignant la recrudescence des vulnérabilités de pratique, IBM X-force [187] révèle en 2022 que l'hameçonnage a été responsable de 38% des intrusions dans les couches OTs d'un ICS. Enfin, SCADAfence [188] révèle à partir d'une étude menée auprès de 3 500 experts de la cybersécurité que les erreurs humaines sont considérées, par 79% de ces derniers, comme le risque principal à l'origine de la compromission de systèmes OT.

Enfin, confirmant leur statut de cibles privilégiées, 33% des ICSs manufacturiers dotés des logiciels Kaspersky [189], [190] ont eu recours en 2022 à l'exclusion de menaces malveillantes à leur encontre tandis que deux nouveaux groupes d'activistes ont récemment été identifiés par Dragos [191] comme acteurs de la cyber-malveillance à l'encontre des systèmes manufacturiers. De surcroit, 58% des cyber-incidents ciblant l'OT auxquels a répondu IBM X-Force étaient relatifs à l'industrie manufacturière [187]. Plus globalement, le secteur manufacturier accumule 24.8% des attaques IT/OT confondues en 2022 contre 17.7% en 2020 [187]. Enfin, l'entreprise Trend-Micro [192] a réalisé en 2021 une étude auprès de 900 entreprises, aux Etats-Unis (300), en Allemagne (300) et au Japon (300), révélant que 90% d'entre elles ont été victimes d'un incident malveillants contre leur système OT, 89% de ceux-ci ont résulté en un arrêt de la production durant, pour 56% des incidents répertoriés, plus de quatre jours.

Les attaques récentes contre les ICSs et en particulier contre ceux du secteur manufacturier d'une part, et la découverte constante de vulnérabilités au sein des composants OTs et des pratiques humaines d'autre part, sont deux démonstrateurs de l'existence d'une cyber-malveillance croissante contre les systèmes manufacturiers. A partir de l'analyse des attaques réelles et des vulnérabilités connues, une définition et une taxinomie des attaques ciblant les ICSs sont proposées.

2.1.2 Définition de l'attaquant : caractéristiques et objectifs

Une cyber-attaque ciblant un ICS se définit de manière opérationnelle d'une part, selon les différentes étapes successives qui la caractérisent, et de manière fonctionnelle d'autre part, suivant l'objectif, les méthodes utilisées, la localisation et les vulnérabilités exploitées pour la réalisation de chaque étape de l'attaque. Dans nos travaux, ces deux représentations opérationnelles et fonctionnelles d'une attaques sont conservées. Elles sont toutes deux introduites dans cette

partie puis adaptées à la caractérisation des actes malveillants ciblant le fonctionnement des niveaux OTs d'un FMS.

Représentation opérationnelle

Dans la littérature, différents auteurs ont proposé une classification des étapes d'une attaque. En 2015, l'institut SANS-Survey propose un modèle de cyber-attaque fondé sur deux phases principales [193] : la préparation et l'exécution de la cyber-intrusion puis le développement et l'exécution de l'attaque contre l'ICS. L'intrusion se décompose selon les étapes de planning, de préparation, l'intrusion et de persistance, tandis que l'attaque se scinde selon les étapes de développement, de validation, et d'exécution de l'attaque en suivant les trois sous étapes d'activation, d'initiation et de support. En conclusion de leur travail, les auteurs présentent les différents objectifs d'une attaque (e.g. vol d'information, interruption de l'ICS, endommagement des composants). La matrice "MITRE ATT&CK Matrix for ICS" éditée par l'institut MITRE [194] propose un référencement de toutes les étapes pouvant constituer une cyber-attaque contre les ICSs. Pour la réussite de chaque étape, la matrice détaille une liste de méthodes, ou vecteurs d'attaque, pouvant être utilisées. Au total, douze étapes sont référencées : accès initial, exécution, persistance, escalade de privilèges, évasion, découverte, mouvement latéral, collection, contrôle commande, inhibition des fonctions de réponse, dégradation de la boucle de contrôle et conséquences de l'attaque. Enfin, Tuptuk et al. [26] définissent le cycle de vie d'une attaque selon les six grandes étapes de pré-intrusion (définition des objectifs, acquisition de compétences et de matériel, conception et implémentation, tests), d'intrusion ("insiders", ingénierie sociale, téléchargement furtif), de propagation (réseau interne, escalade de privilèges), de mise à jour, d'opération (vol de données, sabotage) et de nettoyage (camouflage de traces, non-détectabilité). Cette modélisation séquentielle et opérationnelle d'une attaque se reflète également dans la littérature au travers des approches de cyber-sécurité fondées sur le principe de propagation de l'attaque [195], [196].

Représentation fonctionnelle

Dans un second temps, chacune des étapes d'une attaque peut être définie de manière fonctionnelle. D'après les travaux de [197] s'appuyant sur six taxonomies de la littérature des attaques contre les ICSs manufacturiers, une étape de l'attaque peut ainsi être décrite selon les méthodes et vecteurs qu'elle utilise, selon sa localisation dans l'ICS - à savoir les composants ciblés - selon les vulnérabilités qu'elle exploite et selon son objectif final. La **localisation** et les **vulnérabilités** exploitées par l'attaque seront introduites dans la sous-partie 2.2.1 lorsque une analyse des vulnérabilités des FMSs et de leur architecture de contrôle commande sera conduite. Il est à noter que la localisation d'une étape peut être relative à des entités cybers (e.g. APIs, SCADA), physiques (e.g. machines, actionneurs) ou humaines (e.g. opérateur, "insider"). Dans la littérature, les **méthodes** utilisées par l'attaquant pour atteindre son objectif sont répertoriées dans les catégories suivantes [197] : l'injection de données erronées [198], [199], la suppression de données [198], [199], la manipulation de code ou injection de code [200]-[204], les attaques Man In The Middle (MITM) [205]-[207], les dénis de service (DoS) [26], [205], [207], les attaques de replay [26], [203], [208], la modification de fichiers [26], [27], [209], [210], l'espionnage (Eavesdropping) [207], [211]-[213], et le buffer overflow [205], [207], [214]. Ces différentes catégories d'attaques, utilisées conjointement et ajustées à l'ICS qu'elles ciblent, permettent à un attaquant de réaliser ses objectifs. Traditionnellement, les **objectifs** d'une cyber-

attaque sont catégorisés selon s'ils affectent l'intégrité, la disponibilité ou la confidentialité du système de contrôle commande et du système physique contrôlé [26], [205]. L'intégrité est définie comme l'état d'une entité physique (machine, produit, contrôleur) ou numérique (données, loi de commande), entre état normal, dégradé ou hors-service. La disponibilité caractérise la capacité d'une entité à réaliser correctement les tâches pour lesquelles elle est sollicitée et a été conçue. L'objectif de confidentialité est propre aux données et définit la capacité d'assurer la privacité de ces dernières. Dans le cas des attaques contre les ICSs, les objectifs sont majoritairement relatifs à la dégradation de l'intégrité et de la disponibilité du système physique [156], [199], [215]. Les objectifs ciblant le système physique contrôlé par un ICS manufacturier sont répertoriés par [197] d'après les conséquences ci-contre : la dégradation de la qualité produit, la panne des machines, l'arrêt de la production, les dommages environnementaux, la sûreté humaine, les délais opérationnels et la consommation d'énergie. En outre, les auteurs [197] référencent quatre objectifs ciblant les composants numériques de l'ICS : le vol de données, l'escroquerie financière, la disponibilité des composants et la confidentialité des données.

Notre positionnement

Dans nos travaux, ces deux représentations opérationnelles et fonctionnelles d'une attaque ont été conservées et adaptées aux systèmes d'étude : les niveaux OT des FMSs.

D'une part, une modélisation par étapes d'une attaque contre les ICSs a été sélectionnée et est illustrée dans la figure 2.1. Toutes les étapes y figurent, toutefois nous ne nous intéressons pas dans ce manuscrit aux phases d'intrusion et de propagation car ces dernières n'ont pas pour objectif premier la dégradation de l'intégrité et de la disponibilité du système physique. Nous prenons ainsi l'hypothèse que l'attaque a déjà compromis les composants de contrôle ciblés, et est en phase active. De surcroit, l'étape de conséquences est aussi prise en compte dans nos travaux puisqu'elle reflète les objectifs attendus après l'exécution de l'attaque. Néanmoins, la correspondance entre conséquences et objectifs ne peut pas être vérifiée ; certaines conséquences de l'attaque étant des dégradations supplémentaires sur l'ICS et le système physique non prévues par l'attaquant [197]. Enfin, la phase de préparation de l'attaque est étudiée dans ce manuscrit à des fins de construction de profils d'attaquant [26], [216]. Nous distinguerons un attaquant selon son origine (nation, association terroriste, concurrent industriel, cyber-criminels, hackers amateurs, "Hacktivists", Insiders) [26], [207], [216], ses compétences (expert, confirmé, amateur) et ses moyens humains et matériels (infinis, élevés, limités). Ainsi, il existe une infinité de profils d'attaquant et nous prenons l'hypothèse que, dans le cas ou deux profils distincts partagent le même objectif final, ils ne concevront pas systématiquement la même attaque pour l'atteindre. Les profils d'attaquant sont exploités dans la partie (3.1.1) de ce manuscrit.

D'autre part, la représentation fonctionnelle d'une attaque est restreinte à certaines méthodes et objectifs. Premièrement, seuls les objectifs associés à la dégradation de l'intégrité et de la disponibilité du système physique et de son contrôle sont retenus pour les étapes d'auto-développement et d'exécution. Dans le cas des FMSs, ces objectifs sont précisés dans la section 2.2.1. Deuxièmement, les méthodes d'attaque ne doivent être utilisées que pour la réalisation de ces objectifs. Cette représentation fonctionnelle est illustrée dans la figure 2.1 pour les étapes d'auto-développement et d'exécution.

La préparation et l'exécution des cyber-attaques ciblant les ICSs ont été définies et organisées au travers des méthodes qu'elles utilisent, de leurs localisation et des vulnérabilités

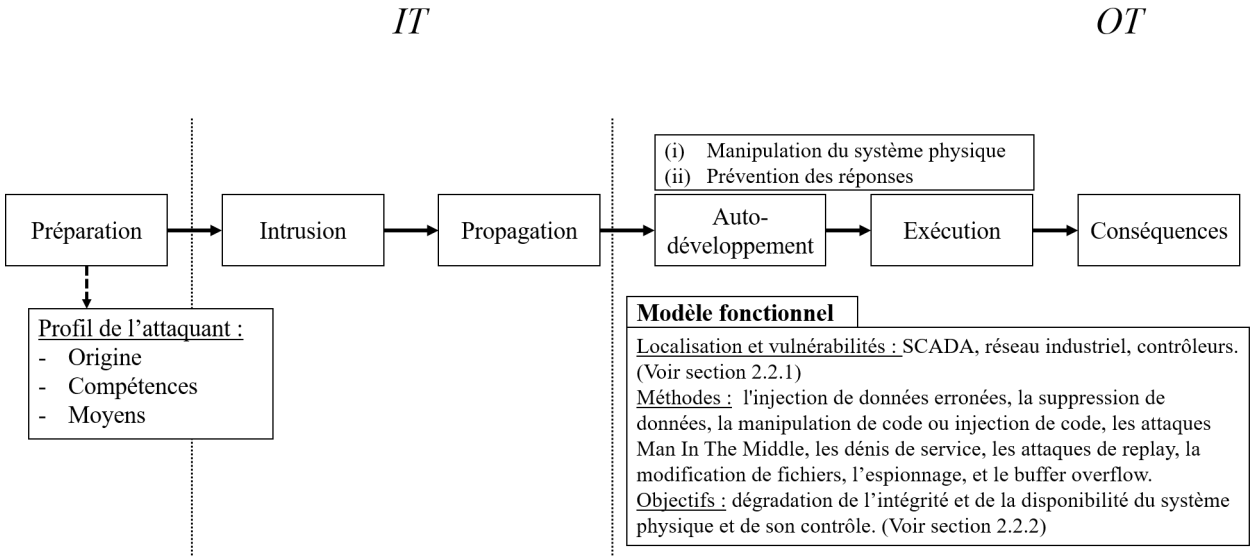


FIGURE 2.1 – Représentation opérationnelle et fonctionnelle d'une attaque ciblant un ICS

qu'elles exploitent, de leurs objectifs et conséquences ainsi qu'au travers des profils d'attaquant répertoriés. Dans la prochaine partie, une présentation des solutions existantes pour faire face à ces attaques est proposée.

2.1.3 Cybersécurité des systèmes industriels

Face aux attaques introduites dans la section précédente (2.1.2), les solutions de cyber-sécurité des ICSs s'organisent par rapport à l'étape de l'attaque à laquelle elles font face. Ainsi, on distingue les solutions d'analyse des vulnérabilités - identifiant les composants vulnérables de l'ICS avant l'attaque - les approches de prévention - dont le rôle est de prévenir l'auto-développement et l'exécution de l'attaque - les approches de détection - observant les comportements de l'ICS afin de détecter des anomalies lors de l'exécution de l'attaque - et les méthodes de réponse - qui lorsqu'un comportement anormal est détecté sont capables de mettre en œuvre une réponse corrective à l'attaque pendant ou après son exécution [154], [198] -. Le modèle par étapes et les localisations de ces dernières met également en exergue le phénomène de propagation de l'attaque à travers l'ICS, depuis l'extérieur de l'entreprise jusqu'au système physique ciblé et via les composants numériques des niveaux OT de l'architecture CIM (niveau supervision et niveau commande). En conséquence, les différentes catégories de solutions de cyber-sécurité sont présentées dans les prochains paragraphes au regard du comportement et/ou composant qu'elles cherchent à protéger, entre comportement continu ou discret et composant réseau ou de commande. Notons que les solutions de cyber-sécurité conçues pour les composants numériques du niveau 2 (e.g. SCADA) ne sont pas étudiées dans nos travaux car elles sont majoritairement implémentées sur des ordinateurs et serveurs relevant du domaine de l'IT. Ainsi seuls les comportements de l'ICS pilotés par le SCADA sont considérés.

L'analyse des vulnérabilités

L'analyse des vulnérabilités caractérise d'une part, la modélisation des attaques exploitant ces vulnérabilités et d'autre part, l'analyse du risque si ces vulnérabilités sont exploitées. La

modélisation d'attaques est développée pour le contrôle continu [198], [217], [218], discret [199], [219]-[224] ainsi que pour les APIs [200], [214], [225] et le réseau industriel de l'ICS [226]. Cette famille d'approches de cyber-sécurité consiste à construire des modèles, méthodes et scénarios d'attaques réalistes afin d'identifier les vulnérabilités des comportements et/ou composants étudiés. L'analyse du risque est une science héritée de la sûreté de fonctionnement cherchant à évaluer les risques et conséquences encourus par l'ICS lorsque des aléas ont lieu [24]. Dans le cas de la cyber-sécurité, ces aléas sont malveillants et s'apparentent à l'exploitation de vulnérabilités connues ou à des scénarios d'attaques réalistes contre le système à protéger. L'analyse du risque face aux aléas malveillants permet de quantifier les conséquences encourues par le système ciblé, d'évaluer et de comparer chaque aléa selon sa probabilité, sa complexité et sa gravité ainsi que d'identifier les composants et/ou comportements de l'ICS les plus critiques au regard des conséquences résultantes de leur compromission, devenant alors les composants et/ou comportements à protéger prioritairement [24], [154], [195], [196], [227], [228].

Prévention des attaques

La prévention des attaques a pour objectif d'empêcher l'occurrence d'une étape de l'attaque ou de l'une de ses méthodes. Dans les comportements continus [217], [229]-[231] et discrets [199], les méthodes de prévention se fondent sur la contrainte ou l'enrichissement d'états du système physique ou de la loi de commande exploités par l'attaquant et qui lui sont nécessaires pour atteindre ses objectifs. Par exemple, pour les comportements des SEDs, ces approches peuvent être appliquées à la prévention de méthodes d'espionnage à travers le principe d'opacité [232]-[235], à la prise en compte des attaques dans les observations pour rendre les systèmes d'estimation d'états robustes aux attaques [199], [236]-[238] ou encore à la contrainte d'états discrets accessibles par le système physique et au travers desquels l'attaque transite sournoisement pour atteindre un état critique du système [199], [239]-[243]. Afin d'éviter la compromission des APIs, les méthodes de prévention réalisent par exemple la vérification de la conformité du firmware [200], [201], [244] et de la loi de commande par rapport à son modèle [225], [245] ou à des règles de sécurité [246]-[249]. Enfin, la majorité des méthodes de prévention protégeant le réseau industriel proviennent du domaine de l'IT [250] et utilisent des technique de cryptographie [211], [251], [252], d'authentification [251], [253], de "défense par cible mouvante" [254], [255] au sein des protocoles de communication ainsi que des approches à base de règles [256]-[259] , n'autorisant que les communications fiables à être émises sur le réseau industriel.

Détection des attaques

Une méthode de détection ou ADS (Anomaly Detection System) est une approche dont l'objectif est de détecter et d'alerter lorsqu'une attaque a lieu sur l'ICS et risque d'endommager son intégrité ou sa disponibilité. Parmi les méthodes de détection, on distingue les méthodes de détection d'anomalies - révélant les comportements anormaux de l'ICS sans pouvoir certifier leurs caractères malveillants - et les méthodes de détection d'attaque - où les anomalies observées peuvent être caractérisées comme d'origine malintentionnée -. D'après les travaux de [260], [261], les ADSs peuvent être classifiés de la manière suivante : les ADSs à base de signatures, les ADSs à base de spécifications et les ADSs à base de modèles comportementaux.

La première catégorie se fonde sur des signatures d'attaques connues que l'approche cherche à détecter [262]-[264]. Ces signatures peuvent par exemple être des assertions ou des

comportements qui, s'ils sont observés et vérifiés, ne peuvent être associés qu'à un attaquant.

Deuxièmement, les ADSs à base de spécifications requièrent comme référentiel pour la détection un ensemble de règles que doit toujours respecter l'ICS [265]. Si une règle n'est pas satisfaite, une détection d'anomalie se produit. Il existe des ADSs à base de spécifications pour les comportements continus, [266]-[270], discrets, [271]-[276] ainsi que pour les composants de commande APIs ([273], [277]) et du réseau industriel (spécifications propres aux protocoles [263], [278]-[281], communications inter-composants [282], [283] et volumétrie du trafic [284], [285]).

Enfin, les ADSs à base de modèles s'appuient sur des modèles comportementaux des ICSs pour effectuer leur détection. Si, en temps réel, une déviation est observée entre le comportement réel de l'ICS et le comportement ordinaire précédemment observé et représenté dans le modèle, alors une anomalie est détectée sans distinction entre attaque et défaillance. Les comportements communément modélisés pour réaliser ce type de détection sont les comportement continus [198], [215], [286]-[290], les comportements discrets [199], [208], [291]-[300], les comportements propres au réseau industriel [261] et les comportements des canaux auxiliaires [301]-[304] - désignant les grandeurs physiques (e.g. température des moteurs/contrôleurs, vibrations) ou numériques (e.g. taux d'utilisation de l'environnement d'exécution, de la mémoire des contrôleurs) non contrôlées ou observées pour le contrôle commande mais dont l'évolution est affectée et peut être révélatrice d'une anomalie ou d'une attaque -. Les comportements des contrôleurs sont ici inclus dans les comportements continus et discrets de leurs lois de commande et des systèmes physiques qu'ils pilotent. Une classification des ADSs comportementaux a été proposée dans [305] pour les ICSs manufacturiers et est illustrée dans la figure 2.2. Cette classification définie un ADS comportemental selon quatre attributs principaux : le comportement modélisé, la nature du modèle, la méthode de modélisation et le cycle de développement de l'ADS.

Communément, les méthodes de détection sont évaluées selon leurs taux de faux négatifs (anomalie non détectée) et de faux positifs (détection d'une anomalie non existante) qu'elles tentent de minimiser. Parmi les trois familles d'approches, les méthodes à base de signatures et à base de spécifications sont écartées de nos travaux car elles ne sont pas capables de détecter les attaques dites "zero-day", autrement dit les attaques inconnues qui ne sont pas spécifiées dans leurs signatures ou couvertes par leurs règles. Ces attaques "zero-day" engendrent des faux-négatifs pour les méthodes à base de signatures et de spécifications, tandis que les approches à base de modèles sont capables de les détecter si ces attaques génèrent des déviations anormales du comportement modélisé au sein des ADSs. Toutefois, les approches à base de modèles ne sont pas capables de différencier une attaque d'une anomalie.

Réponse aux attaques

Les approches de réponses opèrent de manière réactive, dès qu'une détection d'anomalie a lieu, afin d'atténuer, de contenir voire de résoudre les conséquences résultantes de l'attaque. Dans la littérature, cette dernière catégorie de la cybersécurité des ICSs est étudiée de manière clairsemée [198], [272] en raison des craintes à autoriser le contrôle du système physique par un autre logiciel ou programme tiers.

Dans cette première partie du chapitre 2, le contexte de la cyber-malveillance ciblant les systèmes de contrôle commande industriels a été introduit. Un aperçu des attaques récentes et des différentes vulnérabilités affectant les ICSs a permis de mettre en lumière l'existence de cette malveillance. Subséquemment, celle-ci a été circonscrite à travers la définition et la taxonomie

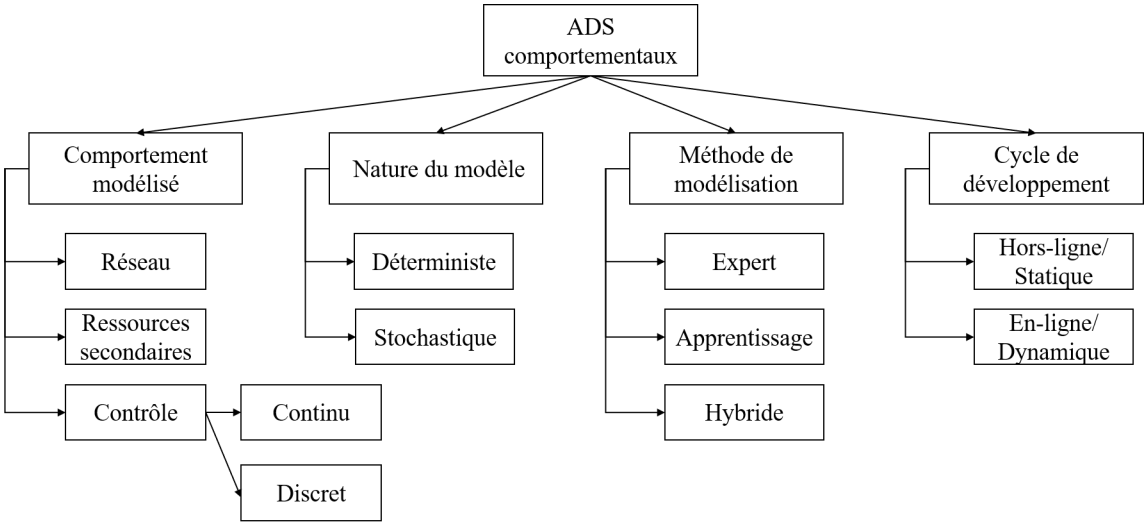


FIGURE 2.2 – Classification des ADSs comportementaux [305]

d'une attaque portée contre les ICSs, caractérisée par les étapes qui la compose et les attributs propres à chaque étape, à savoir : la méthode, les vulnérabilités, la localisation et l'objectif choisis par l'attaquant pour la réalisation de l'étape. Enfin, face à la malveillance OT, les solutions de cyber-sécurité dédiées à la protection des ICSs ont été présentées synthétiquement par une classification scindant la littérature selon les méthodes d'analyse des vulnérabilités, de prévention, de détection et de réponse. Dans la suite de nos travaux, cyber-malveillance, cyber-attaques et cyber-sécurité sont analysées et discutées à l'égard des FMSs.

2.2 Malveillance à l'encontre des FMSs

Les systèmes manufacturiers flexibles, présentés dans le chapitre 1, sont pilotés et supervisés grâce à des systèmes de contrôle-commande industriels démontrés vulnérables aux cyber-malveillance dans la partie 2.1. Dès lors, les FMSs sont aussi vulnérables aux cyber-attaques ciblant spécifiquement l'intégrité et la disponibilités de leurs comportements et composants opérationnels. L'objectif de cette sous-partie est d'étudier les vulnérabilités des FMSs et de leur architecture de contrôle commande, puis de proposer une caractérisation des attaques ciblant les FMSs et d'introduire parmi celles-ci les attaques de blocage du FMS. Enfin, face à cette nouvelle famille d'attaques, une analyse de la robustesse des méthodes de gestion des blocages existantes (sous-partie 1.3.2) est menée et permettra d'orienter notre choix parmi les classes de méthodes de cyber-sécurité introduites dans la section 2.1.3.

2.2.1 Vulnérabilités de l'architecture des FMSs

Initialement introduits dans le chapitre 1, les FMSs sont pilotés pour l'allocation des ressources grâce à une boucle de contrôle commande entre le module de supervision et les contrôleurs locaux, où ordres et observations d'allocation sont communiqués via le réseau industriel. Chacun de ces composants numériques, module de supervision, contrôleurs et réseau industriel, reposent sur un ensemble de technologies dont les vulnérabilités face aux cyber-attaques ont été introduites dans la partie précédente. En guise de synthèse, le tableau 2.1 liste

pour chaque composant les méthodes d'attaque auxquelles il est vulnérable et référence des exemples de vulnérabilités constructeurs exploitables par les différentes méthodes.

TABLEAU 2.1 – Méthodes d'attaque et vulnérabilités des composants numériques OT des FMSs

	Méthodes d'attaque	Exemples de vulnérabilités
Régulateurs	Injection de données, Suppression de données, Injection de code, Attaque de replay, Buffer overflow, Espionnage	[306]-[314]
APIs	Injection de données, Suppression de données, Injection de code, Attaque de replay, Buffer overflow, Espionnage, Modification de fichiers	[315]-[328]
Réseau Industriel	Injection de données, Suppression de données, Man In The Middle, Dénis de service, Espionnage,Attaque de replay	[329]-[336]
Module de supervision (SCADA)	Injection de données, Suppression de données, Injection de code, Attaque de replay, Espionnage, Buffer overflow, Modification de fichiers	[337]-[343]

Cependant, l'architecture de contrôle commande des FMSs n'est pas vulnérable dans sa totalité. **Nous prenons l'hypothèse que la communication bas-niveau entre un contrôleur local et la ressource qu'il contrôle n'est pas vulnérable**. Cette hypothèse est prise pour les deux raisons suivantes. D'une part, la sécurité de la boucle de contrôle locale permettant le contrôle et la régulation d'un système physique de taille limitée par un API ou un régulateur a fait l'objet d'un très grand nombre de travaux dans les cas continus [198], [215], [217], [286], [287] et discrets [199], [208], [238]-[243] (sous-partie 2.1.3). D'autre part, nous considérons le contrôleur local $CTRL_k$ introduit dans la sous-partie 1.1.2 comme le dernier composant numérique vulnérable connecté aux capteurs et actionneurs de la partie opérative, i.e. la ressource. Dans nos travaux, les capteurs et actionneurs ne sont pas des systèmes embarqués "intelligents", ils ne possèdent ni processeur, ni mémoire interne et ne sont qu'uniquement capables de convertir des signaux électriques en action physique et inversement. Ainsi, les signaux électriques envoyés aux actionneurs et générés depuis les capteurs ne peuvent pas être manipulés par un attaquant entre la partie opérative et le contrôleur local.

Remarque 2.2.1. Certains travaux [44] s'intéressent à la vulnérabilité des signaux électriques bas-niveau face aux attaques physiques dont l'objectif est de disposer à proximité des capteurs et actionneurs des générateurs de champs électromagnétiques capables d'ajouter du bruit aux signaux. Néanmoins, ces vulnérabilités étant de l'ordre des aléas malveillants physiques, elles ne sont pas considérées dans nos travaux. ⌟

Afin d'évaluer les conséquences de ces vulnérabilités sur le pilotage d'un FMS, la notion de **fiabilité** d'une donnée extraite est définie comme la fidélité de l'information portée par la donnée par rapport à l'état réel du système physique. Dans le cas d'une commande, elle est dite fiable si elle correspond in fine à un comportement attendu de la partie opérative et dans le cas d'une observation, si elle reflète avec certitude l'état réel de la partie opérative. Cette définition est extrapolée aux différents composants, lesquels sont qualifiés de fiables si les données extractibles

depuis ceux-ci sont fiables. Ainsi, au sein de l'architecture de contrôle commande des FMSs, et selon les vulnérabilités de ses composants présentées dans les paragraphes précédents, seules les données extraites depuis les communications entre contrôleurs et partie opérative ou directement depuis la partie opérative sont fiables. Le reste de l'architecture est considéré comme non fiable. La fiabilité des différents composants de l'architecture est illustrée dans la figure 2.3, où les composants rouges sont non fiables et les verts fiables.

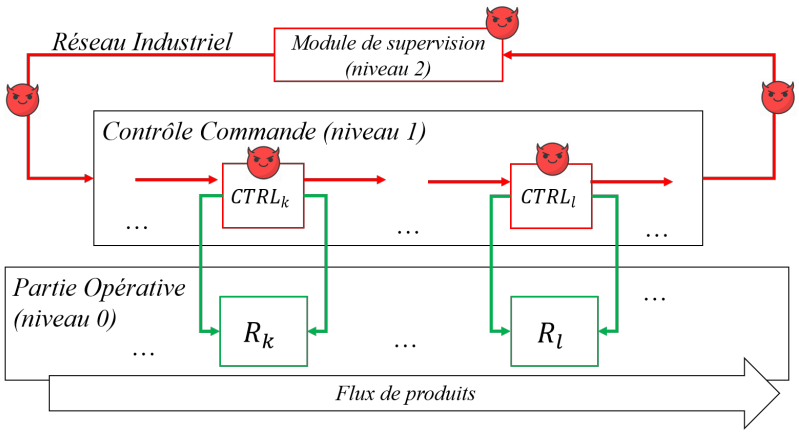


FIGURE 2.3 – Fiabilité des composants de l'architecture de contrôle commande d'un FMS

Dans cette sous-partie, les attributs de l'auto-développement et de l'exécution d'une attaque contre les FMSs, soit sa localisation, les vulnérabilités qu'elle exploite et les méthodes qu'elle utilise, ont été détaillés. Dans la prochaine partie, les différents objectifs d'attaque ciblant les FMSs sont définis et une nouvelle famille d'attaque est introduite : les attaques de blocage.

2.2.2 Objectifs et attaques de blocage

Les objectifs de l'exécution d'une attaque à l'encontre des ICSs ont été précédemment restreints aux conséquences dégradant l'intégrité et la disponibilité du système physique et de son contrôle (sous-partie 2.1.2). Or, au sein des FMSs, le système physique (le flux de produits), la partie opérative agissant dessus (les ressources) ainsi que la boucle rétroactive dédiée à son contrôle via le module de supervision et les contrôleurs, sont responsables de la qualité de la production (sous-partie 1.2.1) ; définie comme "la capacité de l'entreprise à assurer la livraison de ses commandes en temps, en quantité et en qualité désirés par le client en minimisant l'utilisation de ses ressources". Ainsi, la dégradation de l'intégrité et de la disponibilité du système physique d'un FMS et de son contrôle se rapportent à la dégradation de la qualité de la production. Dans la sous-partie 1.2.1, l'ensemble des objectifs de fonctionnement d'un FMS ont été définis comme constituants de l'objectif principal de qualité de la production. Par conséquent, la détérioration d'un objectif de fonctionnement aura pour conséquence la dégradation de l'objectif principal. Ainsi, pour un objectif de fonctionnement donné, le dessein de le dégrader est défini comme un objectif d'attaque. En généralisant, l'ensemble des objectifs d'attaque contre un FMS est défini comme la négation de ses objectifs de fonctionnement. Par exemple, l'objectif de qualité des produits est transposé à l'objectif malveillant de dégradation de la qualité des produits, pouvant être atteint en ciblant les différents objectifs de qualité du FMS en détériorant la prévention de la non qualité, en augmentant le nombre de produits réinjectés ou recyclés (produits défectueux), et en dépréciant la satisfaction client. En se fondant sur cette méthode de construction des

objectifs d'attaque contre les FMSs, toutes les conséquences d'attaque énumérées pour les ICSs manufacturiers [344] dans la sous-partie 2.1.2 peuvent être obtenus, à l'exception des objectifs relevant de l'IT.

Dans nos travaux, nous nous intéressons aux attaques dont l'objectif est la dégradation de la productivité à travers l'occurrence malveillante de blocages. Ce type d'attaque est appelé "attaque de blocage" et vise à manipuler la boucle de contrôle pour l'allocation des ressources du FMS afin d'atteindre un état critique de la zone "morte" $M_{DZ} \in \mathcal{M}_{DZ}$.

Lors de l'exécution d'une attaque de blocage, l'attaquant conduit le FMS d'un état de la zone vivace $M \in \mathcal{M}_{LZ}$ à un état de la DZ. Une attaque de blocage traverse donc une séquence d'états du FMS, appelée trajectoire, depuis M vers M_{DZ}. Lorsque plusieurs trajectoires possibles existent, l'attaquant doit faire un choix parmi celles-ci. Ce choix repose sur un ensemble de sous-objectifs considérés par l'attaquant lors de l'exécution de l'attaque.
Dans la littérature, les sous-objectifs possibles pour un attaquant sont relatifs (1) à la maximisation des conséquences ciblées [212], [242], [345] et (2) à la minimisation des coûts de préparation et d'exécution de l'attaque [223], [346]-[348]. Au regard des attaques de blocage, le sous-objectif de minimisation des coûts conduit l'attaquant à choisir la trajectoire menant à un état de la DZ lui demandant le moins d'investissements (e.g matériels, financiers, temporels, de compétences). Pour sa part, le sous-objectif de maximisation des conséquences mène l'attaquant à bloquer le maximum de recettes du FMS ou à bloquer des recettes prioritaires et à les maintenir bloquées le plus longtemps possible. Ainsi, la maximisation des conséquences se divise en deux sous-objectifs applicables aux attaques de blocages : (1.1) le choix du blocage et (1.2) la prévention de la réponse au blocage.

Premièrement, le choix du blocage est un sous-objectif qui se déduit de la variété des états de la DZ, entre états de pré-blocage, de blocage partiel et de blocage total. Par exemple, dans le cas où aucun état de pré-blocage ou de blocage n'est atteignable par l'attaquant pour des raisons de coûts de l'attaque (par exemple, si trop de contrôleurs locaux différents doivent être compromis), ce dernier pourra cibler les états de blocage partiel avec le plus de recettes bloquées.

Deuxièmement, le sous-objectif de prévention de la réponse désigne les caractéristiques de la trajectoire d'attaque choisie permettant de limiter les réponses des méthodes de cyber-sécurité existantes (2.1.3) à l'encontre de l'occurrence du blocage malveillant et de sa persistance. Ces caractéristiques de la trajectoire peuvent être définies par trois sous-objectifs adaptés aux attaques de blocage : (1.2.1) la sournoiserie, (1.2.2) la maximisation de l'utilisation des ressources et (1.2.3) la rapidité de l'attaque. La sournoiserie [212], [223], [237], [347], en leurrant le superviseur et les méthodes de cyber-sécurité sur le comportement réel du FMS, permet d'empêcher la détection et la prévention de l'attaque. La maximisation du nombre de ressources utilisées par le FMS au sein des états de blocage ciblés permet d'empêcher les méthodes de reprises d'utiliser les ressources non détenues comme stockage temporaire [113]-[115]. Enfin, la rapidité de l'attaque [345], en temps écoulé ou en nombre d'opérations ayant lieu entre le début et la fin de l'attaque, réduit l'intervalle temporel ou discret exploitable par les méthodes de prévention et de détection pour intervenir avant que l'état de blocage ne soit atteint. Tous les sous-objectifs d'une attaque de blocage sont illustrés dans la figure 2.4.

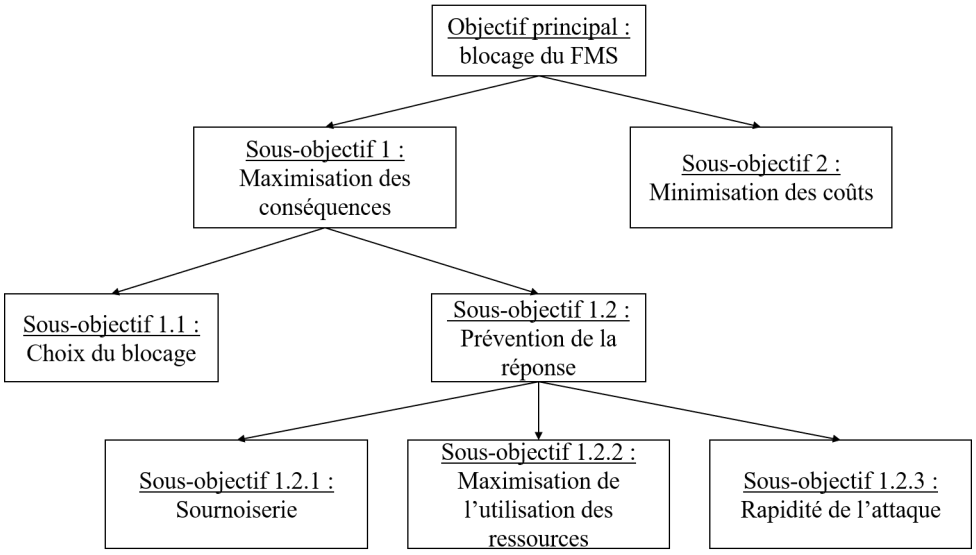


FIGURE 2.4 – Sous-objectifs d'une attaque de blocage

Dans cette partie, les objectifs d'attaque contre les FMSs ont été construits à partir des objectifs de fonctionnement formulés dans le chapitre 1. **Parmi ceux-ci, l'objectif de dégradation de la productivité a été choisi pour nos travaux portant sur les attaques de blocage de l'allocation des ressources d'un FMS. Au delà de cet objectif principal, un ensemble de sous-objectifs appropriés aux attaques de blocage ont été répertoriés. Parmi ces sous-objectifs, un attaquant choisira ceux qui correspondent le mieux à son profil (origine, compétences, moyens)**. Face à cette nouvelle forme d'attaques, la robustesse des méthodes de prévention et d'évitement des états de blocage présentées dans la partie 1.3.2 est analysée dans la prochaine sous-partie.

2.2.3 Robustesse des méthodes existantes face à une attaque de blocage

Les FMSs sont équipés par conception de solutions de gestion des états de blocage, pré-blocage et blocage partiel (sous-partie 1.3.2). Ces méthodes sont comparables dans leur fonctionnement aux approches de cyber-sécurité introduites au début de ce chapitre (sous-partie 1.1.3). Ainsi, les méthodes de prévention des états de blocage, à l'instar des approches de prévention des attaques contre les comportements discrets, cherchent à contraindre un ensemble d'états jugés critiques pour le système physique ou par rapport aux desseins de l'attaquant. Les méthodes d'évitement des états de blocage sont aussi comparables à des méthodes de prévention d'attaque, à la différence de contraindre les états critiques de manière dynamique. Enfin, les méthodes de détection et reprise fonctionnent de la même manière que les approches de détection comportementale et de réponse aux attaques. Toutefois, les approches de détection d'attaques se distinguent contrairement aux méthodes de détection d'états de blocage, par leur capacité à pouvoir détecter une attaque avant qu'elle n'atteigne l'état critique qu'elle cible, à savoir un état de la DZ dans nos travaux [199], [349], [350].

Dans cette partie, notre objectif est d'étudier la robustesse des méthodes de prévention et d'évitement de blocage face aux attaques de blocage. Cette étude permettra d'orienter notre choix vers une solution de cyber-sécurité appropriée à la problématique des attaques de blocage. En préambule à cette étude de robustesse, les définitions des RdPs labellisés, d'une politique de contrôle et d'une attaque contre un RdP sont introduites.

Définition 2.2.1 (RdP labellisé)**.** Soit $N_L = (N, E, l)$ un réseau de Petri labellisé, où N est un RdP, E est un ensemble de symboles , appelé alphabet, $l : T \to (E \cup \varepsilon)$ une fonction assignant pour tout $t \in T$ un symbole $e \in (E \cup \varepsilon)$ avec ε le symbole vide. Ici, (N_L, M_0) représente un N_L marqué avec M_0 le marquage initial de N. ◇

Dans nos travaux, E représente l'ensemble des événements se produisant au sein du système modélisé par N, où un événement est un élément déclencheur du passage de N d'un état discret à un autre état discret. Pour une séquence $\sigma \in T^\star$ et une transition $t \in T$, nous avons $l(\sigma t) = [l(\sigma), l(t)]$. En composant cette propriété itérativement, nous obtenons pour $\sigma = [t_0, t_1, t_2, ..., tk]$, $l(\sigma) = \lambda = [l(t_0), l(t_1), ..., l(t_k)]$. La fonction labellisante l peut ainsi être étendue à $l : T^\star \to E^\star$, avec $E^\star = \{l(\sigma) = \lambda | \sigma \in T^\star\}$. Nous définissons aussi la fonction inverse $l^{-1} : (E \cup \varepsilon) \to T$ tel que $l^{-1}(e) = \{t \in T | l(t) = e\}$. La fonction inverse peut aussi être étendue à $l^{-1} : E^\star \to T^\star$ avec $l^{-1}(\lambda e) = \{[l^{-1}(\lambda), t] | l(t) = e\}$ pour $\lambda \in E^\star, e \in (E \cup \varepsilon)$.

Définition 2.2.2 (Événements contrôlables, incontrôlables, observables et inobservables de E)**.** Nous définissons au sein de E les événements contrôlables E_c, incontrôlables E_{ic}, observables E_o et inobservables E_{io} avec $E = E_c \cup E_{ic} = E_o \cup E_{io}$. ◇

La notion de contrôlable (resp. observable) qualifie un événement que le superviseur est capable de contrôler pour changer l'état discret du système physique (resp. d'observer pour connaitre l'état discret du système physique), à savoir une commande (resp. une observation). On suppose par ailleurs que $E_c \cap E_o = \emptyset$, signifiant que l'application d'un événement de commande par le système physique n'est pas observable par le superviseur via le même événement mais peut toutefois être validée par une observation future. Dans un FMS modélisé par N_L, un RdP de classe S^3PR, toutes les transitions sont labellisées par des événements contrôlables symbolisant des décisions d'allocation et le lancement de nouvelles opérations.

Définition 2.2.3 (Fonctions de projection d'un événement d'observation)**.** Soient $P_o : E \to E_o$ les fonctions de projection d'un événement aux ensembles des événements observables et contrôlables avec $P_o(e) = e$ si $e \in E_o$ et $P_o(e) = \epsilon$ si $e \in E_{io}$. ◇

La fonction peut être étendue à $P_o : E^\star \to E_o^\star$ où pour $\lambda \in E^\star, e\ inE$, on a $P_o(\lambda e) = P_o(\lambda)P_o(e)$. Sa fonction inverse $P_o^{-1} : E_o^\star \to 2^{E^\star}$ est également définie telle que $P_o^{-1}(\lambda_o) = \{\lambda \in E^\star | P_o(\lambda) = \lambda_o\}$.

Définition 2.2.4 (Langage)**.** Le langage généré depuis N_L à partir d'un marquage $M \in \mathcal{R}(N_L, M_0)$ est défini par $\mathcal{L}(N_L, M) = \{\lambda \in E^\star | \exists \sigma \in T^\star, M' \in \mathcal{R}(N_L, M),\ t.q\ M[\sigma > M', l(\sigma) = \lambda\}$. On note $\mathcal{L}(N_L, M_0) = \mathcal{L}(N_L)$. Par ailleurs, nous désignons par $\mathcal{L}_o(N_L, M) = P_o(\mathcal{L}(N_L, M))$, le langage $\mathcal{L}(N_L, M)$ projeté à l'ensemble des événements observables E_o. A la manière de $L(N, M)_{\leq n}$, on définit par $\mathcal{L}(N_L, M)_{\leq n}$ le langage restreint aux mots de taille bornée par $|\lambda| \leq n$. ◇

Définition 2.2.5 (Suffixe et Préfixe)**.** Dans un langage $\mathcal{L}$, les préfixes d'une séquence d'événement

s sont les séquences $s_1 \in \mathcal{L}$ tel que $\exists s_2 \in E^\star, s = s_1 s_2$. On note $s_1 < s$. De la même manière, un suffixe de s dans $\mathcal{L}$ est une séquence $s' \in E^\star$ tel que $ss' \in \mathcal{L}$ et $s < s'$. On note $\mathcal{L} \setminus s = \{s' \in E^\star | ss' \in \mathcal{L}\}$ l'ensemble de tous les suffixes dans $\mathcal{L}$ après s. ◇

Dans nos travaux, nous considérons que **le système étudié a un comportement déterministe** qui peut être déduit d'une séquence d'événements de $E^\star$. Nous assertissons que depuis un marquage $M \in \mathcal{R}(N_L, M_0)$, si une séquence $\lambda \in E^\star$ a lieu, nous avons $\exists! \sigma \in T^\star | l^{-1}(\lambda) = \sigma$ et $M[\sigma > M'$ existe.

Définition 2.2.6 (Politique de contrôle)**.** Dans un superviseur, une politique de contrôle est une fonction $\rho : \mathcal{R}(N, M_0) \to 2^{E_c}$ qui associe à un marquage M des événements de commande autorisés et interdits avec $\rho(M)[i] = 0$ signifiant que le $i^{ème}$ événement de E_c est interdit et $\rho(M)[i] = 1$ que celui-ci est autorisé. Dans le RdP, ρ autorise ou interdit le franchissement de transitions. Dans $\mathcal{R}(N_L, M_0)$, nous définissons $\mathcal{A}_\rho(M) = \{(M, M') | \exists t_i \in T,\ tq\ M[t_i > M'\ et\ \rho(M)[i] = 0\}$ l'ensemble des arcs depuis M qui sont interdits par ρ. Nous notons $N_L|_\rho$ le RdP labellisé N_L controllé par ρ, $\mathcal{R}(N_L, M_0)|_\rho$ ses marquages accessibles, $\mathcal{L}(N_L, M_0)|_\rho$ son langage et $\mathcal{M}_\rho = \mathcal{R}(N_L, M_0) \setminus \mathcal{R}(N_L, M_0)|_\rho$ l'ensemble des marquages interdits par ρ. ◇

Dans les FMSs, les méthodes de prévention et d'évitement des états de la DZ reposent sur une politique de contrôle interdisant pour chaque état $M \in \mathcal{R}(N_L, M_0)$, avec N_L un modèle S^3PR labellisé, les événements d'allocation de ressources conduisant le système vers un état de blocage $M_{DZ} \in \mathcal{M}_{DZ}$. Les méthodes de prévention construisent ρ hors-ligne pour chaque marquage accessible de N_L tandis que les méthodes d'évitement calculent ρ dynamiquement pour le marquage M dans lequel est le FMS en ligne.

Définition 2.2.7 (Attaque contre un SED)**.** Soit un système discret modélisé par le RdP labélisé $N_L = (P, T, F, W, E, l)$. Une attaque contre ce système est définie par M le marquage de lancement de l'attaque, M_c le marquage critique ciblée par l'attaque, et par λ_a une séquence d'événements contenant des événements manipulés par l'attaquant et dont les effets sur le SED, représentés par la séquence d'événement $\lambda \in E^\star$, sont tels que $M[\sigma > M_c$ avec $\sigma = l(\lambda)$. ◇

Dans la littérature, les différentes manipulations d'événements discrets dont est capable un attaquant sont [199] :

- L'insertion d'un événement contrôlable ;
- La suppression d'un événement contrôlable ;
- L'insertion d'un événement observable ;
- La suppression d'un événement observable.

Définition 2.2.8 (Ensemble d'événements attaqués)**.** Soit E_{vul} l'ensemble de tous les symboles de E vulnérables aux attaques, $E_{vul} \in E$. Soit E_i l'ensemble des événements vulnérables aux attaques d'insertion et E_s l'ensemble des événements vulnérables aux attaques de suppression. Nous avons $E_i \in E$, $E_s \in E$ et $E_i \cap E_s$ n'est pas forcément vide. A partir des symboles vulnérables, de nouveaux symboles sont ajoutés à l'alphabet du système attaqué E_a . Ces nouveaux symboles représentent les événements attaqués. L'ensemble des événements insérés est noté $E_+ = \{e_+ | e \in E_i\}$ et l'ensemble des événements supprimés $E_- = \{e_- | e \in E_s\}$. Nous avons alors $E_a = E \cup E_+ \cup E_-$ et nous supposons que $E \cap E_+ \cap E_- = \emptyset$. ◇

Dans E_a, seuls les événements contrôlables et observables sont vulnérables aux attaques car les événements incontrôlables et inobservables n'interviennent pas dans la boucle de contrôle compromise par l'attaquant. Par ailleurs, certains auteurs [221], [348] considèrent une cinquième catégorie d'attaque contre les événements des SEDs appelée attaque de remplacement. Cette dernière consiste au remplacement d'un événement $e_1 \in E$ par un autre événement $e_2 \in E$, $e_1 \neq e_2$. Dans ce manuscrit, nous considérons qu'une attaque de remplacement est une simple succession d'une attaque de suppression et d'une attaque d'insertion symbolisée par $\lambda = e_{1-}e_{2+}$.

Définition 2.2.9 (Attaquant expert)**.** Un attaquant est dit expert si $\forall e \in E_o \cup E_c$, e_+ et e_- existent. ◇

Dans nos travaux, nous considérons l'attaquant toujours expert pour deux raisons. D'une part, nous nous focalisons sur le pire scénario d'attaque afin de protéger au mieux les FMSs. D'autre part, il nous parait plus réaliste qu'un attaquant capable de supprimer un événement est aussi capable de l'insérer, et vice-versa, et est apte à faire ces deux types d'attaques pour n'importe quel autre événement qu'il peut intercepter sur le réseau industriel ou depuis le module de supervision [351]. Pour des raisons de simplification rédactionnelle, une attaque experte désignera une attaque réalisée par un attaquant expert.

Définition 2.2.10 (Fonction de projection d'un événement attaqués)**.** Soit $f_a : E_a \to (E \cup \varepsilon) \times (E \cup \varepsilon)$ la fonction donnant pour chaque événement attaqué $e_a \in E_a$ l'événement correspondant ayant lieu dans le système physique et l'événement correspondant observé par le superviseur. Ainsi, nous définissons f_a par :

$$pour\ les\ évenements\ e \in E_c,\ f_a(e_a) = \begin{cases} (\varepsilon, e), & si\ e_a \in E_- \\ (e, \varepsilon), & si\ e_a \in E_+ \\ (e, e), & si\ e_a \in E \end{cases}$$

$$pour\ les\ évenements\ e \in E_o,\ f_a(e_a) = \begin{cases} (e, \varepsilon), & si\ e_a \in E_- \\ (\varepsilon, e), & si\ e_a \in E_+ \\ (e, e), & si\ e_a \in E \end{cases}$$

◇

Le premier élément de $f_a(e_a)$ est noté $f_a^1(e_a)$ et le second $f_a^2(e_a)$. Cette fonction peut être étendue à une séquence d'événements par $f_a : E_a^\star \to (E^\star \times E^\star)$ tel que $f_a(\lambda_a e_a) = ([f_a^1(\lambda_a), f_a^1(e_a)], [f_a^2(\lambda_a), f_a^2(e_a)])$ avec $\lambda_a \in E_a^\star, e_a \in E_a$. Les inverses des fonctions f_a^1 et f_a^2 sont définit sur $(f_a^1)^{-1} : E^\star \to 2^{E_a^\star}$ et $(f_a^2)^{-1} : E^\star \to 2^{E_a^\star}$ par $\forall \lambda \in E^\star$, on a $(f_a^1)^{-1}(\lambda) = \{\lambda_a \in E_a^\star | f_a^1(\lambda_a) = \lambda\}$ et $(f_a^2)^{-1}(\lambda) = \{\lambda_a \in E_a^\star | f_a^2(\lambda_a) = \lambda\}$. On définit alors la fonction inverse $f_a^{-1} : E^\star \times E^\star \to E_a^\star$ par $\forall \lambda_1, \lambda_2 \in E^\star, f_a^{-1}(\lambda_1, \lambda_2) = (f_a^1)^{-1}(\lambda_1) \cap (f_a^2)^{-1}(\lambda_2)$.

Remarque 2.2.2. Dans la suite de ce manuscrit, le système physique, soit l'allocation des ressources aux opérations et produits des recettes en cours, pourra être désigné par G et le superviseur, soit le module de supervision du FMS, par S. ⌟

Définition 2.2.11 (Langage de l'attaquant)**.** A partir de f_a, pour un RdP labellisé N_L et pour un marquage $M \in \mathcal{R}(N_L, M_0)$, le langage de l'attaquant est défini par $\mathcal{L}_a(N_L, M) = \{\lambda_a \in$

$E_a^\star | \exists \sigma \in T^\star, M' \in \mathcal{R}(N_L, M)$ *t.q.* $f_a^1(\lambda_a) = \lambda = l(\sigma)$ *et* $M[\sigma > M'\}$. Ce langage représente l'ensemble des capacités d'un attaquant à partir d'un état M du système. ◇

Définition 2.2.12 (Attaque sournoise)**.** Dans un système modélisé par N_L, une attaque discrète λ_a déclenchée à un marquage M est dite sournoise ssi $f_a^2(\lambda_a) \in \mathcal{L}(N_L, M)|_\rho$. On note $\mathcal{L}_a(N_L, M)|_{sr} = \{\lambda_a \in E_a^\star | f_a^1(\lambda_a) \in \mathcal{L}(N_L, M)$ *et* $f_a^2(\lambda_a) \in \mathcal{L}(N_L, M)|_\rho\}$ le langage des attaques sournoises depuis M. ◇

Ainsi, on a que $\mathcal{L}(N_L, M) \subset \mathcal{L}_a(N_L, M)|_{sr} \subset \mathcal{L}_a(N_L, M)$, puisqu'un attaquant peut réaliser les séquences d'événements non attaquées de $\mathcal{L}(N_L, M)$. A partir des définitions précédentes, la proposition suivante sur l'existence systématique d'une attaque de blocage experte contre un FMS est introduite puis démontrée.

Proposition 2.2.1. *Soit un superviseur conçu avec la politique de contrôle ρ pour prévenir ou éviter les états de blocage au sein d'un FMS modélisé par (N_L, M_0). Un attaquant expert est capable d'atteindre l'état $M_c \in \mathcal{M}_\rho$ depuis un marquage $M \in \mathcal{R}(N_L, M)|_\rho$ malgré ρ.*

Preuve 2.2.1. Soit $M \in \mathcal{R}(N_L, M_0)|_\rho$ l'état actuel du FMS et $M_c \in \mathcal{M}_\rho$, le marquage du FMS ciblé par l'attaquant. Nous savons que $N_L|_\rho$ est réversible ($\forall M \in \mathcal{R}(N_L, M_0)|_\rho, \exists \sigma \in T^\star | M[\sigma > M_0$) par définition de la DZ. Par conséquent pour $M_c \in \mathcal{R}(N_L, M_0)$, il existe au moins une séquence $\sigma_c \in T^\star$ telle que $M[\sigma_c > M_c$. Par exemple, $\sigma_c = \sigma_1\sigma_2$ où $M[\sigma_1 > M_0$ et $M_0[\sigma_2 > M_c$ est valide. Soit $\lambda_c = l(\sigma_c) = [e_1 e_2 ... e_k], k \in \mathbb{N}$ la séquence d'événements correspondant à la séquence de transition σ_c avec $\forall e \in \lambda_c, e \neq \emptyset$ car toute transition d'un modèle S^{3}PR est labellisée par un événement relatif à une décision d'allocation. L'attaquant étant expert, il peut manipuler tout événement $e \in \lambda_c$. Ainsi, l'attaquant a la capacité de créer une séquence $\lambda_a \in E_a^\star$ telle que $l^{-1}(f_a^1(\lambda_a)) = \sigma_c$ conduisant le système physique vers un état de blocage et $f_a^2(\lambda_a) \in \mathcal{L}(N_L, M)$ respectant une séquence d'observations attendue par le superviseur. Par exemple, la séquence $\lambda_a = [e_{a1} e_{a2} ... e_{ak}]$ avec $e_{ai} = e_{i+}, i \in [1, k]$ si $e_i \in E_c$ et $e_{ai} = e_{i-}, i \in [1, k]$ si $e_i \in E_o$, est une séquence d'attaques respectant ces deux conditions. ⌟

Considérons maintenant un attaquant qui souhaite réduire son coût de préparation. Ce dernier attend que le FMS atteigne un état adjacent à un état de la DZ, i.e. un marquage $M_1 \in \mathcal{R}(N_L, M_0)|_\rho$ tel que $\exists t_1 \in T, M_c \in \mathcal{M}_{DZ}$ and $M_1[t_1 > M_c$. Ces états adjacents sont fréquents dans les FMSs car un état de blocage correspond à un siphon dont les places ressources sont vides, donc détenues par des opérations, et simultanément, car l'objectif de productivité d'un FMS implique que le maximum de ressources soient utilisées par les recettes en cours. Ainsi, nous supposons que l'état adjacent M_1 est systématiquement atteint par le FMS pendant son fonctionnement. Le FMS étant supervisé, la politique de contrôle interdit l'arc (M_1, M_c) ($\rho(M_1)[t_1] = 0 \Leftrightarrow (M_1, M_c) \in \mathcal{A}_\rho(M_1)$). On a donc $e_1 = l(t_1) \in E_c$. L'attaque consiste ainsi à insérer l'événement $e_1(e_{1+})$ quand M_1 est atteint, puis à insérer k_1 événements d'observation $\lambda_{o_1} = e_{o_1}^1 ... e_{o_1}^{k_1}$ ($e_{o_1-}^1 ... e_{o_1-}^{k_1}$) généré par le système physique après e_1 pour respecter la séquence attendue par le superviseur. Notons qu'un marquage atteint après ces événements observables λ_{o_1} reste dans $\mathcal{M}_\rho$ puisqu'un marquage $M \in \mathcal{M}_\rho$ est soit un blocage partiel, soit ne mène qu'à des états de blocage.

Supposons maintenant un superviseur informé de cette attaque et capable de reconfigurer sa politique de contrôle afin d'interdire depuis tout $M_2 \in \mathcal{R}(N_L, M_0)|_\rho$ la transition $t_2, l(t_2) = e_2 \in E_c$ telle que $M_2[\sigma_2 > M_1$ où $\sigma_2 = t_2\sigma_{o_2}$ et $\lambda_{o_2} = l(\sigma_{o_2})$ la séquence d'événements

d'observation ayant lieu après e_2. La politique de contrôle est alors étendue à $\rho(M_2)[t_2] = 0$ ou $(M_2, M_2') \in \mathcal{A}_\rho(M_2)$ avec $M_2[t_2 > M_2'$. Conséquemment, le marquage M_1 n'est plus accessible ; Cependant, l'attaquant peut désormais lancer son attaque depuis M_2 en insérant e_2 (e_{2+}) et en supprimant λ_{o_2} (λ_{o_2-}). La nouvelle attaque est alors $\lambda_a = t_{2+}\lambda_{o_2-}t_{1+}\lambda_{o_1-}$. La politique de contrôle peut à son tour se reconfigurer pour rendre M_2 non accessible. Cela crée une boucle réactive entre l'attaquant et le superviseur ; l'attaquant modifie son attaque pour contourner la politique de contrôle et vice versa, le superviseur étend sa politique de contrôle pour prévenir les modifications de l'attaque. Toutefois, cette boucle se termine systématiquement car l'ensemble des marquages accessibles contraints par le superviseur augmente et est borné par $|\mathcal{R}(N_L, M_0)|$. Le superviseur va à terme contraindre le système physique à ne plus être assez permissif pour garder N_L réversible et vivace. L'attaquant a donc atteint son objectif initial, à savoir le blocage du FMS.

A l'instar des méthodes de prévention et d'évitement, les approches de détection de blocages centralisées sont inefficaces face à un attaquant expert capable de tromper le superviseur sur l'état réel de l'allocation des ressources et en particulier sur le passage de cette dernière dans un état de la DZ. Néanmoins, la détection d'attaques peut être développée à partir de données extraites depuis les différents composants de l'architecture du FMS, en particulier depuis des composants fiables. De surcroit, les méthodes de détection sont des méthodes passives n'ayant pas la capacité d'insérer des événements dans la boucle de contrôle pour l'allocation des ressources. Ces méthodes ne peuvent donc pas être manipulées par l'attaquant pour dégrader l'état du FMS et le conduire à un état de blocage.

Remarque 2.2.3. Dans le cas des méthodes d'évitement et de prévention distribuées [6], [7], [9], où chaque API responsable du contrôle d'une ou plusieurs ressources est doté d'une politique de contrôle globale ou locale, un attaquant expert sera toujours capable de réaliser son attaque de blocage. En effet, un API est un composant vulnérable (section 2.2.1) et les événements d'observation reçus depuis les ressources ou depuis d'autres APIs ainsi que les événements de commande envoyés aux ressources qu'il pilote peuvent être manipulés par un attaquant expert depuis ce même API ou depuis le réseau industriel et les autres APIs qui y sont connectés. ⌟

Dans cette partie, il a été montré qu'une politique de contrôle, même dynamique, n'est pas capable de prévenir ou d'éviter une attaque dont l'instigateur est expert. En conséquence, les méthodes de détection ont été préférées pour répondre à la problématique des attaques de blocage contre les FMSs. Dans la prochaine partie, un état de l'art des méthodes de détection d'attaques dans les SEDs est mené au regard des hypothèses et du positionnement qui ont été pris jusqu'alors.

2.3 Etat de l'art

Dans le chapitre 1 et dans les deux premières parties de ce chapitre, un ensemble de positionnements et d'hypothèses ont été établis afin de délimiter et d'orienter nos travaux.

- Premièrement, nous nous intéressons aux attaques de blocage conduites par un attaquant expert contre le comportement d'allocation de ressources du FMS (sous-partie 2.2.2). Rappelons que ce comportement est assimilable à un SED pouvant être modélisé par une classe des RdPs, les S^{3}PRs, et est piloté de manière centralisé par un module de

supervision équipé d'un ordonnanceur et d'une solution de gestion des blocages (partie 1.1) ;

- Deuxièmement, face aux attaques de blocage, nous avons choisi de nous concentrer sur les approches de détection à base de modèles comportementaux (sous-parties 2.1.3,2.2.3) ;
- Troisièmement, la détection des attaques de blocage est opérée dans un environnement incertain, d'une part en raison de la non fiabilité des données observées (sous-partie 2.2.2), et d'autre part à cause des indisponibilités de ressources (sous-partie 1.2.2). Ces deux sources d'incertitude ne permettent pas une détection totale - sans faux-négatif - et capable d'inférer correctement l'origine de l'anomalie - sans faux positif -.

Dans cette partie, trois états de l'art de la littérature récente sont menés afin de mettre en exergue les attaques de blocage, les méthodes de détection d'attaques expertes au sein des SEDs et les méthodes existantes de diagnostic de l'origine d'une anomalie, entre origine naturelle et malveillante. En conclusion de ce chapitre, ces trois états de l'art nous permettent de dégager une problématique et des verrous scientifiques auxquels nos travaux souhaitent répondre.

2.3.1 Etat de l'art des attaques de blocage des SEDs

Dans la littérature, la problématique des attaques de blocage contre un SED n'est jamais traitée comme le verrou principal auquel répondent les travaux qui s'y intéressent. Ainsi, les attaques de blocage sont introduites dans la littérature, soit comme la conséquence possible d'un autre type d'attaque contre les SEDs, soit comme une propriété que doit respecter un superviseur robuste face aux attaques, à savoir la vivacité du système supervisé ou le caractère non-bloquant du superviseur. Dans cette partie, un état de l'art sur les travaux de la littérature considérant les attaques de blocage selon les deux axes présentés ci-dessus est conduit.

L'objectif de cette première partie de l'état de l'art est de montrer l'absence de travaux considérant le blocage d'un FMS comme l'objectif principal d'une attaque.

Premièrement, différents travaux de la littérature [212], [223], [224], [347] traitent les attaques de blocage comme conséquences possibles d'autres scénarios malveillants. Pour des raisons de synthèse, seuls deux de ces travaux sont introduits dans cette partie.

Ainsi, [223], [347] font référence aux attaques de blocage dans les exemples utilisés pour illustrer une attaque souhaitant conduire le système d'un état M à un état M', où M' est un état de blocage dans leurs exemples. Dans leurs travaux, le système physique est modélisé par un RdP synchronisé avec sortie, dans lequel le franchissement des transitions est dépendant d'événements d'entrée (événements discrets contrôlables, événements discrets non contrôlables mais "irrémédiables") et où l'évolution du marquage de certaines places entrainent la génération d'événements de sortie (événements observables transmis au superviseur). A partir de ce modèle, un automate labellisé fini avec entrées est dérivé à la manière d'un graphe des marquages accessibles. Pour chaque arc du graphe que souhaite franchir l'attaquant pour atteindre l'état M' de manière sournoise, l'attaquant doit d'une part, insérer les événements contrôlables d'entrée et d'autre part, supprimer les événements observables des sortie par lesquels est labellisé cet arc. Afin de construire une attaque pertinente, les auteurs proposent le concept de coût associé à chaque arc du graphe que souhaite forcer l'attaquant. Cette notion de coût permet d'évaluer le

"mérite" qu'a un attaquant d'atteindre M' depuis M. Finalement, dans [347], les coûts associés à chaque arc sont considérés incertains et sont définis non plus par une valeur fixe mais par un intervalle de coûts. Ces intervalles de coûts permettent l'introduction des attaques de robustesse maximale, avec le plus petit coût possible pour atteindre un état interdit M', et de l'indicateur de regret, défini comme la différence entre le coût d'un scenario d'attaque existant avec le coût du scénario maximalement robuste. Dans ces deux références [223], [347], les attaques de blocage contre un système de production sont utilisées comme exemples pour illustrer le calcul du coût d'une attaque et l'étude des scénarios d'attaque maximalement robuste. Toutefois, les états de blocages ne sont considérés que comme des états critiques à interdire au sein d'un ensemble plus large d'états critiques $\mathcal{F}$. Aucune définition et modélisation précises d'une attaque de blocage contre un FMS n'ont été proposées.

Dans un système physique discret G modélisé par un automate, [212] expose une méthode de synthèse d'attaques "secrètes" contre les actionneurs et les capteurs capables d'atteindre un état endommageant de G sans se faire détecter par son superviseur S. Les auteurs construisent successivement un automate représentant le superviseur en présence d'attaques contre les capteurs et les actionneurs, un automate modélisant les capacités des attaques capteurs, un automate modélisant les capacités des attaques actionneurs, un automate composé des séquences d'événements dégradantes pour le système en présence d'attaques, un automate modélisant le moniteur du superviseur capable de détecter une attaque avant qu'elle n'ait lieu et enfin la composition de tous les automates précédents avec l'automate de G, à savoir un automate P modélisant le système supervisé attaqué. A partir de ce dernier automate, un automate d'attaque A peut être construit selon les états ciblés par l'attaquant et les propriétés choisies que doit respecter la composition de P et A. Les auteurs présentent ainsi les propriétés de sournoiserie, d'attaque non bloquante, d'attaque dégradante, d'attaque cupide ou non cupide et enfin d'attaques avec observabilité ou non des événements de commande. La propriété d'attaque non bloquante se réfère à la capacité de l'attaquant de maintenir le système physique hors d'états bloquants, interdits initialement par le superviseur, avant d'atteindre les états qu'il cible. Dans cette troisième référence, les attaques de blocage sont donc présentées comme un objectif que ne souhaite pas réaliser l'attaquant car celles-ci sont des vecteurs possibles de ralentissement et de non sournoiserie de l'attaque. L'inverse de cet objectif de non-blocage signifie que l'attaquant pourrait néanmoins être capable d'atteindre des états bloquants.

Deuxièmement, les attaques de blocages sont référencées dans la littérature de manière indirecte, à travers les approches de prévention construisant un superviseur non-bloquant [241], [352], [353].

Ainsi, [241] propose un algorithme itératif capable de construire un superviseur non-bloquant et le plus permissif possible dans un contexte d'attaques de modifications des événements capteurs, à savoir la suppression ou le remplacement des événements d'observation. Cette approche utilise le graphe des marquages accessibles du RdP modélisant le système physique pour le développement du superviseur en présence d'attaques. Ce dernier est plus restrictif que dans le cas sans attaque car il contraint certains événements dès qu'un doute est émis sur l'état réel du système physique en raison d'attaques ayant pu avoir lieu sur des événements observés. En effet, si dans un des états possibles du système physique, un événement conduit le système dans un état de blocage, alors cet événement est interdit depuis tous les états possibles du système physique. L'auteur précise par ailleurs que dans le cas où l'attaquant a la capacité

de modifier un trop grand nombre d'événements d'observation, alors le superviseur obtenu par leur algorithme peut ne pas pouvoir assurer la vivacité du système supervisé et peut même être "infaisable".

Dans leurs travaux, [353] proposent le principe de changement de superviseurs en ligne pour faire face aux attaques à la manière des méthodes de défense par cibles mouvantes héritées de l'IT. Ils présentent une condition suffisante et une méthode pour la synthèse à partir d'un système physique G et d'une spécification de non-blocage K, deux superviseurs distincts R_1, R_2 respectant la spécification, donc non-bloquants, et deux fonctions de mise à jour π_1, π_2 responsables de la mise à jour du système supervisé en cas de changement de superviseur et d'une fonction de changement Sw qui, pour un état x_g de G, décide si le système doit changer de superviseur ou non. Les attaques d'insertion et de suppression des événements capteurs sont intégrées à cette problématique de synthèse de superviseurs interchangeables. Néanmoins, l'attaquant ne connait pas le superviseur actif lors du lancement de son attaque et par conséquent, son attaque peut être prévenue si le superviseur actif y est robuste. Face à ces attaques, les auteurs définissent la problématique du contrôle par supervision pour la défense par cibles mouvantes et proposent une méthode inspirée de leurs précédents travaux pour la synthèse de superviseurs interchangeables robustes.

Les deux références [241], [353] présentées dans ce paragraphe ainsi que les travaux de [352] proposent la construction de superviseurs robustes face à différents types d'attaques contre les SEDs. L'une des propriétés que doivent respecter ces superviseurs est la vivacité du système supervisé. Cette propriété met en exergue l'existence des états de blocage et la possibilité qu'ils deviennent accessibles lorsqu'une attaque a lieu malgré la présence initiale - sans considération des attaques - d'un superviseur non-bloquant. Néanmoins, les états de blocages ne sont ici pas introduits comme l'objectif premier de l'attaquant, et ne sont pas étudiés dans le contexte des FMSs.

Cette première partie de l'état de l'art a fait le bilan des travaux de la littérature sur les attaques de blocage. Dans ceux-ci, les attaques de blocage sont étudiées en tant que conséquences secondaires d'autres catégories d'attaques ou sont évoquées indirectement à travers la construction de superviseurs robustes et non-bloquants en présence d'attaques. Ainsi, aucune référence ne considère les états de blocage comme l'objectif principal d'un attaquant. Par conséquent, aucune définition ni modélisation des attaques de blocage dans le contexte des FMSs n'ont encore été proposées à ce jour.

2.3.2 État de l'art des méthodes de détection d'attaques expertes dans le SEDs

Dans la section 2.2.3, nous avons démontré la non robustesse des méthodes de prévention et d'évitement face à un attaquant expert capable d'insérer ou de supprimer n'importe quel événement contrôlable ou observable du système. Ainsi, les méthodes de détection ont été préférées pour faire face aux attaques de blocage. Dans la littérature, différentes méthodes de détection de blocages d'une part et de détection d'attaques contre les SEDs d'autre part ont été proposées. Ces méthodes de détection s'apparentent à des méthodes de diagnostic [199] où pour toute attaque (faute), il existe une séquence finie d'événements λ dont la projection λ_o par $P_o : E^\star \to E_o^\star$ aux événements observables permet d'affirmer la présence de l'attaque et de la

détecter, i.e. $(f_a^2)^{-1}(P_o^{-1}(\lambda_o)) \in \mathcal{L}_a \setminus \mathcal{L}$ avec $\mathcal{L}_a$ le langage de l'attaquant contenant toutes les séquences attaquées d'événements qu'il peut exécuter et $\mathcal{L}$ le langage non attaqué du système. A l'inverse, une attaque non diagnosticable ou détectable est sournoise.

L'objectif de cette deuxième partie de l'état de l'art est d'étudier les capacités des méthodes de littérature à détecter des attaques expertes.

Les **méthodes de détection des états de blocage** se définissent par deux étapes : la caractérisation des états de blocage et leur détection. Il a été montré dans une précédente partie (2.2.3), que les méthodes centralisées de détection d'états de blocage [110]-[115] ne permettent pas l'identification des attaques de blocage expertes. Par conséquent, ces dernières ne répondent pas à notre objectif de détection des attaques de blocage expertes.

Les **méthodes de détection des attaques contre les SEDs** sont à l'instar des approches décrites dans la sous-partie précédente (2.3.1) définies selon les types d'attaques auxquelles elles font face, entre suppression et insertion d'événements d'observation capteurs et de commande actionneurs. Dans cet état de l'art, toutes les méthodes de détection des attaques contre les SEDs de la littérature sont introduites.

Précurseurs de la détection d'attaques dans les SEDs, [349] se focalisent sur les attaques d'insertion de commandes et leur détection. Les auteurs proposent une condition de désarmement permettant au système supervisé $S \setminus G$ de détecter et de bloquer une attaque avant qu'elle n'atteigne un état indésirable hors des spécifications du superviseur. Si cette condition ne peut être respectée, une méthode d'évaluation des dommages pouvant être causés par un attaquant est utilisée afin de construire un superviseur minimisant de manière optimale ces dommages et maximisant l'accessibilité d'états dits bénéfiques. L'indicateur de dommages est ici calculé à partir du nombre et du coût des séquences d'événements non désarmables au sein du langage attaqué et de la pondération de la vulnérabilité de chaque événement de commande attaquable.

La notion de désarmabilité est réinterprétée dans [199] où les auteurs étudient de manière indépendante la détection des attaques d'activation des actionneurs (insertions d'événements de commande) et la détection des attaques de suppression et d'insertion d'événements capteurs. Pour chaque type d'attaque, un automate du système physique attaqué G_a et un automate du superviseur attaqué H_a sont construits et composés afin d'obtenir l'automate $G_M = G_a \parallel H_a$ représentatif de la boucle de contrôle attaquée. Dans G_M, un état de l'automate est représenté sous la forme (x, y) avec y l'état réel de G_a et x l'état supervisé par H_a. Les auteurs utilisent successivement un diagnostiqueur et un vérificateur afin de détecter l'attaque et de tester la contrôlabilité A-sûre de G_M, à savoir la capacité de G_M en présence de ces outils à détecter une attaque avant qu'elle n'ait lieu et à pouvoir empêcher par un événement contrôlable non-vulnérable cette dernière d'atteindre un ensemble d'états non-sûrs $\mathcal{F}_X$. Cette méthode est extensible à une combinaison des différents types d'attaques. Cependant, les attaques de suppression de commandes actionneurs ne sont pas considérées et seuls certains événements de G sont vulnérables aux attaques. La contrôlabilité A-sûre d'un système supervisé attaqué est également reprise dans [354] pour la détection des attaques d'insertion d'événements de commande dans un système physique modélisé par un RdP. Cependant, tous les événements de commande ne sont pas considérés simultanément vulnérables à l'insertion, car le cas échéant le système supervisé ne pourrait jamais être contrôlable A-sûre. Similairement aux travaux

présentés dans ce paragraphe, [350] étudie la contrôlabilité du système physique lorsque des attaques d'activation des événements actionneurs et de modification d'événements capteurs sont détectées. Néanmoins, tous les événements observables et contrôlables ne sont par hypothèse pas vulnérables simultanément.

En 2019, [236] proposent une approche de détection où l'attaquant peut utiliser différents dictionnaires d'attaque de manipulation des événements d'observation. Un dictionnaire est une fonction définissant les capacités de l'attaquant : les événements d'observation vulnérables et les types d'attaque auxquels ils sont vulnérables. Notons que cette fonction ne permet pas de construire un dictionnaire d'attaquant expert. Parmi les dictionnaires, deux types d'attaques sont prises en compte ; les attaques constantes, ne pouvant appartenir qu'à un unique dictionnaire et les attaques interchangeables pouvant appartenir à plusieurs dictionnaires dont le dictionnaire du cas sans-attaque. La détection de ces attaques et de leurs dictionnaires est finalement développée à l'aide de la théorie des observateurs et des diagnostiqueurs des SEDs.

A l'instar de [354], les RdPs ont également été choisis par [208], [300], [355] pour la détection d'attaques. Un RDP à signaux interprété modélise le fonctionnement d'un API et tous les événements d'entrée et de sortie de l'API (événements capteurs et actionneurs) sont vulnérables. Dans leur premier travail [208], [300], les attaques de répétition des événements capteurs et les attaques secrètes - soient la manipulation d'événements actionneurs et parallèlement d'événements capteurs pour rester sournois - sont considérées. Ces attaques ne sont pas initialement détectables par l'API car elle insèrent des séquences d'événements d'entrée admissibles pour ce dernier. Néanmoins, elles s'appuient sur une connaissance a priori des entrées et sorties de l'API. Puis, une fois lancées, les séquences d'événements attaqués ne peuvent plus être modifiées. Ainsi, pour permettre la détection de telles attaques, les auteurs proposent la permutation en ligne des entrées et sorties afin de rendre caduques les modèles du système physique, du contrôleur et des signaux d'entrée/sortie déployés par l'attaquant. Lorsque la permutation a lieu, les modèles caduques utilisés par l'attaquant deviennent détectables par la méthode.

Dans leur seconde contribution, [355] conservent les RdPs à signaux interprétés et s'intéressent à la détection temporelle des attaques d'insertions d'événements capteurs et des attaques secrètes. Tous les événements sont vulnérables. La méthode s'appuie sur les RdPs temporisés dont les transitions sont labellisées par trois temps, un temps minimum τ_{min}, un temps maximum τ_{max} et un temps d'expiration $\tau_{timeout}$. Si une transition est franchie hors de l'intervalle $[\tau_{min}, \tau_{max}]$ un nombre de fois supérieur à un seuil donné, une détection a lieu. Pour sa part, le dépassement de $\tau_{timeout}$ est détecté comme une attaque car il signifie qu'un événement est très anormalement en retard par rapport aux spécifications du système supervisé. Notons que la capacité de détection temporelle d'une attaque est rendue possible par les deux hypothèses suivantes : dans le cas des injections d'observations, l'attaquant n'a pas une maitrise totale de la temporalité de la boucle de contrôle entre G et S, tandis que dans le cas des attaques secrètes, le cycle de l'API est mis en pause du début à la fin de l'attaque conduisant à des dérives temporelles de ce dernier.

Finalement, [356] s'intéressent à la détection d'attaque de modification des événements capteurs au sein d'une boucle de contrôle où le système physique possède un comportement stochastique. Les auteurs soulèvent la limite des travaux de [199], [350] sur la non-détectabilité des attaques respectant le langage du système supervisé projeté sur E_o et proposent d'enrichir

la détection par l'analyse des informations probabilistes inhérentes aux traces d'attaques. Soit une séquence d'événements observables pouvant conduire en cas d'attaque à un état suspect où une détection devrait avoir lieu avant d'atteindre un état critique. La probabilité d'observer cette séquence dans le cas sans attaque est comparée à la probabilité que cette séquence résulte d'une attaque. Si la première probabilité est largement inférieure à la seconde, une suspicion peut-être émise sur l'origine malveillante de la séquence. Un système supervisé est alors dit ϵ-sûr, $\epsilon \in (0.5, 1]$, si toutes les traces conduisant à un état suspect sont globalement plus attribuables probabilistiquement à un attaquant qu'à un comportement normal du système physique stochastique. Un vérificateur est utilisé pour valider la propriété ϵ-sûr d'un système. Contrairement aux autres méthodes, cette dernière ne permet pas de détecter de manière déterministe une attaque mais offre une alternative pour la détection des attaques sournoises.

A notre connaissance, toutes les méthodes SED de détection d'attaques ont été présentées dans cette sous-partie. Les travaux traitant des attaques sournoises cités dans la précédente sous-partie (2.3.1) [212], [223], [237], [347] peuvent être également pris en compte dans notre état de l'art car la présence d'une attaque sournoise implique la non-détectabilité de celle-ci par n'importe quel diagnostiqueur, vérifieur ou observateur.

Ainsi, nous observons que, à l'exception de [208], [300], [355], aucune méthode de détection ne considère la détection d'attaques conçues par un attaquant expert. En effet, dans le cas où la méthode de détection est positionnée au niveau du superviseur et a la même observabilité que ce dernier, une attaque sournoise dont la projection sur E_o appartient au langage $\mathcal{L}_o$ du système supervisé ne pourra jamais être détectée. Un attaquant expert capable de manipuler tous les événements d'observations et connaissant les séquences d'observations admissibles par le langage du superviseur est donc capable de créer une attaque sournoise. Dans [208], [355], la détection est rendue possible en modifiant [208] ou en prenant certaines hypothèses [355] sur les connaissances du langage observable détenues par l'attaquant expert. Cependant, les limites suivantes de ces méthodes ne permettent pas de les appliquer à nos travaux.

Premièrement, [208] nécessite un composant actif capable d'inter-changer les canaux de communication entrées/sorties du superviseur et de rendre obsolètes et erronées les connaissances de l'attaquant. Outre un coût de déploiement élevé d'une telle approche, le caractère actif de la méthode n'est pas conservé dans nos travaux pour les raisons évoquées en conclusion de 2.2.3, à savoir la capacité de manipulation des méthodes actives par un attaquant maitrisant leurs fonctionnements.

Deuxièmement, [355] ne requiert pour sa part qu'un composant passif d'observation des événements d'entrée/sortie du superviseur. La détection est rendue possible grâce à un enrichissement des modèles d'observation avec les comportements temporels normaux de la boucle de contrôle supposés non connus par l'attaquant. Cependant ces comportements temporels enrichissant le modèle de détection sont inadaptés au contexte des FMSs. En effet, l'environnement incertain des FMSs dans lequel les ressources peuvent devenir indisponibles ne permet pas l'estimation précise de seuils temporels robustes aux faux-positifs. Dans la prochaine sous-partie, cet environnement incertain est étudié au regard de l'état de l'art des méthodes de diagnostic d'attaques et de fautes.

2.3.3 État de l'art des méthodes de diagnostic de fautes et d'attaques

Le diagnostic de fautes est utilisé dans les SEDs pour la détection de défaillances capteurs, actionneurs et des erreurs de fonctionnement du système physique (e.g. débordement d'un réservoir) ou du contrôleur [10], [235], [357], [358]. Par définition, une faute est assignée à un événement non observable $e_f \in E_o$ et est diagnostiquée lorsque $\exists \lambda_o \in \mathcal{L}_o(N, M_o) | \forall \lambda \in P_o^{-1}(\lambda_o), e_f \in \lambda$. La faute e_f est alors dite diagnosticable si $\exists n_0 \in \mathbb{N}, \lambda_1 \in (E \setminus e_f)^\star, \lambda_2 \in E^\star$ tel que pour $\lambda = \lambda_1 e_f \lambda_2 \in \mathcal{L}(N, M_0)$ on a $|\lambda_2| < n$ et $e_f \in P_o^{-1}(P_o(\lambda))$. En d'autres termes, la faute peut être diagnostiquée systématiquement après son occurrence grâce à une séquence finie d'observations.

Dans le cas des FMSs, le diagnostic de faute peut-être nécessaire si les événements symbolisant l'indisponibilité d'une ressource, l'ordre de maintenance préventive et le compte-rendu de défaillance échangés entre le superviseur et les contrôleurs locaux ne sont pas observables. Nous supposons dans nos travaux que les événements de défaillance sont observables et appartiennent à E_o. Les événements de maintenance sont eux contrôlables et appartiennent à E_c. Le superviseur considère que les événements de maintenance sont toujours correctement exécutés par les contrôleurs locaux des ressources ciblées. Les événements relatifs à l'indisponibilité d'une ressource sont donc diagnosticables par le superviseur puisque ce dernier connait systématiquement leur occurrence et est capable de les prendre en compte dans son pilotage du FMS. Cependant, leur appartenance à E_o et E_c les rend vulnérables aux attaques. Par conséquent, ils peuvent être manipulés par un attaquant expert pour atteindre un état de blocage du FMS.

Dans la précédente partie (2.3.2), nous avons montré que la détection d'attaque, solution de cybersécurité sélectionnée pour lutter contre les attaques de blocage, repose sur le diagnostic des séquences propres aux attaques. Face à l'observation de l'indisponibilité d'une ressource, la problématique du diagnostic entre réelle indisponibilité et fausse indisponibilité manipulée par l'attaquant se pose alors. A notre connaissance, aucune référence de la littérature ne s'intéresse à la problématique du diagnostic commun des attaques et des fautes à partir d'une séquence anormale d'observations. Néanmoins, un ensemble de travaux permettent d'introduire cette problématique selon deux axes majeurs : **les approches de diagnostic de fautes en présence d'attaques et les approches de détection dont les modèles enrichis permettent de caractériser les attaques par rapport aux comportements normaux et défaillants des SEDs. Ces deux axes et leur littérature associée sont présentés dans cette sous-partie**.

> L'objectif de cette dernière partie de l'état de l'art est d'étudier les travaux introduisant la problématique de diagnostic commun des attaques et des fautes.

Le diagnostic de faute en présence d'attaque se définit comme la capacité à diagnostiquer une faute malgré la présence d'un attaquant capable de modifier des événements influençant ou nécessaires au diagnostic. En 2022, [348] proposent une méthode de diagnostic robuste aux attaques pondérées par un coût. Les auteurs considèrent les attaques d'insertion, de suppression et de remplacement d'événements observables dont un ensemble restreint est vulnérable. Ils définissent la notion de diagnosabilité d'un système supervisé en présence d'attaques à coût illimité et limité. Chaque attaque sur un événement possède un coût unitaire exploité ensuite pour le calcul du coût de n'importe quelle séquence. Un automate non déterministe dont les

états sont des pairs (x, c) avec x l'état du système physique attaqué et c le coût d'attaque de la séquence d'événements menant à x depuis l'état initial x_0 est utilisé pour répondre à la problématique de diagnosabilité robuste. Notons que $c = 0$ signifie que la séquence d'événements ne contient aucune attaque. A partir de ces travaux liminaires, [346] présentent une méthode de construction d'un diagnostiqueur dans le cas où les attaques ont un coût illimité. A partir de ce diagnostiqueur, les auteurs montrent qu'une faute initialement diagnosticable peut ne plus l'être en présence d'attaques.

En 2021, [359] s'intéressent à la capacité d'observabilité des états critiques d'un système discret en l'absence ou en présence d'attaque. Différemment des travaux de [346], la faute n'est pas définie par un événement e_f mais par un ensemble d'états critiques du système physique que l'on souhaite pouvoir diagnostiquer de la manière suivante ; soit Ω l'ensemble des état critiques, et une paire $(\lambda_1, \lambda_2) \in E^\star \times E^\star$ tels que $P_o(\lambda_1) = P_o(\lambda_2) = \lambda$ et menant aux états discrets (x_1, x_2). Le système est dit Ω-observable si $(x_1, x_2) \in (\Omega \times \Omega) \cup (\overline{\Omega} \times \overline{\Omega})$, à savoir si (x_1, x_2) sont deux états critiques ou deux états non critiques. Cette définition est étendue aux systèmes discrets en présence d'attaque dans le but de détecter les attaques d'une part, en supposant $\Omega = X_a$ l'ensemble des états résultant d'une attaque, et d'autre part au diagnostique des états critiques de Ω en présence d'attaques manipulant les séquences λ_1 et λ_2.

Cependant, notre problématique est différente de celle proposée par ces travaux car nous supposons que le système n'est pas diagnosticable en présence d'attaques. En effet, une séquence d'observation associée à une faute peut aussi correspondre à une attaque dans le cas des événements d'indisponibilité d'une ressource du FMS pour deux raisons. **D'une part, les événements de défaillance et de maintenance sont vulnérables en appartenant à E_o et à E_c et il n'est par conséquent pas possible de diagnostiquer si leur origine est naturelle ou malveillante**. D'autre part, en ayant accès à une observabilité fiable (section 2.2.1) de l'état des ressources et de l'allocation de celles-ci à partir de données extraites entre les contrôleurs et les ressources, un événement e_f symbolisant l'indisponibilité de R_k n'est pas observable car échangé entre $CTRL_k$ et S. Cet événement est donc considéré non diagnosticable en présence d'un attaquant. En d'autres termes, pour $e_f \in E_{uo}$, il n'existe pas de séquence d'observation λ_o telle que $\forall w \in P_0^{-1}(\lambda_o), \lambda_o \in w$ et $(f_a^2)^{-1}(\lambda_0) = \emptyset$ bien que λ_o soit reconnue comme anormale par une observation fiable, à travers par exemple l'absence d'événements communiqués entre $CTRL_k$ et R_k alors que l'allocation de R_k à un produit en cours est attendue.

Néanmoins, les capacités de différenciation entre attaque et faute peuvent être améliorées en enrichissant les modèles discrets de diagnostic du système supervisé et de l'attaque par de nouvelles connaissances. Par exemple, le principe de coût d'une attaque présenté par [348] pourrait permettre de distinguer une faute d'une attaque lorsque le coût d'une séquence d'observations anormale est supérieur à un seuil défini pour le cas d'une attaque. L'état de l'art des méthodes utilisant des modèles enrichis est introduit dans le prochain paragraphe.

Un modèle discret enrichi est défini dans ce manuscrit comme un modèle où des informations supplémentaires sont intégrées à ce dernier à travers l'ajout de nouveaux états et événements ou par l'addition d'informations (e.g. temps, probabilités, coût) à des états et événements déjà existants. Dans le domaine de la cyber-sécurité des SEDs, les modèles discrets du système physique, du superviseur et de l'attaque ont pu être enrichis dans le but de proposer des

modèles d'attaque réalistes, des superviseurs robustes et des approches de détection assurant la contrôlabilité du système hors de ses états critiques. L'état de l'art des références proposant des modèles discrets enrichis est présenté dans ce paragraphe au regard de l'entité considérée, entre système physique, superviseur et attaque. Pour chaque référence, une analyse de son intérêt pour le diagnostic conjoint d'attaques et de fautes est développée.

Premièrement, le modèle du système physique est enrichi dans la littérature par des connaissances temporelles et stochastiques. Introduits dans la section précédente (2.3.2), les travaux de [355] définissent pour chaque transition d'un RdP un intervalle de temps $[\tau_{min}, \tau_{max}]$ et un temps d'expiration $\tau_{timeout}$. Ces variables temporelles représentent et bornent le temps qui doit normalement s'écouler entre deux événements successifs reçus et/ou envoyés par la loi de commande du superviseur. Notons que ce temps est relatif au comportement du système physique car il représente le temps nécessaire à ce dernier pour transiter entre deux de ses états discrets. Si l'intervalle de temps ou le seuil d'expiration ne sont pas respectés, une détection a lieu. Cependant, aucune distinction n'est faite entre attaque et faute et le non respect de ces temps de référence pourrait correspondre à ces deux anomalies. Hors du domaine de la cyber-sécurité, [360] propose un vérificateur pour la détection de traces temporelles aux seins des SEDs temporisés modélisés par des automates. A partir d'un vérificateur temporel où les transitions sont labellisées par des événements et franchissables uniquement sur des intervalles de temps donnés, une méthode de construction d'un vérificateur logique, où l'incrémentation du temps est modélisé par les transitions, est exposée. L'auteur suggère l'utilisation de ce vérificateur pour l'étude de l'opacité temporelle d'un SED face aux attaques. Dans le cadre de la détection, un tel vérificateur pourrait être utilisé pour identifier et comparer les trace temporelles relatives aux fautes et celle relatives aux attaques.

L'enrichissement stochastique du modèle du système physique est introduit dans les travaux de [242], [356] présentés précédemment (2.3.2). L'utilisation de transitions stochastiques pour modéliser le comportement du système physique permet de comparer les probabilités qu'une séquence suspicieuse d'observations soit le résultat d'une attaque ou d'un comportement normal. Dans le cas du diagnostic entre fautes et attaques, ce modèle pourrait être enrichi par des probabilités relatives aux fautes afin d'inférer stochastiquement l'origine d'un comportement anormal observé. En 2006, [349] exploite aussi un modèle probabiliste, cette fois-ci pour évaluer les conséquences d'une attaque sur le système physique. Un coût aléatoire compris entre $[-1, 0)$ est alloué aux états non désirables, un coût entre $(0, 1]$ pour les états désirables et un cout nul pour les autres états. A partir d'un état x, toutes les transitions de sortie $t \in x^\bullet$ sont associées à une probabilité et la somme de toutes ces probabilités est égale à 1. Enfin, chaque événement contrôlable est affilié à une probabilité comprise entre $[0, 1]$ et représentative de la vulnérabilité de l'événement où une probabilité de 1 modélise une haute vulnérabilité. A partir de toutes ces informations stochastiques intégrées au modèle du système physique, un coût représentatif des conséquences d'une attaque est calculé pour chaque séquence du langage, un coût proche de -1 signifiant de grandes conséquences, un coût proche de 0 aucune conséquence, et un coût proche de 1 des conséquences favorables. Ce coût propre à chaque séquence pourrait être utilisé pour représenter un attaquant souhaitant optimiser les dégâts qu'il inflige au système physique en exploitant les composants les plus vulnérables.

Deuxièmement, le modèle du superviseur peut être enrichi par la définition au sein de l'ensemble des états du système physique de nouvelles catégories d'états élargies au delà des états autorisés et interdits par le superviseur. Au sein des méthodes de détection cherchant à assurer

la contrôlabilité du système physique entre l'état de la détection et les états interdits [199], [349], [350], [354], l'identification d'états de détection robustes permettant cette contrôlabilité est comparable à l'enrichissement du modèle du superviseur par une nouvelle catégorie d'état. Cependant, il a été montré [356] qu'il n'existe pas un état de détection robuste pour chaque séquence d'attaque possible. Similairement, [224] définissent des états dit détecteurs qu'ils intègrent au superviseur via la construction d'un observateur. Si l'accès à un état détecteur est observé, une attaque a forcément lieu. Cependant, ces états détecteurs ne sont pas définis pour chaque type d'attaque et ne prennent pas en compte les défaillances pouvant elles aussi conduire à ces états détecteurs.

Un modèle de supervision peut aussi être affiné par la définition au sein des états autorisés d'états favorables et d'états non favorable mais sûrs. Dans leurs travaux, [349] définissent des états favorables auxquels ils assignent un coût positif représentant les bénéfices obtenus si ces états sont atteints ou traversés par le système. Bien que ces états favorables ne soient pas totalement exploités par les auteurs, ils pourraient permettre une suspicion d'attaque lorsqu'une déviation hors des trajectoires favorables est observée. A des fins de construction d'un superviseur robuste, [361] définissent trois catégories d'états accessibles par le système physique : les états désirés, les états tolérables et les états non-sûrs. L'objectif de leur superviseur est de maximiser le langage au sein des états désirés, interdire les séquences menant aux états non-sûrs, et d'autoriser tout en minimisant les séquences traversant les états tolérables non désirés. L'originalité de cette méthode est d'utiliser les états tolérables non-désirés comme états pouvant être soit interdits, soit autorisés sans contraintes afin d'assurer la robustesse du système supervisé face aux attaques d'insertion de commandes. Dans le cas de la détection, ces états tolérables non-désirés peuvent être révélateurs d'une anomalie non critique puisqu'il ne respectent pas la politique de contrôle du superviseur autorisant uniquement les états désirés.

Au reste, le modèle du superviseur peut être enrichi par le modification en ligne des statuts de supervision des états et séquences d'événements du système physique. En effet, dans leurs travaux introduits précédemment (2.3.1), [353] inter-changent deux superviseurs distincts lors du fonctionnement en ligne du système supervisé. Par conséquent, les états ou séquences interdits peuvent devenir autorisés et inversement. Un attaquant n'étant pas capable de prendre en compte ces changements sera détecté lorsque, par exemple, son attaque traverse un état devenu interdit.

Troisièmement, le modèle de l'attaque peut aussi être amélioré à des fins de diagnostic par l'ajout d'informations sur le profil d'attaquant, le coût de l'attaque, sa rapidité, les dommages qu'elle occasionne et sa sournoiserie. La notion de profil d'attaquant n'est pas explicitement présente dans la littérature mais peut être associée aux dictionnaires d'attaques introduits dans [236] (voir section 2.3.2) ou aux fonctions d'attaque définies dans [242] par les capacités d'insertion et de suppression d'événements observables par un profil d'attaquant. En effet, différents dictionnaires (ou fonctions) d'attaques peuvent modéliser des attaquants n'ayant pas le même objectif ou les mêmes capacités. Le diagnostic en ligne des dictionnaires d'attaques représentant toutes les capacités d'un attaquant pourrait permettre d'éliminer l'origine malveillante d'une détection d'anomalie lorsqu'aucun dictionnaire ne correspond aux séquences anormales observées même si ces dernières appartiennent au langage global de l'attaquant. En d'autres termes, l'attaque n'est pas retenue comme origine de l'anomalie car peu réaliste par rapport aux objectifs attendus d'un attaquant.

Ces profils d'attaquants peuvent être construits à partir des différentes caractéristiques d'un attaquant pouvant être intégrées à son modèle d'attaque. La première caractéristique d'un attaquant étudiée dans la littérature est le coût d'une attaque pour un attaquant [223], [346]-[349]. Dans [223], [346]-[348], un coût est associé à chaque séquence attaquée. Plus un coût est élevé, plus l'attaque est complexe à réussir et requiert des moyens importants pour l'attaquant. Dans [349], le coût est associé aux événements contrôlables et représente la vulnérabilité de ces événements. Plus le coût est élevé, plus l'événement est vulnérable et facile à exploiter par un attaquant. Par ailleurs, l'idée de coût d'une attaque est aussi développée dans [345] à travers une fonction coût que cherche à minimiser un attaquant. Dans [345], il s'agit de l'énergie consommée par un attaquant lorsqu'il réalise une attaque d'insertion de commande (activation d'un actionneur). Les auteurs démontrent comment créer une attaque secrète optimale capable d'atteindre son état objectif tout en minimisant sa consommation d'énergie. Dans [242], le coût d'une attaque est pondéré à la probabilité qu'elle puisse avoir lieu au sein d'un SED stochastique. Ainsi, parmi les fonctions d'attaques atteignant les états non sûrs du système, la fonction avec le plus petit coût est choisie. La résolution de ces objectifs d'accessibilité d'états non sûrs et de minimisation de coût par un attaquant est rendue possible par un processus de décision Markovien dont les états modélisent conjointement l'état du système physique, l'état du superviseur et le dernier événement exécuté par le système physique tandis que les actions du processus représentent les actions de l'attaquant.

La deuxième caractéristique, la rapidité d'une attaque contre un SED est introduite par [345]. Ce sous-objectif d'une attaque (sous-partie 2.2.2) est représentatif de la volonté d'un attaquant de causer des dommages le plus rapidement possible au système attaqué. En outre, les auteurs considèrent peu réaliste un attaquant prenant un temps très élevé voire sans limite pour mener à bien son attaque. Ainsi, à l'instar de la minimisation de l'énergie, [345] proposent une méthode de construction d'une attaque secrète optimale minimisant une fonction temps allouant à chaque événement, attaqué ou non, un temps d'exécution.

Enfin, le modèle d'attaque contre les SEDs est restreint à de nombreuses reprises dans la littérature aux attaques sournoises [199], [212], [223], [237], [242], [345], [347] et infligeant des dommages au système physique [199], [212], [242], [345], [352]. Ces deux caractéristiques d'une attaque sont nécessaires à la létalité de celle-ci. En effet, la sournoiserie d'une attaque empêche sa détection et par conséquent, la limitation de ses dommages par une méthode de cyber-sécurité ou un superviseur. Pour sa part, une attaque n'infligeant pas de dommages a peu d'intérêt pour un attaquant car l'objectif qu'il cible n'est pas atteint. Dans les SEDs déterministes, [199], [212], [345], [352] considèrent qu'une attaque inflige des dommages si elle atteint un état critique du système. Dans les travaux de [242] considérant les SEDs stochastiques, cette caractéristique d'affliction de dommage est quantifiée plus finement par une fonction win_A associant à une profil d'attaquant A une probabilité qu'il occasionne l'occurrence d'une séquence d'événements non-sûre pour le système physique. Ainsi, plus la probabilité est élevée, plus il y a de chance que l'attaque inflige des dommages, l'attaquant cherchant consciemment à maximiser cette probabilité win_A.

Toutes les caractéristiques présentées dans les paragraphes précédents et restreignant la taille du modèle de l'attaque ont pour conséquence la réduction de la taille du langage de l'attaquant $\mathcal{L}_a$ et de la projection de ce langage sur l'ensemble des événements observables $P_o(\mathcal{L}_a)$. Ainsi, l'intersection entre $P_o(\mathcal{L}_a)$ avec le langage des séquences d'observations pour le diagnostic de fautes pourra être réduite, conduisant à une meilleure distinction entre attaques

et fautes.

Dans ce dernier état de l'art, le diagnostic conjoint des attaques et des fautes a été étudié. Les méthode de diagnostic de fautes en présence d'attaques ont été présentées et ont permis de mettre en lumière l'absence de méthode capables de différencier systématiquement attaques et fautes à partir d'une séquence d'observation anormale. Face à cette problématique, à notre connaissance non résolue, différentes approches fondées sur l'enrichissement des modèles du système physique, du superviseur et de l'attaquant ont été exposées. Pour chacune de ces approches, une discussion sur l'intérêt de leur modèle enrichi pour la caractérisation des attaques a été menée. La combinaison de connaissances et d'informations apportées aux différents modèles pourrait à terme permettre la différenciation complète entre fautes et anomalies. En conclusion de ce chapitre, les trois états de l'art exposés dans cette partie sont examinés à l'aune du contexte des FMSs et nous permettent d'introduire une problématique de recherche originale.

2.3.4 Problématique et verrous scientifiques

La détection des attaques de blocage expertes ciblant les FMSs dans un contexte incertain a été définie comme l'objectif principal de ces travaux. Dans cette partie 2.3, trois états de l'art de la littérature récente sur la cyber-sécurité des SEDs ont été menés au regard de cet objectif. A partir de ces états de l'art, de notre positionnement, des hypothèses de travail que nous avons choisies et de cet objectif, les verrous scientifique et les limites suivantes de la littérature peuvent être identifiées :

1. L'objectif de blocage d'un SED n'a pas été analysé en tant qu'objectif principal d'un attaquant. En particulier, aucun travail ne s'intéresse aux attaques de blocage ciblant les FMSs (sous-partie 2.3.1).
2. Parmi les méthodes de détection d'attaques dédiées aux SEDs, seulement deux méthodes définissent un attaquant expert(def. 2.2.9), capable d'insérer et de supprimer tous les événements du système (sous-partie 2.3.2). Une troisième méthode, tentant d'élargir le spectre des attaques détectables considérées dans la littérature, pourrait également être pertinente pour traiter les attaquants experts. Ces trois méthodes de détection ne sont cependant pas, en l'état, adaptées aux FMSs.
   (a) La première [208], [300] s'appuie sur une méthode active capable de modifier aléatoirement le comportement des canaux de communication des entrées/sorties d'un contrôleur afin de tromper un attaquant et le détecter. Cette méthode requiert plusieurs composants numériques responsables de ces modifications sur les communications. Cette méthode est vulnérable aux attaques en raison du caractère numérique de ces composants car celle-ci peut être manipulée par un attaquant capable de comprendre son fonctionnement.
   (b) La deuxième méthode de détection [355] s'appuie sur le comportement temporel du système physique. Néanmoins, dans le cas des FMSs, les seuils temporels définis pour chaque transition du RdP ne sont pas applicables car ces seuils ne prennent pas en compte les indisponibilités des ressources et les retards qu'elles occasionnent.
   (c) La troisième méthode de détection [356] est fondée sur l'analyse du comportement stochastique du système physique, où une séquence d'événements ayant une faible

probabilité d'être exécutée lors du fonctionnement normal du système devient suspicieuse lorsqu'elle est observée. Cependant, cette méthode n'est pas adaptée aux FMSs car le comportement pour l'allocation des ressources est un comportement déterministe puisque celui suit un ordonnancement fixe.

3. Au sein de l'environnement incertain des FMSs, une méthode de diagnostic capable de détecter et de différencier les attaques et les fautes (indisponibilités des ressources) à partir de l'observation d'une séquence anormale est nécessaire. Aucune référence de la littérature récente ne propose une telle méthode (sous-partie 2.3.3).
4. Un tel diagnostic conjoint entre fautes et attaques requiert des connaissances supplémentaires caractérisant les attaques, les indisponibilités et les comportements du FMS afin de pouvoir les distinguer. Différents travaux proposent des modèles du système physique, du superviseur et des attaques enrichis par ces connaissances pour des applications de cyber-sécurité (sous-partie 2.3.3). Néanmoins, aucun n'a étudié les connaissances pouvant être intégrées aux modèles propres aux FMSs pour des applications de cyber-sécurité et en particulier de diagnostic commun de fautes et d'attaques.

Face à ces différentes limites et verrous scientifiques, nous souhaitons répondre dans nos travaux à la problématique de recherche suivante :

Comment détecter, caractériser et diagnostiquer une attaque experte de blocage d'un FMS dans un contexte incertain où attaques et indisponibilités des ressources co-existent ?

Dans les prochains chapitres, ce manuscrit répond à cette problématique par les contributions suivantes :

1. Chapitre 3 : diagnostic des attaques dans un contexte certain ;
   (a) Proposition d'une définition et d'une modélisation des attaques de blocage contre les FMSs (partie 3.1). Introduction de différents profils d'attaquants fondés sur les notions de coût et de rapidité.
   (b) Enrichissement des modèles de G et de S par la définition d'un ordonnanceur, imposant un comportement optimal d'allocation des ressources au sein du FMS, et d'une méthode de supervision, interdisant les états critiques de la DZ (partie 3.2).
   (c) Construction d'un module de diagnostic des attaques de blocage exploitant les données fiables du FMS (partie 3.3). Diagnostic des attaques de blocage à partir des différents profils d'attaquants et des comportements optimaux et critiques du FMS.
2. Chapitre 4 : diagnostic conjoint des attaques et des indisponibilités de ressources ;
   (a) Extension des méthodes de supervision et d'ordonnancement au contexte incertain.
   (b) Extension du modèle d'attaques et des profils d'attaquant au contexte incertain et aux attaques contre les événements symbolisant l'indisponibilité des ressources (partie 4.1).
   (c) Extension du module de diagnostic à ces nouvelles attaques et au changement de modes. Diagnostic de l'origine d'un changement de mode (partie 4.3) entre indisponibilité réelle ou attaque.

3. Chapitre 5 : Application expérimentale de ces méthodes sur une plateforme manufacturière (chapitre 5).

L'ensemble des contributions de diagnostic de ce manuscrit sont illustrées dans la figure 2.5 ci-contre. Ainsi, trois diagnostics peuvent être distingués, celui du profil d'attaquant, celui des comportements du FMS, et celui du changement de mode. Ces différents diagnostics seront construits séparément puis fusionnés afin d'obtenir une solution de détection et de diagnostic des attaques de blocage contre les FMSs dans des contextes certain et incertain.

Dans toutes nos contributions, les modules et modèles utilisés pour la détection et le diagnostic des attaques de blocage sont construits à partir des connaissances sur le fonctionnement, l'architecture, l'environnement, la boucle de contrôle commande et les vulnérabilités face aux attaques des FMSs exposés dans les deux premiers chapitres. Ainsi, contrairement à la littérature des SEDs s'efforçant de résoudre des problématiques globales de cybersécurité des SEDs, nos travaux s'intéressent à un type particulier de systèmes discrets, les FMSs, et tentent d'exploiter les caractéristiques propres de ces systèmes dans l'objectif de proposer une méthode de cyber-sécurité adaptée aux FMSs et à leurs enjeux.

Conclusion

Dans le chapitre 2, le contexte de la cyber-malveillance à l'encontre des FMSs a été exposé. Premièrement, une introduction sur les attaques réelles et théoriques, les vulnérabilités et les approches de cyber sécurité dédiées aux systèmes de contrôle commande industriels a permis de mettre en exergue la problématique de cyber-malveillance à laquelle font face ces systèmes. Dans un deuxième temps, cette introduction, associée à la présentation des FMSs du chapitre 1, a été le support d'une analyse des vulnérabilités et attaques pouvant cibler les FMSs. Ainsi, parmi les attaques identifiées, l'objectif malveillant de blocage du FMS par un attaquant expert a été retenu pour nos travaux de recherche. Face à ces attaques dites expertes, nous avons prouvé que les méthodes de détection étaient les seules solutions applicables aux dépens des méthodes de prévention et d'évitement des états de blocage. En conclusion de ce chapitre, un état de l'art a été mené sur les trois sujets suivants : les attaques de blocage, la détection d'attaques expertes et le diagnostic commun des attaques et des fautes au sein de la littérature des systèmes à événements discrets. L'état de l'art a permis d'établir la problématique de recherche suivante : "Comment détecter, caractériser et diagnostiquer une attaque experte de blocage d'un FMS dans un contexte incertain où attaques et indisponibilités des ressources co-existent ?". Dans le prochain chapitre (3), une première réponse est apportée à cette problématique à travers la définition des attaques de blocages, leur modélisation et la construction d'un module de diagnostic de ces dernières dans un contexte certain, à savoir sans considération des indisponibilités des ressources. Le contexte incertain sera développé dans le chapitre 4 de ce manuscrit.

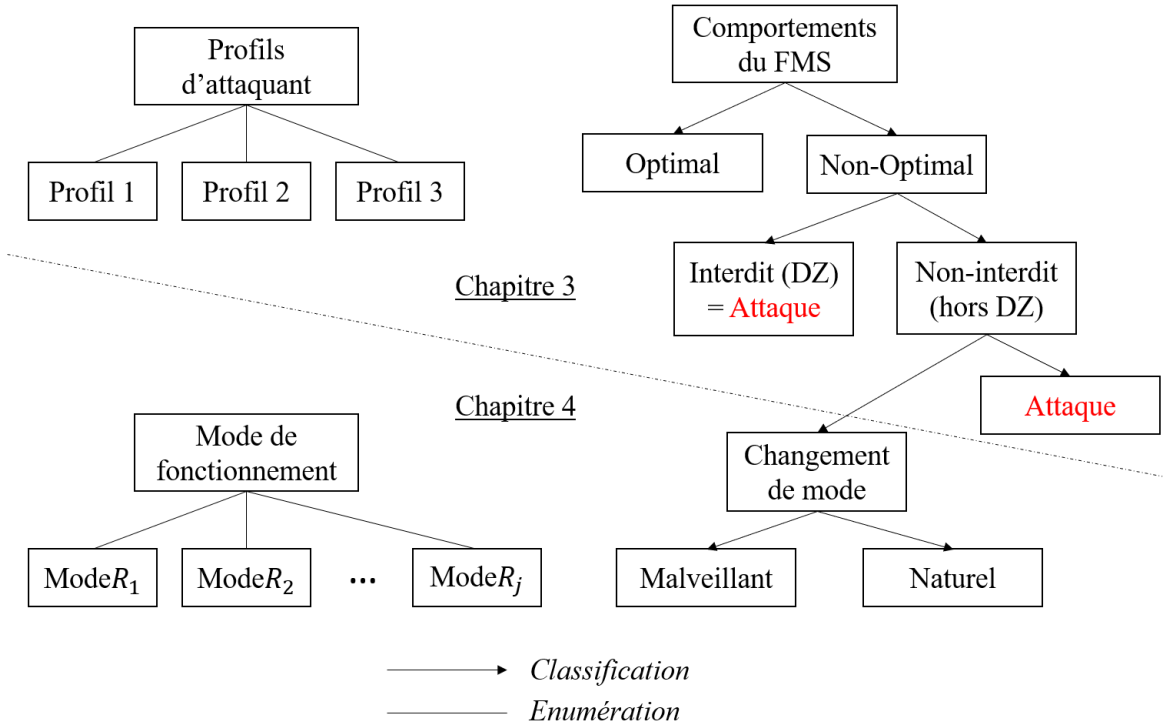


FIGURE 2.5 – Méthodes de diagnostic développées dans nos travaux

Chapitre 3

Proposition d'une méthode de diagnostic des attaques de blocage

Introduction

Dans ce chapitre, un module de diagnostic des attaques de blocage et des profils d'attaquant dans un contexte certain est développé. Ce module de diagnostic est déployé au sein de l'architecture de contrôle commande d'un FMS et doit être capable de réaliser, en ligne, un diagnostic fondé sur la base d'une part, des données fiables extraites depuis les composants du FMS et d'autre part, de diagnostiqueurs temps-réel construits selon les comportements normaux et attaqués. Le fonctionnement général de ce module de diagnostic est illustré dans le diagramme d'activité UML 3.1 ci-dessous.

Les différentes parties de ce chapitre, indiquées dans les cadres jaunes, ont pour objectif de développer les différents blocs du diagramme 3.1 nécessaires au module de diagnostic. Dans la première partie de ce chapitre (3.1), les comportements d'attaque sont introduits au regard de trois profils d'attaquant (**partie 3.A.1**) et d'un modèle des attaques de blocage (**parties 3.A.2 et 3.A.3**) permettant le calcul de ces profils depuis n'importe quel état du FMS. Dans la deuxième partie (3.2), le pilotage des comportements optimaux et la gestion des états critiques de la DZ du FMS sont mis en place par la construction d'un algorithme d'ordonnancement (**partie 3.B.2**) et le choix d'une méthode de prévention (**partie 3.B.1**) appropriés à l'existence des attaques de blocage. De fait, la maîtrise et la connaissance des comportements du FMS générés par ces méthodes sont nécessaires au diagnostic des attaques de blocage ainsi qu'au calcul des profils d'attaquant. Un algorithme original est développé pour le calcul des profils d'attaquant en conclusion de cette partie (**partie 3.B.3**). Dans la troisième et dernière partie de ce chapitre (3.3), le module de diagnostic est présenté. Tout d'abord, son positionnement dans l'architecture du FMS et sa synchronisation avec le module de supervision est discuté (**partie 3.C.1**). Une condition de déclenchement du calcul de l'ordonnancement est proposée de manière commune entre les modules de supervision et de diagnostic (**Partie 3.C.1**). Puis, les méthodes de diagnostic des attaques de blocage et des profils d'attaquant sont développées à partir de la théorie des diagnostiqueurs, avant d'être intégrées au sein d'un seul et même diagnostiqueur (**partie 3.C.2**). Enfin, le fonctionnement global du module de diagnostic est illustré par un exemple détaillé (**Partie 3.C.3**).

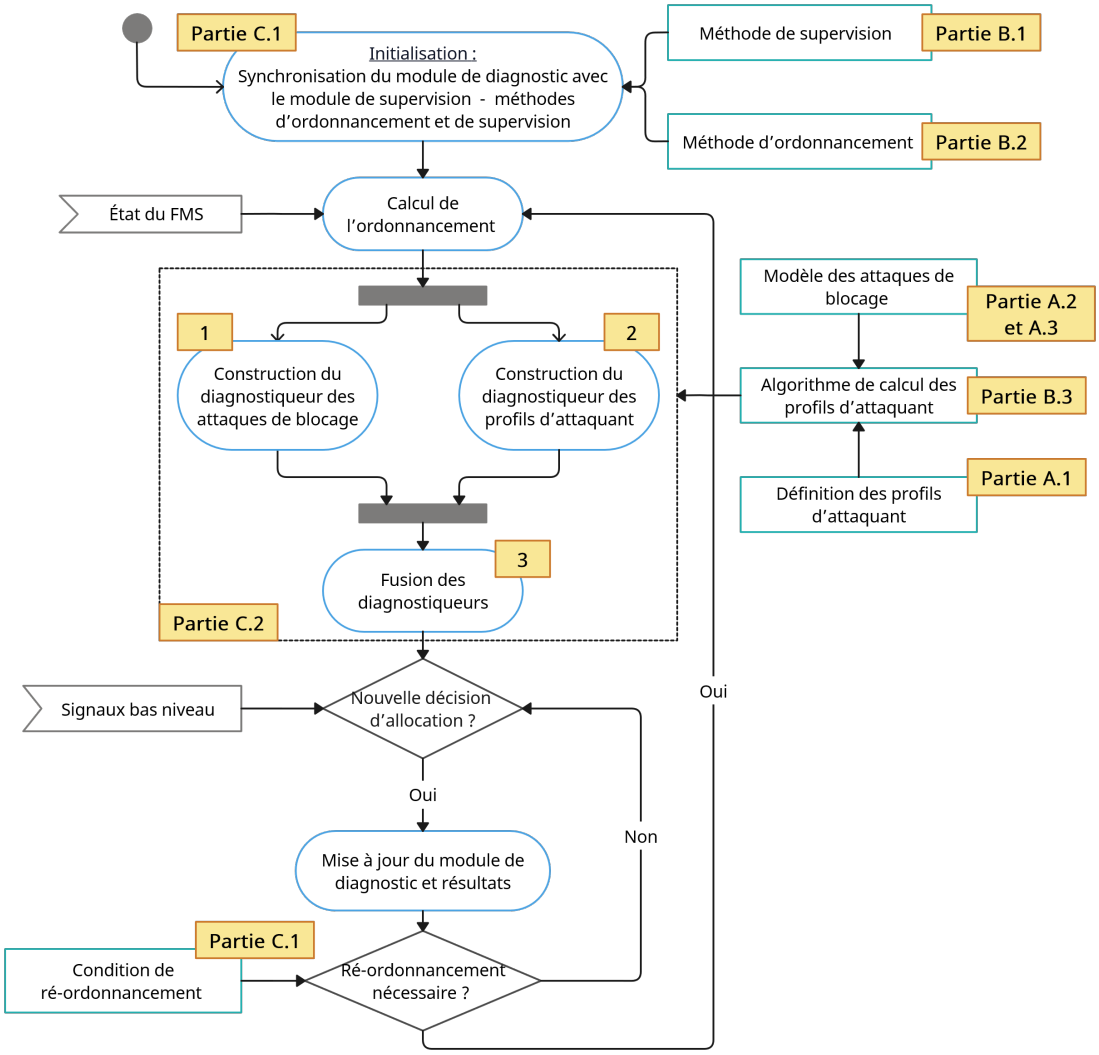


FIGURE 3.1 – Diagramme d'activité UML global du module de diagnostic

3.1 Attaque de blocage : définition et modélisation

La détection d'attaques à partir de modèles comportementaux nécessite une modélisation des comportements du système attaqué d'une part et de l'attaque d'autre part en tant que comportements de référence à diagnostiquer. Le premier modèle, caractérisant le système attaqué - les FMSs, a déjà été réalisé dans le chapitre 1. Toutefois, le choix d'une méthode de gestion des blocages et d'une méthode d'ordonnancement est nécessaire et sera présenté dans la partie 3.2 lorsque les comportements interdits et optimaux seront introduits pour la diagnostic. Le second modèle, celui de l'attaque, est développé dans cette partie. Ce développement comporte trois étapes :

1. Une définition d'une attaque de blocage est proposée et plusieurs profils d'attaquant sont choisis et justifiés ;
2. Un modèle des attaques de blocage reposant sur les RdPs et la classe des S^3PRs est décrit. Une méthode de construction de ce modèle d'attaque est présentée et les différents sous-objectifs d'un attaquant sont explicités à partir de ce modèle et d'extensions de

celui-ci.

3. En conclusion de cette partie, un modèle réduit du modèle d'attaque sera proposé pour répondre à la problématique d'explosion des états inhérente à l'extension d'un modèle S^3PR au modèle d'attaque. Ce modèle réduit sera prouvé équivalent au modèle d'attaque original pour le calcul des profils d'attaquant.

3.1.1 Attaques de blocage et profils d'attaquant

Dans cette première sous-partie, une définition des attaques de blocage est proposée et trois profils d'attaquant sont définis. Dans le diagramme représentant le fonctionnement global du module de diagnostic (figure 3.1), ces définitions sont requises pour la construction du modèle des attaques de blocage et pour le développement de l'algorithme de calcul des profils d'attaquant. En préambule, cinq définitions sur les attaques contre les SEDs présentées dans la partie 2.2.3 sont mobilisées dans cette sous-partie :

- La politique de contrôle ρ d'un FMS ;
- Les attaques d'insertion e_+ et de suppression e_- d'événements ;
- L'ensemble des événements d'un système attaqué E_a et des séquences attaquées $E_a^\star$;
- Le langage d'un attaquant $\mathcal{L}_a$ et le langage sournois d'un attaquant $\mathcal{L}_a|_{sr}$;
- La fonction f_a de projection des événements d'une attaque par rapport aux événements observés par le superviseur (f_a^2) et par le système physique (f_a^1).

A partir de ces éléments théoriques et de la proposition 2.2.1, la définition d'une attaque de blocage est proposée.

Définition 3.1.1 (Attaque de blocage)**.**
Dans un RdP labellisé N_L, soit un marquage $M \in \mathcal{M}_{LZ}$ et un ensemble d'états ciblés par l'attaquant $\mathcal{M}_c \subset \mathcal{M}_{DZ}$. Une attaque de blocage est une séquence $\lambda_a \in \mathcal{L}_a(N_L, M)$ avec $l^{-1}(f_a^1(\lambda_a)) = \sigma$ telle que $\exists M_c \in \mathcal{M}_c$ et $M[\sigma > M_c$. ◇

Définition 3.1.2 (Langage des attaques de blocage)**.**
Le langage de toutes les attaques de blocage λ_a existantes depuis $M \in \mathcal{M}_{LZ}$ vers un état de $\mathcal{M}_c$ est noté $\mathcal{L}_a(N_L, M, \mathcal{M}_c)$. ◇

Définition 3.1.3 (Attaque minimale de blocage)**.**
Soit une séquence d'attaque $\lambda_{a1} \in \mathcal{L}_a(N_L, M, \mathcal{M}_c)$.
λ_{a1} est dite minimale si $\nexists \lambda_{a2} \in \mathcal{L}_a(N_L, M, \mathcal{M}_c)$ tel que $\lambda_{a2} < \lambda_{a1}$, i.e. λ_{a1} ne possède pas de suffixe appartenant au langage des attaques de blocage depuis M. On définit par $\mathcal{L}_a(N_L, M, \mathcal{M}_c)|_{min}$ le langage de toutes les séquences d'attaque minimales. ◇

En considérant $\mathcal{M}_c = \mathcal{M}_{DZ}$, un attaquant est ainsi capable d'exécuter à partir d'un état $M \in \mathcal{M}_{LZ}$ toutes les attaques $\lambda_a \in \mathcal{L}_a(N_L, M, \mathcal{M}_c)$ afin d'atteindre un état de la DZ. Cependant, toutes ces séquences d'attaque λ_a ne sont pas pertinentes pour un attaquant. Par exemple, une séquence d'attaque possible $\lambda_a = \lambda_{a1}\lambda_{a2}$ avec $f_a^1(\lambda_{a1}) \in \mathcal{L}(N_L, M)|_\rho$ et $|\lambda_{a1}| > n \in \mathbb{N}$, n beaucoup plus grand que les cardinaux des éléments de $\mathcal{L}_a(N_L, M, \mathcal{M}_c)$, est une attaque peu appropriée pour un attaquant cherchant à atteindre rapidement un état de la DZ et à investir le minimum de moyens dans son attaque. Afin de restreindre $\mathcal{L}_a(N_L, M, \mathcal{M}_c)$ aux attaques satisfaisant les objectifs d'un attaquant, différents profils d'attaquants sont définis. Ces derniers

sont définis selon les objectifs, sous-objectifs et capacités de l'attaquant (sous-partie 2.2.2) et sont inspirés de la notion de dictionnaires ou fonctions d'attaques introduits par [236].

Définition 3.1.4 (Profil d'attaquant)**.**
Soit 2^Q l'ensemble de tous les sous-ensembles d'un ensemble Q. On définit par $\mathcal{P} : \mathcal{R}(N_L, M_0)|_\rho \rightarrow 2^{\mathcal{L}_a(N_L, M, \mathcal{M}_c)}$ un profil d'attaquant qui associe à un état M appartenant à l'ensemble des états accessibles dans (N_L, M_0) et selon la politique de contrôle ρ, un sous-langage d'attaque $\mathcal{P}(M) \subset \mathcal{L}_a(N_L, M, \mathcal{M}_c)$ choisi par l'attaquant. ◇

Pour chaque profil d'attaquant, sa fonction $\mathcal{P}$ est définie selon les caractéristiques de l'attaquant. Dans nos travaux, une caractéristique est une règle que suit systématiquement un profil d'attaquant lorsqu'il construit sa séquence d'attaque. Une caractéristique peut être définie à partir de l'objectif, d'un sous-objectif (sous-partie 2.2.2) ou d'un attribut lié à l'origine, aux compétences et aux moyens de l'attaquant (sous-partie 2.1.2). Il existe des caractéristiques communes à tous les profils et des caractéristiques propres aux profils d'attaquant considérés. Ci-après, les caractéristiques communes sont présentées, puis les différents profils d'attaquants que nous avons choisis sont formulés et justifiés au regard de leurs caractéristiques spécifiques.

Premièrement, cinq caractéristiques communes à tous les profils d'attaquants ont été choisies. Elles représentent les pré-requis nécessaires à la réussite d'une attaque de blocage et sont les suivantes :

1. L'attaquant prépare une unique séquence d'attaque à partir d'un état M. Cela signifie que l'ensemble des images d'une fonction $\mathcal{P}$ peut être restreint à $\mathcal{L}_a(N_L, M, \mathcal{M}_c)$ puisque l'attaquant ne choisit qu'un élément du langage d'attaque qu'il souhaite exécuter depuis M. Cette caractéristique est fondée sur le principe qu'un profil d'attaquant est déterministe et prépare une unique attaque qu'il déclenche depuis M afin d'être certain d'atteindre son objectif dans $\mathcal{M}_c$ et de respecter ses sous-objectifs.
2. L'attaque choisie est minimale (définition 3.1.3).
3. L'attaquant est expert (définition 2.2.9), i.e. $\forall e \in E_o \cup E_c$, e_+ et e_- existent.
4. L'attaquant est sournois par rapport au superviseur (définition 2.2.12). Il est clair qu'un profil d'attaquant non sournois a peu d'intérêt pour nos travaux puisqu'il serait directement détecté par le superviseur et ne requerrait donc pas l'utilisation d'une méthode de diagnostic.
5. L'attaquant connait l'ordonnancement calculé et piloté par le module de supervision. Par conséquent, il est capable de tromper les décisions d'allocations prises par le module de supervision afin de rester sournois.

En s'appuyant sur la définition d'une attaque et d'un attaquant ciblant l'OT (sous-partie 2.1.2) et des objectifs et sous-objectifs d'une attaque de blocage (sous-partie 2.2.2), trois profils d'attaquants distincts $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ sont construits. Pour chaque profil, une description de l'origine de l'attaquant, de ses moyens et de son objectif global est tout d'abord exhibée, puis les différentes caractéristiques du profil sont énumérées à l'égard de la description liminaire.

Profil 1 : Le premier profil d'attaquant a pour objectif le blocage total du FMS, sans possibilité de réponse, et possède des moyens très élevés (financiers, matériels, humains etc.). Il

pourrait s'apparenter à un attaquant étatique ou à une entreprise concurrente puissante. Nous définissons pour ce profil d'attaquant les caractéristiques suivantes :

1. Il cible spécifiquement un état de blocage. L'ensemble des états ciblés par un attaquant est restreint à $\mathcal{M}_{c_1} = \{M \in \mathcal{R}(N_L, M_0) | \nexists t \in T, M' \in \mathcal{R}(N_L, M_0)\ t.q.\ M[t > M'\}$ et le profil d'attaquant devient $\mathcal{P}_1 : \mathcal{R}(N_L, M)|_\rho \rightarrow \mathcal{L}_a(N_L, M, \mathcal{M}_{c_1})|_{min}$.
2. L'attaquant est capable d'investir un coût d'attaque très élevé grâce à ses moyens. Autrement dit, le coût d'une attaque $\lambda_a = \mathcal{P}_1$ n'est pas borné. La définition des origines d'un coût d'attaque, sa modélisation et son calcul seront présentés ultérieurement sous-partie (3.1.2).
3. L'attaquant cherche à prévenir la capacité de réponse du système à un état de blocage via une méthode de reprise. A cette fin, il met en place deux règles au sein de son profil d'attaquant :
   (a) La séquence d'attaque λ_a évite de faire transiter le FMS dans des états de la DZ hors états de blocages, soit dans des états de pré-blocage ou de blocage partiel. En effet, ces états de pré-blocage, si détectés, laissent au FMS des états transitoires pour répondre à l'attaque avant l'état de blocage total.
   (b) L'attaquant cible des états de blocage dans lesquels les ressources sont les plus utilisées. En effet, les méthodes de reprise [112]-[115] s'appuient dans les modèles S^{3}PR sur des réseaux de reprise déplaçant de manière temporaire un produit bloqué vers une ressource non bloquante disponible afin de libérer la ressource détenue par ce produit. Si peu de ressources sont disponibles, ces méthodes de reprises deviennent moins efficaces voire inapplicables.
4. Enfin, après avoir appliqué toutes les règles précédentes, si plusieurs séquences d'attaques sont encore éligibles dans $\mathcal{L}_a(N_L, M, \mathcal{M}_{c_1})|_{min}$, l'attaquant choisit l'attaque la plus économe en nombre d'opérations puis en temps d'opération si un choix subsiste.

Profil 2 : Le deuxième profil d'attaquant cherche à bloquer complètement le FMS, en atteignant un état de blocage ou de pré-blocage indifféremment, mais possède des moyens limités. Ce profil d'attaquant pourrait appartenir à une entreprise concurrente modeste ou à des hacktivistes souhaitant dégrader la production et l'image de l'entreprise via une tentative d'attaque de blocage. Nous attribuons à ce profil les caractéristiques suivantes :

1. Il cible spécifiquement un état de blocage ou de pré-blocage. L'ensemble des états ciblés par un attaquant est restreint à $\mathcal{M}_{c_2} = \{M \in \mathcal{R}(N_L, M_0) | \nexists t \in T, M' \in \mathcal{R}(N_L, M_0)\ t.q.\ M[t > M'\} \cup \{M \in \mathcal{R}(N_L, M_0) | \forall M' \in \mathcal{R}(N_L, M_0), \sigma \in T\ t.q. M[\sigma > M' \Rightarrow M' \in \mathcal{M}_{c_1} \cup \mathcal{M}_{c_2}\}$ et le profil d'attaquant devient $\mathcal{P}_2 : \mathcal{R}(N_L, M)|_\rho \rightarrow \mathcal{L}_a(N_L, M, \mathcal{M}_{c_2})|_{min}$.
2. L'attaquant ne peut investir qu'un coût borné dans son attaque puisque ce dernier possède des moyens limités. Soit $C^2_{max} \in \mathbb{R}$ la borne supérieure de l'attaque $\lambda_a = \mathcal{P}_2$.
3. Après avoir appliqué toutes les règles précédentes, si plusieurs séquences d'attaques sont encore éligibles dans $\mathcal{L}_a(N_L, M, \mathcal{M}_{c_2})|_{min}$, l'attaquant choisit l'attaque ayant le rapport $(coût) \times (nombre\ opérations)$ le plus faible. Cela signifie que l'attaquant cherche à atteindre un état de la DZ en minimisant conjointement le nombre d'opérations pour y parvenir et le coût de réalisation de son attaque.

4. Enfin, si un choix subsiste encore, l'attaquant choisit l'attaque avec le plus petit coût.

Profil 3 : Le troisième profil d'attaquant cible les états de blocage partiel avec l'objectif de dégrader la production, en bloquant certains circuits, mais sur une durée longue en ne créant pas de blocage total. En effet, ce profil considère qu'un FMS opérant partiellement n'alertera pas immédiatement les opérateurs puisque certaines ressources s'exécutent encore malgré l'attaque. L'antagoniste pourrait être lié à une entreprise concurrente souhaitant dégrader sur la durée les performances du FMS. Les caractéristiques suivantes sont attribuées à ce dernier profil :

1. L'attaquant cible spécifiquement un état de blocage partiel. L'ensemble des états ciblés par un attaquant est restreint à $\mathcal{M}_{c_3} = \mathcal{M}_{DZ} \setminus \mathcal{M}_{c_2}$ et le profil d'attaquant devient $\mathcal{P}_2 : \mathcal{R}(N_L, M)|_\rho \rightarrow \mathcal{L}_a(N_L, M, \mathcal{M}_{c_3})|_{min}$.
2. L'attaquant ne peut investir qu'un coût borné dans son attaque. Soit $C^3_{max} \in \mathbb{R}$ la borne supérieure de l'attaque $\lambda_a = \mathcal{P}_3$. On suppose que $C^3_{max} > C^2_{max}$.
3. Après avoir appliqué toutes les règles précédentes, si plusieurs séquences d'attaques sont encore éligibles dans $\mathcal{L}_a(N_L, M, \mathcal{M}_{c_3})|_{min}$, l'attaquant choisit l'attaque ayant le rapport $(co\hat{u}t) \times (dur\acute{e}e)$ le plus faible. Cela signifie que l'attaquant souhaite atteindre rapidement un état de la LZ tout en réduisant le coût de son attaque.
4. Enfin, si un choix subsiste encore, l'attaquant choisit l'attaque avec le plus petit coût.

Dans cette partie, trois profils d'attaquant ont été définis selon les objectifs, origines et caractéristiques qui les singularisent. **Par rapport aux travaux de la littérature définissant un attaquant selon ses capacités d'insertion, de suppression et de remplacement d'événements observables et/ou contrôlables, capacités supposées expertes dans nos recherches, l'objectif des profils d'attaquant est de restreindre les séquences d'attaques étudiées à des scénarios réalistes**. Dans nos travaux, les profils choisis se veulent représentatifs de scénarios aux caractéristiques diversifiées. Ainsi, tous les états de la DZ ont été inclus dans les profils sélectionnés et différentes variations de la borne supérieure du coût C^i_{max} sont proposées afin de représenter les contrastes de moyens parmi les attaquants. Dans le reste de ce manuscrit, les méthodes de diagnostic seront construites et évaluées à partir de ces 3 profils d'attaquants. En outre, à partir d'une détection d'un comportement d'attaque, le diagnostic du profil d'attaquant sera mené afin d'identifier le ou les profils, s'ils existent, pouvant être à l'origine de l'attaque. A l'inverse, si aucun profil ne peut être reconnu, le diagnostic explicitera alors la singularité défiante du comportement d'attaque observé. Ainsi, sans équivoque, le diagnostic des profils d'attaquant permettra de caractériser l'attaque et d'enrichir les informations de détection transmises aux opérateurs.

3.1.2 Modèle d'une attaque de blocage

Dans cette deuxième sous-partie, un modèle des attaques de blocage contre les FMSs est développé. Au sein du diagramme représentant le fonctionnement global du module de diagnostic (figure 3.1), le modèle d'attaque est requis par l'algorithme de construction des profils

d'attaquant et pour la génération du diagnostiqueur des attaques de blocage.

Les attaques de blocage ont été jusqu'à présent définies et étudiées au regard des événements qu'elles sont capables de manipuler (définitions 2.2.7, 2.2.8) et aux langages malveillants qu'elles peuvent générer (définitions 2.2.11, 3.1.2). Or, afin de construire les différents profils d'attaquant depuis un état M en tenant compte des caractéristiques de chaque profil et sans requérir la génération de tout le langage d'attaque, un modèle incluant ces caractéristiques et les événements vulnérables est nécessaire. Dans cette partie, un modèle des attaques de blocage contre un FMS modélisé par le S^{3}PR N est introduit en cinq étapes.

1. Les événements observables et contrôlables du FMS pour l'allocation des ressources sont identifiés et intégrés dans une nouvelle classe de RdP, extension des S^{3}PRs ;
2. A partir de cette extension, un modèle RdP composant le modèle du système physique G et le modèle du superviseur S est développé. Cette composition permettra lors de l'intégration d'événements attaqués à cette dernière de modéliser à la fois les effets de l'attaque sur G et sur S ;
3. Chaque événement attaqué $e_a \in (E^+ \cup E^-)$ est intégré au modèle composé à travers une transition d'attaque. Le modèle ainsi obtenu est le modèle des attaques de blocage ;
4. La notion de coût d'une attaque est définie plus en détail et intégrée au modèle d'attaque ;
5. Enfin, les caractéristiques temporelles d'une attaque sont modélisées via la temporisation des places du modèle d'attaque.

Événements pour l'allocation des ressources et extension du modèle N à N_L

Dans un modèle S^{3}PR noté $N = (P_A \cup P^0 \cup P_R, T, F)$, le franchissement d'une transition symbolise une décision d'allocation prise par le module de supervision : l'opération O_{i+1} requérant la ressource R_l est débutée, libérant alors la ressource R_k détenue par l'opération précédente O_i. Néanmoins, une décision d'allocation n'est pas un événement singulier communiqué entre le module de supervision et un contrôleur mais une séquence d'événements de commande ordonnant l'application de la décision au sein du FMS et d'événements d'observation validant l'exécution de la décision. Du reste, par définition, un attaquant manipule exclusivement les événements observables et contrôlables d'un SED. Par conséquent, les transitions de N ne peuvent être labellisées par une décision d'allocation afin de construire le modèle de l'attaque mais, a contrario, la construction de ce modèle requiert que les différents événements constitutifs d'une décision d'allocation soient identifiés, classifiés entre commandes et observations, et modélisés par l'ajout à N de transitions labellisées par ces derniers.

Ainsi, une décision d'allocation peut être décomposée en deux événements de commande et deux événements d'observation validant l'exécution de ces commandes. La première commande est l'ordre de début de l'opération O_{i+1} envoyé au contrôleur $CTRL_l$ pilotant la ressource R_l. Cet événement de commande est noté $e_{deb}(O_i)$. La seconde commande est l'événement de libération de la ressource R_k, détenue jusqu'alors par l'opération O_i. Cette commande est envoyée au contrôleur $CTRL_k$ et est notée $e_{lib}(R_k)$. En réponse différée à $e_{deb}(O_{i+1})$, le contrôleur $CTRL_l$ notifie le module de supervision de la fin de l'opération O_{i+1} par l'événement $e_{fin}(O_{i+1})$. Similairement, le contrôleur $CTRL_k$ envoie au module de supervision un événement d'observation $e_{dis}(R_k)$ validant la libération et la disponibilité de la ressource R_k.

Au sein d'un FMS, après la prise d'une décision d'allocation par le module de supervision, les événements de commande de début d'opération et de libération de ressource peuvent être envoyés simultanément ou successivement selon les types des ressources R_l et R_k et leurs interactions physiques. Par exemple, dans le cas où un bras de robot R_k transporte et dépose un produit sur le socle dédié d'une machine R_l, la commande de libération de la ressource R_k doit être réalisée en premier, puis, une fois le produit correctement libéré et le bras de robot écarté de la machine, l'opération O_{i+1} peut être débutée par la machine R_l. A l'inverse, lorsqu'une opération O_i est terminée par la machine R_k et attend que le bras de robot R_l vienne se saisir du produit et le transporter (opération O_{i+1}), la commande de début $e_{deb}(O_{i+1})$ est envoyée avant $e_{lib}(R_k)$ afin que le robot se saisisse du produit, évacue la machine R_k de ce dernier et permette à la commande de libération de réinitialiser la machine R_k et de la préparer à recevoir un nouveau produit. Afin de modéliser cette incertitude sur l'ordre d'envoi de ces deux commandes tout en conservant leur simultanéité relative à une unique décision d'allocation, elles sont regroupées au sein d'un unique événement de commande labellisé $e_{dec}(O_{i+1}, R_k)$ incluant $e_{deb}(O_{i+1})$ et $e_{lib}(R_k)$ et représentant la décision d'allocation occasionnant le début de O_{i+1} et la libération de R_k. Dans le cas où les deux commandes se succèdent, le délai temporel entre l'envoi de celles-ci sera pris en compte lors de la temporisation du RdP expliquée dans la sous partie 3.1.2.

Remarque 3.1.1. Notons que dans nos travaux, le superviseur ne doit pas envoyer une demande de réservation de la ressource, et se la voir accorder par le contrôleur, avant de prendre une décision d'allocation. De fait, dans notre architecture de contrôle commande, le module de supervision est le seul capable de réserver une ressource. Par exemple, la réservation d'une ressource pour une opération de maintenance préventive est réalisée via le module de supervision. ⌟

Soit $N_L = (P_A^L \cup P^0 \cup P_R^L, T_L, F_L, E, l)$ l'extension de N incluant de nouvelles transitions labellisées par les événements présentés précédemment. Prenons dans N_L une transition $t \in T$; elle représente une décision d'allocation prise par le module de supervision et possède deux arcs de sortie (t, p_{a_i}) et (t, p_{r_k}) avec $p_{a_i} \in P_A$ représentant l'opération O_i et $p_{r_k} \in P_R$ la disponibilité de la ressource R_k. L'arc (t, p_{a_i}) représente alors la commande $e_{deb}(O_i)$, et (t, p_{r_k}) la commande $e_{lib}(R_k)$. Ces deux arcs modélisent naturellement le parallélisme souhaité pour modéliser l'occurrence de ces deux événements de commande suite à la décision d'allocation associée à t. Par conséquent, la transition $t \in T$ représentant la décision d'allocation peut-être labellisée par $e_{dec}(O_i, R_k)$ tandis que l'événement $e_{fin}(O_i)$ (resp. $e_{dis}(R_k)$) est ajouté dans N_L après l'arc (t, p_{a_i}) (resp. (t, p_{r_k})). Mathématiquement, nous proposons d'inclure l'événement $e_{fin}(O_i)$ (resp. $e_{dis}(R_k)$) au sein d'une extension de la place p_{a_i} (resp. p_{r_k}) ayant les mêmes arcs d'entrée et de sortie que p_{a_i} (resp. p_{r_k}).

Soit $N_L(p_{a_i}) = (P_{a_i}, T_{a_i}, F_{a_i}, E_{a_i}, l_{a_i})$ le sous-réseau de Petri labellisé de N_L extension de la place $p_{a_i} \in P_A$. On suppose que la place p_{a_i} représente l'opération O_i. On a $P_{a_i} = \{p_{a_i}^1, p_{a_i}^2\}$ et $T_{a_i} = \{t_{a_i}\}$. Les arcs reliant les places de P_{a_i} aux transitions de T_{a_i} sont définis par $F = \{(p_{a_i}^1, t_{a_i}), (t_{a_i}, p_{a_i}^2)\}$. Afin de représenter les événements correspondants à l'opération O_i, on définit $E_{a_i} = \{e_{fin}(O_i)\}$ puis $l(t_{a_i}) = e_{fin}(O_i)$, le franchissement de la transition t_{a_i} signifiant la fin de l'opération O_i. Dans $N_L(p_{a_i})$, la place $p_{a_i}^1$ correspond à l'exécution de O_i tandis que $p_{a_i}^2$ représente l'état d'attente de la prochaine décision d'allocation une fois O_i terminée.

Soit $N_L(p_{r_k}) = (P_{r_k}, T_{r_k}, F_{r_k}, E_{r_k}, l_{r_k})$ le sous-réseau de Petri labellisé de N_L extension de la place $p_{r_k} \in P_R$. On suppose que la place p_{r_k} représente la disponibilité de la ressource R_k. On a $P_{r_k} = \{p_{r_k}^1, p_{r_k}^2\}$ et $T_{r_k} = \{t_{r_k}\}$. Les arcs reliant les places de P_{r_k} aux transitions de T_{r_k} sont définis

par $F = \{(p^1_{r_k}, t_{r_k}), (t_{r_k}, p^2_{r_k})\}$. Afin de représenter les événements correspondants à la libération de la ressource R_k, on définit $E_{r_k} = \{e_{dis}(R_k))\}$ puis $l(t_{r_k}) = e_{dis}(R_k)$, le franchissement de la transition t_{r_k} signifiant la fin de la libération de R_k. Dans $N_L(p_{r_k})$, la place $p^1_{r_k}$ correspond à l'opération de libération de R_k tandis que $p^2_{r_k}$ représente la disponibilité de la ressource R_k, en état d'attente de le prochaine décision d'allocation requérant R_k.

Définition 3.1.5 (Modèle étendu labellisé N_L)**.** A partir des définitions des sous-réseaux $N_L(p_{a_i})$ et $N_L(p_{r_k})$, on construit $N_L = (P^L_A \cup P^0 \cup P^L_R, T_L, F_L, E, l)$ de la manière suivante :

- $P_L = P^L_A \cup P^0 \cup P^L_R$ avec
  - $P^L_A = \bigcup_{i=1}^{|P_A|} P_{a_i}$;
  - $P^L_R = \bigcup_{k=1}^{|P_R|} P_{r_k}$;
- $T_L = T \cup T^a_L \cup T^r_L$ avec :
  - $T^a_L = \bigcup_{i=1}^{|P_A|} T_{a_i}$;
  - $T^r_L = \bigcup_{k=1}^{|P_R|} T_{r_k}$;
- $F_L = F^a_L \cup F^r_L$ avec :
  - $F^a_L = \bigcup_{i=1}^{|P_A|}(F_{a_i} \cup \{(t, p^1_{a_i}), \forall t \in {}^\bullet p_{a_i}\} \cup \{(p^2_{a_i}, t), \forall t \in p^\bullet_{a_i}\})$;
  - $F^r_L = \bigcup_{k=1}^{|P_R|}(F_{r_k} \cup \{(t, p^1_{r_k}), \forall t \in {}^\bullet p_{r_k}\} \cup \{(p^2_{r_k}, t), \forall t \in p^\bullet_{r_k}\})$;
- $E = (\bigcup_{i=1}^{|P_A|} E_{a_i}) \cup (\bigcup_{k=1}^{|P_R|} E_{r_k}) \cup (\bigcup_{t \in T} e_{dec}(O_i, R_k))$ avec :
  - $e_{dec}(O_i, R_k) = \{e_{deb}(O_i), e_{lib}(R_k)\}$ si $t \in T \cap {}^\bullet p_{a_i} \cap {}^\bullet p_{r_k}$;
  - $e_{dec}(O_i, R_k) = e_{dec}(O_i, -) = e_{deb}(O_i)$ si $t \in T \cap {}^\bullet p_{a_i} \cap (P^0_e)^\bullet$;
  - $e_{dec}(O_i, R_k) = e_{dec}(-, R_k) = e_{lib}(R_k)$ si $t \in T \cap {}^\bullet p_{r_k} \cap {}^\bullet(P^0_e)$;
- $l : T_L \to (E \cup \varepsilon)$ définit par :
  - $l(t) = l_{a_i}(t) = e_{fin}(O_i)$ si $t \in T_{a_i}, i \in [1, |P_A|]$;
  - $l(t) = l_{r_k}(t) = e_{lib}(R_k)$ si $t \in T_{r_k}, k \in [1, |P_R|]$;
  - $l(t) = e_{dec}(O_i, R_k) = \{e_{deb}(O_i), e_{lib}(R_k)\}$ si $t \in T \cap {}^\bullet p_{a_i} \cap (P^0_e)^\bullet$;
  - $l(t) = e_{dec}(O_i, -) = e_{deb}(O_i)$ si $t \in T \cap {}^\bullet p_{a_i} \cap (P^0_e)^\bullet$;
  - $l(t) = e_{dec}(-, R_k) = e_{lib}(R_k)$ si $t \in T \cap {}^\bullet p_{r_k} \cap {}^\bullet(P^0_e)$;

  Un marquage initial M^L_0 de N_L est qualifié d'acceptable ssi :
- $M^L_0(p^0) \geq 1$ avec $p^0 \in P^0_e$;
- $M^L_0(p_a) = 0$ avec $p_a \in P^e_A$;
- $M^L_0(p_r) = 1$ si $p_r = p^2_{r_k}, k \in [1, |P_R|]$ et $M^L_0(p_r) = 0$ sinon.

◇

Pour une opération O_i détenant la ressource R_k, les sous-réseaux de N_L modélisant les événements relatifs au début et à la fin de O_i ainsi qu'à la libération de R_k sont illustrés dans la figure 3.2. Dans la suite de ce manuscrit, un événement e associé à la transition t sera noté $t(e)$ pour des raisons de clarté au sein des différentes figures.

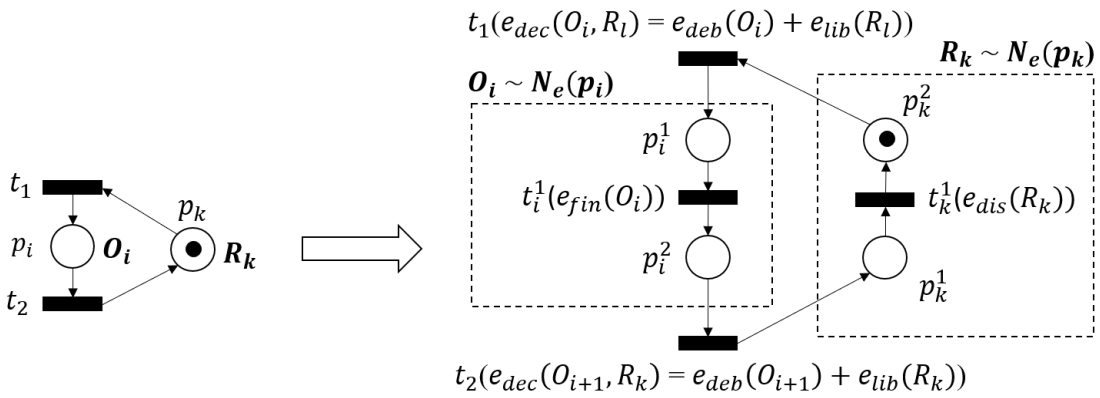


FIGURE 3.2 – Exemple d'extension de N à N_L

Composition des modèles du système physique et du superviseur

Le modèle S^{3}PR étendu N_L présenté dans les paragraphes précédents permet la représentation de tous les événements observables et contrôlables propres aux décisions d'allocation des ressources du FMS. L'intégration de ces événements au modèle du FMS est nécessaire à la modélisation des attaques de blocage capable de supprimer ou d'insérer ces événements. Néanmoins, la définition 2.2.12 d'une attaque sournoise proposée dans la section 2.2.3 ainsi que les caractéristiques des profils d'attaquant présentés précédemment (sous-partie 3.1.1) requièrent la modélisation des effets de l'attaque sur le système physique G et sur le module de supervisions S. En effet, une attaque de blocage est fondée sur les séquences malveillantes d'événements conduisant le système physique à un état bloquant tandis que le sous-objectif de sournoiserie d'une attaque est, par définition, dépendant du langage observable attendu par le superviseur. L'objectif de cette sous-partie est d'introduire un modèle inspiré des travaux de [222] incluant les comportements pour l'allocation des ressources de G et de S.

Soit $N_G = (P_G, T_G, F_G, E_G, l_G)$ le modèle représentant l'état réel du FMS pour l'allocation des ressources et $N_S = (P_S, T_S, F_S, E_S, l_S)$ celui modélisant l'état du FMS observé et piloté par le module de supervision. On définit initialement ces deux modèles par $N_G = N_S = N_L$. La composition de N_G et N_S notée $N_{GS} = N_G \parallel N_S = (P_{GS} = P_G \cup P_S, T, F_{GS} = F_G \cup F_S, E, l)$ est définie en annexe de ce document. La composition de modèles $\parallel$ conserve les places des deux sous-modèles N_G et N_S et fusionne leurs transitions selon la méthode proposée par [222]. Ainsi, les arcs de F_{GS} résultant de cette composition connectent les places distinctes de N_G et N_S avec les transitions fusionnées au sein de T_{GS}.

Le choix de composer ces deux modèles via leurs transitions et événements communs se justifie par le fonctionnement de la boucle de contrôle dans un contexte sans attaque. En effet, on suppose que l'absence d'attaque signifie qu'un événement contrôlable généré par S atteint est systématiquement exécuté par G, tandis qu'un événement observable émis par G est systématiquement reçu et observé par S. Par conséquent, le franchissement d'une transition $t \in T$ labellisée par un événement $e = l(t) \in E$ entraine la mise à jour simultanée des marquages des places de N_G et de N_S connectés à t puisque l'on considère que l'occurrence de e existe pour S et pour G. A contrario, bien qu'identiques, les ensembles de places P_G et P_S ne sont pas fusionnés au sein de P_{GS} afin de représenter distinctement l'état réel de l'allocation des ressources et l'état observé par le module de supervision. Cette distinction permettra, lors de la

modélisation des attaques contre les événements du FMS, de différencier les effets malveillants sur G et sur S selon le type d'attaque (insertion ou suppression) et le type d'événement attaqué (contrôlable ou observable). On note p^G une place $p \in P_G \cup P_{GS}$, p^S une place $p \in P_S \cup P_{GS}$ et $M|_G$ (resp. $M|_S$) un marquage de N_{GS} réduit aux place de P_G (resp. P_S).

Remarque 3.1.2. Dans le cas où une méthode de gestion des états de blocage conduit à la construction de places moniteurs $p_c \in P_c$ reliées par des arcs F_c aux transitions du modèle S^{3}PR afin de contraindre leurs franchissements, ces dits places et arcs sont intégrés dans un premier temps au modèle du superviseur N_S puis par composition à N_{GS} en suivant les étapes décrites ci-dessus. On a alors $P_c \subset P_S$, $F_c \subset F_S$, et $M_0(p_c), \forall p_c \in P_c$ défini dans M_0^S à partir de la méthode de prévention. ⌟

Dans la continuité de l'exemple de la figure 3.2, le modèle N_{GS} de ce dernier est illustré dans la figure 3.3. Précisément, les modèles N_G et N_S sont représentés séparément en haut de la figure puis composés en N_{GS} en bas.

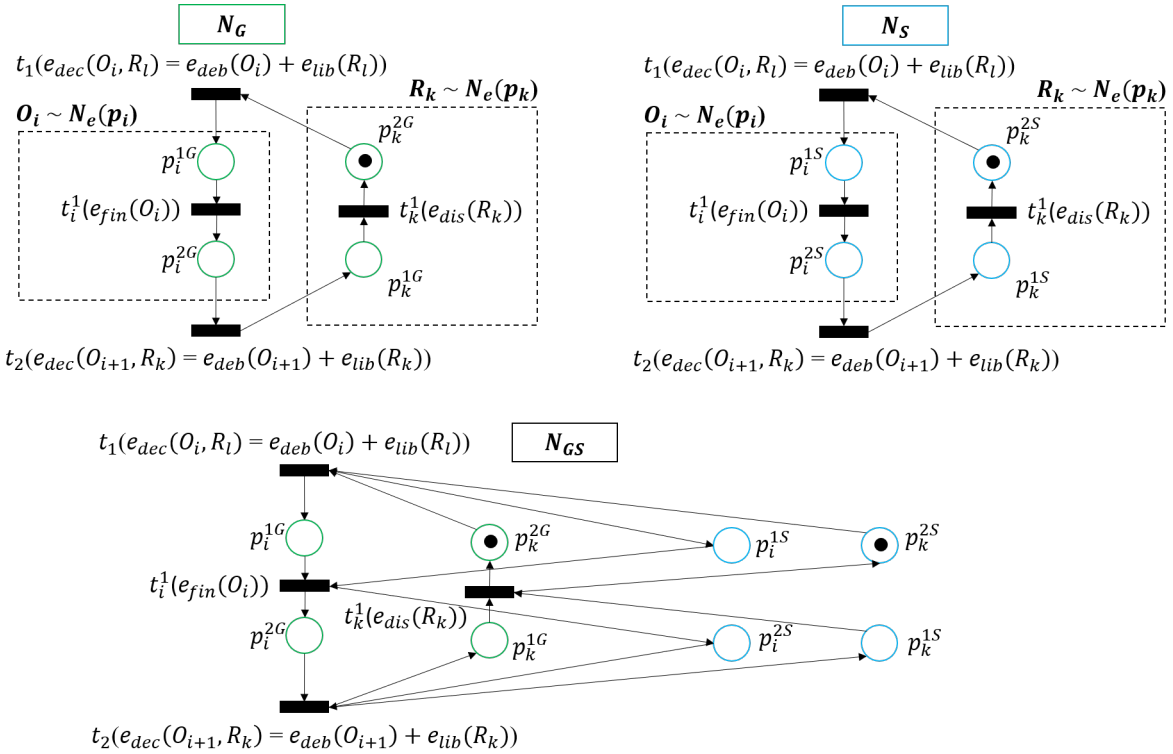


FIGURE 3.3 – Exemple de composition de N_G et N_S en N_{GS}

Modèle des attaques de blocage

Le modèle des attaques de blocage a pour objectif de modéliser toutes les séquences d'événements, manipulées ou non, exploitables par un attaquant pour atteindre un état de la DZ. Au sein du modèle N_{GS}, seuls les événements non attaqués sont représentés à travers les transitions qu'ils labellisent. Par conséquent, afin de construire le modèle d'attaque, chaque événement attaqué doit être intégré à N_{GS} par l'ajout de nouvelles transitions labellisées.

Définition 3.1.6 (Modèle des attaques de blocage N_a)**.** Soit $N_a = (P_{GS}, T_a, F_a, E_a, l_a)$ le modèle d'attaque. Il est défini de la manière suivante :

- Le modèle d'attaque contient toutes les places $P_{GS} = P_G \cup P_S$.
- $E_a = E \cup E_+ \cup E_-$ avec :
  - $E_- = \{e_- | e \in E\}$;
  - $E_+ = \{e_+ | e \in E\}$;

  E_a est défini ainsi car tous les événements de E sont vulnérables aux attaques d'insertion et de suppression (attaquant expert).
- $T_a = T \cup T_- \cup T_+$ avec :
  - $T_- = \{t^-, \forall t \in T\}$;
  - $T_+ = \{t^+, \forall t \in T\}$;

  T_- et T_+ sont les ensembles des transitions représentant respectivement la suppression (e_-) et l'insertion (e_+) de l'événement e lié à $t \in T$ par $l(t) = e$.
- Soit $\tau_a : T_a \to T$ la fonction qui associe à chaque transition $t_a \in T_a$ la transition non attaquée $t \in T$ correspondante, i.e. $\tau_a(t) = t, \tau_a(t^+) = t$ et $\tau_a(t^-) = t$. Cette fonction peut être étendue à $\tau_a : T_a^\star \to T^\star$ par $\tau_a(\sigma_a t_a) = \tau_a(\sigma_a)\tau_a(t_a)$ avec $\sigma_a \in T_a^\star$.
- Soit $l_a : T_a \to (E_a \cup \varepsilon)$ la fonction de labellisation des transitions de T_a définit par :
  - $l_a(t) = l(t)$ si $t \in T$;
  - $l_a(t) = e_+$ si $t \in T_a^+, l(\tau_A(t)) = e$;
  - $l_a(t) = e_-$ si $t \in T_a^-, l(\tau_A(t)) = e$;
- $F_a = F_{GS} \cup F_{T_a}$ avec :
  - $F_{T_a} = \{(p, t_a)/(t_a, p) | t_a \in T_a^- \cup T_a^+, \tau_A(t_a) = t, l_a(t) = e, l_a(t_a) = e_a, (p, t)/(t, p) \in F_{GS}$ *et* $((p \in P_G) \wedge (f_a^1(e_a) = e)) \vee ((p \in P_S) \wedge (f_a^2(e_a) = e))\}$
- Un marquage initial M_0^a est qualifié d'acceptable s'il est acceptable pour N_{GS}.

◇

La définition de F_{T_a} proposée ci-dessus signifie qu'une transition d'attaque $t_a \in T_a$ telle que $\tau_A(t_a) = t, l_a(t_a) = e_a, l_a(t) = e$ est connectée aux places liées à t et représentant l'état de G (resp. de S), à savoir les places $p \in P_G \cap ({}^\bullet t \cup t^\bullet)$ (resp. $p \in P_S \cap ({}^\bullet t \cup t^\bullet)$) si et seulement si l'effet de l'attaque $f_a^1(e_a)$ sur G (resp. $f_a^2(e_a)$ sur S) existe, est égale à e et n'est pas l'élément vide ε. A l'inverse, lorsque $f_a^1(e_a) = \varepsilon$ (resp. $f_a^2(e_a) = \varepsilon$), G (resp. S) considère qu'aucun événement n'a lieu, et par conséquent les places de P_G (resp. P_S) ne sont pas connectées à la transition t_a et conservent le même marquage quand t_a est franchie.

Le modèle d'attaque est illustré dans la figure 3.4 reprenant l'exemple de la figure 3.3. Dans cette figure, les transitions d'attaque sont représentées en rouge et notées $t^+(e_+)$ et $t^-(e_-)$. Par ailleurs, dans ce modèle d'attaque, les événements $e_{deb}(O_i)$ et $e_{lib}(R_k)$ composés en $e_{des}(O_i, R_k)$ sont systématiquement insérés ($t^+(e_{dec}(O_i, R_k)_+)$) et supprimés ($t^-(e_{dec}(O_i, R_k)_)-$) ensemble par l'attaquant puisqu'il a été présenté dans la sous-partie précédente (3.1.2) que ces événements sont interdépendants, i.e. une opération ne peut pas être débutée sans que la ressource détenue par l'opération précédente ne soit libérée et inversement. Ainsi, un attaquant devra supprimer ces deux événements de commande s'il souhaite tromper une décision d'allocation et les insérer s'il souhaite débuter une opération ou libérer une ressource.

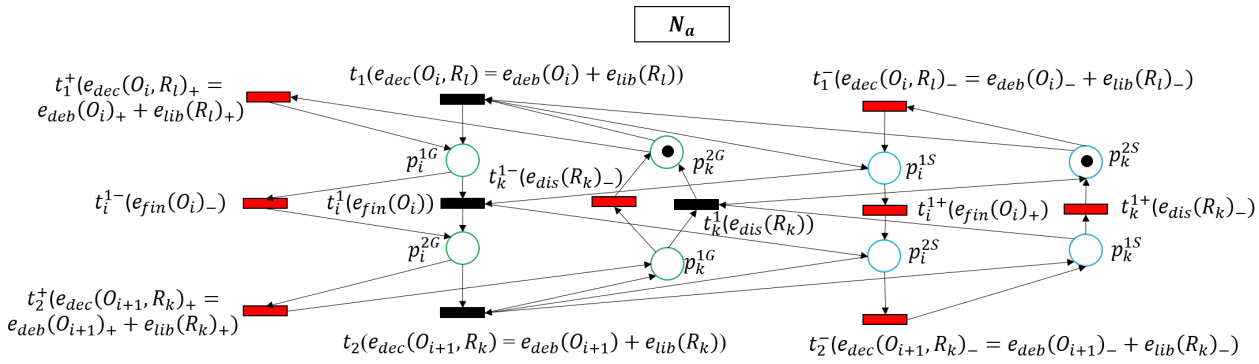


FIGURE 3.4 – Exemple de construction de N_a à partir de l'exemple de la figure 3.3

Proposition 3.1.1. *Soient N_L un S^3PR labellisé et N_a le modèle d'attaque construit à partir de N_L. On a que $\mathcal{L}_a(N_L, M_0^L)|_{sr} = \mathcal{L}(N_a, M_0^a)$. Autrement dit, le langage généré à partir du modèle N_a est équivalent au langage des attaques sournoises expertes défini initialement (2.2.12).*

Preuve 3.1.1.
Cette propriété peut être prouvée en considérant les événements séparément. Lors de la construction de N_a, l'ajout lors des étapes 3,4 et 5 d'une transition d'attaque labellisée par un événement $e_a \in E_a \setminus E$ respecte l'effet de l'attaque sur G et S selon le type d'attaque (insertion ou suppression) et le type d'événement (observation ou commande). Précisément, pour une transition $t_a \in T_a, l(t_a) = e_a, \tau(t_a) = t$, la création des arcs à l'étape 5 permet de garantir ces effets puisque t_a est reliée aux places de G si $f_a^1(e_a) = e$ et aux places de S si $f_a^2(e_a) = e$ correspondantes aux places de P_G et P_S auxquelles t est reliée. Notons que la sournoiserie est modélisée dans N_a par la prise en compte des effets $f_a^2(e_a)$ sur S dans la construction des transitions d'attaque. Ainsi, puisque tous les événements de E_a sont considérés lors de la construction des transitions de T_a et des arcs associés, on a que $\mathcal{L}_a(N_L, M_0^L) \subset \mathcal{L}(N_a, M_0^a)$. Si une politique de contrôle ρ est appliquée à N, les places "moniteurs" sont intégrées à P_S et sont connectées aux transitions d'attaque t_a tel que $f_a^2(l(t_a) = e_a) = e$ et tel qu'il existe des arcs reliant ces moniteurs aux transitions $t = \tau(t_a)$. On obtient alors $\mathcal{L}_a(N_L, M_0^L)|_\rho \subset \mathcal{L}(N_a, M_0^a)$. La réciproque de cette preuve est vraie si l'on considère que la méthode de construction des transitions d'attaque est exhaustive et n'ajoute pas de nouveaux événements ou de nouvelles attaques. Pour $e_a \in \mathcal{L}(N_a, M_0^a)$, on a donc que $e_a \in \mathcal{L}_a(N_L, M_0^L)|_\rho$. Par extension à une séquence d'événements attaqués, on obtient que $\mathcal{L}(N_a, M_0^a) \subset \mathcal{L}_a(N_L, M_0^L)|_\rho$. **Par inclusion mutuelle, nous prouvons donc que** $\mathcal{L}(N_a, M_0^a) = \mathcal{L}_a(N_L, M_0^L)|_\rho$.

⌟

La proposition 3.1.1 permet de valider l'objectif ciblé lors de la construction du modèle N_a, à savoir sa capacité à représenter toutes les attaques (de blocage) sournoises expertes à partir de son langage. Ce modèle permettra de calculer de manière exacte les différents profils d'attaquant à partir de tous les états M accessibles de N_L. Dans les deux prochaines sous-parties, le calcul des profils d'attaquant est rendu possible grâce à la définition et à la modélisation du coût d'une attaque et à la temporisation du RdP.

Remarque 3.1.3. Dans le modèle N_a, l'attaquant pourrait créer un blocage en supprimant une décision d'allocation $e_{dec}(O_i, R_k)_-$ envoyée depuis le superviseur et en insérant les observations

$e_{fin}(o_i)_+, e_{dis}(R_k)_+$ de bonne réalisation de la décision, puis ne plus manipuler aucun événement. Cette attaque désynchroniserait alors S et G et bloquerait le FMS puisque G serait dans l'attente d'une décision d'allocation qu'il ne recevra jamais. Cependant, cette attaque ne crée pas de blocage de G, défini comme l'objectif des attaques de blocage considérées, et n'est pas modélisée dans N_a par un marquage bloquant puisque le transition symbolisant l'insertion de la décision préalablement supprimée est franchissable. Ces attaques de blocage par désynchronisation ne sont pas étudiées dans ces travaux et relèvent plus du domaine de la protection des protocoles de communication. ⌟

Coût d'une attaque

Le coût d'une attaque représente les moyens (humains, financiers, matériels etc.) qu'un attaquant doit investir pour la réaliser [223], [346]-[349]. Ainsi, plus le coût d'une attaque est élevée, plus elle requiert des moyens élevés de la part de l'attaquant pour être exécutée. Le calcul du coût d'une attaque s'appuie sur un coût unitaire associé à chaque événement attaqué de E_a.

Définition 3.1.7 (Coût d'un événement)**.** Soit $Ct : E_a \to \mathbb{N}$ la fonction coût qui associe à chaque événement de N_a un coût d'attaque. Pour $e \in E$, $Ct(e) = 0$ puisque l'occurrence d'un événement non attaqué ne requiert aucune intervention de l'attaquant. A l'inverse, pour $e_a \in E_+ \cup E_-$, on a $Ct(e_a) > 0$ car un événement attaqué coûte systématiquement à l'attaquant. ◇

D'après cette définition, le coût unitaire associé à un événement $e_a \in E_+ \cup E_-$ n'est pas dépendant des capacités et moyens de l'attaquant. Dans nos travaux, nous supposons en effet que le coût unitaire $Ct(e_a)$ traduit uniquement la vulnérabilité de l'événement non attaqué $e \in E_e$ face au type d'attaque considéré par e_a. De fait, l'évaluation de la vulnérabilité d'un événement peut dépendre de la vulnérabilité des composants par lesquels il transite au sein de la boucle de contrôle commande, dans lesquels il peut être compromis par un attaquant. Elle peut aussi résulter du type de l'événement (commande, observation) ainsi que du type de l'attaque (insertion, suppression) considérés dans e_a. Par exemple, considérons deux événements e_i et e_j envoyés respectivement par S aux contrôleurs $CTRL_k$ et CRL_l des ressources R_k et R_l. Si l'on suppose, indépendamment des vulnérabilités de S et du réseau industriel, que $CTRL_k$ est plus vulnérable que $CTRL_l$ en raison de vulnérabilités non corrigées du composant $CTRL_k$ (sous-partie 2.2.1) ou d'une meilleure sécurisation de $CTRL_L$ (sous-partie 2.1.3), on évalue alors que $Ct(e_{i+/-}) \leqslant Ct(e_{j+/-})$, puisqu'il est plus aisé pour l'attaquant de compromettre R_k que R_l.

Une méthode précise d'évaluation des coûts unitaires de chaque événement attaqué e_a pourrait faire l'objet d'un travail de recherche approfondi prenant en compte des facteurs multiples tels que l'architecture du FMS, les vulnérabilités de ses composants ou encore les protocoles de communication utilisés. Le développement d'une telle méthode ne fait pas partie du sujet de nos travaux. Néanmoins, afin de garantir le réalisme de notre modélisation des FMSs et des profils d'attaquant choisis, nous appliquerons trois contraintes, présentées ci-après, lors de la construction des coûts associés aux événements $e_a \in E_a$.

Premièrement, pour deux ressources R_k et R_l, on suppose que le contrôleur de l'une est systématiquement plus vulnérable que celui de l'autre. Par conséquent, si $CTRL_k$ est plus vulnérable que $CTRL_l$, on aura pour deux événements $(e_i, e_j) \in \{(e_{deb}(O_i)_+, e_{deb}(O_j)_+), (e_{deb}(O_i)_-, e_{deb}(O_j)_-), (e_{lib}(O_i)_+, e_{lib}(O_j)_+), (e_{lib}(O_i)_-, e_{lib}(O_j)_-), (e_{fin}(O_i)_+, e_{fin}(O_j)_+), (e_{fin}(O_i)_-,$

$e_{fin}(O_j)_-), (e_{dis}(R_k)_+, e_{dis}(R_l)_+), (e_{dis}(R_k)_-, e_{dis}(R_l)_-)\}$ manipulés par R_k et R_l pour les opérations O_i et O_j respectivement que $Ct(e_i) \leqslant Ct(e_j)$.

Deuxièmement, on suppose qu'une ressource R_k requise par plus d'opérations qu'une ressource R_l, i.e. $|H_{r_k}| > |H_{r_l}|$ avec H_{r_k} l'ensemble des places opérations au sein du modèle S^{3}PR requérant R_k, est moins vulnérable que R_l. En effet, en tant que ressource essentielle au fonctionnement du FMS, R_k est par conception plus sécurisée que R_l afin de prévenir un dysfonctionnement naturel ou malveillant de celle-ci.

Enfin, l'ordre de grandeur des coûts unitaires doit être globalement cohérent, à savoir qu'un coût unitaire ne peut pas être extrêmement élevé par rapport aux autres coûts non nuls. Cette contrainte se justifie par l'idée qu'un événement invulnérable n'existe pas pour un attaquant expert et qu'un coût trop élevé serait équivalent à l'invulnérabilité d'un événement e lorsque e_+ ou e_- ne sont jamais utilisés par les profils d'attaquant car trop coûteux. Une fonction coût Ct est dite acceptable par rapport à cette contrainte si elle respecte la condition $\forall e_a \in E^2_{atq} = \{e_-, e_+ | e \in \{e_{deb}(O_i), e_{lib}(R_k), e_{fin}(O_i), e_{dis}(R_k)\}, i \in [1, |P_A|], k \in [1, |P_R|]\}, Ct(e_a) \in [\overline{Ct_a} - K.\sigma_{Ct_a}; \overline{Ct_a} + K.\sigma_{Ct_a}]$ avec $K \in \mathbb{R}$ une constante,

$$\overline{Ct_a} = \frac{1}{|E^2_{atq}|} \sum_{e_a \in E^2_{atq}} Ct(e_a) \text{ et } \sigma_{Ct_a} = \sqrt{\frac{1}{|E^2_{atq}|} \sum_{e_a \in E^2_{atq}} (Ct(e_a) - \overline{Ct_a})^2}$$

la moyenne et l'écart type de tous les coûts des événements attaqués. Dans nos travaux, la valeur de K sera toujours inférieure à 3 pour respecter la cohérence des ordres de grandeur souhaitée.

Remarque 3.1.4. Notons, pour un événement $e_{des}(O_i, R_k)$, qu'un coût unitaire distinct est attribué aux événements $e_{deb}(O_i)$ et $e_{lib}(R_k)$ puisqu'ils dépendent de deux ressources différentes. Le coût de $e_{des}(O_i, R_k)$ correspond alors à la somme de ces deux coûts unitaires, i.e. $Ct(e_{des}(O_i, R_k)) = Ct(e_{deb}(O_i)) + Ct(e_{lib}(R_k))$.

⌟

A partir de la définition et de l'évaluation des coût unitaires de chaque événement $e_a \in E_a$, le coût d'une attaque $\lambda_a \in \mathcal{L}(N_a, M_a)$ est définit de la manière suivante :

Définition 3.1.8 (Coût d'une attaque)**.** La fonction Ct est étendue à une séquence d'événements par la fonction de coût d'une attaque $CT : E^\star_a \to \mathbb{N}$ où pour $\lambda_a \in E^\star_a$:

$$CT(\lambda_a) = \sum_{e_a \in \lambda_a} Ct(e_a)$$

◇

Ainsi, dans N_a, une séquence de transitions franchissable $\sigma_a \in T^\star_a$ se voit associer un coût d'attaque $CT(l(\sigma_a))$. Si $\sigma_a \in T^\star$, on a alors $CT(l(\sigma_a)) = 0$. Cette fonction coût CT ne prend pas en compte les différences de compétences et de moyens entre les profils d'attaquants mais uniquement les vulnérabilités unitaires de chaque événement attaqué. Les différences de compétences et de moyens entre profils d'attaquant sont intégrées dans les bornes supérieures de coûts C_{max} introduits dans la partie 3.1.1 lors de la définition des trois profils d'attaquant $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$.

Exemple 3.1.1.
La fonction coût unitaire Ct est représentée au sein de la figure 3.5 où le coût de chaque

événement est noté en rouge à coté de la transition qu'il labellise. Par ailleurs, le coût d'un événement $e_{des}(R_k, O_i)$ est noté $Ct(e_{des}(O_i, R_k)) = Ct(e_{deb}(O_i)) + Ct(e_{lib}(R_k))$. Dans cet exemple, deux opérations O_i et O_{i+1} requérant les ressources R_k et R_m respectivement sont modélisées.

En l'absence du modèle complet de N_a, on suppose arbitrairement $|H_{r_k}| > |H_{r_m}|$ (contrainte 2 de Ct) et on obtient que R_m est plus vulnérable que R_k. On ne s'intéresse qu'aux coûts des événements relatifs aux ressources R_k et R_m, ceux liés aux ressources requises par O_{i-1} et O_{i+2} ne sont pas considérés. La contrainte 1 de la fonction coût unitaire Ct est respectée dans cette exemple puisque pour e_i et e_j deux événements attaqués relatifs à R_k et R_m respectivement et identiques au regard du type d'attaque et de l'événement initialement attaqué, on vérifie $Ct(e_i) \geq Ct(e_j)$. La troisième contrainte est aussi validée car $\overline{Ct_a} = 9.25$, $\sigma_{Ct_a} = 4.1553$ et pour $K = 2$, on a que $\forall e_a \in (E^+ \cup E^-), Ct(e_a) \in [0.9394; 17.5606]$.

Soit la séquence de transitions franchissable $\sigma_a = t_1^+ t_i^{1-} t_2^+ t_{i+1}^{1-} t_k^{1-}$ et représentant l'attaque prenant la décision malveillante d'allouer R_k à O_i puis R_l à O_{i+1} en libérant R_k. On obtient $l(\sigma_a) = e_{dec}(O_i, R_l)_+ e_{fin}(O_i)_- e_{dec}(O_{i+1}, R_k)_+ e_{fin}(O_{i+1})_- e_{dis}(R_k)_-$ et la fonction CT nous donne $CT(\sigma_a) = 14 + 17 + 14 + 13 + 4 = 57$. ■

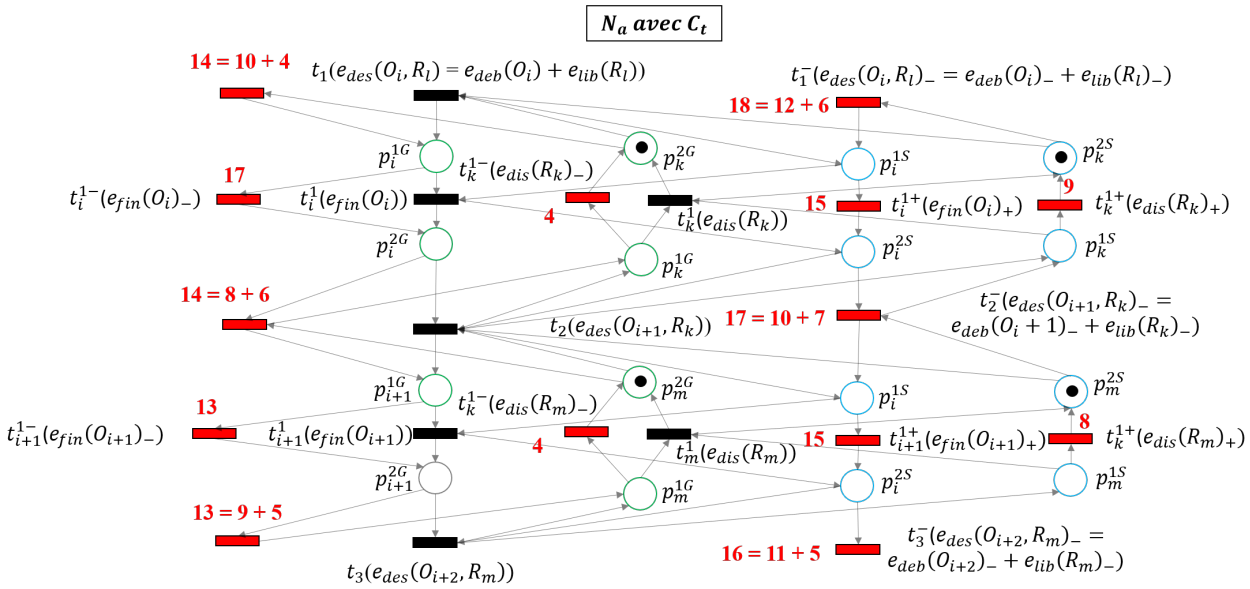


FIGURE 3.5 – Exemple de l'extension de N_a à la fonction coût C_t

Spécificités temporelles du modèle d'attaque

Dans nos travaux, la temporisation des modèles S^{3}PR et du modèle d'attaque est nécessaire au calcul de l'ordonnancement du FMS et à la construction des profils d'attaquant. Dans la littérature SED, les RdPs peuvent être temporisés à travers leurs places (RdP P-temporisé), leurs transitions (RdP T-temporisé) et leurs arcs. La temporisation des transitions agit comme un retardataire [10], [41], [120], [355] ; dès lors que la transition devient franchissable au regard de ses places d'entrée, le retardataire s'enclenche et la transition ne peut être réellement franchie que lorsque le retardataire dépasse un seuil temporel donné. En outre, le retardataire peut être enclenché selon d'autres conditions logiques (e.g. le franchissement de la première transition du RdP) et le seuil temporel d'une transition peut être évolutif (e.g. il change à chaque nouveau

franchissement de la transition). Pour sa part, la temporisation des places repose sur des délais d'indisponibilité attribués aux jetons [18], [64] ; lorsqu'un jeton est ajouté à une nouvelle place, il devient indisponible et ne peut la quitter qu'après un délai donné. La temporisation des places permet ainsi de délayer plusieurs jetons dans une même place.

Dans le domaine de l'ordonnancement, ces deux paradigmes sont utilisés pour représenter les temps des différentes opérations. Cependant, dans le modèle d'attaque N_a, la temporisation des places et celle des transitions n'ont pas les mêmes significations et implications. En effet, les places de N_a étant indépendantes des attaques et représentant uniquement l'état réel de l'allocation des ressources G et l'état observé par S, leur temporisation suppose que l'attaquant respecte systématiquement les contraintes et caractéristiques temporelles propres au FMS non attaqué. A contrario, les transitions de N_a pouvant être labellisées par des événements attaqués, leur temporisation impliquerait une possible différence entre caractéristiques temporelles du FMS sans attaque et en présence d'attaques. Dans nos travaux, nous prenons l'hypothèse que l'attaquant respecte les contraintes et caractéristiques temporelles du FMS afin de ne pas pouvoir être détecté par les méthodes de détection temporelles de la littérature [355]. De surcroit, dans le modèle N_L, les différentes places symbolisent distinctement les étapes de l'allocation des ressources (3.1.2). La temporisation des places permet donc de modéliser explicitement les temps de chaque étape, dont les temps des opérations et de la libération des ressources. **Pour ces raisons, les RdPs P-temporisés sont retenus dans nos travaux**.

Définition 3.1.9 (RdP P-temporisé)**.** Un RdP P-temporisé est un 5-tuplets $N = (P, T, F, W, D)$ où P, T, F, W sont respectivement les places, transitions, arcs et la fonction de pondération du RdP et $D : P \to \mathbb{R}^+$ la fonction qui associe à chaque place de P un temps de délai. ◇

Cette définition des RdPs P-temporisés peut être appliquée aux modèles S^3PR, N_L, N_G, N_S, N_{GS} et N_a introduits dans ce manuscrit. Néanmoins, au sein de N_L, puis par extension au sein des modèles construits à partir de N_L, seules certaines places $p \in P_L$ ont un délai non nul, i.e. $D(p) > 0$. En effet, parmi les étapes symbolisées par les places de N_L, certaines de ces étapes ne possèdent pas de contraintes de durée et sont considérées comme instantanées, à savoir qu'un jeton arrivant dans ces places peut immédiatement être utilisé pour franchir une transition de sortie dès lors qu'elle est franchissable.

Ainsi, pour une opération O_i, la place p_i^1 désignant l'exécution de l'opération O_i est temporisée avec $D(p_i^1) > 0$ la durée de l'opération O_i, et la place p_i^2 représentant l'attente de la prochaine décision d'allocation est instantanée ($D(p_i^2) = 0$) dès lors que la décision d'allocation peut être prise sitôt la fin de O_i. Pour une ressource R_k, la place p_k^1 modélisant la libération de R_k est temporisée avec $D(p_1^k) > 0$ la durée de libération de R_k, et la place p_k^2 décrivant la disponibilité de la ressource R_k est instantanée ($D(p_k^2) = 0$) dès lors que la ressource R_k est immédiatement allouée à une nouvelle opération O_j. Finalement, pour une place $p^0 \in P^0$, $D(p^0) = 0$ puisque cette place ne symbolise aucune étape mais une place d'attente pour les produits en entrée du FMS. Dans N_{GS} et N_a, la fonction D est étendue aux places de P_G et P_S avec $\forall p \in P_L, D(p) = D(p^G) = D(p^S)$ tandis que les places moniteurs de P_S ont des temps de délai nul.

Dans nos travaux, nous supposons D déterministe, à savoir qu'une durée fixe est assignée à chaque place $p \in P_e$. Ce déterminisme signifie qu'une opération et une libération de ressource ont des durées d'exécution connues et invariables. Cet invariabilité repose sur l'hypothèse d'absence

de fonctionnement dégradé des ressources en cas de défaillance (voir sous-partie 1.2.2).

Remarque 3.1.5. Dans la sous-partie 3.1.2, la séquentialité pouvant exister entre les événements de début d'une opération O_i et de libération de la ressource R_k détenue par l'opération précédente a été exposée. Le délai temporel entre ces deux événements est pris en compte dans le modèle temporisé de la manière suivante : si O_i débute avant la libération de R_k, le temps $D(p_k^1)$ associé à la libération de R_k inclut le délai entre l'événement de début d'opération $e_{deb}(O_i)$ et celui de libération $e_{lib}(R_k)$. Dans le cas contraire, si R_k est libérée avant le début de O_i, le temps $D(p_i^1)$ associé à l'opération O_i est incrémenté du délai entre $e_{lib}(R_k)$ et $e_{deb}(O_i)$. ⌟

Les RdPs P-temporisés diffèrent des RdPs traditionnels par leur dynamique de transition depuis un état M vers un second état M'. En effet, dans un RdP P-temporisé, une transition devient franchissable uniquement si les temps de délai des places précédents la transition ont été respectés par les jetons qui les composent. Cette évolution temporelle des états de N est définie dans l'annexe 6.3 à partir des travaux de [18], [64]. Les éléments et notations clés de cette méthode d'évolution temporelle d'un RdP P-temporisé sont présentés ci-dessous.

Définition 3.1.10 (Matrice des temps de délai restants)**.** Soit une matrice R définie pour représenter l'état temporel des jetons à chaque état du FMS. Si on suppose que les places activités de N sont κ-bornés, $\kappa \in \mathbb{N}$, R est alors une matrice de dimensions $|P| \times \kappa$ où un élément $R(p_i, k)$ de R, $k \in [1, \kappa]$, désigne le temps de délai restant du $k^{ième}$ jeton présent dans la place $p_i \in P$. Si il n'existe pas de $k^{ième}$ jeton dans p_i, $R(p_i, k) = 0$.

Dans les S^{3}PR ordinaires que nous considérons, on a $\kappa = 1$. Par conséquent, R est un vecteur de dimensions $|P| \times 1$. ◇

Définition 3.1.11 (Temps écoulé entre deux états d'un RdP temporisé)**.** Soit (M, R) un état d'un RdP temporisé $(N, (M_0, R_0))$ tel que $\exists \sigma \in T^\star, M_0[\sigma > M'$. On note $\Gamma_{(M_0,R_0)}$ le temps écoulé depuis le début de l'exploration de N à partir de (M_0, R_0) et jusqu'à (M, R). La méthode de calcul de $\Gamma_{(M_0,R_0)}$ est présentée en annexe. ◇

Définition 3.1.12 (Fonction de transition dans un RdP P-temporisé)**.** La méthode de franchissement temporisé de $t_j \in T$ dans N présentée en annexe est désignée par la fonction $\Omega_N : T \times \mathcal{R}(N, (M_0, R_0)) \times \mathbb{R}^+ \to \mathcal{R}(N, (M_0, R_0)) \times \mathbb{R}^+$ et notée $\Omega_N(t_j, M, R, \Gamma) = (M', R', \Gamma')$.

La fonction Ω_N peut être étendue à $T^\star$ avec pour $\sigma_2 = \sigma_1 t \in T^\star$, $\Omega_N(\sigma_2, M, R, \Gamma) = \Omega_N(t, \Omega_N(\sigma_1, M, R, \Gamma))$. ◇

Dans la suite de ce manuscrit, on désignera un état du RdP P-temporisé N par le couple $(M, R) \in \mathcal{R}(N, (M_0, R_0)) = \{(M, R) | \exists \sigma \in T^\star, (M_0, R_0)[\sigma > (M, R)\}$ où M_0 est un marquage initial acceptable du cas non temporisé et R_0 est la matrice initiale des temps de délai restant. Le temps écoulé depuis (M_0, R_0) est initialisé à $\Gamma_{(M_0,R_0)} = 0$ et dans un modèle S^{3}PR, R_0 est qualifié d'acceptable s'il s'agit du vecteur nul $R_0 = 0_{|P|\times 1}$ (aucune opération n'a été débutée).

La définition de l'ensemble des états accessibles est extensible à $\mathcal{R}(N, (M, R))$ où $(M, R) \in \mathcal{R}(N, (M_0, R_0))$. On note alors $\Gamma_{(M,R)}$ le temps écoulé depuis (M, R) avec $\Gamma_{(M,R)} = 0$ initialement.

Si N est labellisé par (l, E), on notera $\mathcal{L}(N,(M,R)) = \{\lambda \in E^\star | \exists (M',R') \in \mathcal{R}(M,R), \exists \sigma \in T\star,\ t.q.\ l(\sigma) = \lambda, (M,R)[\sigma > (M',R')\}$ le langage de N depuis l'état (M,R). Les profils d'attaquants sont ainsi étendus à $\mathcal{P}(M,R) : \mathcal{R}(N,(M_0,R_0)) \rightarrow \mathcal{L}_a(N,(M,R),\mathcal{M}_c)$ où $\mathcal{M}_c$ est l'ensemble des marquages ciblés par le profil d'attaquant indépendamment des temps restants. Pour sa part, le marquage initial du modèle N_a P-temporisé devient égal à $(M_0^i R_0^a)$.

Proposition 3.1.2 (Durée d'une attaque)**.** *Dans un modèle d'attaque P-temporisé* $N_a = (P_{GS}, T_a, F_a, E_a, l_a, D)$*, la durée d'une attaque sournoise* $\lambda_a = l(\sigma_a)$ *débutée à un état* $(M,R) \in \mathcal{R}(N_a,(M_0^a,R_0^a))$ *et ciblant un état* (M_c,R_c) *tel que* $\Omega_{N_a}(\sigma_a,(M,R),0) = (M_c,R_c,\Gamma_{(M,R)})$ *est égale à* $\Gamma_{(M,R)}$*.*

Preuve 3.1.2. Soit $\sigma_a = t_{a_1} t_{a_2} ... t_{a_n}$ avec $n = |\sigma_a|$. A chaque franchissement d'un transition $t_{a_k} \in \sigma_a$, l'étape 5. de la méthode de franchissement Ω_N présentée dans cette partie ajoute à $\Gamma_{(M,R)}$ le temps écoulé depuis le franchissement de la précédente transition $t_{a_{k-1}}$. Ainsi, si σ_a inclus correctement toutes les transitions franchies pendant l'attaque, $\Gamma_{(M,R)}$ est la somme de tous les temps écoulés entre chaque paire de transition $t_{a_k} t_{a_{k+1}}, k \in [1, n-1]$ et représente donc la durée totale de l'attaque de M jusqu'à M_c. ⌟

Exemple 3.1.2.
Au sein de la figure 3.6, l'exemple de la figure 3.5 est étendu aux RdPs P-temporisés avec les durées $D(p)$ représentées en bleu. On considère l'attaque $\sigma_a = t_1^+ t_i^{1-} t_2^+ t_{i+1}^{1-} t_k^{1-}$ labellisée par $\lambda_a = l(\sigma_a) = e_{dec}(O_i,R_l)_+ e_{fin}(O_i)_- e_{dec}(O_{i+1},R_k)_+ e_{fin}(O_{i+1})_- e_{dis}(R_k)_-$. Si l'on applique la méthode de franchissement des transitions Ω_{N_a} présentée précédemment, on observe que les places temporisées p_k^1 et p_{i+1}^1 seront marquées simultanément après la transition t_2^+ et leurs jetons auront des valeurs dans R positives. On peut ainsi estimer le temps nécessaire à l'exécution de l'attaque à $D_{\lambda_a} = D(p_i^{2G}) + D(p_{i+1}^{2G}) = 10.5 + 9.1 = 19.6$, durée inférieure au résultat naïf $D_{\lambda_a} = D(p_i^{2G}) + D(p_{i+1}^{2G}) + D(p_k^{2G}) = 22.3$ puisque l'opération O_{i+1} et la libération de R_k sont exécutées simultanément. Cependant, il ne s'agit que d'une estimation de la durée de l'attaque étant donné que le franchissement de la transition t_2^+ est dépendant de la disponibilité de la ressource R_m. En effet, si la ressource R_m est détenue par une opération débutée préalablement par la méthode d'ordonnancement de S et n'est pas immédiatement disponible pour l'attaque, alors la durée réelle de l'attaque sera supérieure à $D_{\lambda_a} = 19.7$.

■

Ainsi, cet exemple met en évidence la problématique suivante : **la construction des profils d'attaquant $\mathcal{P}_{1,2,3}$ nécessite de connaître l'ordonnancement du FMS pour estimer précisément la durée d'une attaque**. Cette connaissance de l'ordonnancement du FMS est aussi indispensable au calcul exact du coût de l'attaque. En effet, un profil d'attaquant doit connaître quelles décisions d'allocations sont prises par S sur la durée de l'attaque dans le but de les tromper et de maintenir son état sournois. Précisément, ces décisions sont manipulées par l'attaquant à travers le franchissement des transitions $t^- \in T^-, l(t) = t^-(e_{des}(O_i,R_k)_-)$, se traduisant par le franchissement de transitions pondérées par un coût unitaire non nul, à savoir les transitions t^-, t_i^{1+} et t_k^{1+}. Ces transitions doivent être ajoutées à σ_a et sont alors responsables de l'augmentation du coût réel de l'attaque, niant possiblement l'optimalité initiale

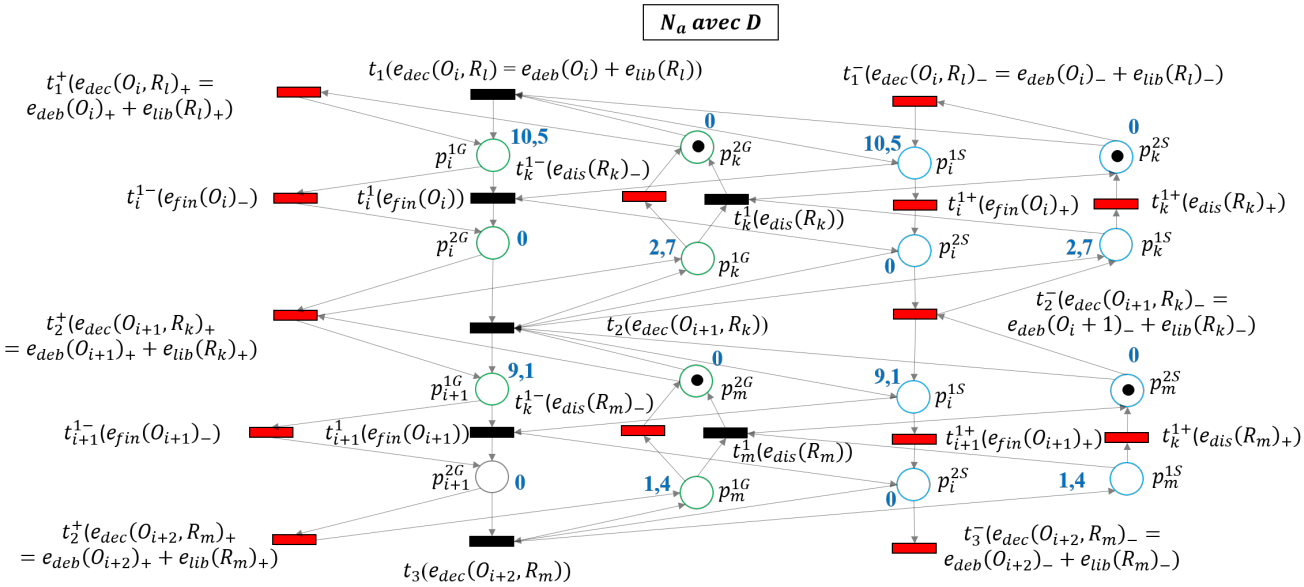


FIGURE 3.6 – Exemple de l'extension de N_a à la fonction de temporisation des places D

de la séquence pour un profil d'attaquant donné.

Dans le paragraphe précédent, il a donc été montré que l'ordonnancement du FMS est nécessaire au calcul des profils d'attaquant. Toutefois, la méthode d'ordonnancement sélectionnée dans nos travaux n'est introduite que dans la partie 3.2.2 dans laquelle nous étudierons les comportements optimaux et critiques du FMS à des fins de détection et de diagnostic. Par conséquent, la méthode de calcul des profils d'attaquant ne sera introduite que consécutivement dans la partie 3.2.3. Dans la prochaine partie (3.1.3), une autre problématique est traitée : la simplification du calcul des profils d'attaquants à travers la réduction du modèle d'attaque N_a.

3.1.3 Réduction du modèle d'attaque

L'extension des modèles S^{3}PR aux modèles N_L, N_G, N_S, N_{GS} et N_a a permis de modéliser de manière réaliste les comportements événementiels normaux et attaqués de l'allocation des ressources au sein d'un FMS. Cependant, ces extensions ont conduit à la création pour une place initiale $p \in P_A \cup P_R$ de 1 place et 1 transition au sein de N_L, puis 2 places et 2 transitions dans N_{GS}, et enfin 2 place et 4 transitions dans N_a. Dans ce dernier modèle, on observe une augmentation du nombre de places $|P_a| = 4.(|P| - |P_0|) + |P_0|$ et de transitions $|T_a| = 9.|T|$ ayant pour conséquence l'explosion du nombre d'états accessibles de $\mathcal{R}(N_a, M_0^a)$ et l'incapacité de développer une méthode performante d'exploration de ce modèle, en particulier à des fins de calcul des profils d'attaquant. De plus, outre l'augmentation du nombre de places et de transitions, la temporisation des places de N_a mène également à l'explosion du nombre d'états accessibles puisqu'un état de N_a est dorénavant associé au doublet (M, R) où pour un marquage M donné peuvent correspondre plusieurs états temporisés de N_a.

Dans cette partie, un modèle réduit de N_a incluant le coût des transitions attaquées et la durée des places opérations et libérations de ressources est proposé. Ce modèle réduit, noté N_α, doit être équivalent à N_a pour le calcul des profils d'attaquant. Ainsi, la notion d'équivalence pour le calcul des profils d'attaquant est définie dans un premier temps, puis le modèle N_α est

introduit. A partir de ces deux définitions préliminaires, une preuve de l'équivalence de N_α avec N_a est apportée.

Au sein du fonctionnement global du module de diagnostic (diagramme 3.1), le modèle N_α est le modèle d'attaque qui sera utilisé pour la construction des profils d'attaquant et des diagnostiqueurs.

Équivalence de modèles pour le calcul des profils d'attaquant

Définition 3.1.13 (Équivalence pour les profils d'attaquants)**.**
Un modèle réduit $N_\alpha = (P_\alpha, T_\alpha, F_\alpha, E_\alpha, l_\alpha, Ct_\alpha, D_\alpha)$ est dit équivalent à N_a pour le calcul des profils d'attaquants ssi pour un état $(M, R) \in \mathcal{R}(N_a, (M_0^a, R_0^a))$ issu d'une séquence d'événements non attaqués $\lambda_1 \in E^\star$ et un profil d'attaquant $\mathcal{P}(M, R) = \lambda_a \in \mathcal{L}(N_a, (M, R))$ avec $\sigma_a = l_a^{-1}(\lambda_a) \in T_a^\star$ et $(M, R)[\sigma_a > (M_c, R_c) \in \mathcal{R}(N_a, (M_0^a, R_0^a))$ l'état ciblé par le profil, on a :

1. Les recettes, les circuits et les ressources allouées à chaque opérations sont identiques entre N_a et N_α ;
2. Pour un état de N_a, noté $(M, R) \in \mathcal{R}(N_a, (M_0^a, R_0^a))$, il existe un unique état de N_α, noté $(M_\alpha, R_\alpha) \in \mathcal{R}(N_\alpha, (M_0^\alpha, R_0^\alpha))$, tel que :
   (a) Pour une opération O_i, son état réel dans G (resp. son état observé par S) est identique entre les deux modèles N_a et N_α et si O_i est en cours pour G (resp. S) le délai de temps restant jusqu'à sa fin est aussi identique entre N_a et N_α.
   En d'autres termes, soient $P_{a_i}^G$ et $P_{a_i}^{G\alpha}$ (resp. $P_{a_i}^S$ et $P_{a_i}^{S\alpha}$) les ensembles de places représentant l'état de O_i dans G (resp. dans S) dans N_a et N_α respectivement. On a que $M(P_{a_i}^G) = M_\alpha(P_{a_i}^{G\alpha})$ et $R(P_{a_i}^G) = R_\alpha(P_{a_i}^{G\alpha})$ (resp. $M(P_{a_i}^S) = M_\alpha(P_{a_i}^{S\alpha})$ et $R(P_{a_i}^S) = R_\alpha(P_{a_i}^{S\alpha})$) ;
   (b) Pour une ressource R_k, sa disponibilité réelle pour G (resp. sa disponibilité observée par S) est identique entre les deux modèles N_a et N_α et si R_k est en train d'être libérée dans G (resp. S) le temps de délai restant jusqu'à sa libération est aussi identique.
   En d'autres termes, soit $P_{r_k}^G$ et $P_{r_k}^{G\alpha}$ (resp. $P_{r_k}^S$ et $P_{r_k}^{S\alpha}$) les ensembles de places représentant l'état de R_k dans G (resp. dans S) dans N_a et N_α respectivement. On a que $M(P_{r_k}^G) = M_\alpha(P_{r_k}^{G\alpha})$ et $R(P_{r_k}^G) = R_\alpha(P_{r_k}^{G\alpha})$ (resp. $M(P_{r_k}^S) = M_\alpha(P_{r_k}^{S\alpha})$ et $R(P_{r_k}^S) = R_\alpha(P_{r_k}^{S\alpha})$) ;
   (c) Pour une recette B, le nombre de produits en entrée de la recette est conservé entre N_a et N_α.
   En d'autres termes, soit $p_0^{B,G}$ (resp. $p_0^{B,S}$) sa place d'entrée dans P_G (resp. P_S) et $p_0^{B,G\alpha}$ (resp. $p_0^{B,S\alpha}$) celle dans $P_G^\alpha \subset P_\alpha$ (resp. $P_S^\alpha \subset P_\alpha$). On a que $M(p_0^{B,G}) = M(p_0^{B,G\alpha})$ (resp. $M(p_0^{B,S}) = M(p_0^{B,S\alpha})$) ;
   (d) Cette équivalence d'états présentée dans les points $(a), (b), (c)$ reste vraie entre (M_0^a, R_0^a) et (M_0^α, R_0^α) pour que ce second marquage initial soit acceptable pour N_α.
3. A partir de l'état (M_α, R_α), la séquence d'attaque $\mathcal{P}(M, R) = \lambda_a$ considérée doit aussi appartenir au langage $\mathcal{L}(N_\alpha, (M_\alpha, R_\alpha))$ et conduire à un état équivalent à (M_c, R_c) en terme d'opérations en cours et de disponibilité des ressources.
   Autrement dit, $\exists! \sigma_\alpha \in (T_\alpha)^\star$ tel que $l_\alpha(\sigma_\alpha) = \lambda_\alpha = \lambda_a$ où $\lambda_\alpha \in E_\alpha^\star$ et $(M_\alpha, R_\alpha)[\sigma_\alpha > (M_c^\alpha, R_c^\alpha) \in \mathcal{R}(N^\alpha, (M_0^\alpha, R_0^\alpha))$ avec (M_c, R_c) et (M_c^α, R_c^α) équivalents selon le point 2. ;

4. Les coûts de l'attaque $\lambda_a = \lambda_\alpha$ calculés à partir de N_a et de N_α respectivement doivent être égaux, i.e. $CT(\lambda_a) = CT(\lambda_\alpha)$;
5. Les durées de l'attaque λ_a calculées à partir de N_a et de N_α respectivement doivent être égales, i.e. $\Gamma_{(M,R)} = \Gamma_{(M_\alpha,R_\alpha)}$ avec $\Omega_a(\sigma_a, M, R, 0) = (M_c, R_c, \Gamma_{(M,R)})$ et $\Omega_\alpha(\sigma_\alpha, M_\alpha, R_\alpha, 0) = (M_c^\alpha, R_c^\alpha, \Gamma_{(M_\alpha,R_\alpha)})$.

◇

Les différents points de cette définition permettent de calculer de manière égale les différentes caractéristiques communes ou spécifiques d'un profil d'attaquant $\mathcal{P}$ présentées dans la partie 3.1.1.

Premièrement, les événements de la séquence d'attaque sont identiques entre N_a et N_α d'après le point 3. Cette égalité permet de garantir la sournoiserie de l'attaque puisque $f_a^2(\lambda_a)$ reste inchangée et que l'attaque occasionne les mêmes conséquences sur l'allocation des ressources (blocage, pré-blocage, blocage partiel) puisque $f_a^1(\lambda_a)$ reste aussi inchangée. Cette égalité des séquences d'attaque préserve également le caractère expert de l'attaque et la taille de l'attaque en nombre d'opérations.

Deuxièmement, lorsque l'attaque traverse un état spécifique M_s de N_a (pré-blocage, blocage-partiel) après une sous-séquence $\lambda_{a_1} = l(\sigma_{a_1})$ de λ, il s'agit aussi du même type d'état dans N_α d'après le point 3. En effet, en remplaçant M_c par cet état spécifique M_s dans 3, on sait qu'il existe un état M_s^α après l'exécution de λ_{a_1} depuis (M^α, R^α) équivalent à M_s au regard d'une allocation identique des ressources aux mêmes opérations. Cette équivalence entre M_s et M_s^α jointe au point 1 de la définition permet de démontrer que si M_s est un état de pré-blocage ou de blocage partiel, M_s^α l'est aussi. Cette propriété valide la caractéristique de minimalité de l'attaque car l'ensemble des états ciblés $\mathcal{M}_c$ est correctement identifié dans N_α. De surcroit, cette propriété garantit pour le profil 1 que le minimum d'états de pré-blocage sera parcouru par l'attaque et que l'état M_c ciblé à un taux d'utilisation des ressources maximum.

Troisièmement, l'égalité du coût de l'attaque entre N_a et N_α est garantie par le point 4.

Enfin, l'uniformité de la durée de l'attaque est prouvée par le point 5. En reprenant les caractéristiques des profils d'attaquant présentées dans la partie 3.1.1, on observe que toutes ont été prises en compte dans cette définition de l'équivalence de deux modèles pour le calcul des profils d'attaquant.

Modèle d'attaque N_α

Le modèle d'attaque réduit proposé dans cette partie repose sur l'agrégation au sein des transitions $t \in T$ labellisées par les événements $e_{dec}(O_i, R_k)$ des transitions t_i^1 et t_k^1 symbolisant respectivement la fin de l'opération $O_i(e_{fin}(O_i))$ et la disponibilité de la ressource $R_k(e_{dis}(R_k))$. Cette agrégation est de la même manière réalisée pour les transitions t^+ et t^- fusionnées respectivement avec les paires de transitions (t_i^{1-}, t_k^{1-}) et (t_i^{1+}, t_k^{1+}). Cette agrégation de transitions entraine alors la fusion des places p_i^1 et p_i^2 et des places p_k^1 et p_k^2 dans les ensembles P_G et P_S.

Définition 3.1.14 (Modèle d'attaque réduit N_α)**.** Soit $N_\alpha = (P_\alpha, T_\alpha, F_\alpha, E_\alpha, l_\alpha, Ct_\alpha, D_\alpha)$ le modèle d'attaque réduit définit par :

- $P_\alpha = P_0^a \cup P_A^{G\alpha} \cup P_A^{S\alpha} \cup P_R^{G\alpha} \cup P_R^{S\alpha}$ avec :
  - $P_A^{G\alpha} = \{p_{a_i}^G | \{p_{a_i}^{1G}, p_{a_i}^{2G}\} = P_a \cap P_{a_i}^G\}$;

  - $P_A^{S\alpha} = \{p_{a_i}^S | \{p_{a_i}^{1S}, p_{a_i}^{2S}\} = P_a \cap P_{a_i}^S\}$;
  - $P_R^{G\alpha} = \{p_{r_k}^G | \{p_{r_k}^{1G}, p_{r_k}^{2G}\} = P_a \cap P_{r_k}^G\}$;
  - $P_R^{S\alpha} = \{p_{r_k}^S | \{p_{r_k}^{1S}, p_{r_k}^{2S}\} = P_a \cap P_{r_k}^S\}$;
  - $P_A^{\alpha} = P_A^{G\alpha} \cup P_A^{S\alpha}$, $P_R^{\alpha} = P_R^{G\alpha} \cup P_R^{S\alpha}$, $P^{G\alpha} = P_A^{G\alpha} \cup P_R^{G\alpha} \cup P_0^G$ et $P^{S\alpha} = P_A^{S\alpha} \cup P_R^{S\alpha} \cup P_0^S$

  Ainsi, pour chaque opération et pour chaque ressource, une unique place est définie pour G et pour S ;
- $T_\alpha = T \cup T_-^\alpha \cup T_+^\alpha$ avec :
  - $T_-^\alpha = \{t_- | t \in T\}$;
  - $T_+^\alpha = \{t_+ | t \in T\}$;
- On définit par ailleurs la fonction $\tau_\alpha : T_\alpha \to T$ par $\tau_\alpha(t) = \tau_\alpha(t_+) = \tau_\alpha(t_-) = t$;
- $F_\alpha = F_G^\alpha \cup F_S^\alpha \cup F_-^\alpha \cup F_+^\alpha$ avec :
  - $F_G^\alpha = \{(p_{a_i}^G, t), (p_{r_k}^G, t) | (p_{a_i}^{2G}, t), (p_{r_k}^{2G}, t) \in F_a, t \in T, p_{a_i}^G \in P_A^{G\alpha}, p_{r_k}^G \in P_R^{G\alpha}\}$
    $\cup \{(t, p_{a_i}^G), (t, p_{r_k}^G) | (t, p_{a_i}^{1G}), (t, p_{r_k}^{1G}) \in F_a, t \in T, p_{a_i}^G \in P_A^{G\alpha}, p_{r_k}^G \in P_R^{G\alpha}\}$;
  - F_S^α est identique à F_G^α en remplaçant les G par des S ;
  - $F_-^\alpha = \{(p_{a_i}^S, t_-), (p_{r_k}^S, t_-) | (p_{a_i}^S, t), (p_{r_k}^S, t) \in F_\alpha\} \cup \{(t_-, p_{a_i}^S), (t_-, p_{r_k}^S) | (t, p_{a_i}^S), (t, p_{r_k}^S) \in F_\alpha\}$;
  - $F_+^\alpha = \{(p_{a_i}^G, t_+), (p_{r_k}^G, t_+) | (p_{a_i}^G, t), (p_{r_k}^G, t) \in F_\alpha\} \cup \{(t_+, p_{a_i}^G), (t_+, p_{r_k}^G) | (t, p_{a_i}^G), (t, p_{r_k}^G) \in F_\alpha\}$;
- $E_\alpha = E^\alpha \cup E_-^\alpha \cup E_+^\alpha$ avec :
  - $E^\alpha = \{e^\alpha(O_i, R_k) = e_{deb}(O_i) e_{lib}(R_k) e_{fin}(O_i) e_{dis}(R_k) | e_{dec}(O_i, R_k) \in E_e\}$;
  - $E_-^\alpha = \{e^\alpha(O_i, R_k)_- = e_{deb}(O_i)_- e_{lib}(R_k)_- e_{fin}(O_i)_+ e_{dis}(R_k)_+ | e^\alpha(O_i, R_k) \in E^\alpha\}$;
  - et $E_+^\alpha = \{e^\alpha(O_i, R_k)_+ = e_{deb}(O_i)_+ e_{lib}(R_k)_+ e_{fin}(O_i)_- e_{dis}(R_k)_- | e^\alpha(O_i, R_k) \in E^\alpha\}$.
- La fonction de labellisation $l_\alpha : T_\alpha \to E_\alpha$ est définie par :
  - $l_\alpha(t) = e^\alpha(O_i, R_k)$ pour $t \in T$;
  - $l_\alpha(t^+) = e^\alpha(O_i, R_k)_+$ pour $t^+ \in T_+^\alpha$;
  - $l_\alpha(t^-) = e^\alpha(O_i, R_k)_-$ pour $t^- \in T_-^\alpha$;
- La fonction coût unitaire $Ct_\alpha : E_\alpha \to \mathbb{N}$ est définie par :
  - $Ct_\alpha(e^\alpha(O_i, R_k)) = 0$;
  - $Ct_\alpha(e^\alpha(O_i, R_k)_-) = Ct_a(e_{deb}(O_i)_-) + Ct_a(e_{lib}(R_k)_-) + Ct_a(e_{fin}(O_i)_+) + Ct_a(e_{dis}(R_k)_+)$;
  - $Ct_\alpha(e^\alpha(O_i, R_k)_+) = Ct_a(e_{deb}(O_i)_+) + Ct_a(e_{lib}(R_k)_+) + Ct_a(e_{fin}(O_i)_-) + Ct_a(e_{dis}(R_k)_-)$;
- La fonction temps de délai $D_\alpha : P_\alpha \to \mathbb{R}$ est définie par :
  - $D_\alpha(p_0^a) = 0$;
  - $\forall p_{a_i}^G, p_{a_i}^S \in P_A^\alpha, D_\alpha(p_{a_i}^G) = D_\alpha(p_{a_i}^S) = D_a(p_{a_i}^{1G}) = D_a(p_{a_i}^{1S})$ avec $p_{a_i}^{1G}, p_{a_i}^{1S} \in P_a$;
  - $\forall p_{r_k}^G, p_{r_k}^S \in P_A^\alpha, D_\alpha(p_{r_k}^G) = D_\alpha(p_{r_k}^S) = D_a(p_{r_k}^{1G}) = D_a(p_{r_k}^{1S})$ avec $p_{r_k}^{1G}, p_{r_k}^{1S} \in P_a$;
- Un marquage initial (M_0^α, R_0^α) de N_α est qualifié d'acceptable ssi pour un marquage initial (M_0^a, R_0^a) acceptable dans N_a :
  - $\forall p_0^a \in P_0^a, M_0^\alpha(p_0^a) = M_0^a(p_0^a)$;
  - $\forall p_{a_i} \in P_A^\alpha, M_0^\alpha(p_{a_i}) = 0$, aucune opération n'est en cours ;

– $\forall p_{r_k} \in P_R^\alpha, M_0^\alpha(p_{r_k}) = 1$, toutes les ressources sont disponibles ;
– $R_0^\alpha = 0_{|P_\alpha| \times 1}$, aucune opération n'a débuté.

◇

Remarque 3.1.6. La fonction $f_a = (f_a^1, f_a^2)$ de projection des événements attaqués sur G et sur S est étendue aux événements de E_a et de E_α en conservant la distinction entre événements contrôlables et événements observables ainsi que celle entre insertion et suppression de ces événements. ⌟

Définition 3.1.15 (Fonction de conversion f_α)**.** A partir du modèle N_α, on définit la fonction de conversion $f_\alpha : \mathbb{N}_+^{|P_a|} \times \mathbb{R}_+^{|P_a|} \to \mathbb{N}_+^{|P_\alpha|} \times \mathbb{R}_+^{|P_\alpha|}$ qui à partir d'un marquage (M_a, R_a) de N_a construit le marquage correspondant (M_α, R_α) dans N_α. Cette fonction peut être définie en considérant les places de P_a séparément.

- Pour $p \in P_a^0 \subset (P_a \cap P_\alpha)$, $f_\alpha(M_a(p), R_a(p)) = (M_\alpha(p), R_\alpha(p))$ avec $M_\alpha(p) = M_a(p)$ et $R_\alpha(p) = R_a(p)$ puisque les places d'entrée sont conservées entre N_a et N_α ;
- Pour une opération O_i et $\{p_{a_i}^1, p_{a_i}^2\} \in (P_{a_i}^G \cup P_{a_i}^S) \cap P_a)$, on a $f_\alpha(M_a(\{p_{a_i}^1, p_{a_i}^2\}), R_a(\{p_{a_i}^1, p_{a_i}^2\})) = (M_\alpha(p_{a_i}), R_\alpha(p_{a_i}))$ avec $p_{a_i} \in P_A^\alpha$ et $M_\alpha(p_{a_i}) = M_a(p_{a_i}^1) + M_a(p_{a_i}^2)$, $R_\alpha(p_{a_i}) = R_a(p_{a_i}^1)$.
  Par conséquent, une place activité est marquée dans N_α si l'une des deux places dans N_a dont elle est issue est marquée. Dans N_α, on conserve la distinction entre $p_{a_i}^1$ et $p_{a_i}^2$ à travers la valeur de $R_\alpha(p_{a_i})$ qui est non nulle uniquement si l'opération est en cours dans $p_{a_i}^1$, i.e. si $p_{a_i}^1$ est marquée, car dès que $R_\alpha(p_{a_i}^1)$ est nul, la transition $t_{a_i} \in T_a$ de fin d'opération est franchie ;
- Pour une ressource R_k, la définition de f_α est identique à une opération O_i en remplaçant tous les a_i par des r_k. Similairement, l'information sur l'état "en cours de libération" de R_k est conservée dans N_α à travers une valeur non nulle de $R_\alpha(p_{r_k})$ avec $p_{r_k} \in P_R^\alpha$;

La fonction inverse f_α^{-1} peut être définie à partir de la valeur de $R_\alpha(p_{ik})$ avec $p_{ik} \in P_A^\alpha \cup P_R^\alpha$ un place activité ou ressource de N_α. Si cette valeur est non nulle, alors la place correspondante dans N_a à savoir p_{ik}^1 est marqué et $R_a(p_{ik}^1) = R_\alpha(p_{ik})$. Sinon, p_{ik}^2 est marquée et $R_a(p_{ik}^1) = R_a(p_{ik}^2) = 0$.

◇

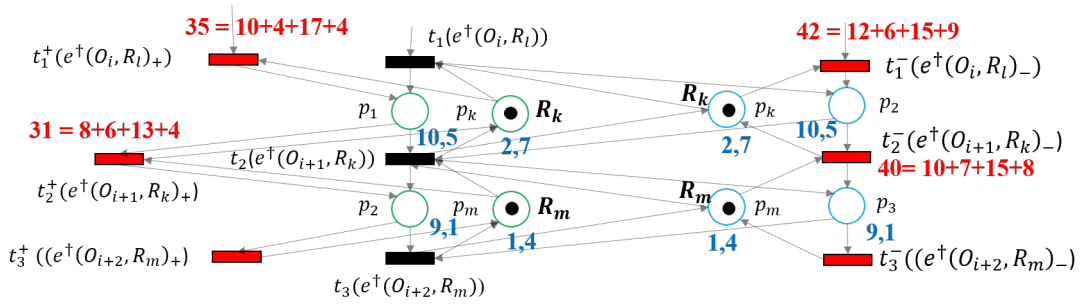
La fonction f_α est bijective puisque la correspondance entre p_{ik}^1, p_{ik}^2 et p_{ik} est unique et $P_a^0 \subset (P_a \cap P_\alpha)$. Le modèle N_α est illustré dans la figure 3.7. Dans cette figure, les opérations O_i et O_{i+1} modélisées initialement dans la figure 3.4 sont représentées.

Équivalence entre N_a et N_α

Proposition 3.1.3. *Le modèle N_α proposé ci-dessus est équivalent au modèle N_a selon la définition 3.1.13.*

Preuve 3.1.3. Le point 1 de la définition est vraie pour N_α puisque les transitions sont héritées du modèle initial N et les arcs entre une opération ou un ressource avec l'une de ces transitions sont conservés depuis N_a à travers la définition de F_α.

Le point 2 est vrai dans N_α si l'on applique la fonction bijective f_α à (M, R) pour obtenir l'unique état correspondant (M_α, R_α).



$$l(t_2^+) = e^\dagger(O_{i+1},R_k)_+ = e_{deb}(O_2)_+ e_{lib}(R_k)_+ e_{fin}(2)_- e_{dis}(R_k)_-$$
$$l(t_2^-) = e^\dagger(O_{i+1},R_k)_- = e_{deb}(O_2)_- e_{lib}(R_k)_- e_{fin}(2)_+ e_{dis}(R_k)_+$$

FIGURE 3.7 – Exemple du modèle réduit N_α

Le point 3 de la définition peut être prouvé en considérant une unique décision, en traitant séparément les cas $e_{dec}(O_i,R_k)$,$e_{dec}(O_i,R_k)_-$ et $e_{dec}(O_i,R_k)_+$, puis en généralisant à n'importe quelle séquence d'attaque. Soit (M,R) et (M_α,R_α) les états équivalents dans N_a et N_α d'après le point 2 et la fonction f_α. Dans N_a, t_1,t_1^- et t_1^+ sont possiblement franchissables depuis (M,R) avec $t_1 \in T$.

Si $t_1^-(e_{dec}(O_i,R_k)_-)$ est franchie par l'attaquant, celui-ci est forcé de franchir $t_{a_i}^+(e_{fin}(O_i)_+)$ et $t_{r_k}^+(e_{dis}(R_k)_+)$ afin de rester sournois puisque t_{a_i} et t_{r_k} ne sont pas franchissables. On obtient alors $\sigma_a = t_1^- t_{a_i}^+ t_{r_k}^+$ avec $l_a(\sigma_a) = e_{dec}(O_i,R_k)_- e_{fin}(O_i)_+ e_{dis}(R_k)_+ = \lambda_a$. Dans N_α, cela correspond à franchir $t_1^-(e^\alpha(O_i,R_k)_-)$ avec $l_\alpha(t_1^-) = \lambda_\alpha = \lambda_a$.

De la même manière si $t_1^+(e_{dec}(O_i,R_k)_+)$ est franchie, $\sigma_a = t_1^+ t_i^{1-} t_k^{1-}$ a lieu dans N_a avec $l_a(\sigma_a) = e_{dec}(O_i,R_k)_+ e_{fin}(O_i)_- e_{dis}(R_k)_- = \lambda_a$ égale à $l_\alpha(t_1^+(e_\alpha(O_i,R_k)_+)$ dans N_α.

Enfin, si t_1 est franchie, t_i et t_k seront forcément franchies car les événements d'observation ont systématiquement lieu dans G et sont attendus par S après t_a. Ces deux transitions sont franchies au détriment des transitions attaquées puisqu'elle sont franchissables et n'engendrent pas un coût supplémentaire pour le profil d'attaquant contrairement au franchissement équivalent de $t_i^{1+}, t_i^{1-}, t_k^{1+}, t_k^{1-}$ en terme d'événements et d'états atteints dans G et dans S. Dans N_α, cela correspond au franchissement de $t^1(e^\alpha(O_i,r_k))$ avec $e^\alpha(O_i,r_k) = e_{deb}(O_i)e_{lib}(R_k)e_{fin}(O_i)e_{dis}(R_k) = l_a(t_1 t_i t_k)$.

On a donc prouvé ici l'existence lorsqu'une transition $t|\pi_a(t) \in T$ est franchie dans N_a d'une transition unique dans T_α telle que les séquences d'attaques générées par le franchissement de ces deux transitions λ_a et λ_α sont égales. Notons que les autre transitions de N_a (dans T_L^a et T_L^a) sont inclues dans λ_a puisque leur franchissement a été prouvé comme induit par le franchissement des transitions $t|\pi_a(t) \in T$. Après l'occurrence de λ_a depuis (M,R) et (M_α,R_α), les deux marquages atteints M' et M'_α sont équivalents selon f_α d'après le point 1 et la définition de F_α, autrement dit les mêmes opérations requérant les mêmes ressources ont été débutées puis finies, les mêmes ressources ont été libérées et les mêmes décisions d'allocation sont éligibles ; i.e. les transitions franchissables débutent les mêmes opérations, requièrent et libèrent les mêmes ressources.

Cependant, les valeurs de R' etR'_α ne sont pas les mêmes puisque $R'_\alpha(p_{a_i})$, représentant le temps restant de l'opération O_i, n'est décrémenté que lorsque la prochaine transition de décision

d'allocation est franchie. Toutefois, cette décrémentation a finalement toujours lieu puisque une transition franchissable succédant l'opération sera tôt ou tard franchie par le profil d'attaquant pour atteindre son objectif hormis pour l'opération succédant la dernière transition menant à M_c. **Face à cette problématique, on suppose que l'attaque a atteint son objectif lorsque la dernière décision d'allocation menant au blocage est prise et les temps de l'opération et de la libération de ressources débutées par cette décision ne sont pas pris en compte dans le calcul de** $\Gamma_{(M,R)} = \Gamma_{(M_\alpha,R_\alpha)}$. Dans ce cas là, la différence de temps entre R_c et R_c^α n'est pas préjudiciable au calcul de la durée du profil. Pour la même raison, l'équivalence entre λ_a et λ_α est vérifiée si les événements d'observation $e_{fin}(O_i)$ et $e_{lib}(R_k)$ de la dernière décision d'allocation prise par le profil d'attaquant sont retirés de la séquence d'attaque λ_α.

Le point 4 de la définition est vérifiée car l'égalité $\lambda_a = \lambda_\alpha$ signifie que les mêmes événements ont lieu et la somme de leurs coûts unitaires permet d'obtenir le même coût final $CT_a(\lambda_a) = CT_a^(\lambda_\alpha)$.

Enfin, le point 5 de la définition est aussi vrai dans N_α puisque les temps des opérations et de libération des ressources sont conservés dans N_α et les temps de délai restant sont garantis entre (M, R) et (M_α, R_α) selon le point 2. Une unique différence entre N_a et N_α apparait lorsque le marquage M_c est atteint. De fait, dans N_a, le blocage est atteint lorsque la dernière opération O_i est finie et $e_{fin}(O_i)$ a été émis tandis que dans N_α, l'opération n'est pas réellement finie, i.e. $R_c^\alpha(p_{a_i}) > 0$, puisqu'aucune transition n'est franchie après M_c^α, n'entraînant pas la décrémentation du temps de délai de l'opération modélisée par p_{a_i}. Néanmoins, l'hypothèse prise pour le point 3. permet de répondre à cette problématique (on suppose que l'attaque a atteint son objectif lorsque la dernière décision d'allocation menant au blocage est prise et les temps de l'opération et de la libération de ressources débutées par cette décision ne sont pas pris en compte dans le calcul de $\Gamma_{(M,R)} = \Gamma_{(M_\alpha,R_\alpha)}$).

Les cinq points de la définition 3.1.13 ont été prouvés, le modèle N_α est donc équivalent à N_a pour le calcul des profils d'attaquant.

⌟

Dans cette première partie du chapitre 3, les attaques de blocage ont été définies et modélisées dans le but d'introduire précisément la problématique malveillante à laquelle notre méthode de diagnostic souhaite répondre. Différents profils réalistes d'attaquants on été présentés et un modèle du FMS attaqué a été proposé. A l'aide de ce modèle, toutes les attaques de blocage sournoises existantes et définies à travers le langage d'attaque $\mathcal{L}_a(N_L, M_0)|_{sr}$ peuvent être construites. Une extension de ce modèle incluant le coût d'une attaque et la temporisation des opérations du FMS a été développée pour supporter le calcul des différents profils d'attaquant à partir de n'importe quel état du FMS. Enfin, un modèle réduit équivalent pour la construction des profils d'attaquant a été choisi pour réduire le temps de calcul et de simulation de ces profils.

Dans la prochaine partie, les différents comportements du FMS et de l'attaquant nécessaires au diagnostic des attaques de blocage sont définis et estimés par voie algorithmique. Ainsi, les comportements critiques de blocage du FMS seront étudiés dans la sous-partie 3.2.1 tandis que des méthodes de calcul des comportements optimaux du FMS et des profils d'attaquant seront développées dans les sous-parties 3.2.2 et 3.2.3 respectivement.

3.2 Estimation des comportements critiques, optimaux et des profils d'attaquant

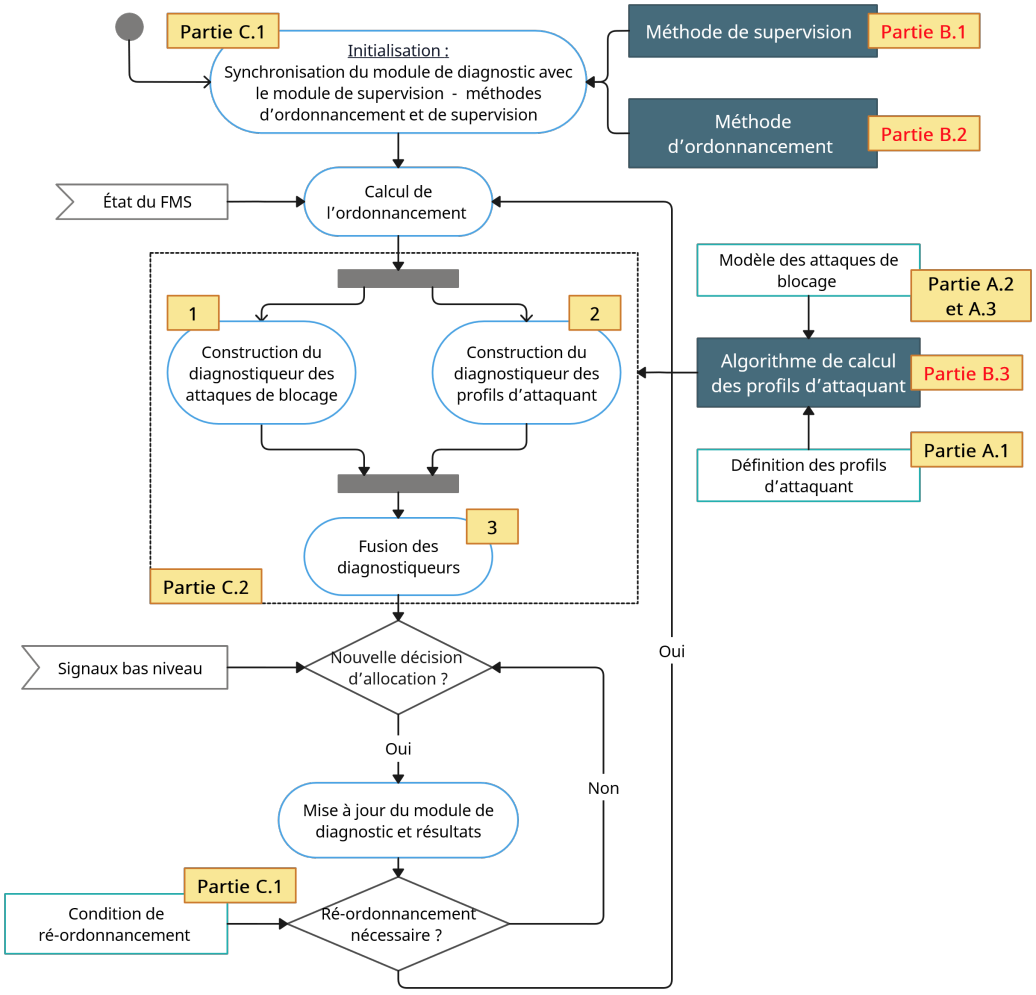


FIGURE 3.8 – Diagramme d'activité UML global du module de diagnostic

La méthode de diagnostic des attaques de blocage proposée dans ce manuscrit et dont le fonctionnement est illustré dans la figure 2.5 est fondée d'une part sur l'identification en ligne des comportements critiques et optimaux propres à un FMS et d'autre part sur la détection des comportements des différents profils d'attaquant introduits dans la partie précédente (partie 3.1). Au sein d'un FMS, les comportements critiques sont prévenus et évités grâce aux méthodes de gestion des états de blocage (sous-partie 1.3.2) tandis que le comportement optimal est, lui, piloté par une méthode d'ordonnancement (sous-partie 1.3.1). Ainsi, dans les sous-parties 3.2.1 et 3.2.2, une méthode de gestion des blocages et une méthode d'ordonnancement seront choisies au regard de la problématique suivante ; soit l'hypothèse prise en introduction de ce chapitre stipulant que la solution de diagnostic n'a accès qu'aux données fiables du FMS, à savoir les données échangées entre les contrôleurs locaux et les ressources. Elle n'a par conséquent pas accès ni à la politique de contrôle, ni à l'ordonnancement exact calculé depuis le module de supervision. **Les méthodes de gestion des blocages et d'ordonnancement du FMS sélectionnées et développées dans ces deux sous-parties devront donc permettre**

à notre solution de diagnostic de les estimer précisément en ligne à partir des données fiables bas-niveau. En conclusion de cette partie, une méthode de calcul des profils d'attaquant sera développée dans la sous partie 3.2.3 à partir du modèle d'attaque réduit N_α en exploitant les comportements critiques et optimaux préalablement obtenus.

Les différents travaux présentés dans cette partie sont illustrés par la re-mobilisation (blocs grisés de la figure 3.8 ci-dessus) du diagramme d'activité UML du fonctionnement global du module de diagnostic proposé en introduction de ce chapitre (figure 3.1).

3.2.1 Choix d'une méthode de supervision du FMS

Au sein du fonctionnement global du module de diagnostic (diagramme 3.8), la méthode de supervision est requise lors de l'initialisation du module afin d'être déployée de manière synchronisée dans les modules de supervision et de diagnostic. Puis, cette méthode est nécessaire en ligne pour le calcul de l'ordonnancement afin de faire éviter à ce dernier les états interdits de la DZ. Dans cette partie, l'objectif est de choisir une méthode de prévention de la littérature adaptée à l'implémentation de notre module de diagnostic.

Les différentes méthodes de gestion des états de blocage ont été introduites dans le chapitre 1 (sous-partie 1.3.2) et se subdivisent entre les méthodes de prévention, d'évitement et de détection et reprise. Parmi celles-ci, seules les méthodes de prévention et d'évitement permettent de répondre à notre problématique de diagnostic d'attaque, les méthode de détection ne permettant pas la distinction entre l'origine naturelle ou malveillante d'un état interdit de la DZ atteint par le FMS et détecté par ces méthodes (voir état de l'art 2.3.2). **De fait, la méthode de diagnostic proposée dans ces travaux doit être capable de différencier sans équivoque un état interdit, révélateur d'une attaque car non atteignable autrement, d'un état autorisé par le module de supervision.**

Dans la littérature, les méthodes de prévention et d'évitement permettent d'interdire un ensemble d'états incluant les états critiques de la DZ. Parmi celles-ci, on souhaite implémenter dans le module de supervision une méthode ayant les trois propriétés suivantes.

Premièrement, la méthode doit être totalement permissive, à savoir qu'elle n'interdit aucun état non critique hors de la DZ. Cette propriété est choisie pour garantir une grande flexibilité à la méthode d'ordonnancement. En particulier, les états non critiques interdits par les méthodes non totalement permissives sont les états adjacents à la DZ. Or, ces états peuvent être des états pertinents dans l'optimisation de l'ordonnancement. En effet, au sein de ceux-ci le taux d'utilisation des ressources est élevé puisque par définition, un état de blocage M implique qu'il existe un siphon $S \in \Pi$ tel que $M(S) = 0$ signifiant que toutes les ressources $p_r \in S \cap P_R$ sont détenues par des opérations. Ainsi, un état M' adjacent à la DZ et interdit par une méthode non totalement permissive est proche d'un état de blocage. Par conséquent on a $M'(S)$ proche de 0 signifiant une utilisation partielle mais élevée des ressources $p_r \in S \cap P_R$. Dans de nombreuses méthodes d'ordonnancement, une haute utilisation des ressources permet par exemple la réduction de critères d'optimisation temporels tels que la cadence, le temps d'exécution des produits ou le respect des dates d'échéances.

Deuxièmement, la méthode doit être complète en interdisant tous les états de la DZ de manière certaine. Si ce n'est pas le cas, la problématique d'incapacité de diagnostic de l'origine

d'un état critique associé aux méthodes de détection est aussi imputée aux états de la DZ que la méthode n'interdit pas. Jointe à la première propriété, cela signifie que l'ensemble des états interdits par la méthode choisie doit être exactement égale à $\mathcal{M}_\rho = \mathcal{M}_{DZ}$.

Troisièmement, afin de pouvoir conserver la validité de la priorité 3.1.1, la méthode choisie devra directement intégrer sa politique de contrôle dans le modèle du FMS et ses extensions, à savoir N_S, N_{GS}, N_a et N_α ainsi que dans les modèles futurs utilisés pour le calcul de l'ordonnancement. De fait, le langage des attaques sournoises sera conservé dans le modèle d'attaque puisque la sournoiserie par rapport à la politique de contrôle sera directement intégrée au modèle initial N_S.
Parmi les méthodes de prévention et d'évitement, seules les méthodes de prévention fondées sur le graphe des marquages accessibles possèdent ces trois propriétés. En effet, les méthodes d'évitement ne respectent pas systématiquement les propriétés 1 et 2 [1] et ne satisfont jamais la propriété 3. Quant aux méthodes de prévention fondées sur des approches structurelles (siphons, circuits), elles ne sont jamais totalement permissives (propriété 1) [1].

Dans nos travaux, la méthode de prévention d'Uzam [50], respectant ces trois propriétés, a été choisie pour être implémentée au sein du module de supervision du FMS N afin de lui interdire l'accès aux états de sa DZ. Dans cette méthode, la construction des places moniteurs ou places de contrôle $p_c \in P_c$ est réalisée de manière itérative où à chaque itération un nouveau moniteur est ajouté au modèle S^{3}PR supervisé N_i pour former un nouveau modèle supervisé N_{i+1}. Au sein d'une itération i , le graphe des marquages accessibles $\mathcal{RG}(N_i, M_{0_i}) = \mathcal{RG}_i$ du modèle S^{3}PR N_i munis des places moniteurs construites lors des précédentes itérations est exploré. Si cette exploration conclut sur la vivacité du RdP, l'algorithme se termine et garantit que tous les états de la DZ ont été interdits (propriété 2). Sinon, le "First Bad Marking" (FBM) est identifié, à savoir le premier marquage M_{FBM} de la DZ découvert lors de l'exploration de $\mathcal{RG}_i$. Un P-invariant I est alors défini à partir des places activités marquées de M_{FBM}(son support $\| I \|$ est composé de ces places) et une place de contrôle p_{c_i} de I ainsi que son marquage initial $M_{0_i}(p_{c_i})$ sont obtenus selon les travaux de [362]. Enfin, le modèle $(N_{i+1}, M_{0_{i+1}})$ est obtenu en ajoutant $(p_{c_i}, M_{0_i}(p_{c_i}))$ à (N_i, M_{0_i}) et l'itération $i+1$ est débuté jusqu'à terminaison de l'algorithme. Les auteurs ont prouvé que cette algorithme se termine en un nombre fini d'itérations et permet d'obtenir un système supervisé totalement permissif (**propriété 1**) grâce aux places de contrôles $p_c \in P_c$ ajoutées au modèle initial (**propriété 3**).

Les travaux d'Uzam sont appliqués à notre méthode à travers l'exemple précédent de la figure 1.6. Cet exemple est étendu et muni de places de contrôle. Ce FMS supervisé noté est représenté par la figure 3.9 ci-contre où les places de contrôles sont exposées séparément du modèle S^{3}PR et coloriées en bleu pour des raisons de clarté. La méthode d'Uzam [50] appliqué à ce modèle S^{3}PR conduit à la création de 13 places moniteurs $p_1^c, p_2^c, ..., p_{13}^c \in P_c$ dont les arcs les connectant aux transitions du modèle sont indiqués à l'extrémité des arcs en entrée et en sortie de chacune de ces places.

3.2.2 Choix d'une méthode d'ordonnancement du FMS

Au sein du fonctionnement global du module de diagnostic (diagramme 3.8), la méthode d'ordonnancement est requise lors de l'initialisation du module afin de permettre à ce dernier de calculer en ligne l'ordonnancement de manière synchronisée avec le module de supervision. En d'autres termes, l'ordonnancement calculé par le module de supervision et l'ordonnancement cal-

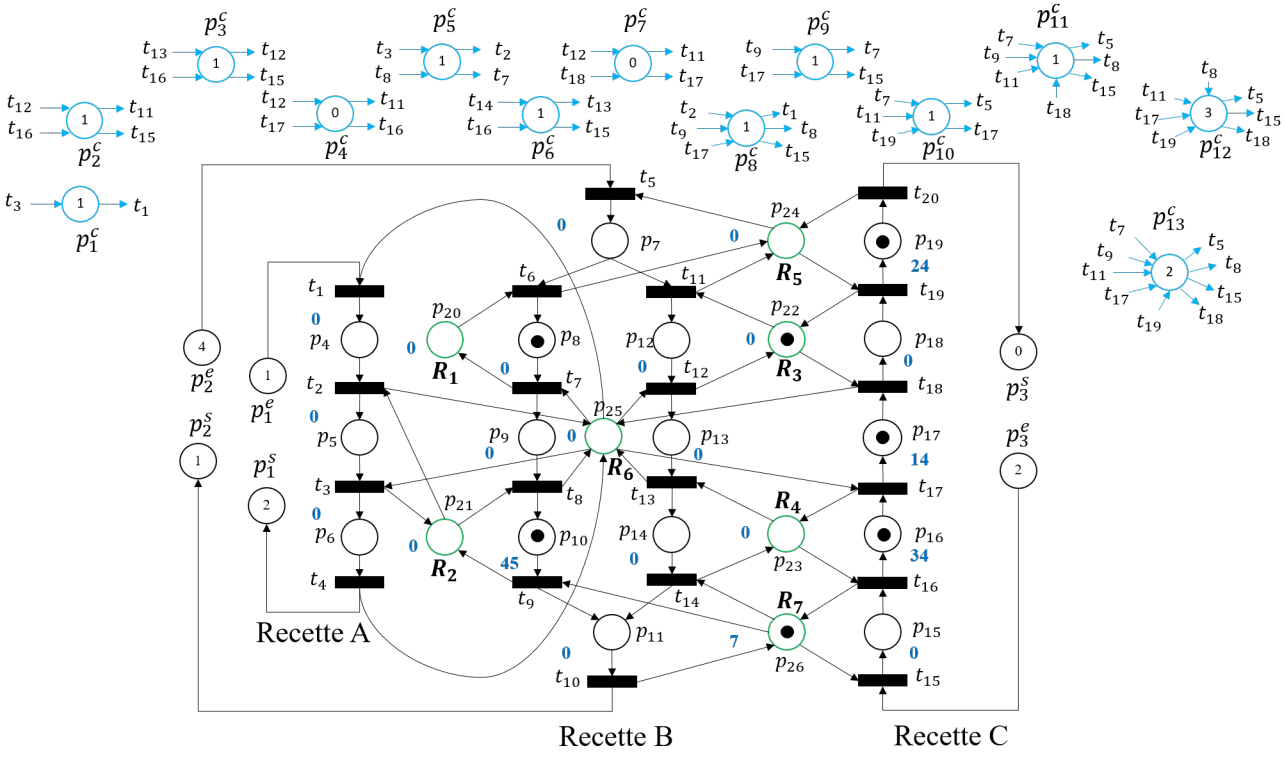


FIGURE 3.9 – Exemple d'un modèle S^{3}PR supervisé doté de places moniteurs $p_c \in P_c$

culé par le module de diagnostic doivent être égaux et obtenus simultanément. Dans cette partie, une méthode d'ordonnancement de la littérature est par conséquent choisie selon cet objectif de synchronisation. A la suite de ce choix, une extension des S^{3}PRs, une fonction heuristique et un algorithme de la méthode d'ordonnancement choisie sont présentés succinctement. En conclusion de cette partie, un exemple d'application de cette méthode est proposé afin d'illustrer ses résultats.

Choix d'une méthode d'ordonnancement

Dans la partie 1.3.1, les différentes méthodes d'ordonnancement des FMSs fondées sur la théorie des RdPs ont été introduites et ont été comparées selon les caractéristiques suivantes : la taille en nombre de décisions d'allocation de la séquence d'ordonnancement calculée, son optimalité par rapport aux critères choisis, la taille de l'espace d'états exploré, et le déterminisme de la méthode.

Dans nos travaux, la méthode d'ordonnancement choisie doit permettre la mise en place de la méthode de diagnostic des attaques de blocages. **Pour rappel, cette dernière doit être capable d'estimer à partir des données fiables bas-niveau l'ordonnancement normalement calculé par le module de supervision afin de diagnostiquer le comportement optimal ou non du FMS**. Considérant cet objectif de diagnostic, la méthode d'ordonnancement choisie doit posséder les caractéristiques suivantes :

1. La séquence d'ordonnancement calculée doit avoir une taille partielle ou totale. En effet, il a été montré la nécessité pour le calcul des profils d'attaquant d'avoir une séquence d'ordonnancement suffisamment grande afin de savoir quelles décisions d'allocation vont

devoir être trompées par l'attaquant. Par ailleurs, une séquence d'ordonnancement au minimum partielle calculée à partir d'un état M du FMS permettra la construction d'un diagnostiqueur sur plusieurs états d'allocation successifs capables, grâce à cette projection, d'établir la diagnosabilité d'un profil d'attaquant $\mathcal{P}(M)$.

2. L'objectif principal de cette méthode n'est pas d'obtenir une solution optimale mais de garantir le fonctionnement du diagnostiqueur.
3. Le temps de calcul d'une séquence d'ordonnancement par la méthode choisie doit être faible. Si, en ligne, la solution de diagnostic doit recalculer une séquence d'ordonnancement, elle doit pouvoir le faire en un temps réduit, idéalement avant que la prochaine décision d'allocation ne soit prise par le module de supervision. Cependant, le temps de calcul n'est pas un indicateur pouvant être déterminé mathématiquement à partir de l'algorithme utilisés par la méthode mais doit être obtenu par simulation. Il peut toutefois être estimé d'une méthode à l'autre en comparant la complexité et la taille de l'espace des états exploré des algorithmes des deux méthodes.
4. La méthode d'ordonnancement doit être déterministe. Si elle ne l'est pas, la méthode de diagnostic pourrait obtenir une estimation de l'ordonnancement différente de l'ordonnancement réel lorsque l'orientation aléatoire de la recherche de solutions ne conduit pas le diagnostiqueur et le module de supervision vers le même ordonnancement final.
5. Enfin, le critère ou les critères de production à optimiser par la méthode d'ordonnancement ne sont pas précisément définis dans ces travaux puisqu'ils dépendent de décisions prises au niveau de la stratégie de l'entreprise (niveau 4-5 de la pyramide CIM). Par défaut, le temps total d'exécution (makespan) sera le critère à optimiser puisqu'il s'agit du critère traité en majorité dans la littérature [18], [53]-[55].

Le tableau 1.2 de classification des méthodes d'ordonnancement présenté dans la partie 1.3.1 permet d'éliminer les méthodes fondées sur des règles de dispatching, les méthodes de méta-heuristiques et les méthodes d'apprentissage du périmètre des méthodes conformes aux caractéristiques listées ci-dessus. De fait, la première catégorie de méthodes ne respecte pas la contrainte de taille, générant des solutions unitaires, tandis que les deux autres catégories ne garantissent pas le déterminisme de la méthode. Par conséquent, **les méthodes de recherche sont choisies pour l'ordonnancement des FMSs équipés d'une solution de diagnostic des attaques de blocage**. Cependant, la famille des méthodes de recherche pour le calcul de l'ordonnancement des FMSs modélisés par des RdPs est très diversifiée et le choix final d'une tel méthode de recherche est orienté par la contrainte suivante : la méthode doit être capable de calculer un ordonnancement en un temps réduit de l'ordre de la durée minimum d'une opération du FMS (caractéristique 3.). Cette contrainte peut être respectée au détriment de l'optimalité de la solution.

Face à cette contrainte, la méthode retenue est une méthode hybride combinant la recherche $A^\star$ avec le retour sur trace appelée recherche $A^\star BT$. Cette hybridation utilisée à de nombreuses reprises dans la littérature [363]-[367] possède plusieurs versions dans lesquelles les appels aux méthodes de recherche $A^\star$ et de retour sur trace au sein de l'algorithme hybride diffèrent. Parmi ces différentes déclinaisons de la méthode $A^\star BT$, celle proposée par [365] a été retenue, l'auteur prouvant par simulation dans un ouvrage récent [64] la supériorité de sa version par rapport aux autres en terme de temps de calcul et d'optimalité de la solution finale. Cette méthode déploie

la recherche $A^\star$ localement et le retour sur trace globalement, permettant ainsi de combiner l'optimalité locale de la première avec l'efficacité de convergence vers une solution finale de la seconde. Il s'agit d'une méthode déterministe dont l'ordonnancement final calculé, noté σ_f, est total. Le choix de cette méthode pour répondre à notre problématique de temps de calcul faible résulte de l'existence d'un paramètre $K_{max} \in \mathbb{N}$ définissant la taille maximum de l'espace exploré lors d'une recherche $A^\star$ local et permettant de moduler les performances de la méthode entre optimalité de l'ordonnancement et temps de calcul.

Par conséquent, cette méthode de recherche $A^\star BT$ est introduite en trois étapes. Dans un premier temps, le critère à optimiser, à savoir le temps total d'exécution, impose d'une part la définition d'un modèle de RdP temporisé adapté au calcul de l'ordonnancement et d'autre part la construction d'une heuristique permettant son optimisation. Dans un deuxième temps et à partir de ce nouveau modèle, l'algorithme de la méthode $A^\star BT$ sélectionnée est présenté. Enfin, une application de cet algorithme pour un exemple donné permettra de valider les performances temporelles attendues de la méthode.

Modèle SC-net

Le modèle conçu pour l'ordonnancement des FMSs et optimisant le temps total d'exécution des produits est appelé un *SC-net* [365] et est noté $N_t = (P_t = P_A \cup P_R \cup P_E \cup P_S, T, F_t, D)$. Il s'agit d'un RdP P-temporisé obtenu à partir d'un modèle S^3PR P-temporisé $N = (P_A \cup P_R \cup P^0, T, F, D)$ par la scission des places de P^0 en des places d'entrée P_E et des places de sortie P_S. Ce modèle est défini dans l'annexe 6.4 de ce manuscrit et est illustré par la figure 3.11. Un *SC-net* N_t permet le calcul de l'ordonnancement par la recherche $A^\star BT$ grâce à sa capacité à définir un marquage objectif (M_{obj}, R_{obj}) de ses places de sortie. La définition d'un marquage objectif est introduit dans l'annexe 6.4 et correspond à un état du FMS où tous les produits en entrée du FMS et tous les produits en cours de fabrication dans le FMS ont été terminés selon les recettes qu'ils suivent.

Dans le reste de nos travaux, le modèle d'attaque N_α est lui aussi étendu aux places d'entrée et de sortie des SC-nets. Ainsi, on a $P_0^\alpha = P_0^a = P_0^G \cup P_0^S = (P_E^G \cup P_S^G) \cup (P_E^S \cup P_S^S)$ avec P_E^G et P_S^G les places d'entrée et de sortie dans G et P_E^S et P_S^S les places d'entrée et de sortie dans G. Cette modification de N_α est requise pour le calcul des profils d'attaquant. D'une part, elle permet de définir précisément le nombre de produits présents en entrée des recettes du FMS, produits pouvant être débutés et manipulés par un attaquant pour la construction d'un profil. D'autre part, l'ajout des places d'entrée et de sortie à G et à S permet la projection des états de N_α sur des états de N_t dans G et dans S. Cette capacité de projection sera requise par l'algorithme de calcul des profils d'attaquant dans un contexte incertain pour le calcul d'ordonnancements depuis des états attaqués de N_α.

Fonction heuristique

L'algorithme de recherche $A^\star BT$ cherche à trouver un chemin sous-optimal d'un état initial noté (M_{init}, R_{init}) jusqu'à (M_{obj}, R_{obj}). Néanmoins, la recherche $A^\star BT$ requiert la définition d'une fonction heuristique dont le rôle est d'orienter l'algorithme vers les chemins d'ordonnancement les plus prometteurs par rapport au critère à optimiser. Les définitions d'une heuristique et de ses propriétés sont introduites dès à présent car elles seront requises lors du développement de l'algorithme de calcul des profils d'attaquant dans la partie 3.2.3.

Définition 3.2.1 (Heuristique)**.** Pour un état $(M, R) \in \mathcal{R}(N_t, M_0^t)$ appartenant à un chemin d'ordonnancement candidat, une heuristique h estime le coût minimum restant exact $h^\star(M, R)$ entre (M, R) et (M_{obj}, R_{obj}) par une valeur estimée $h(M, R)$. Dans notre travail, le coût minimum correspond au temps total d'exécution minimum de l'ordonnancement. On note $h^\star(M, R)$ la valeur réel du coût de (M, R) à (M_{obj}, R_{obj}). ◇

En omettant le critère à optimiser, le choix d'une heuristique h est communément justifié par les quatre propriétés définies ci-après.

Définition 3.2.2 (Admissibilité, cohérence, informativité et référentiel d'une heuristique)**.**

- Admissibilité : L'admissibilité d'une heuristique h est définie par$\forall (M, R) \in \mathcal{R}(N_t, M_0^t)$, $h(M, R) < h^\star(M, R)$;
- Cohérence : La consistance d'une heuristique est définie par $\forall (M, R), (M', R') \in \mathcal{R}(N_t, M_0^t)$ $|\exists t \in T_t, (M, R)[t > (M', R'), h(M, R) \leq h(M', R') + c((M, R), (M', R'))$ où c est le coût de la transition entre les deux états (M, R) et (M', R') ;
- Informativité : Pour deux heuristiques admissibles h, h', h est plus informée que h', ssi $\forall (M, R) \in \mathcal{R}(N_t, M_0^t), h(M, R) \leq h'(M, R))$. L'heuristique plus informée h est capable de mieux estimer le coût restant depuis (M, R) que h' ;
- Référentiel : Enfin, une heuristique h se définit par l'entité du FMS servant de référentiel au calcul de h à savoir les ressources, les opérations ou les recettes du FMS [365].

◇

Dans nos travaux, l'admissibilité et la consistance de la fonction heuristique ne sont pas des propriétés obligatoires puisque l'optimalité de la solution finale, ce que garantissent ces deux propriétés à la recherche $A^\star$, n'est pas recherchée. Pour sa part, l'informativité de l'heuristique doit être maximisée dans le cas où il est admissible car plus ce dernier est informé plus il orientera la recherche vers des solutions plus optimales qu'il aura correctement estimées. Enfin, l'entité de référence pour le calcul de l'heuristique choisie sont les ressources, afin de prendre en considération les temps de libération des ressources et en anticipation du chapitre 4 où l'indisponibilité de celles-ci sera prise en compte.

L'heuristique choisie a été proposée en 2020 par [64] et cherche à estimer pour un état (M, R) le maximum parmi toutes les ressources du systèmes du temps total minimum nécessaire à une ressource donnée pour finir tous les produits en entrée du FMS ou en cours. Cette heuristique est admissible et est qualifiée par l'auteur de "très" informée par rapport aux autres heuristiques admissibles de la littérature déployées dans le cadre de la recherche $A^\star$ appliquée aux RdPs. Il est présenté dans la formule ci-contre.

$$h(M, R) = \max_{p_r \in P_R} \{ \sum_{p_i \in P \setminus P_R} M(p_i).WRT(p_i, p_r) + R(p_i, 1) + D(p_r).\varphi(p_r) \}$$

Avec,

$$\varphi(p_r) = \begin{cases} (\sum\limits_{\substack{p_i \in P \setminus P_R \\ M(p_i) \neq 0 \\ WRT(p_i, p_r) \neq 0}} M(p_i)) - 1, & si \sum\limits_{\substack{p_i \in P \setminus P_R \\ M(p_i) \neq 0 \\ WRT(p_i, p_r) \neq 0}} M(p_i) \geq 2 \\ 0, & sinon \end{cases}$$

Dans cette formule, WRT est une matrice de dimensions $|P \setminus P_R| \times |P_R|$ où un élément $WRT(p_i, p_r)$ désigne le temps minimum d'opération par la ressource p_r nécessaire à un produit en p_i pour être terminé. Il est à noter que dans $WRT(p_i, p_r)$, seuls les temps des opérations par p_r ayant lieu après l'opération modélisée par p_i sont pris en compte.

Par ailleurs, le terme $D(p_r).\varphi(p_r)$ de cette heuristique a été ajouté par rapport à la version d'origine pour inclure les temps de libération associés aux places ressources. En effet, pour une ressource p_r, son temps total minimum de libération est égal à son temps unitaire de libération multiplié par le nombre minimum moins 1 de produits non finis requérant obligatoirement cette ressource afin d'être terminés. Cette nouvelle version de h est admissible puisque la forme initiale est admissible selon [64] et le second terme représente le temps total nécessaire à la ressource p_r pour se libérer entre deux opérations devant obligatoirement être exécutées par p_r pour la réalisation de produits non finis. Ce second terme est admissible car il n'est en aucun supérieur au temps de libération réel requis par la ressource sur la durée de l'ordonnancement. La nouvelle version est donc plus informative que l'originale.

Définition 3.2.3 (Fonction f d'estimation du coût d'une solution d'ordonnancement)**.**
Soit (M_{init}, R_{init}) l'état initial pour le calcul de l'ordonnancement.
Soit (M, R) l'état actuel de la solution d'ordonnancement étudiée.
Soit $g(M, R)$ le coût réel observé entre l'état de début d'ordonnancement (M_{init}, R_{init}) et (M, R).
Soit $h(M, R)$ l'estimation par l'heuristique h du coût minimum restant entre (M, R) et l'état objectif (M_{obj}, R_{obj}).

L'estimation du coût total d'une solution d'ordonnancement est notée $f(M, R)$ et est égale à $f(M, R) = g(M, R) + h(M, R)$. Le coût total réel est noté $f^\star(M, R)$. ◇

Algorithme $A^\star BT$

L'algorithme de recherche $A^\star BT$ utilise l'estimation f pour prioriser les chemins d'ordonnancement les plus prometteurs, soit ceux avec les plus petites valeurs de f. Cet algorithme est présenté par son pseudo-code dans l'annexe 6.5 de ce manuscrit. Le fonctionnement itératif de cet algorithme peut être résumé à l'aide de la figure 3.10.

Dans cette figure, chaque triangle représente un ensemble d'états (M, R) du FMS explorés lors la $i^{ème}$ itération de la recherche $A^\star$ local. Un état (M, R) correspond à un ordonnancement unique parmi ceux existant depuis (M_{init}, R_{init}) jusqu'à (M, R) et possède des coûts f, g et h propres. Si la taille de l'espace exploré i dépasse une constante choisie $K_{max} \in \mathbb{N}$, la recherche sur trace est appliquée globalement et l'état le plus prometteur de l'espace i est conservé, à savoir l'état possédant le plus petit coût f, et une itération $i + 1$ de la recherche $A^\star$ est débutée localement à partir de cet élément (croix rouge dans la figure). La recherche sur place permet de surcroît lorsqu'un ensemble exploré i devient vide (triangle en pointillés) de retourner aux espaces précédents $j < i$ afin de sélectionner le meilleur état de tous les espaces j et de recommencer la $i^{ème}$ recherche à partir de ce nouvel état. Dans la figure, la croix verte représente un tel état, avec le deuxième plus petit coût f de l'espace i, depuis lequel une nouvelle itération $i + 1$ est débutée (triangle en ligne pleine de l'itération $i + 1$). Notons qu'un espace i peut devenir vide si les états qu'il explore ont déjà été visités précédemment et étaient plus optimaux en raison de valeurs de coût passé g plus faibles. L'algorithme de recherche $A^\star BT$ se termine lorsque l'état objectif (M_{obj}, R_{obj}) est atteint et retourne alors l'ordonnancement final σ_f ou lorsqu'aucun ordonnancement ne peut être trouvé.

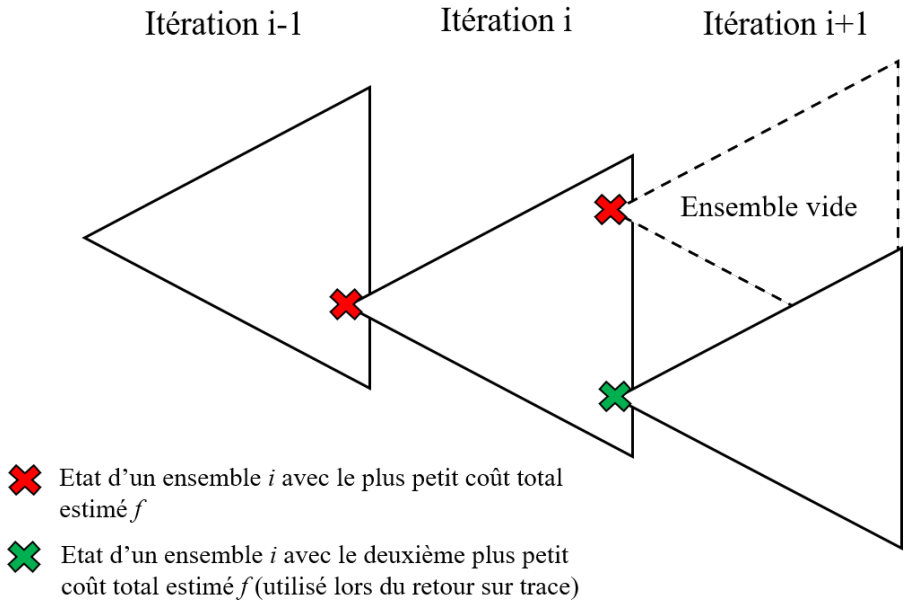


FIGURE 3.10 – Fonctionnement schématique de la recherche $A^{\star}BT$ [64]

L'intérêt de la recherche $A^{\star}BT$ est de pouvoir moduler la valeur de K_{max} afin de trouver un compromis entre temps de calcul de l'algorithme et optimalité de l'ordonnancement final obtenu. Notons que si $K_{max} = 0$, l'algorithme est équivalent à un algorithme de recherche en profondeur [365]. Les effets de K_{max} sur le temps de calcul et l'optimalité de la recherche $A^{\star}BT$ sont illustrés par l'exemple ci-après.

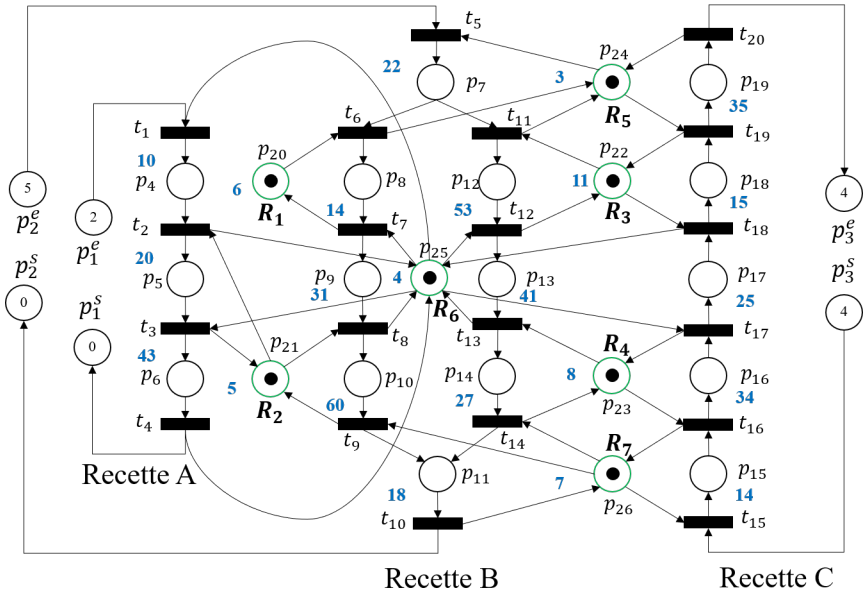


FIGURE 3.11 – Exemple d'un modèle $SC - net$

Exemple

Soit N_t le modèle *SCnet* construit à partir de l'exemple des figures 1.6 et 3.9. N_t est illustré dans la figure 3.11. Dans cette figure, les temps de chaque place $p \in P_A \cup P_R$ sont indiqués en bleu et les places contrôle de la figure 3.9 sont inclues dans cet exemple bien qu'elles soient omises de la figure pour des raisons de clarté. Les simulations de cet exemple sont réalisées sur le logiciel Matlab par un processeur quatre cœurs *Intel(R)Core(TM)i5-9400H*.

L'algorithme de recherche $A^\star BT$ est appliqué à ce modèle N_t pour trois marquages initiaux différents, à savoir $M_0^1(p_1^e, p_2^e, p_3^e) = (2, 5, 4)$, $M_0^2(p_1^e, p_2^e, p_3^e) = (4, 8, 6)$, $M_0^3(p_1^e, p_2^e, p_3^e) = (7, 14, 10)$, impliquant des états objectifs et des solutions d'ordonnancement distincts, et selon différentes valeurs de $K_{max} \in \{10, 50, 100\}$. Les résultats de taille de σ_f, de coût (temps total d'exécution en second), de temps de calcul et de nombre d'états (M, R) explorés de l'algorithme selon les marquages initiaux choisis et les valeurs de K_{max} sont présentés dans le tableau 3.1. Dans celui-ci, les résultats obtenus par la recherche $A^\star BT$ en utilisant l'heuristique originale proposée par [64] sont données entre parenthèses dans chaque cellule.

A partir de ce tableau, on observe tout d'abord que notre heuristique est globalement plus efficace que l'heuristique originale en terme de coût, de temps et de nombre d'états visités, illustrant sa meilleur informativité.

Deuxièmement, ces résultats montrent que la valeur de K_{max} influe sur le temps de calcul et le nombre d'états visités, plus K_{max} est faible, plus ces derniers le sont aussi. Les deux graphiques à droite de la figure 3.12 montre cette influence pour des valeurs de K_{max} variant de 5 en 5 entre 15 et 510. Cependant, aucune conclusion ne peut être faite à partir des résultats du tableau sur l'influence de K_{max} sur l'optimalité de la séquence d'ordonnancement finale. Toutefois, pour des valeurs de K_{max} variant de 5 en 5 entre 15 et 510, on obtient la première courbe de la figure 3.12. Ainsi, en réalisant une régression linéaire à partir des coûts calculés pour les différentes valeurs de K_{max}, une tendance globale de diminution du coût final est constatée à travers le coefficient directeur négatif de la droite estimée, justifiant ainsi le gain d'optimalité obtenu lorsqu'un plus grand nombre d'ordonnancements sont explorés localement par la recherche $A^\star BT$. Néanmoins, au détriment de l'optimalité, le choix final de K_{max} doit en priorité permettre un temps de calcul faible pour permettre le ré-ordonnancement en ligne des modules de supervision et de diagnostic et garantir leur synchronisation. Précisément, les temps de calcul du tableau 3.1 sont tous inférieurs ou de l'ordre de la durée minimum des opérations du FMS (10s pour p_4) à l'exception de la recherche depuis M_0^3 pour $K_{max} = 100$.

Enfin, ce tableau met en exergue le gain en temps de calcul et en nombre d'états explorés de la recherche $A^\star BT$ réalisé dès lors que le nombre de produits à réaliser en entrée des recettes diminue. Cette dernière observation sera importante lors du déploiement de la méthode d'ordonnancement et la construction de conditions de déclenchement synchronisé d'un calcul de ré-ordonnancement. En effet, le nombre de produits en entrée de chaque recette pris en compte dans la définition du marquage initial (M_{init}, R_{init}) de l'algorithme $A^\star BT$ sera borné dans l'optique de réduire la taille de l'ordonnancement final et conséquemment le temps de calcul de l'algorithme.

L'algorithme de recherche $A^\star BT$ choisi et présentée dans cette partie permet d'obtenir une séquence d'ordonnancement sous-optimale σ_f à partir de n'importe quel état (M_{init}, R_{init})

TABLEAU 3.1 – Résultats de la recherche $A^\star BT$ sur l'exemple de la figure 3.11

M initial	M_0^1			M_0^2			M_0^3		
$Kmax$	10	50	100	10	50	100	10	50	100
Taille σ_f	62			100			172		
Coût	612s (629s)	613s (674s)	645s (584s)	977s (1028s)	984s (988s)	896s (974s)	1592s (1715s)	1679s (1742s)	1591s (1594s)
Temps de calcul	0.42s (0.59s)	1.79s (2.18s)	5.8s (6.21s)	0.75s (1.12s)	3.49s (5.97s)	10.40s (9.77s)	1.47s (2.25s)	9.08s (16.62s)	21.74s (26.31s)
Nombre d'états	230 (270)	650 (650)	1000 (800)	360 (390)	1000 (1000)	1400 (1600)	600 (670)	1800 (1800)	2500 (2700)

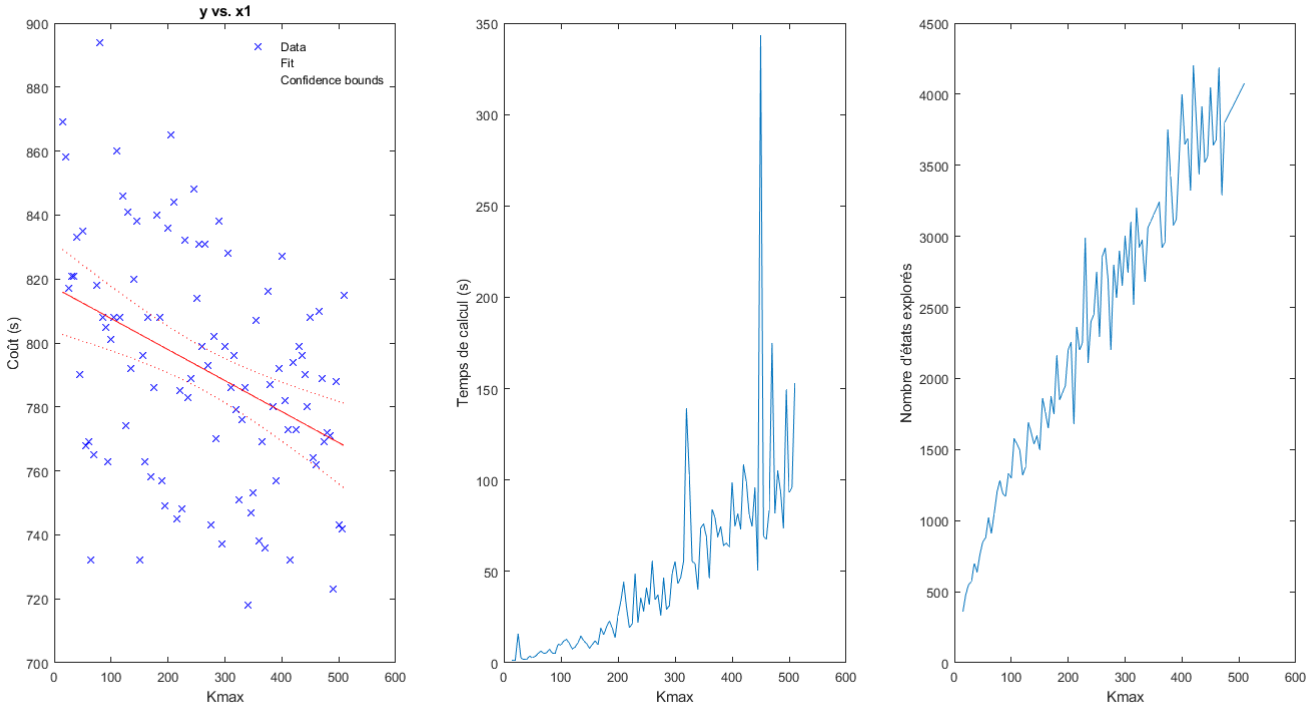


FIGURE 3.12 – Evolution du coût, du temps de calcul et du nombre d'états explorés de l'algorithme $A^\star BT$ selon la valeur de K_{max}

du modèle $SC - net$ du FMS. En modulant la valeur de K_{max}, cette méthode termine son calcul d'ordonnancement en un temps faible conformément aux contraintes introduites précédemment. Dans la prochaine partie, les algorithmes de calcul des profils d'attaquant sont développés à partir de σ_f et en conservant la méthodologie de la recherche $A^\star$.

3.2.3 Méthode de calcul des profils d'attaquant

Au sein du fonctionnement global du module de diagnostic (diagramme 3.8), l'algorithme de calcul des profils d'attaquant est requis pour la construction du diagnostiqueur des profils d'attaquant. Toutes les données nécessaires en entrée de cet algorithme ont été introduites dans

ce chapitre : les profils d'attaquant (sous-partie 3.1.1), le modèle des attaques de blocage N_α (sous-parties 3.1.2, 3.1.3) et les méthodes de supervision et d'ordonnancement (sous-parties 3.2.1,3.2.2). L'objectif de cette partie est de présenter cet algorithme et son application aux trois profils d'attaquant, deux contributions originales de nos travaux.

Les trois profils d'attaquant présentés dans la partie 3.1.1 ont été retenus pour leur capacité à modéliser différents types d'attaquants réalistes aux origines, moyens, objectifs et sous-objectifs distincts. Le diagnostic de ces profils d'attaquant, à savoir la capacité pour un état $(M, R) \in \mathcal{R}(N_L, M_0)$ succédé par une séquence d'événements observés λ_o d'inférer ou non la possible présence du profil, requiert de connaître la séquence d'attaque λ_a générée par un profil $\mathcal{P}$ pour atteindre ses objectifs.

Pour chaque profil, l'algorithme de calcul de λ_a est introduit dans cette partie. Tout d'abord, une présentation générale de ces algorithmes est menée à travers la méthode de recherche globale utilisée pour chacun d'eux. Ensuite, profil par profil, chaque algorithme est détaillé au regard des ses spécificités, à savoir ses objectifs, sa fonction coût, ses heuristiques et ses contraintes propres. Enfin, un exemple permettra d'analyser les résultats obtenus par chacun de ces algorithmes.

Algorithme général de calcul des profils d'attaquant

Le calcul d'un profil d'attaquant équivaut à la recherche de la séquence de transitions optimale $\sigma_\alpha = l_\alpha^{-1}(\lambda_\alpha) \in T_\alpha^\star$ à partir d'un état initial $(M_{init}, R_{init}) \in \mathcal{R}(N_\alpha, M_0^\alpha)$ depuis lequel l'attaquant cherche à atteindre un état de G parmi l'ensemble des états critiques ciblés $\mathcal{M}_c$ selon les caractéristiques communes et propres de son profil. La recherche $A^\star$ permet de trouver une telle séquence optimale en utilisant un heuristique admissible. Elle est par conséquent retenue et adaptée dans nos travaux à l'objectif de calcul des profils d'attaquant.

Soit $A^\star\mathcal{P}$ l'algorithme de recherche inspiré de la méthode $A^\star$ pour le calcul des profils d'attaquants. Sa structure est illustrée dans le diagramme d'activités UML 3.13 ci-dessous et est détaillée par son pseudo-code dans l'annexe 6.6. Dans ce diagramme, (M, R) est l'état traversé par l'attaque en cours d'exploration, $OPEN$ désigne la liste des états à explorer et $CLOSED$ la liste des états déjà explorés. Dans cette sous-partie, la forme générale de l'algorithme $A^\star\mathcal{P}$ est introduite au regard des ses spécificités par rapport à un algorithme de recherche $A^\star$ traditionnel, à savoir **la paramétrisation de l'algorithme selon le profil d'attaquant** et l'implémentation de **la propriété de sournoiserie d'un profil**. En conclusion de cette sous-partie, l'algorithme $A^\star\mathcal{P}$ est prouvé conforme au respect des caractéristiques communes à tous les profils introduites précédemment (3.1.1).

Premièrement, la forme générale de $A^\star\mathcal{P}$ est applicable au calcul de tout profil $\mathcal{P}$ par la paramétrisation des états objectifs $\mathcal{M}_c$, de la fonction coût c, de l'heuristique h et des contraintes de rejet d'une séquence ou d'un état. Dans le diagramme d'activité 3.13, la paramétrisation est rendue possible lors de quatre étapes distinctes au sein desquelles les paramètres propres au profil sont indiqués en rouge :

- A l'étape 4, l'état exploré (M, R) est comparé à l'ensemble $\mathcal{M}_c$ des états ciblés par l'attaquant. Si (M, R) appartient à $\mathcal{M}_c$, le profil d'attaquant $\mathcal{P}$ est construit à partir des arcs de $F_\mathcal{P}$ et l'algorithme se termine. Notons que cette comparaison nécessite la projection

de (M, R) et $\mathcal{M}_c$ sur $P_A^G \cup P_R^G$;

- A l'étape 11, la fonction coût c et l'heuristique h du profil sont requis pour le calcul du coût passé g et de l'estimation du coût total f depuis un nouvel état (M', R') atteint par le franchissement d'une transition d'attaque $t \in T_{fr}$ (étape 8 et 10). La valeur de f ainsi calculée permet le tri de $OPEN$ à l'étape 12.
- A l'étape 11, les contraintes du profil sont appliquées au nouvel état atteint (M', R'). Si une contrainte est vraie, l'état n'est ajouté ni à $OPEN$, ni à $CLOSED$ et est rebouté de l'exploration. Une contrainte est une condition interdite sur l'état (M', R') ou sur la séquence de transitions de (M_{init}, R_{init}) à (M', R').
- A l'étape 12, outre la consigne de tri de $OPEN$ selon les valeurs croissantes de f, des consignes de tri propres au profil sont appliquées. Dans le cas où une consigne de tri n'est pas la première à être appliquée, elle ne concerne que le tri des états voisins de $OPEN$ égaux selon toutes les consignes précédentes.

Deuxièmement, l'algorithme $A^\star\mathcal{P}$ diffère de l'algorithme de $A^\star$ par l'implémentation en son sein de la propriété de sournoiserie des profils d'attaquant.

En préambule à la propriété de sournoiserie, les attaques unitaires depuis (M, R) permettant à l'attaquant d'approcher un état ciblé de $\mathcal{M}_c$ dans G, soit un état de blocage, de pré-blocage ou de blocage partiel du FMS, sont définies au sein de l'étape 5 : l'insertion malveillante d'une décision d'allocation $(t_+ \in T_+^\alpha)$ ou l'exécution non altérée de la prochaine décision d'allocation de σ_f si la transition associée $t_{\sigma_f} = \tau(t_-)$ est franchissable dans N_α depuis (M, R). Ces attaques sont alors modélisées par l'ensemble T_{fr} de leurs transitions associées.

Puis, lors de l'étape 6 de la figure 3.13, la séquence $T-$ des transitions de suppression des décisions d'allocation successives ayant lieu depuis l'état $(M, R) \in \mathcal{R}(N_\alpha, (M_{init}, R_{init}))$ et franchissable avant ou simultanément à la transition d'attaque $t \in T_{fr}$ est calculé. A partir de $T-$ l'algorithme prend la décision suivante : si $T-$ est non vide et qu'il existe des décisions d'allocation ayant lieu avant que le prochain état (M', R') ciblé par l'attaquant ne soit atteint, alors, pour rester sournois, l'attaquant doit supprimer toutes les décisions de $T-$ préalablement à son attaque.

Ainsi, pour chaque transition $t \in T_{fr}$ et à partir de l'ensemble $T-$ calculé depuis t, la décision de l'étape 6 doit être prise. Si $T-$ est non vide, alors chaque $t_- \in T-$ doit être franchie avant t et les étapes 7 et 8 sont exécutées : toutes les transitions de $T-$ sont franchies successivement, $(M, R)[t_-^1 > (M_1'', R_1'')[t_-^2 > (M_2'', R_2'')...[t_- > (M'', R'')$, puis t est franchie, $(M'', R'')[t > (M', R')$. Tous les arcs associés au franchissement de ces transitions t_- et t sont ajoutés à $F_\mathcal{P}$ et tous les états intermédiaires $(M_1'', R_1''), (M_2'', R_2''), ..., (M'', R'')$ sont intégrés aux listes $OPEN$ ou $CLOSED$ s'ils leur appartiennent déjà et présentent un coût passé g plus faible. Sinon, l'étape 10 est exécutée et seule la transition t est franchie, $(M, R)[t > (M', R')$, et l'arc associé est ajouté à $F_\mathcal{P}$. D'un point de vue mathématique, l'ordre de franchissement entre les transitions de $T-$ et t est obtenu grâce à des appels à la fonction Ω pour chaque transition $t_- \in T-$ et pour t, soient $\Omega_{N_\alpha}(t, M, R, 0) = (M', R', \Gamma')$ et $\Omega_{N_\alpha}(t_-, M, R, 0) = (M'', R'', \Gamma'')$, puis à la comparaison de Γ' avec Γ''. **Ainsi, grâce à la décision lors de l'étape 6 de supprimer systématiquement les décisions d'allocation ayant lieu au cours d'une**

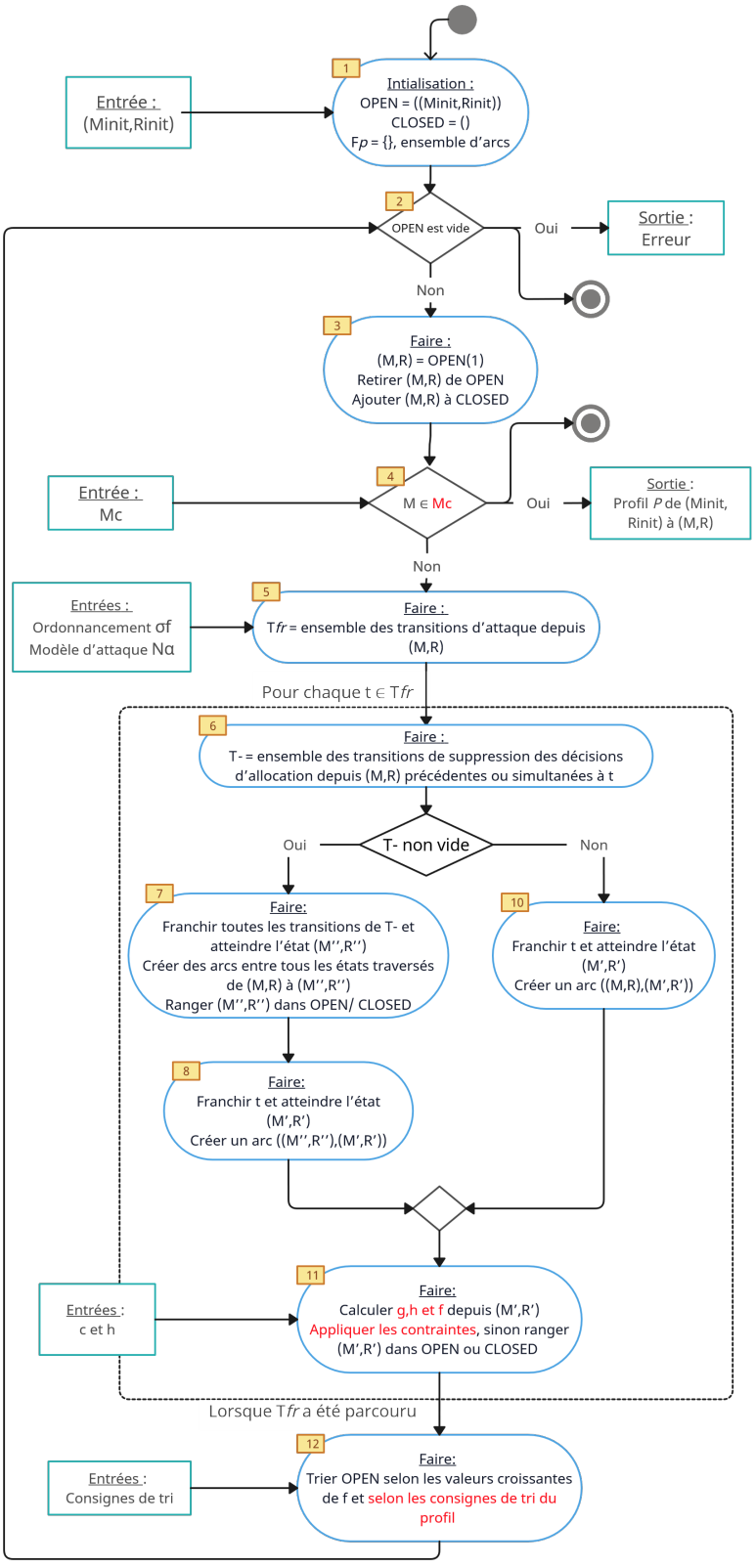


FIGURE 3.13 – Diagramme d'activité UML de l'algorithme $A^{\star}\mathcal{P}$

séquence d'attaque, la sournoiserie du profil est garantie.

Troisièmement, dans cet algorithme $A^\star\mathcal{P}$, les caractéristiques communes aux profils introduites dans la partie 3.1.1 sont respectées. En effet, le profil en sortie de l'algorithme est une séquence unitaire (caractéristique 1), l'attaque est minimale (caractéristique 2) car la recherche s'arrête au premier état de $\mathcal{M}_c$ atteint, elle est experte (caractéristique 3) par définition de N_α, et elle est sournoise (caractéristique 4), N_α ne modélisant que les attaques sournoises et la recherche $A^\star\mathcal{P}$ connaissant et trompant l'ordonnancement réel σ_f du FMS (caractéristique 5). **Le respect des ces cinq caractéristiques communes à tous les profils combiné aux quatre paramétrisations spécifiques à un attaquant permettent ainsi à l'algorithme $A^\star\mathcal{P}$ de pouvoir calculer n'importe quel profil d'attaquant**.

Remarque 3.2.1.
Identiquement à la recherche $A^\star$, la séquence d'attaque $\mathcal{P}$ calculée par l'algorithme $A^\star\mathcal{P}$ est optimale selon les caractéristiques propres au profil d'attaquant considéré grâce au tri de $OPEN$ réalisé à l'étape 12. En effet, lors de cet étape, l'état le plus optimal selon f et selon les autres consignes de tri est mis en tête de $OPEN$ afin d'être le prochain état à être exploré par l'algorithme. Toutefois, l'optimalité de cet état requiert l'admissibilité de l'heuristique h du profil. Les trois prochaines sous-parties introduisent les heuristiques des profils choisis dans la section 3.1.1 et prouvent leurs admissibilités. ⌟

Spécificités algorithmiques pour le profil d'attaquant $\mathcal{P}_1$

Le profil d'attaquant $\mathcal{P}_1$ possède quatre caractéristiques propres. La caractéristique 2 n'est pas paramétrée dans l'algorithme $A^\star\mathcal{P}$ puisqu'elle justifie l'absence de toute contrainte de coût. Les trois autres caractéristiques spécifiques sont présentées dans cette sous-partie.

Au regard de la **caractéristique 1**, l'ensemble des états ciblés a été défini précédemment par $\mathcal{M}_{c_1} = \{M \in \mathcal{R}(N, M_0) | \nexists t \in T, M' \in \mathcal{R}(N, M_0)\ t.q.\ M[t > M'\}$, i.e. l'attaquant cible l'ensemble des états de blocage hors états de pré-blocage ou de blocage partiel.

La **caractéristique spécifique 3.a** du profil 1, stipulant que l'attaquant minimise le nombre d'états de pré-blocage ou de blocage partiel que son attaque traverse, est définie à travers une consigne de tri. Avant que le tri selon les valeurs de f croissantes n'ait lieu, l'algorithme trie les éléments de $OPEN$ selon le nombre croissant d'états de $\mathcal{M}_{DZ} \setminus \mathcal{M}_{c_1}$ que leur trajectoire depuis (M_{init}, R_{init}) contient. Le tri selon f ne s'applique alors qu'aux états ayant le même nombre d'états de $\mathcal{M}_{DZ} \setminus \mathcal{M}_{DD}$ dans leurs trajectoires. Cette consigne de tri prioritaire s'applique aussi à l'étape 11 du diagramme 3.13 lorsque pour un nouvel état (M', R'), celui-ci est déjà présent dans $OPEN$ ou dans $CLOSED$ par un état (M^{oc}, R^{oc}), $M' = M^{oc}$ et $R' = R^{oc}$. On compare alors dans un premier temps le nombre d'état de pré-blocage ou de blocage partiel traversés avant de comparer les valeurs de g de ces deux états et de décider lequel des deux conserver dans $OPEN$ ou dans $CLOSED$ (c.f. annexe 6.6).

La **caractéristique spécifique 3.b** du profil 1, spécifiant que l'attaque cible les états de blocage dans lesquels les ressources sont le plus détenues, est définie par la modification de $\mathcal{M}_{c_1}$. La constante $U_{max} = \min_{M_{c_1} \in \mathcal{M}_{c_1}}(M_{c_1}(P_R))$ représentant le nombre maximum de ressources détenues au sein des états de blocage du FMS est calculée et permet de restreindre $\mathcal{M}_{c_1}$ à $\mathcal{M}^2_{c_1} = \{M_c \in \mathcal{M}_{c_1} | M_{c_1}(P_R) = U_{max}\}$, soit l'ensemble des états de blocage pour lesquels U_{max}

ressources sont détenues.

Afin d'appliquer la **caractéristique 4**, la fonction coût g_1 de ce profil calcule le nombre d'opérations ayant eu lieu entre l'état (M_{init}, R_{init}) et l'état (M, R) dans G. On a donc $c_1((M,R),(M',R')) = 1$, $c_1((M,R),(M'',R'')) = 0$ et $c_1((M'',R''),(M',R')) = 1$. L'heuristique h_1 doit alors estimer le nombre minimum d'opérations restantes de (M, R) à un état de $\mathcal{M}^2_{c_1}$. Il est développé à partir de l'étude des marquages des siphons stricts minimaux de N, calculés à partir des travaux de [93] et rassemblés au sein de l'ensemble Π_{SMS}. Cette heuristique h_1 se présente sous la forme ci-dessous :

Définition 3.2.4 (Heuristique h_1)**.**

$$h_1(M,R) = \min_{S \in \Pi_{SMS}} (M(S))$$

◇

Proposition 3.2.1. *h_1 est admissible pour la fonction coût c_1 et $h_1^\star$.*

Preuve 3.2.1. Il a été prouvé dans le théorème 1.2.2 que si un marquage de blocage M_{DD} est atteint, l'ensemble de places $Q \subset P$ tel que $M_{DD}(Q) = 0$ est un siphon. Or, puisque tous les siphons dans Π_{SMS} sont minimaux, cela signifie $\exists S \in \Pi_{SMS}$ tel que $S \subset Q$ et $M_{DD}(S) = 0$. Afin de vider ce siphon S à partir d'un état M, il faut franchir une séquence de transitions faisant perdre $M(S)$ jeton à S. Cette séquence de transition représente les opérations devant être réalisées par le FMS pour vider S. L'heuristique h_1 cherche à être une borne inférieure pour n'importe quel $S \in \Pi_{SMS}$ de la taille de cette séquence de transitions. Cependant, pour avoir une égalité entre $M(S)$ et le nombre minimum de transitions nécessaires pour vider S, il faut prouver que dans un modèle S^{3}PR ordinaire le franchissement d'une transition ne peut pas enlever plus de un jeton à S, sinon, dans le cas contraire, il pourrait suffire de franchir moins de $M(S)$ transitions pour vider S.

Cette preuve peut être apportée à partir des travaux de [93] associant les siphons SMS aux circuits ressources transitions maximales parfaits. Par l'absurde, si l'on suppose qu'une transition $t \in (T \setminus ((P^0)^\bullet \cup {}^\bullet(P^0)))$ faisant perdre deux jetons à S existe, on aurait alors que $p_r^2 = P_R \cap {}^\bullet t \in S$ et $p_a^1 = P_A \cap {}^\bullet t \in S$ mais $p_r^1 = P_R \cap t^\bullet \notin S$ et $p_a^2 = P_A \cap t^\bullet \notin S$. Ces quatre places, $p_a^1, p_a^2, p_r^1, p_r^2$ sont les seules places connectées à t dans N. Or, les auteurs dans [93] ont prouvé que si pour un SMS S, $p_r \in P_R$ et $H_r \cap S \neq \emptyset$ alors $p_r \in S$. Cette propriété n'est pas respectée par les places $p_a^1, p_a^2, p_r^1, p_r^2$ puisqu'on a $p_a^1 \in H_{r1} \cap S$ mais $p_r^1 \notin S$. Il n'existe donc pas de transition $t \in (T \setminus ((P^0)^\bullet \cup {}^\bullet(P^0))$ dont le franchissement fait perdre deux jetons à un siphon S. Le marquage $M(S)$ est ainsi une borne inférieure du nombre d'opérations nécessaires pour vider totalement S et h_1, en étant égal au minimum de $M(S)$ parmi tous les siphons $S \in \Pi_{SMS}$, est donc admissible pour estimer le nombre minimum d'opérations nécessaires pour atteindre un état de blocage depuis M. ⌟

Enfin, la deuxième partie de la **caractéristique 4** du profil 1 est appliquée par une dernière consigne de tri ordonnant que si des égalités de f subsistent entre deux états voisins $(M,R),(M',R')$ de $OPEN$, ceux-ci doivent être triés par valeur croissante de temps global de (M_{init}, R_{init}) à (M,R) et (M',R').

Spécificités algorithmiques pour le profil d'attaquant $\mathcal{P}_2$

Le profil d'attaquant $\mathcal{P}_2$ possède quatre caractéristiques propres. Ces quatre caractéristiques spécifiques sont présentées dans cette sous-partie.

Au regard de la **caractéristique 1**, l'ensemble des états ciblés a été défini précédemment par $\mathcal{M}_{c_2} = \{M \in \mathcal{R}(N, M_0) | \nexists t \in T, M' \in \mathcal{R}(N, M_0)\ t.q.\ M[t > M'\} \cup \{M \in \mathcal{R}(N, M_0) | \forall M' \in \mathcal{R}(N, M_0), \sigma \in T^\star\ t.q. M[\sigma > M' \Rightarrow M' \in \mathcal{M}_{c_1} \cup \mathcal{M}_{c_2}\}$, regroupant les états de blocage et de pré-blocage indifféremment. De surcroît, afin de ne pas explorer les états de blocage partiel lors de la recherche, la contrainte Ctr_{2a} : "$M \in \mathcal{M}_{DZ} \setminus \mathcal{M}_{PDD}$" est inclue à CTR_2 l'ensemble des contraintes du profil 2.

La **caractéristique 2**, bornant le coût d'une attaque à une valeur maximale C^2_{max}, est aussi définie par une contrainte Ctr_{2b} : "$CT_\alpha(\lambda_a) \geq C^2_{max}$" avec $\lambda_a = l_\alpha(\sigma_a) | \sigma_a \in T^\star_\alpha$ et $(M_{init}, R_{init})[\sigma_a > (M, R)$. Cette contrainte Ctr_{2b} est ajoutée CTR_2.

Afin d'appliquer la **caractéristique 3**, la fonction coût c_2 du profil 2 doit permettre le calcul du produit $(coût) \times (nombre\ d'opérations)$ de l'attaquant. Ainsi, la fonction coût passé g_2 est définie par $g_2(M, R) = g^a_2(M, R) \times g^b_2(M, R)$ avec $g^a_2 = g_1$ la fonction coût passé qui compte le nombre d'opérations ayant eu lieu depuis (M_{init}, R_{init}) à l'aide de la fonction coût c_1 et g^b_2 la fonction coût passé d'une attaque définie par $g^b_2 = CT_\alpha(\lambda_a)$ avec $\lambda_a = l_\alpha(\sigma_a) | \sigma_a \in T^\star_\alpha$ et $(M_{init}, R_{init})[\sigma_a > (M, R)$, où σ_a représente le chemin dans N_α entre (M_{init}, R_{init}) et (M, R).

La fonction g^b_2 peut aussi être calculée à partir de l'état précédent par $g^b_2(M', R') = g^b_2(M, R) + c^b_2((M, R), (M'', R'')) + c^b_2((M'', R''), (M', R'))$ ou par $g^b_2(M', R') = g^b_2(M, R) + c^b_2((M, R), (M', R'))$ dans le cas où aucune décision d'allocation ne doit être supprimée. Dans cette formule $c^b_2((M, R), (M'', R'')) = Ct_\alpha(t_-)$ et $c^b_2((M'', R''), (M', R')) = c^b_2((M, R), (M', R')) = Ct_\alpha(t)$ où t et t_- sont définies dans la sous partie précédente.

La fonction heuristique associée à ce coût passé g_2 est noté h_2 et est définie par :

Définition 3.2.5 (Heuristique h_2)**.**

$$h_2(M, R) = h^a_2(M, R) \times h^b_2(M, R)$$

avec

$$h^a_2(M, R) = \begin{cases} \min\limits_{S \in \Pi_{SMS}} [M(S) - \max\limits_{M_c \in \mathcal{M}_{c_2}} (M_c(S))] & , si \geq 1 \\ 0 & , si M \in \mathcal{M}_c \\ 1 & , sinon \end{cases}$$

et

$$h^b_2(M, R) = \begin{cases} 0 & , si M \in \mathcal{M}_c \\ \min\limits_{\substack{l_\alpha(e_a) = t_a \\ t_a \in (T^\alpha_+ \cap L(N_\alpha, (M,R))|_1)}} Ct_\alpha(e_a) & , sinon \end{cases}$$

$\diamond$

Dans cette formule, h^a_2 est similaire à l'heuristique h_1 mise à part qu'un second terme constant y est soustrait. Celui-ci estime pour un siphon $S \in \Pi_{SMS}$ donné et pour tous les états

de blocage et de pré-blocage du FMS, le marquage maximum que peut avoir S lorsque de tels états sont atteints. Il peut en effet exister des états de pré-blocage pour lesquels S n'est pas vide. Il faut donc prendre en compte ces états dans l'estimation h_2^a du nombre d'opérations restantes jusqu'à un état ciblé de M_{c_2}. De surcroit, deux autres cas définissent h_2^a. D'une part, si $M \in \mathcal{M}_c$, l'heuristique est nulle puisqu'aucune opération n'a lieu une fois un état ciblé atteint. D'autre part, si la première estimation de h_2^a est inférieure à 1 et $M \notin \mathcal{M}_c$, alors $h_2^a(M,R) = 1$ car au moins une opération doit encore être débutée dans G via l'insertion d'une décision d'allocation avant d'atteindre depuis (M,R) un état ciblé de blocage ou de pré-blocage.

Quant à h_2^b, il estime le coût restant de l'attaque. Cette estimation est égale à 0 si $M \in \mathcal{M}_c$ puisque l'attaquant a atteint son objectif. Sinon, h_2^b est égale à une borne inférieure du coût restant, à savoir la plus petite valeur non nulle de Ct_α pour les attaques d'insertion modélisées par les transitions franchissables $t \in (T_+^\alpha \cap L(N_\alpha,(M,R))|_1)$. Cette borne a été choisie car il a été démontré (sous-partie 2.2.3) que pour atteindre un état de la DZ, l'attaquant doit réaliser au moins l'une de ces attaques d'insertion afin d'outrepasser le contrôle du module de supervision.

Proposition 3.2.2. *h_2 est admissible pour la fonction coût c_2 et $h_2^\star$.*

Preuve 3.2.2. On a prouvé que h_1 était admissible et on peut prouver de la même manière que h_2^a l'est aussi car son second terme soustrait est une borne supérieure du nombre de jeton restants dans le siphon strict minimal S pour tout état de pré-blocage ou de blocage. Il reste donc depuis un état (M,R) au minimum $h_1(M,R)$ moins cette borne supérieure opérations pour atteindre un état de blocage ou de pré-blocage relatif à S. Par définition, h_2^b est admissible par rapport au coût réel restant de l'attaque mais est peu informé. Puisque h_2^a et h_2^b sont admissibles, leur produit h_2 l'est aussi. ⌟

Ainsi, à partir de $g_2 = g_2^a \times g_2^b$ et $h_2 = g_2^a \times g_2^b$, la fonction coût totale estimé f_2 peut-être calculée. Cependant, l'utilisation de fonctions g_2 et h_2 sous la forme de produits modifie la formule de f_2. En effet, on sait que $\forall (M,R) \in \mathcal{R}(N_\alpha, M_0^\alpha), \forall (M_c,R_c) \in \mathcal{M}_c$, on a $f_2(M,R) \leq g_2(M_c,R_c) = g_2^a(M_c,R_c) \times g_2^b(M_c,R_c)$ puisque h_2 est admissible. Soient $f_2^a(M,R) = g_2^a(M,R) + h_2^a(M,R)$ une estimation du nombre d'opérations nécessaires pour atteindre $\mathcal{M}_c$ depuis (M_{init},R_{init}) et $f_2^b(M,R) = g_2^b(M,R) + h_2^b(M,R)$ une estimation du coût total de ce chemin. L'admissibilité de h_2^a et h_2^b par rapport à g_2^a et g_2^b permet d'établir les inégalités $f_2^a(M,R) \leq g_2^a(M_c,R_c)$ et $f_2^b(M,R) \leq g_2^b(M_c,R_c)$. On peut alors définir la formule de f_2 suivante :

$$f_2 = f_2^a \times f_2^b = (g_2^a + h_2^a) \times (g_2^b + h_2^b) = g_2^a.g_2^b + h_2^a.h_2^b + g_2^a.h_2^b + g_2^b.h_2^a \leq g_2 = g_2^a \times g_2^b \quad (3.1)$$

L'ajout des termes croisés $g_2^a.h_2^b$ et $g_2^b.h_2^a$ transforme ainsi $f_2(M,R)$ en une meilleure estimation de $g_2(M_c,R_c)$.

Enfin, la **caractéristique 4** du profil 2 est appliquée à travers une consigne de tri qui ordonne les éléments de $OPEN$ ayant la même valeur de f_2 par valeur croissante de coût d'attaque (g_2^b).

Spécificités algorithmiques pour le profil d'attaquant $\mathcal{P}_3$

Le profil d'attaquant $\mathcal{P}_3$ possède quatre caractéristiques propres. Ces quatre caractéristiques spécifiques sont présentées dans cette sous-partie.

La **première caractéristique** de $\mathcal{P}_3$, l'ensemble de ses états objectifs, soient les états de blocage partiel, est définie par $\mathcal{M}_{c_3} = \mathcal{M}_{DZ} \backslash \mathcal{M}_{c_2}$. Parallèlement, afin d'éviter les états de blocage et de pré-blocage lors de la recherche, la contrainte suivante est définie : Ctr_3^a : "$M \in \mathcal{M}_{c2}$".

La **caractéristique 2** du profil 3, une contrainte de coût C_{max}^3, a déjà été présentée pour le profil 2 et prend donc la forme d'une contrainte Ctr_3^b : "$CT_\alpha(\lambda_a) \geq C_{max}^3$" avec $\lambda_a = l_\alpha(\sigma_a) | \sigma_a \in T_\alpha^\star$ et $(M_{init}, R_{init})[\sigma_a > (M, R)$.

Au regard de la **caractéristique 3**, la fonction coût du profil 3 est égale au produit $(coût) \times (durée)$ de l'attaque. La fonction coût passé peut alors être défini par $g_3 = g_3^a \times g_3^b$ avec $g_3^a = g_2^b$ la sous-fonction calculant le coût de l'attaque et $g_3^b = g$ la sous-fonction calculant la durée de l'attaque (c.f. l'algorithme $A^\star BT$ 3.2.1). La définition de g_3^b diffère légèrement de g par l'existence des états intermédiaires (M'', R'') et on définit alors la fonction coût c_3^b par $c_3^b((M,R),(M',R')) = \Gamma'$, $c_3^b((M,R),(M'',R'')) = \Gamma''$ et $c_3^b((M'',R''),(M',R')) = \Gamma' - \Gamma''$. L'heuristique h_3 est le produit $h_3 = h_3^a \times h_3^b$ avec $h_3^a = h_2^b$, l'heuristique d'estimation du coût restant de l'attaque, tandis que la nouvelle heuristique h_3^b est définie de manière séquentielle dans les prochains paragraphes.

L'heuristique h_3^b cherche à estimer depuis un état (M, R) le temps minimum restant pour atteindre un état objectif, ici un état de blocage partiel. Pour faire cette estimation, l'heuristique h_3^b calcule le plus petit temps nécessaire pour qu'un siphon strict minimal $S \in \Pi_{SMS}$ atteigne un marquage M' où $M'(S) = KLL(S)$ avec $KLL(S)$ une constante propre à S représentant le marquage maximal que peut avoir S lorsqu'un état de blocage partiel de $\mathcal{M}_{c_3}$ est atteint. Pour estimer ce temps minimum, les étapes suivantes sont nécessaires :

Premièrement, $KLL(S)$ doit être défini :

Définition 3.2.6 ($KLL(S)$: marquage maximal d'un siphon S pour un état $M \in \mathcal{M}_{c_3}$)**.**

$$K_{LL}(S) = \max_{M_c \in \mathcal{M}_{c_3}} (M_c(S)),$$

◇

Deuxièmement, une matrice représentant pour un produit présent dans une place activité ou d'entrée $p_i \in P_A \cup P^0$ le temps minimum qui lui est nécessaire pour atteindre une place activité atteignable $p_k \in P_A$ est requise. Pour un siphon $S \in \Pi_{SMS}$, cette matrice permettra de connaître le temps nécessaire à un produit pour atteindre, si possible, chaque place puits de S, soit les places activité faisant perdre un jeton à S. Cette matrice est dépendante du temps de délai restant R des jetons des places activité $p_i \in P_A$ et est par conséquent noté $MSK(R)$. Cette matrice est définie par :

Définition 3.2.7 ($MSK(R)$: matrice des temps requis entre deux places activités d'un S^3PR)**.** Soit $MSK(R)$ une matrice de dimension $|P_A \cup P^0| \times |P_A|$.

Pour une place $p_j \in P_A$ atteignable depuis une place $p_i \in P_A \cup P^0$ en suivant le chemin de places activité $p_i p_{i_1} p_{i_2} ... p_j$, MSK est égal à :

$$MSK(R)(p_i, p_j) = max(R(p_i), R(p_{i_1})) + D(p_{i_1}) + D(p_{i_2}) + ... + D(p_{j_{-1}})$$

avec $p_{j_{-1}}$ la place précédent p_j.

Le premier terme de cette somme représente le temps de délai minimum restant à attendre au produit dans la place p_i pour atteindre la place activité suivante p_{i_1} en considérant qu'un produit peut être présent dans p_{i_1}. Si p_i est une place d'entrée on a $R(p_i) = 0$. Le reste de la somme représente le temps total de toutes les opérations que va suivre le produit dans les places activité $p_{i_1}, p_{i_2}, ..., p_{j_{-1}}$.

Si plusieurs chemins de places activité existent entre p_i et p_j en raison d'un parallélisme dans le modèle S^{3}PR, $M(R)(p_i, p_j)$ est égal au temps total d'exécution minimum parmi ces chemins. Enfin, si p_j n'est pas atteignable depuis p_i via un chemin de places activité, alors $M(R)(p_i, p_j) = \{\varepsilon\}$.

◇

Troisièmement, pour un siphon S et l'état (M, R) du FMS, un ensemble regroupant grâce à $MSK(R)$ toutes les combinaisons possibles des temps nécessaires pour amener des produits présents dans les places activités marquées de M vers des places puits de S et ainsi obtenir un marquage de S égal à $KLL(S)$ doit être construit. Cet ensemble, dépendant de (M, R) et de S, est noté $U(MSK(R), M, S)$ et est défini par :

Définition 3.2.8 $(U(MSK(R), M, S))$**.**
Soit $P_{sk}(S) = \{p \in P_A | p \in H_r, p_r \in S, p \notin S\}$ l'ensemble des places puits du siphon S lui enlevant un jeton. A partir de cet ensemble $P_{sk}(S)$, $U(MSK(R), M, S)$ est défini par :

$$U_{MSK(R)}(M,S) = \{\{MSK(R)(p_{i_1}, p_{j_1}), ..., MSK(R)(p_{i_\beta}, p_{j_\beta})\} \textit{ tel que } \forall (m_1, m_2) \in [1, \beta], m_1 < m_2, \beta = M(S) - KLL(S), (p_{i_{m_1}}, p_{i_{m_2}}) \in (P_A \cup P^0), (p_{j_{m_1}}, p_{j_{m_2}}) \in P_{sk}(S), MSK(R)(p_{i_{m_1}}, p_{j_{m_1}}) \neq \{\varepsilon\}, MSK(R)(p_{i_{m_2}}, p_{j_{m_2}}) \neq \{\varepsilon\}, \textit{ on a } p_{j_{m_1}} \neq p_{j_{m_2}}, M(p_{j_{m_1}}) = M(p_{j_{m_1}}) = 0$$

$$\textit{et } \forall p_{i_{m_1}} \in \{p_{i_1}, ..., p_{i_\beta}\}, \sum_{\substack{p_i = p_{i_{m_1}} \\ p_i \in \{p_{i_1}, ..., p_{i_\beta}\}}} 1 \leq M(p_i)\}$$

Dans cette définition, un ensemble de temps $\{MSK(R)(p_{i_1}, p_{j_1}), ..., MSK(R)(p_{i_\beta}, p_{j_\beta})\}$ doit respecter les différentes conditions listées. La première ligne de conditions définit les places $p_{i_{m_1}}, p_{i_{m_2}}, p_{j_{m_1}}, p_{j_{m_2}}$ et rappelle que les valeurs de $MSK(R)$ associées doivent être différentes de $\{\varepsilon\}$. La deuxième ligne oblige les deux places puits $p_{j_{m_2}}$ et $p_{j_{m_1}}$ ciblées par deux produits différents dans les places $p_{i_{m_1}}$ et $p_{i_{m_2}}$ à être différentes et vides initialement. Notons que $p_{i_{m_1}}$ et $p_{i_{m_2}}$ peuvent être identiques dans le cas d'une place d'entrée contenant plusieurs jetons, i.e. plusieurs produits. Quant à la troisième ligne, elle conditionne la présence d'un produit disponible dans toutes les places de départ $p_i \in \{p_{i_1}, ..., p_{i_\beta}\}$. Ainsi le nombre de places égale à p_i dans $\{p_{i_1}, ..., p_{i_\beta}\}$ doit toujours être inférieur au nombre initial de produits dans p_i, soit $M(p_i)$.

◇

Quatrièmement, à partir des trois première définitions, l'heuristique h_3^b en tant quel peut être défini par :

Définition 3.2.9 (Heuristique h_3^b)**.**

$$h_3^b(M,R) = \min_{S\in\Pi_{SMS}}(\min_{U\in U(MSK(R),M,S)}(\max(U)))$$

◇

Proposition 3.2.3. *h_3 est admissible pour la fonction coût c_3 et $h_3^\star$.*

Preuve 3.2.3. On sait que $h_3^a = h_2^b$ est admissible puisque h_2^b l'est. L'admissibilité de la fonction h_3^b peut être prouvée en décomposant ses différents termes.

Soit un siphon $S \in \Pi_{SMS}$, on sait qu'il doit perdre au minimum $M(S) - KLL(S)$ jeton pour atteindre un état de blocage partiel avec $KLL(S)$ calculé de manière similaire au second terme de la première ligne de h_2^a. L'ensemble $U(MSK(R), M, S)$ dans h_3^b énumère tous les groupes de $M(S) - KLL(S)$ chemins différents pouvant permettre à un produit présent dans une place activité ou d'entrée du FMS d'atteindre une place puits $p_k \in P_{sk}(S)$ du siphon S, à savoir une place faisant perdre un jeton à S quand elle est marquée. Au sein d'un groupe $U \in U(MSK(R), M, S)$ de chemins distincts devant avoir lieu pour atteindre un état de blocage partiel en vidant S de $M(S) - KLL(S)$ jetons, on choisit le chemin avec le temps d'exécution le plus long car ce temps est dans tous les cas nécessaire pour atteindre l'état de blocage partiel $(max(U))$. Notons que l'on choisit le maximum des temps de tous les chemins de U et non la somme de ces temps car ces chemins peuvent être parcourus en parallèle par le FMS si par exemple ils ne font pas partie de la même recette.

Puis, parmi tous les groupes U de $U(MSK(R), M, S)$, on prend le plus petit maximum à travers la première application de la fonction min. Il s'agit pour un siphon S d'une borne inférieure du temps nécessaire pour lui faire atteindre un marquage égal à $KLL(S)$.

Enfin, en prenant le minimum des bornes inférieures des temps nécessaires pour atteindre un état de blocage partiel parmi tous les siphons $S \in \Pi_{SMS}$ (seconde application de la fonction min), une valeur de h_3^b inférieure au temps minimum réel nécessaire à l'attaquant pour atteindre un état de blocage partiel est obtenue. **Ainsi, h_3^b et admissible est par produit de fonctions admissibles, $h_3 = h_3^a \times h_3^b$ l'est aussi.**

⌟

A l'instar du profil 2, la formule de f_3 inclut des termes croisés, soit $f_3 = f_3^a \times f_3^b = (g_3^a + h_3^a) \times (g_3^b + h_3^b) = g_3^a.g_3^b + h_3^a.h_3^b + g_3^a.h_3^b + g_3^b.h_3^a$.

Enfin, la **caractéristique 4** du profil 3 se traduit par une consigne de tri stipulant que si une égalité subsiste entre les valeurs de f_3 de deux états voisins dans $OPEN$, il doivent être triés selon la valeur croissante de *coût* de l'attaque.

Exemple

Dans les prochaines paragraphes, l'algorithme de construction des profils d'attaquant est appliqué aux trois profils choisis afin d'illustrer sur un exemple les contributions exposées dans cette deuxième partie du chapitre 3.

Soit N_α le modèle d'attaque construit à partir du modèle S^{3}PR N de l'exemple 1.6 où la temporisation des places est identique à celle de la figure 3.11. La construction de N_α est

TABLEAU 3.2 – Coûts unitaires de chaque événement attaqué et coûts totaux des différentes transitions attaquées dans l'exemple 3.2.3

	Ressource libérée	**Ressource requise**	**Attaque**	$e_{deb-}\|e_{deb+}$	$e_{fin+}\|e_{fin-}$	$e_{lib-}\|e_{lib+}$	$e_{dis+}\|e_{dis-}$	**Coût total**
t_1	/	R_6	$t_-(t_{21})$	41	33	/	/	74
			$t_+(t_{41})$	53	44	/	/	97
t_2	R_6	R_2	$t_-(t_{22})$	36	26	40	43	145
			$t_+(t_{42})$	45	39	48	41	173
t_3	R_2	R_6	$t_-(t_{23})$	40	35	32	34	141
			$t_+(t_{43})$	54	42	42	36	174
t_4	R_6	/	$t_-(t_{24})$	/	/	40	43	83
			$t_+(t_{44})$	/	/	48	41	89
t_5	/	R_3	$t_-(t_{25})$	24	19	/	/	43
			$t_+(t_{45})$	35	28	/	/	63
t_6	R_3	R_1	$t_-(t_{26})$	20	16	26	25	87
			$t_+(t_{46})$	30	25	32	28	115
t_7	R_1	R_6	$t_-(t_{27})$	40	36	22	20	118
			$t_+(t_{47})$	55	43	27	24	149
t_8	R_6	R_2	$t_-(t_{28})$	35	25	40	43	143
			$t_+(t_{48})$	44	39	48	41	172
t_9	R_2	R_7	$t_-(t_{29})$	22	18	32	34	106
			$t_+(t_{49})$	33	27	42	36	138
t_{10}	R_7	/	$t_-(t_{30})$	/	/	24	23	47
			$t_+(t_{50})$	/	/	30	25	55
t_{11}	R_3	R_4	$t_-(t_{31})$	38	33	26	25	122
			$t_+(t_{51})$	49	41	32	28	150
t_{12}	R_4	R_6	$t_-(t_{32})$	39	38	36	38	151
			$t_+(t_{52})$	54	46	45	39	184
t_{13}	R_6	R_5	$t_-(t_{33})$	26	23	40	43	132
			$t_+(t_{53})$	36	31	48	41	156
t_{14}	R_5	R_7	$t_-(t_{34})$	23	18	28	29	98
			$t_+(t_{54})$	31	26	37	31	125
t_{15}	/	R_7	$t_-(t_{35})$	/	/	24	23	47
			$t_+(t_{55})$	/	/	30	25	55
t_{16}	R_7	R_5	$t_-(t_{36})$	26	24	24	23	97
			$t_+(t_{56})$	35	30	29	26	120
t_{17}	R_5	R_6	$t_-(t_{37})$	42	37	28	29	136
			$t_+(t_{57})$	53	45	37	31	166
t_{18}	R_6	R_4	$t_-(t_{38})$	38	34	40	43	155
			$t_+(t_{58})$	48	40	48	41	177
t_{19}	R_4	R_3	$t_-(t_{39})$	24	20	36	38	118
			$t_+(t_{59})$	36	25	45	39	145
t_{20}	R_3	/	$t_-(t_{40})$	/	/	26	25	51
			$t_+(t_{60})$	/	/	32	28	60

TABLEAU 3.3 – Évolution des valeurs de f, g et h pour les fonctions coût de chaque profil d'attaquant

Etats		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mathcal{P}_1$	g_1	0	0	0	0	0	1	1	1	1	1	1	1	2	3	4	5	\	\	\	\
	h_1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	\	\	\	\
	f_1	0	1	1	1	1	2	2	2	2	2	2	2	3	4	5	**5**	\	\	\	\
$\mathcal{P}_2$	g_2	0	0	0	291	\	\	\	\	\	\	\	\	\	\	\	\	\	\	\	\
	h_2	0	110	165	0	\	\	\	\	\	\	\	\	\	\	\	\	\	\	\	\
	f_2	0	110	165	**291**	\	\	\	\	\	\	\	\	\	\	\	\	\	\	\	\
$\mathcal{P}_3$	g_3	0	0	0	0	0	0	0	0	0	0	4140	7490	19404	23688	28106	40208	55440	66960	73080	107778
	h_3	0	385	770	1320	1870	1870	2475	2860	3245	3465	3795	3850	4235	5170	5170	6160	6160	6600	6600	0
	f_3	0	385	770	1320	1870	1870	2475	2860	3245	3465	7935	11340	23639	28858	33276	46368	61600	73560	79680	**107778**

détaillée dans la première partie 3.1 de ce chapitre et seul la fonction coût unitaire Ct_α reste à définir. Il existe 20 transitions $t \in T$ attaquables dans N à partir desquelles nous construisons les transitions attaquées T^α_+ et T^α_-. Ainsi, $\forall t \in T, Ct_\alpha(t) = 0$ tandis que le coût unitaire des 40 transitions attaquées est présentée dans le tableau 3.2 ci-contre. Cette fonction coût unitaire est prouvée acceptable selon les contraintes présentées dans la sous-partie 3.1.2 en fixant l'ordre des vulnérabilités par $R6 > R4 > R2 > R5 > R3 > R7 > R1$, en calculant $\overline{Ct_\alpha} = 33.87$, $\sigma_{Ct_\alpha} = 7.79$ et en observant que toutes les valeurs du tableau sont comprises pour $K = 3$ entre $[33.87 - 3 \times 7.79 = 10.5; 33.87 + 3 \times 7.79 = 57.23]$. Aucune figure modélisant N_α dans sa totalité n'est proposée pour des raisons de lisibilité.

Nous simulons l'algorithme de la figure 3.13 décliné pour les trois profils d'attaquant à partir des conditions initiales suivantes : l'état d'entrée de l'algorithme est (M, R) avec $M = 3p^e_1 + 5p^e_2 + 2p^e_3 + p_8 + p_{10} + p_{16} + p_{17} + p_{19} + p_{22} + p_{26} + 3p^e_{27} + 5p^e_{28} + 2p^e_{29} + p_{34} + p_{36} + p_{42} + p_{43} + p_{45} + p_{48} + p_{52} + p_{53} + p_{54} + p_{55} + p_{57} + p_{58} + p_{60} + p_{61} + p_{62} + p_{63} + 3p_{64} + 2p_{65}$ et R défini par $R(p_{10}) = R(p_{36}) = 45$, $R(p_{16}) = R(p_{42}) = 34$, $R(p_{17}) = R(p_{43}) = 14$, $R(p_{19}) = R(p_{45}) = 24$, $R(p_{26}) = R(p_{52}) = 7$ et $R(p_i) = 0$ pour toutes les autres places de P_α, l'ordonnancement réduit à partir de (M, R) est égal $\sigma_f = t_{15}t_{18}t_{20}t_{17}t_{19}t_{16}t_9t_{18}t_7t_{10}$ pour une valeur de $K_{max} = 20$, et pour le profil 2, $C^2_{max} = 400$ tandis que pour le profil 3, $C^3_{max} = 800$. Au sein de ces conditions initiales, on observe dans (M, R) que les places communes entre G et S sont marquées identiquement et possèdent les mêmes temps de délai, que les transitions franchies au sein de σ_f sont uniquement des transitions non attaquées appartenant à T et que la contrainte $C^2_{max} < C^3_{max}$ est bien respectée. A partir de ce paramétrage, on obtient les trois profils d'attaquant suivants représentés par les séquences de transitions correspondantes.

$$\sigma_{\mathcal{P}_1(M,R)} = t_{35}t_{58}t_{38}t_{41}t_{20}t_{45}t_{55} = t_{15-}t_{18+}t_{18-}t_{1+}t_{20}t_{5+}t_{15+}$$

$$\sigma_{\mathcal{P}_2(M,R)} = t_{15}t_{18}t_{41} = t_{15}t_{18}t_{1+}$$

$$\sigma_{\mathcal{P}_3(M,R)} = t_{15}t_{18}t_{20}t_{17}t_{19}t_{16}t_9t_{18}t_7t_{60}t_{30}t_{59}t_8t_{35}t_{60}t_{37}t_{45}t_{40}t_{51}$$

$$= t_{15}t_{18}t_{20}t_{17}t_{19}t_{16}t_9t_{18}t_7t_{20+}t_{10-}t_{19+}t_8t_{15-}t_{20+}t_{17-}t_{5+}t_{20-}t_{11+}$$

Pour chaque profil, le marquage atteint projeté sur G aux places de $P^{G\alpha}_A \cup P^{G\alpha}_R$ appartient à l'ensemble interdit ciblé par chacun d'eux projeté sur $P_A \cup P_R$. Ainsi, le profil 1 atteint dans $P_A \cup P_R$ le marquage $M_{c_1} \in \mathcal{M}_{c_1} = p_4 + p_7 + p_8 + p_{10} + p_{15} + p_{16} + p_{18}$, le profil 2 atteint $M_{c_2} \in \mathcal{M}_{c_2} = p_8 + p_{10} + p_{15} + p_{16} + p_{18} + p_{19}$ et le profil 3 atteint $M_{c_3} \in \mathcal{M}_{c_3} =$

$p_{10} + p_{11} + p_{12} + p_{16} + p_{20} + p_{24} + p_{25}$. En utilisant le modèle S^{3}PR de la figure 3.11, il est possible de valider visuellement les catégories de marquages interdits de M_{c_1}, M_{c_2} et M_{c_3}. Pour le profil 1, il a par ailleurs été par vérifié par simulation qu'aucun état de pré-blocage n'était traversé pour atteindre l'état M_{c_1} et que $M_{c_1}(P_R) = 0$, validant ainsi les contraintes de recherche associées au profil. Au regard de leurs séquences de transitions, ces profils d'attaquant suppriment correctement les décisions d'allocation prises par le module de supervision. Par exemple, le profil 1 supprime dans l'ordre les décisions modélisées par les transitions t_{15} et t_{18} en franchissant les transitions t_{35} et t_{38}. Pour sa part, l'admissibilité des différentes heuristiques h_1,h_2 et h_3 peut être validée par le tableau 3.3 présenté ci-dessous représentant l'évolution de g, h et f calculés pour les états parcourus par les profils d'attaquant.

Pour les trois profils, on observe que les valeurs de f pour chaque état intermédiaire sont toujours inférieures à la valeur finale $f^\star$ (en rouge dans le tableau), prouvant ainsi l'admissibilité des heuristiques. Cependant, ce tableau révèle aussi la faible informativité des différents heuristiques proposées puisque leurs valeurs intermédiaires prises sont éloignées de la valeur réelle de $h^\star = f^\star - g$. L'amélioration de ces heuristiques pourrait ainsi faire l'objet de futur travaux de recherche. Enfin, dans cette exemple de simulation, les contraintes de coûts des profils 2 et 3 sont vérifiées puisque $CT_\alpha(l_\alpha(\sigma_{\mathcal{P}_2(M,R)})) = 97 < 400$ et $CT_\alpha(l_\alpha(\sigma_{\mathcal{P}_3(M,R)})) = 759 < 800$. Pour le profil 3, cette contrainte de coût l'oblige à attendre que le système atteigne un état propice avant de modifier son premier événement comme l'illustre l'égalité entre les neuf premières transitions franchies par σ_f et $\sigma_{\mathcal{P}_3(M,R)}$. Cette attente de l'état propice est aussi une conséquence des états de blocage partiel ciblés par le profil 3. En effet, ces derniers nécessitent de conserver des circuits du FMS non bloqués et par conséquent conduisent le profil 3 à libérer le FMS de certains produits en cours de fabrication en franchissant par exemple la transition t_{20+}.

Dans la partie 3.2 du chapitre 3, les différents comportements du FMS et de l'attaquant, nécessaires au diagnostic des attaques de blocage et des profils d'attaquant (voir diagramme 3.8), ont été définis, modélisés et simulés à l'aide d'algorithmes dédiés. Premièrement, les états interdits du FMS, à savoir les états de blocage, de pré-blocage et de blocage partiel, ont été introduits et rendus inaccessibles dans le cas sans-attaque depuis les états autorisés grâce à une méthode de supervision fondée sur des places de contrôle P_c ajoutées au modèle S^{3}PR du FMS. Deuxièmement, les comportements optimaux du FMS ont été présentés à travers le choix d'une méthode d'ordonnancement partagée et adaptée à la coopération entre le module de supervision et celui de diagnostic. Un algorithme de la méthode sélectionnée, la recherche $A^\star BT$, a été présenté et simulé dans cette partie. Enfin, les comportements de l'attaquant ont été simulés afin d'obtenir les profils d'attaquant présentés dans la partie 3.1.1. Un algorithme a ainsi été développé pour permettre le calcul des profils d'attaquant pour n'importe quel état temporisé (M, R) du FMS. Dans la prochaine partie, une méthode de diagnostic des attaques de blocage et des profils d'attaquant sera positionnée et construite à l'aide de ces différents comportements.

3.3 Diagnostic des attaques de blocage

La détection des attaques de blocage consiste au diagnostic de celles-ci et de leurs origines parmi les différents profils d'attaquants présentés précédemment. Dans cette partie, un module de diagnostic des attaques de blocage dans un contexte certain est développé. Premièrement (3.3.1), le positionnement de ce module dans l'architecture de contrôle commande des FMSs

(1.1.2) est défini à partir des données fiables requises pour un diagnostic performant et non vulnérable aux attaques. Le positionnement ainsi choisi requiert l'élaboration de conditions de ré-ordonnancement communes aux modules de supervision et de diagnostic présentées dans cette même première partie. Puis (3.3.2), à partir des données fiables et des différents modèles comportementaux présentés dans la partie précédente (3.2), la méthode de diagnostic est construite dans le but de détecter systématiquement les attaques de blocage et de diagnostiquer les profils pouvant y être associés. Enfin (3.3.3), la mise en application du module de diagnostic est présentée et illustrée par un exemple complet et critique concluant ce chapitre.

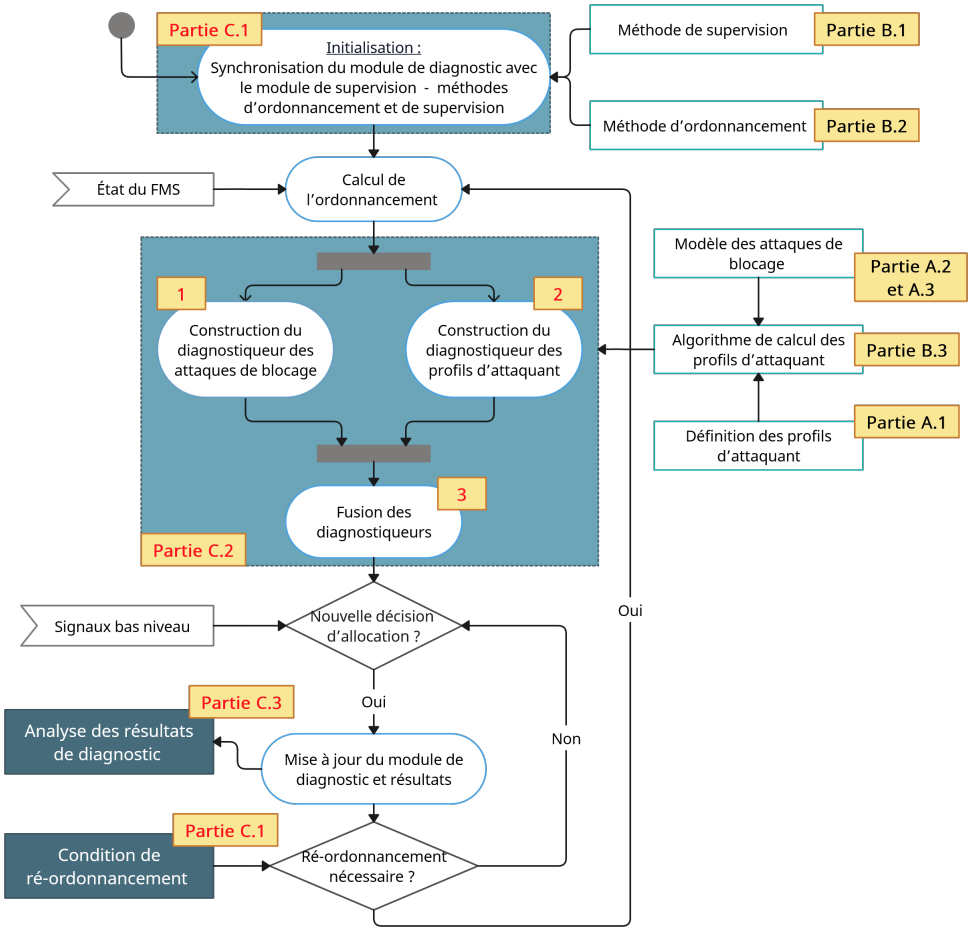


FIGURE 3.14 – Diagramme d'activité UML global du module de diagnostic

Les différents travaux présentés dans cette partie sont illustrés par la re-mobilisation (blocs grisés de la figure 3.14 ci-dessus) du diagramme d'activité UML du fonctionnement global du module de diagnostic proposé en introduction de ce chapitre (3.1).

3.3.1 Positionnement du module de diagnostic

Positionnement du module de diagnostic

L'objectif du module de diagnostic est de détecter et de diagnostiquer les attaques de blocage en apportant des informations fiables et interprétables aux opérateurs en cas d'anomalie. Outre les modèles comportementaux présentés précédemment (3.2), un tel module nécessite un

ensemble de données du FMS représentatif du fonctionnement en temps réel de l'allocation des ressources. En effet, ces données de fonctionnement doivent être analysées et interprétées par le module à des fins de détection d'attaques et de profils d'attaquant.

Cependant, la partie 2.2.1 a montré la vulnérabilité des composants numériques du FMS aux attaques expertes et la non fiabilité des données extraites depuis ceux-ci. Ainsi, ces données deviennent non représentatives du fonctionnement du FMS puisque l'attaquant est capable de les modifier, par insertion et par suppression, en amont et en aval de leur point d'extraction. Par exemple, un module de diagnostic positionné sur le réseau industriel extrait des événements de commande ayant pu être manipulés par l'attaquant au niveau du SCADA pour paraître conforme aux états de supervision et d'ordonnancement internes au module de diagnostic. Puis en aval du module, ces événements peuvent de nouveau être modifiés par l'attaquant au sein d'un API compromis pour conduire le FMS vers un état de blocage. Dans cette configuration, le module de diagnostic est donc lui-même vulnérable aux attaques expertes.

Dès lors, le module de diagnostic développé dans nos travaux doit avoir accès à des données fiables de l'architecture du FMS représentatives de l'état réel de l'allocation des ressources. Il a été défini et justifié dans la partie 2.2.1 que les seules données fiables de l'architecture des FMSs sont celles prélevées au niveau de la communication entre les ressources et leurs contrôleurs locaux (niveau 0-1). Ainsi, seules ces données sont exploitables par le module de diagnostic, instituant son positionnement entre les ressources et les contrôleurs locaux tel qu'illustré dans la figure 3.15.

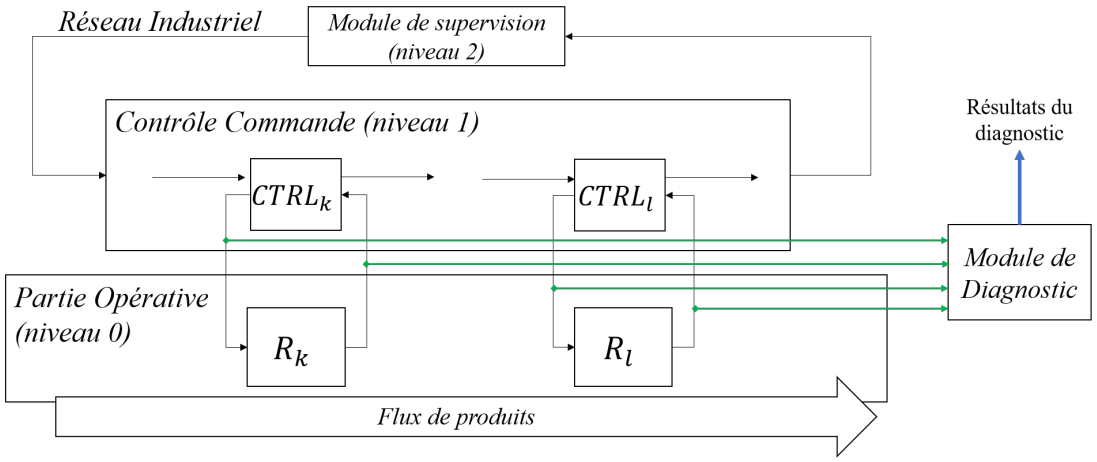


FIGURE 3.15 – Positionnement du module de diagnostic au sein de l'architecture du FMS

A partir de ces données "bas-niveau", le module de diagnostic a une visibilité fiable sur l'état réel de l'allocation des ressources de G. Néanmoins, les événements d'observation et de commande propres à l'allocation des ressources, à savoir $e_{deb}(O_i), e_{lib}(R_k), e_{fin}(O_i), e_{dis}(R_k)$, ne se présentent pas sous cette forme lorsqu'ils sont échangés entre les contrôleurs et les ressources. En effet, il s'agit majoritairement de signaux électriques analogiques (0-24V ou 4-20mA) représentatifs de commandes et d'observations continues ou discrètes. Par exemple, le début d'une opération peut correspondre au passage de 0V à 24V d'un signal électrique échangé entre un API et un actionneur tout ou rien ou à l'augmentation du signal de courant envoyé d'un régulateur à un moteur continu. **Nous supposons donc dans nos travaux que le module de diagnostic est capable d'interpréter tous les événements pour l'allocation des**

ressources à partir des données bas-niveau et ainsi de convertir un signal bas niveau en l'un de ces événements. Le module de diagnostic est alors en mesure de mettre à jour ses modèles (N_L, N_{GS}, N_α, N_t) afin que leurs places représentatives de l'état de G correspondent à l'état réel de l'allocation des ressources.

Cependant, le positionnement du module de diagnostic présenté dans la figure 3.15 pose la problématique suivante : comment synchroniser le module de supervision et le module de diagnostic pour le calcul d'une même séquence d'ordonnancement σ_f ? De fait, le module de diagnostic n'est pas autorisé à avoir un accès direct à σ_f depuis S en raison de sa non fiabilité. Or, cette séquence d'ordonnancement est nécessaire puisque le diagnostic du comportement optimal ou non optimal du FMS est requis pour la détection des attaques de blocage (voir figure 2.5). Le module de diagnostic doit donc calculer l'ordonnancement du FMS indépendamment de S mais de manière synchronisée avec ce dernier pour obtenir la même séquence σ_f dans le cas non attaqué, soit le comportement optimal attendu du FMS. Toutefois, deux contraintes à cette synchronisation inter-modules sont identifiées. D'une part, le diagnostiqueur ne peut pas avoir accès à l'information de déclenchement d'un ré-ordonnancement opéré par le module de supervision car cette information est également non fiable. D'autre part, l'ordonnancement ne possède pas de régime permanent au sein duquel une séquence d'ordonnancement cyclique se répète, mais évolue et est recalculé selon l'arrivée de nouveaux produits à fabriquer en entrée du FMS. Face à ces deux contraintes, la solution suivante est proposée : la définition de conditions de déclenchement d'un calcul d'ordonnancement. Ces conditions seront partagées entre le module de supervision et celui de diagnostic et paramétrées uniquement à partir des données représentatives de l'état de l'allocation des ressources et interprétables depuis les signaux bas-niveau.

Conditions de ré-ordonnancement

Les conditions de ré-ordonnancement sont dépendantes de l'évolution des produits en entrée des recettes, ce qui correspond mathématiquement au changement des marquages des places d'entrées $p_e \in P_E$ du modèle SC-net N_t (3.2.2). **Ainsi, l'hypothèse stipulant que le module de diagnostic connaît le nombre de produits en entrée des recettes doit être prise**. Cette connaissance peut par exemple être obtenue par un capteur tout ou rien à partir duquel un front montant est interprété par le module de diagnostic comme l'arrivée d'un nouveau produit en entrée de la recette A associée à ce capteur, i.e. l'incrémentation du marquage $M(p_e)$, tandis que le début de la première opération de la recette A entraîne la décrémentation de ce marquage. On suppose donc dans cet exemple que **le module de diagnostic est paramétré initialement avec les nombres exacts de produits en entrée des différentes recettes**.

Dans ces travaux, les conditions de déclenchement du calcul d'un ordonnancement doivent permettre la maximisation de l'optimalité de l'ordonnancement, la représentativité des quantité de produits en entrée de chaque recette par rapport au nombre de produits simultanés pouvant être opérés par chacune d'entre elles ainsi que la rapidité de calcul de σ_f par l'algorithme $A^\star BT$.

Soit M_{init} le marquage du modèle N_t lorsque l'ordonnancement σ_f est débuté, M le marquage normalement atteint après une sous-séquence σ de σ_f, $M_{init}[\sigma > M, \sigma < \sigma_f$, et M_{rl} le marquage réellement atteint par le FMS après la séquence σ et incluant l'arrivée de nouveaux

produits. On a $M(p) = M_{rl}(p), \forall p \in P_t \setminus P_E$ et $M(p) \leq M_{rl}(p), \forall p \in P_E$ avec P_E l'ensemble des places d'entrée du modèle $SC - net$. A partir d'une définition liminaire, la condition Cn_d suivante est ainsi proposée :

Définition 3.3.1 ($Mx(A)$)**.**
Soit $Mx(A)$ le nombre de produits maximum pouvant être fabriqués simultanément pour une recette A. On définit mathématiquement $Mx(A)$ par :

$$Mx(A) = \sum_{\substack{p_r \in P_R \\ \exists p_a \in H_r \cap \bigcup C_A^i}} M_0^t(p_r)$$

Pour rappel, C_A^i désigne le circuit i de la recette A dans le modèle S^{3}PR d'origine et H_r l'ensemble des places activités requérant la ressource r. $Mx(A)$ est donc une borne supérieure du nombre maximum de produits pouvant être réellement produit simultanément par A.

◇

Définition 3.3.2 (Condition de ré-ordonnancement Cn_d)**.**

$$Cn_d = Cn_d^1 \vee Cn_d^2 \vee Cn_d^3$$

avec

$$Cn_d^1 =< \sigma_f \text{ a été } totalement\ réalisée > ,$$

$$Cn_d^2 =< \forall (p_A^e, P_B^e) \in P_E \times P_E, p_A^e \neq P_B^e, (\frac{M_{rl}(p_A^e)}{M_{rl}(p_B^e)} \geqslant K_{cn_d}^1 . \frac{M_{init}(p_A^e)}{M_{init}(p_B^e)}) \wedge (\frac{M_{rl}(p_A^e)}{M(p_A^e)} \geq 1 + K_{cn_d}^2) >$$

et

$$Cn_d^3 =< \exists p_A^e, (M(p_A^e) \leq K_{cn_d}^3 . Mx(A)) \wedge (M_{rl}(p_A^e) \geq K_{cn_d}^4 . M(p_A^e))$$

◇

Dans cette formule, le premier terme de la deuxième sous-condition Cn_d^2, reflétant une variation des quantités relatives de produits en entrée de deux recettes A et B entre l'état initial M_{init} et l'état réellement atteint M_{rl}, est validé si le nombre de produits en entrée de A a connu un pourcentage d'augmentation significativement plus important que la recette B. Le second terme de Cn_d^2 est, lui, nécessaire pour infirmer que cette augmentation relative de la quantité de produits en entrée de A par rapport aux produits en entrée de B n'est pas due à l'ordonnancement σ_f. Dans Cn_d^2, la constante $K_{cn_d}^1 > 1$ régule la sensibilité de la variation relative autorisée entre A et B, autrement dit plus $K_{cn_d}^1$ est élevé plus le pourcentage d'augmentation des quantités de produits en entrée de A par rapport à ceux en entrée de B doit être élevé pour déclencher un ré-ordonnancement. La constante $K_{cn_d}^2 > 0$ permet, elle, de choisir la variation suffisante entre le nombre réel de produits en entrée de A et le nombre attendu pour considérer que l'ordonnancement ne prend plus en considération l'état réel du FMS et de ses places d'entrée.

La troisième condition Cn_d^3 est un garde-fou permettant de débuter un ré-ordonnancement lorsque le nombre de produits en entrée d'une recette A devient singulièrement faible dans

l'état attendu M par rapport à $Mx(A)$ alors que la quantité réelle de produit dans M_{rl} est plus élevée. Cette condition permet d'optimiser l'occupation des places activités de la recette A et d'éviter que cette recette ne soit pas débutée alors que des produits sont en attente en entrée de celle-ci. Dans Cn_d^3, la constante $K_{cn_d}^3$ permet de définir le seuil minimal de produits en entrée de A considéré comme "alarmant" pour la validité de l'ordonnancement tandis la constante $K_{cn_d}^4$ permet de moduler la validation de ce seuil alarmant selon si la quantité de produits réellement en entrée de A est plus ou moins importante par rapport à la quantité attendue. Il existe des cas où les conditions Cn_d^2 et Cn_d^3 sont vérifiées simultanément.

Lorsque Cn_d est vraie, un nouvel ordonnancement est calculé à partir de l'état réel (M_{rl}, R_{rl}) du FMS. Cependant, afin d'obtenir des performances de calcul adaptées au mode en ligne, les marquages des places d'entrée $p_e \in P_E$ sont redéfinis et réduits si nécessaires. Ainsi n'importe quel marquage M_{init} utilisé en entrée de l'algorithme d'ordonnancement est défini par $M_{init}(p) = M_{rl}(p)$ pour les places hors entrée $p \in P_t \setminus P_E$ et pour toutes les places d'entrée $\forall p_A^e \in P_E$, si $\exists p_e \in P_E | M_{rl}(p_e) > 0$, on a :

$$M_{init}(p_A^e) = min(M_{rl}(p_A^e), arr[(\frac{M_{rl}(p_A^e)}{\sum_{p_B^e \in P_E} M_{rl}(p_B^e)} - \frac{Mx(A)}{\sum_{p_B^e \in P_E} Mx(B)}) + 1) \times K_{cn_d}^5 * Mx(A)]) \quad (3.2)$$

Dans cette définition de M_{init}, la constante $K_{cn_d}^5$ permet de moduler le nombre de jetons mis dans les places entrée du FMS, une valeur faible de cette constante entraînant un plus petit nombre de produits à fabriquer par l'ordonnancement, une taille plus petite de σ_f et donc un temps de calcul de ce dernier plus faible. Dans cette formule, le terme central entre parenthèses permet de conserver les rapports de quantités relatives de produits en entrée des différentes recettes lorsque le nombre de jetons dans les places entrée est réduit dans M_{init}. Le terme arr désigne ici la fonction arrondie permettant d'obtenir un nombre entier de jetons. **Au cours du fonctionnement en ligne du module de diagnostic, et lorsque la condition Cn_d sera vraie, le ré-ordonnancement du FMS sera dorénavant débuté à partir de l'état** (M_{init}, R_{rl}). Dans la prochaine partie, la méthode de diagnostic des attaques de blocage et des profils d'attaquant est introduite.

3.3.2 Méthode de diagnostic

La méthode de diagnostic des attaques de blocage dans un contexte certain proposée dans cette partie est fondée sur la théorie des diagnostiqueurs de la littérature [10], [368], [369] et sur les travaux de la littérature recourant à un diagnostiqueur pour la détection d'attaque contres les SEDs [199], [346], [348]. Premièrement, la méthode de diagnostic est présentée dans sa globalité selon les différents diagnostiqueurs qui la compose. Une méthode standard de construction de diagnostiqueurs est ainsi proposée à partir de la littérature. Puis, chacun de ces diagnostiqueurs, celui des attaques de blocage et celui des profils d'attaquant, est développé. Enfin, le diagnostiqueur commun, fusionnant les deux précédents diagnostiqueurs, est présenté et les états composés ainsi obtenus sont analysés.

La construction du diagnostiqueur des attaques de blocage, du diagnostiqueur des profils d'attaquant et leur fusion en un diagnostiqueur commun est une contribution originale de nos travaux.

Fonctionnement global

Un diagnostiqueur d'un RdP labellisé $(N_l, M_0) = (P, T, F, E, l)$ est un automate modélisant l'occurrence d'événements de fautes au sein de N_l à travers l'utilisation de labels de fautes associés aux états de $\mathcal{R}(N_l, M_0)$ atteints consécutivement à ces fautes.

Définition 3.3.3 (Label d'un diagnostiqueur)**.** Soit δ un label caractérisant un état M de $\mathcal{R}(N_l, M_0)$. L'ensemble de tous les labels est noté Δ et se scinde entre le label $\{O\}$ symbolisant l'absence de fautes et les labels $\Delta_f = \{\delta_1, ..., \delta_n\}$ représentant l'occurrence d'une faute labellisée $\delta_k \in \Delta_f$ pouvant avoir lieu dans le système modélisé par N_l. On obtient donc $\Delta = \{O\} \cup \Delta_f$.

Dans N_l, l'occurrence d'une faute est représentée par un événement, noté e_{δ_k}, et est associée au label δ_k. ◇

Ainsi, à partir de la définition d'un label et d'un état initial $M_{init} \in \mathcal{R}(N_l, M_0)$ depuis lequel le diagnostiqueur est construit, une paire $(M, \delta) \in \mathcal{R}(N_l, M_{init}) \times \Delta$ est créée et définie par l'existence d'une séquence d'événements observables $\lambda_o \in E_o$ reliant M_{init} à M, i.e. $\exists \lambda \in E^\star, \sigma \in T^\star | \lambda \in P_o^{-1}(\lambda_o), \sigma = l^{-1}(\lambda)$ *et* $M_{init}[\sigma > M$. Dans cette paire, selon la valeur de δ associée à M, 4 cas se présentent :

- Si $\delta = O$ aucun événement inobservable de faute n'a pu avoir lieu entre M_0 et M, i.e. $\nexists \lambda \in E^\star, P_o(\lambda) = \lambda_o, \mathfrak{l}^{-1}(\lambda) = \sigma | \exists e_\delta \in E_{io}, \delta \in \Delta_f$ *t.q.* $e_\delta \in \lambda$ *et* $M_0[\sigma > M$.
- Si $\delta \in \Delta_f$, l'événement de faute $e_\delta \in E_{io}$ a eu lieu, i.e. $\exists \lambda \in P_o^{-1}(\lambda_o), l^{-1}(\lambda) = \sigma | e_\delta \in \lambda$ *et* $M_0[\sigma > M$.
- Si un état M est atteint par une séquence λ contenant plusieurs événements de fautes, la paire résultante est notée $(M, \{\delta_1, \delta_2, ..., \delta_k\})$ avec $\delta_1, \delta_2, ..., \delta_k \in \Delta_f$ les fautes dont les événements $e_{\delta_1}, e_{\delta_2}, ..., e_{\delta_k}$ appartiennent à λ.
- Enfin, si un état M peut être atteint par deux séquences $\lambda_1, \lambda_2 \in P_o^{-1}(\lambda_o)$ contenant deux ensembles différents de fautes labellisés Δ_1 et $\Delta_2 \subset \Delta$, alors deux paires (M, Δ_1) et (M, Δ_2) sont créées. Ce cas est extensible à n'importe quel nombre de séquences $\lambda_1, \lambda_2, ...$ différentes.

A partir de la définition des labels et de la signification de leurs associations à des états atteignables du FMS, un diagnostiqueur peut être construit de la manière suivante.

Définition 3.3.4 (Diagnostiqueur)**.** Un diagnostiqueur est un automate $Diag(N_l, M_{init}) = (X_d, E_d, f_d, x_0^d)$.

X_d est l'ensemble de ses nœuds et chaque état $x_d \in X_d$ du diagnostiqueur est un sous ensemble de paires (M, Δ_k), i.e. $x_d \in 2^{(\mathcal{R}(N_l, M_{init}) \times \Delta)}$, représentant l'ensemble des états atteignables depuis l'état M_{init} et après une séquence λ_o en explorant toutes les séquences $\lambda \in P_o^{-1}(\lambda_o)$ et en les associant aux fautes contenues dans la séquence λ reliant M_{init} et l'état atteint M.

E_d représente l'ensemble des événements labellisant les arcs de $Diag(N_l, M_{init})$, soit les événements observables de N_l. On a donc $E_d = E_o$.

Les arcs sont définis au sein de $Diag(N_l, M_{init})$ par la fonction transition $f_d : (X_d, E_d) \to E_d$ avec $f_d(x_d^1, e_o) = x_d^2$ si x_d^1 et $x_d^2 \in X_d$ sont connectés par un arc labellisé par $e_o \in E_d$.

Enfin, l'état initial de $Diag(N_l, M_{init})$ est noté $x_0^d \in X_d$. ◇

A partir d'un tel diagnostiqueur $Diag(N_l, M_{init})$, la diagnosabilité d'une faute dans le système modélisé par N_l est alors définie de la manière suivante. Cette diagnosabilité permet de déterminer si la faute est systématiquement détectable ou non par le diagnostiqueur [10]. Dans le cas des attaques de blocage, cela revient à déterminer si une telle attaque sera systématiquement détectée et correctement identifiée ou non par notre module de diagnostic.

Définition 3.3.5 (Diagnosabilité)**.** La diagnosabilité d'une faute ou d'une attaque est définie comme l'existence d'une taille finie $n \in \mathbb{R}$ des séquences d'événements observables par le module de diagnostic à partir d'un état $x_d \in X_d$ pour laquelle la présence de la faute ou de l'attaque est diagnostiquée à partir de n'importe quelle séquence $\lambda_o \in E_d^\star, |\lambda_o| \geq n$ ◇

Dans nos travaux, trois diagnostiqueurs sont construits. Le premier (3.3.2) a pour objectif le diagnostic des attaques de blocage et le deuxième (3.3.2) le diagnostic des différents profils d'attaquant. Finalement, le troisième (3.3.2) est la composition des deux premiers. L'élaboration des deux premiers diagnostiqueurs est fondée sur l'algorithme proposé par [10] dont la mission est la construction de l'automate $Diag(N_l, M_{init})$ à partir de l'exploration du modèle d'attaque N_α selon les événements observables et non-observables par le module de diagnostic. Le pseudo-code de l'algorithme général est proposé dans l'annexe 6.7 et cet algorithme est illustré par le diagramme d'activité 3.16 ci-contre.

Dans cet algorithme, la construction du diagnostiqueur repose sur l'initialisation de x_0^d depuis M_{init} lors des étapes 1 à 5 et est poursuivie tant que de nouveaux états $x_d \in X_d$ sont découverts (étape 6). Lors d'une itération de la boucle d'exploration débutée à l'étape 7, chaque état découvert jusqu'alors $x_d \in X_d$ et chaque événement observable $e \in E_o$ sont considérés. Ainsi, à l'étape 8, pour un état x_d et un événement observable e, l'ensemble $\mathcal{M}_e$ des marquages atteignables depuis les paires $(M_1, \Delta_1) \in x_d$ suite à l'occurrence de e est préalablement calculé. Le label Δ_1 est aussi associé aux marquages $M_e \in \mathcal{M}_e$ puisqu'aucun événement inobservable de faute n'appartient à e. Ce label est noté $\Delta_e = \Delta_1$. Puis, au sein de l'étape 8, l'ensemble $UR(\mathcal{M}_e)$ des marquages atteignables depuis un état $M_e \in \mathcal{M}_e$ après l'occurrence d'une séquence non observable $\lambda_{io} \in E_{io}^\star$ est construit. Les ensembles UR et $\mathcal{M}_e$ sont détaillés mathématiquement dans l'annexe 6.7. Chaque marquage de $UR(\mathcal{M}_e)$ atteint après une séquence $\lambda_{io} \in E_{io}^\star$ depuis un état $M_e \in \mathcal{M}_e$ est alors labellisé lors de l'étape 9 et ajouté au nouvel état atteint x_d^{new}. Au sein de cette labellisation, quatre cas se présentent :

a) Si le label de M_e contient des fautes, soit $\Delta_e \in \Delta_f$, et des fautes associées au label Δ_2 ont lieu dans λ_{io}, alors la paire $(M, \Delta_e \cup \Delta_2)$ est ajoutée à x_d^{new} ;

b) Si le label de M_e contient des fautes, soit $\Delta_e \in \Delta_f$, et aucune faute n'a lieu dans λ_{io}, alors la paire (M, Δ_e) est ajoutée à x_d^{new}.

c) Si le label de M_e est O et des fautes associées au label Δ_2 ont lieu dans λ_{io}, alors la paire (M, Δ_2) est ajoutée à x_d^{new}

d) Si le label de M_e est O et aucune faute n'a lieu dans λ_{io}, alors la paire (M, O) est ajoutée à x_d^{new}.

Enfin, à l'étape 11, x_d^{new} est ajouté à X_d s'il ne lui appartient pas encore et un arc entre x_d et x_d^{new} labellisé par e est créé à l'étape 12 du diagramme d'activité.

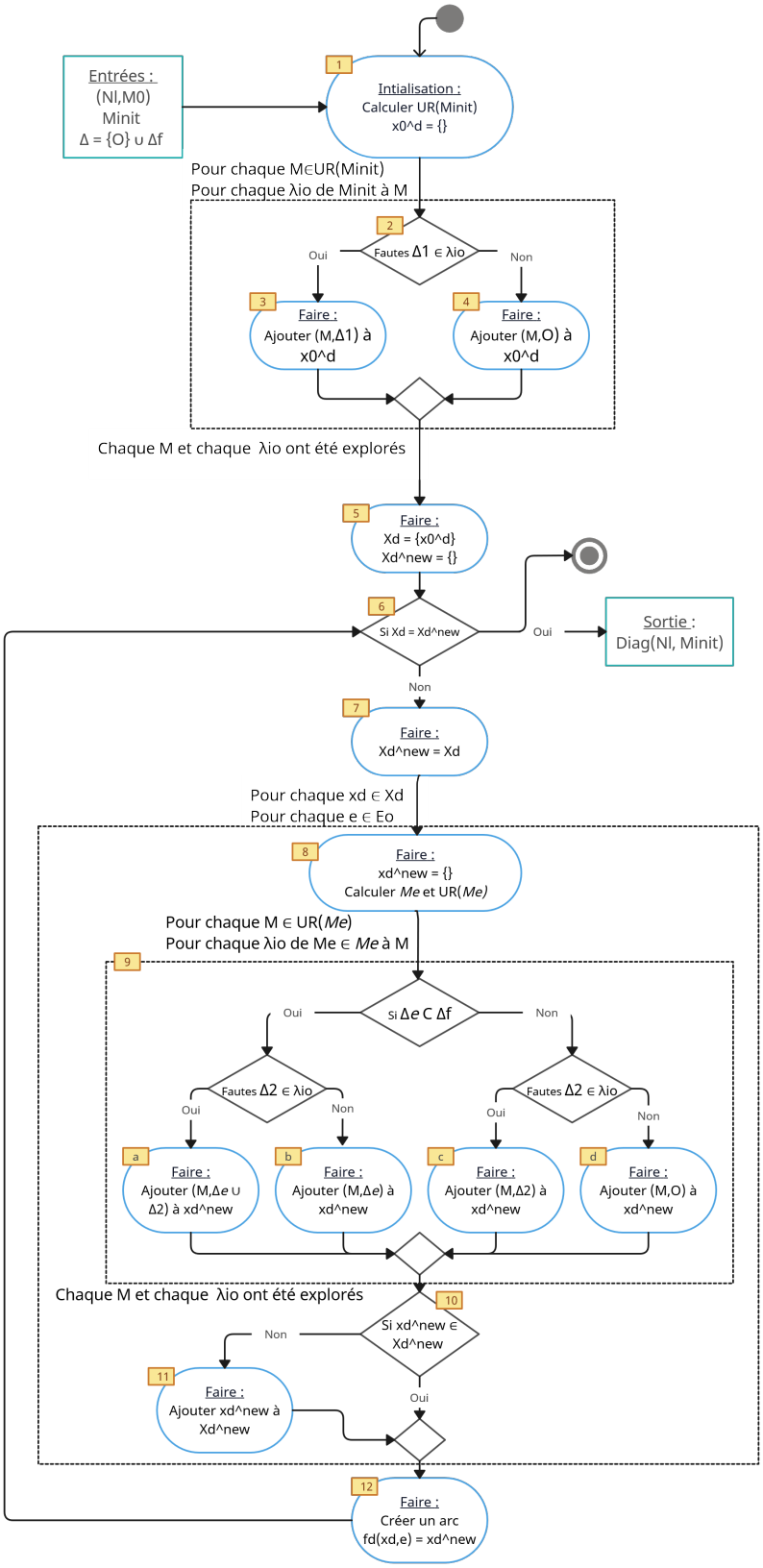


FIGURE 3.16 – Diagramme d'activités UML de l'algorithme de construction de $Diag(N, M_{init})$

Cependant, pour un RdP N_l possédant un grand nombre d'états atteignables $\mathcal{R}(N_l, M_{init})$, la construction complète d'un diagnostiqueur entraîne une explosion du nombre de ses états dans X_d [10]. Face à cette problématique, les différents diagnostiqueurs de notre méthode de diagnostic sont construits de manière partielle grâce aux trois contraintes ci-après :

- Premièrement, l'exploration du diagnostiqueur est stoppée lorsque un état $x_d \in X_d$ ne contenant que des labels relatifs aux attaques et aux profils est atteint. En effet, l'objectif du diagnostiqueur est alors rempli : une attaque et/ou des profils sont diagnostiqués ;
- Deuxièmement, le diagnostiqueur est exploré uniquement pour les états du FMS traversés par son ordonnancement. Cette contrainte concerne le diagnostiqueur des profils d'attaquant pour lequel les profils ne sont calculés que depuis ces états normalement traversés par le FMS. En effet, un attaquant ne lance son attaque que depuis un état atteint grâce à l'ordonnancement.
- Troisièmement, puisque l'observabilité du diagnostiqueur via les signaux bas-niveau est réduite à l'état de G, l'espace des marquages explorés Me et $UR(Me)$ est restreint aux places de G soit à $P^{G\alpha} = P_A^{G\alpha} \cup P_R^{G\alpha} \cup P_G^0$ dans N_α avec $P_G^0 = P_G^e \cup P_G^s$. Cette contrainte pourra permettre de diminuer le nombre d'états explorés.

A partir de ces trois contraintes de construction d'un diagnostiqueur partiel et des conditions de ré-ordonnancement des modules de supervision et de diagnostic, le fonctionnement global de la méthode de diagnostic des attaques de blocage et des profils d'attaquant dans un contexte certain est illustré dans le diagramme d'activité 3.17 ci-contre. Ce diagramme est similaire à celui exposé en introduction de cette partie (figure 3.14) au sein duquel les entrées de la méthode de diagnostic ont été précisées. Cette méthode peut être résumée par le flux d'étapes suivant :

1) Initialisation du module de diagnostic :
   (a) Le module de supervision et le module de diagnostic sont paramétrés identiquement : méthode de supervision des états de blocage identique (3.2.1), méthode d'ordonnancement identique (3.2.2), même condition de ré-ordonnancement Cn_d et même méthode de réduction du marquage des places d'entrées (3.3.1) ;
   (b) Le module de diagnostic est initialisé en parallèle du FMS. Il connaît l'état initial M_0^t (modèle $CS - net$) de ce dernier et le nombre de produits présents en entrée des différentes recettes $(M_0^t(p_e), p_e \in P_E)$;
   (c) Le marquage initial des places d'entrée est réduit d'après l'équation 3.2. Un marquage réduit M_{init} est obtenu. Le temps restant est défini par $R_{init} = 0_{|P_t|\times 1}$;
2) L'ordonnancement σ_f est calculé à partir de l'algorithme 0 en prenant (M_{init}, R_{init}) comme état de début et la terminaison de tous les produits comme état objectif.
3) Le premier diagnostiqueur est calculé (3.3.2) pour la détection d'attaques de blocage ;
4) Le second diagnostiqueur est calculé (3.3.2) pour le diagnostic des trois profils d'attaquant ;
5) Le diagnostiqueur commun fusion des deux précédents diagnostiqueur est construit (3.3.2) ;
6) A chaque nouvelle décision d'allocation au sein du FMS (interprétée depuis les signaux bas-niveau) :
   (a) Mise à jour des modèles N_t (pour le calcul de l'ordonnancement) et N_α (pour le calcul des profils d'attaquant) ;

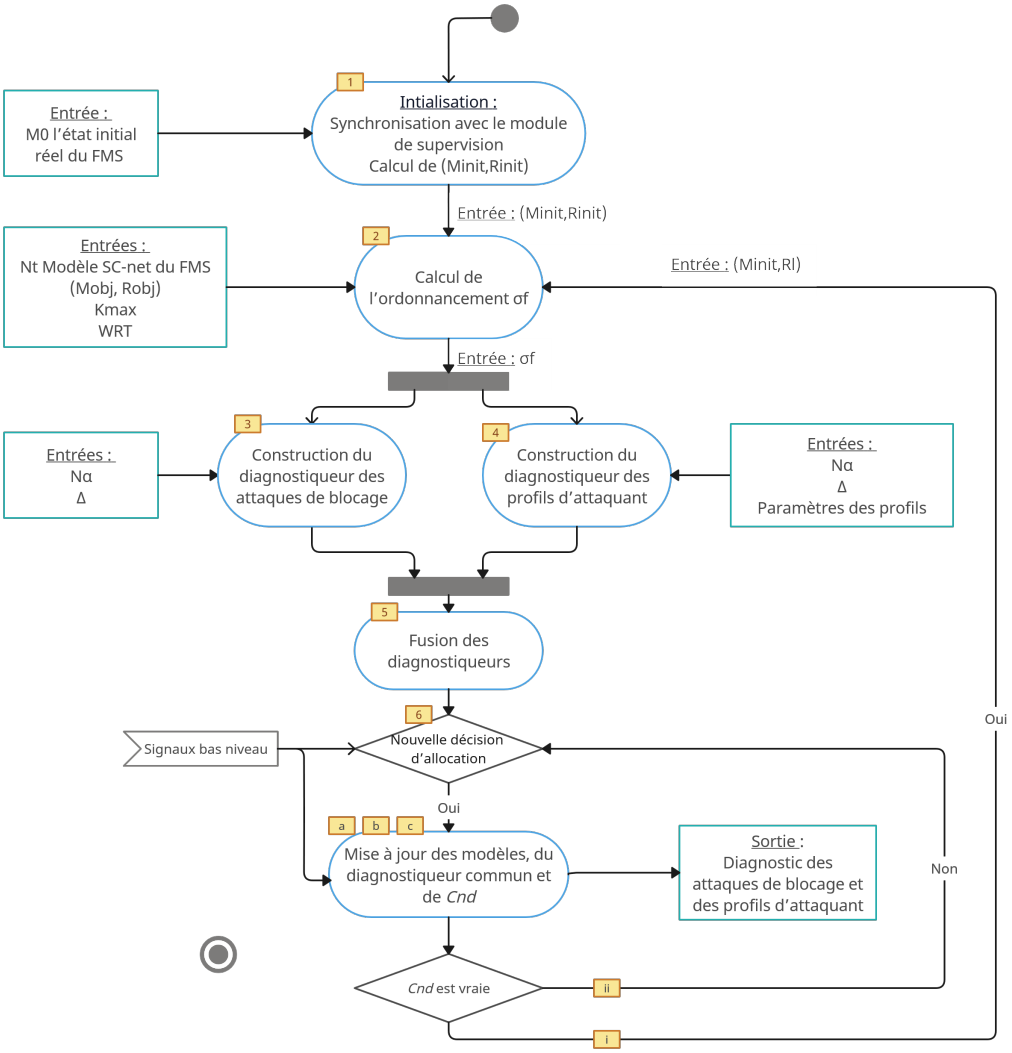


FIGURE 3.17 – Diagramme d'activités UML de la méthode de diagnostic dans un contexte certain

(b) Mise à jour de l'état actuel du diagnostiqueur commun. En continu, les opérateurs sont informés de son état, notamment sur les profils d'attaquant potentiellement en présence. Si une attaque est détectée, les opérateurs sont alertés mais le module de diagnostic est maintenu actif pour permettre d'établir le diagnostic d'un ou plusieurs profils d'attaquant et d'enrichir ainsi les informations de détection transmises ;

(c) La condition Cn_d est calculée en prenant en compte les nouveaux produits en entrée des recettes :

i. Si Cn_d est vraie, l'état (M_{init}, R_{rl}) est calculé à partir de l'équation de réduction 3.2 et le processus retourne à l'étape 2. en prenant $(M_{init}, R_{init}) = (M_{init}, R_{rl})$ comme état initial de l'ordonnancement ;

ii. Sinon, le processus retourne à l'étape 6 dans l'attente d'une nouvelle décision d'allocation.

Dans les prochaines parties, les étapes 2, 3 et 4, à savoir la construction des diagnostiqueurs des attaques de blocage et des profils d'attaquant ainsi que leur fusion au sein d'un diagnostiqueur commun, sont détaillées. Ces étapes prennent en compte les contraintes de partiellité imposées aux diagnostiqueurs et modifient en conséquence l'algorithme 3.16. En particulier, la labellisation des états au sein de cet algorithme (étape 9) est adaptée aux événements d'attaque et de profils que l'on souhaite diagnostiquer.

Diagnostic des attaques de blocage

Soit $Diag_a(N_\alpha, M^\alpha_{init}) = (X^a_d, E^a_d, f^a_d, x^a_0)$ le diagnostiqueur construit pour la détection des attaques de blocage (étape 3 de la figure 3.17). L'ensemble des labels définis pour $Diag_a$ est noté $\Delta_a = \{O, \delta_{A1}, \delta_{A2}\}$ avec O le label symbolisant un état optimale du FMS, δ_{A1} un état attaqué lorsque le système suit une trajectoire non optimal et δ_{A2} un état attaqué lorsqu'un état de la DZ a été atteint. Il est clair que le label δ_{A2} inclut le label δ_{A1}.

L'ensemble des événements observables par le module de diagnostic sont les événements $E^a_d = E^\alpha_o = E^\alpha \cup E^\alpha_+$, autrement dit les événements $e^\alpha(O_i, R_k)$ relatifs aux décision d'allocation opérées par le module de supervision débutant l'opération O_i et libérant la ressource R_k et les événements $e^\alpha(O_i, R_k)_+$ symbolisant l'insertion d'une décision d'allocation par un attaquant. Par conséquent, les événements non observables par le module de diagnostic sont les événements $e^\alpha(O_i, R_k)_-$ appartenant à E^α_- et représentant la suppression par un attaquant d'une décision d'allocation. Cependant, la méthode de construction de $Diag_a$ diffère de celle proposée dans l'algorithme 3.16 selon trois aspects : le calcul de $\mathcal{M}_e$, le calcul de $UR(\mathcal{M}_e)$ et l'attribution des labels aux états de $UR(\mathcal{M}_e)$.

Premièrement, pour le calcul de $\mathcal{M}_e$, le module de diagnostic ne fait pas la différence entre les deux événements $e^\alpha(O_i, R_k)$ et $e^\alpha(O_i, R_k)_+$ qu'il observe puisque $f^1_a(e^\alpha(O_i, R_k)) = f^1_a(e^\alpha(O_i, R_k)_+) = e^\alpha(O_i, R_k)$. Soit $x_d \in X^a_d$ un état déjà exploré et $e \in E^\alpha$ un événement non attaqué de N_α observé par le diagnostiqueur. On définit alors $\mathcal{M}_e = \{M \in \mathcal{R}(N_\alpha, M^\alpha_0) | \exists (M_1, \Delta_1) \in x_d, \Delta_1 \subset \Delta_a, t \in (l_\alpha)^{-1}((f^1_a)^{-1}(e)), M_1[t > M\}$ que l'on projette ensuite sur P^{G^α} afin d'obtenir l'ensemble $\mathcal{M}_e = \mathcal{M}_e|_{P^{G^\alpha}}$. Cette projection assure le respect de la troisième contrainte de construction d'un diagnostiqueur partiel.

Deuxièmement, l'ensemble $UR(\mathcal{M}_e)$ est égal à $\mathcal{M}_e$ pour $Diag_a$ car ce dernier ne cherche qu'à détecter les attaques conduisant le système physique, à savoir l'allocation des ressources, dans un état hors de l'ordonnancement optimal et/ou dans un état de la DZ. Or, les effets des événements d'attaque inobservables sont nuls sur G puisque $f^1_a(e^\alpha(O_i, R_k)_-) = f^1_a(e_{deb}(O_i)_-) f^1_a(e_{lib}(R_k)_-) f^1_a(e_{fin}(O_i)_+) f^1_a(e_{dis}(R_k)_+) = \{\varepsilon\}\{\varepsilon\}\{\varepsilon\}\{\varepsilon\}$.

Troisièmement, l'attribution des labels aux marquages dans $Diag_a$ diffère de la méthode présentée dans le diagramme 3.16 car les labels de Δ_a ne sont pas affiliés à des événements non-observables de N_α mais à un événement observable n'appartenant pas à l'ordonnancement pour δ_{A1} et à l'appartenance de $M_e \in \mathcal{M}_e$ à la DZ pour δ_{A2}. Cette labellisation est une altération de l'étape 9 du diagramme 3.16 et est illustrée dans la figure 3.18 ci-contre. Le pseudo-code de l'algorithme de construction de $Diag_a$ est présenté dans sa totalité dans l'annexe 6.8. Dans la figure 3.18, M_1 est l'état précédant M_e et est labellisé par Δ_1. La première contrainte du diagnostiqueur partiel est appliquée via cette labellisation puisque si $\Delta_1 \neq \{O\}$, à avoir si M_1 n'appartient pas à la trajectoire optimale ou appartient à la DZ, alors une attaque a déjà été

diagnostiquée en M_1 et l'exploration du diagnostiqueur $Diag_a$ s'arrête, i.e. aucune action n'est réalisée par l'algorithme dans ce cas-ci.

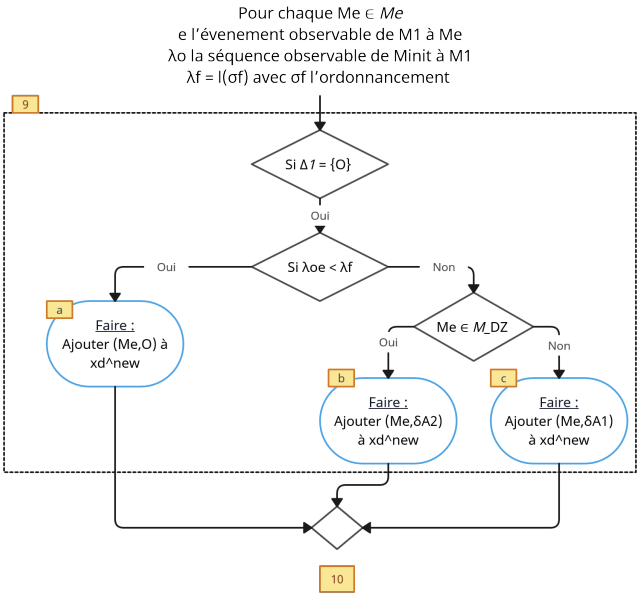


FIGURE 3.18 – Diagramme d'activités UML de la labellisation des états dans $Diag_a$

Les attaques de blocages sont toutes diagnosticables par le diagnostiqueur $Diag_a$ puisqu'elle seront au plus tard détecter lorsqu'un état de $\mathcal{M}_{DZ}$ est atteint par le FMS.

Remarque 3.3.1. Le diagnostiqueur $Diag_a$ n'est pas construit pour détecter le non respect des contraintes temporelles imposées dans N_α par les temps des opérations et les temps de libération des ressources. Des références de méthodes développant une telle détection sont présentées dans [355]. Par conséquent, lors de la construction de $\mathcal{M}_e$, le franchissement des transitions n'est pas conditionné par les délais restants R et les marquages sont ainsi définis par M dans X_d et non par (M, R). ⌟

Diagnostic des profils d'attaquant

Soit $Diag_{\mathcal{P}}(N_\alpha, M^\alpha_{init}) = (X^{\mathcal{P}}_d, E^{\mathcal{P}}_d, f^{\mathcal{P}}_d, x^{\mathcal{P}}_o)$ le diagnostiqueur construit pour le diagnostic des profils d'attaquant (sous-parties 3.1.1, 3.2.3) (étape 4 de la figure 3.17). L'ensemble des labels définis pour $Diag_{\mathcal{P}}$ est noté $\Delta_{\mathcal{P}} = \{O_{\mathcal{P}}, \delta_{\mathcal{P}_1}, \delta_{\mathcal{P}_2}, \delta_{\mathcal{P}_3}\}$ avec $O_{\mathcal{P}}$ le label symbolisant un état ne correspondant à aucun profil d'attaquant, et $\delta_{\mathcal{P}_1}, \delta_{\mathcal{P}_2}, \delta_{\mathcal{P}_3}$ les labels indiquant que l'état est traversé par le profil d'attaquant correspondant. Pour la construction de $Diag_{\mathcal{P}}$, la définition des ensembles des événements observables et non observables par le module de diagnostic, ainsi que la construction de $\mathcal{M}_e$ et $UR(\mathcal{M}_e)$ sont identiques à celles de $Diag_a$. Deux différences notoires existent cependant : l'état initial et l'assignation des labels.

Premièrement, l'état initial de $Diag_{\mathcal{P}}$ est défini par $x^{\mathcal{P}}_o = (M^\alpha_{init}, \{\delta_{\mathcal{P}_1}, \delta_{\mathcal{P}_2}, \delta_{\mathcal{P}_3}\})$. Cet état initial signifie que les trois profils d'attaquant débutent leur attaque à partir du marquage M_{init}. S'ils décident de débuter leurs attaques depuis un autre marquage M traversé par la trajectoire optimale σ_f, le diagnostiqueur $Diag_{\mathcal{P}}(N^\alpha, M)$ devra alors être construit en utilisant la même méthode. Il apparaît donc que les diagnostiqueurs des attaques de blocage et des profils

d'attaquant n'ont pas la même dynamique, le premier ne devant être reconstruit que lorsqu'un nouvel ordonnancement est calculé tandis que le second doit l'être dès qu'un marquage à partir duquel une attaque peut-être lancée est atteint, soit dans notre cas tous les états traversés par la trajectoire d'ordonnancement. Cette problématique correspond à la deuxième contrainte de partialité du diagnostiqueur. Elle est traitée dans la prochaine sous-partie (3.3.2).

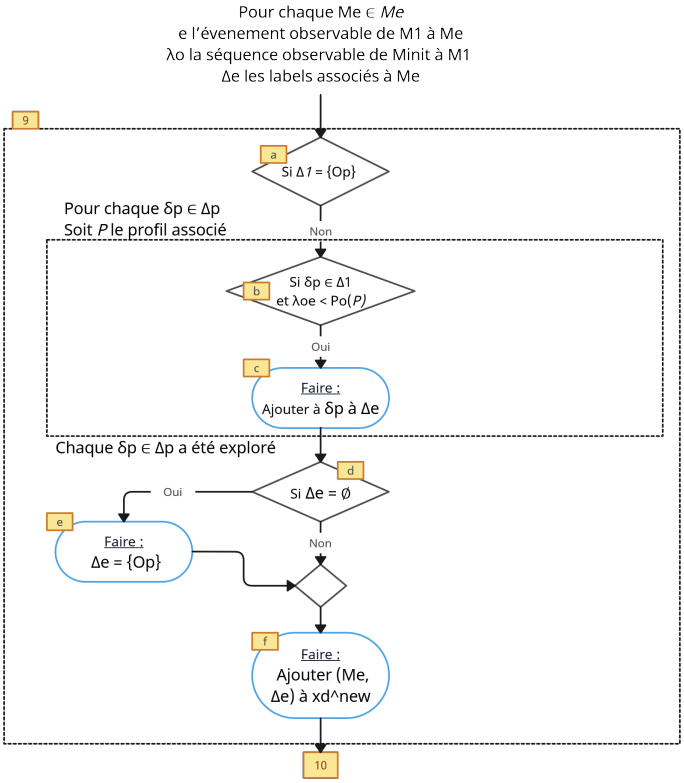


FIGURE 3.19 – Diagramme d'activités UML de la labellisation des états dans $Diag_p$

Deuxièmement, l'attribution des labels aux marquages dans $Diag_{\mathcal{P}}$ est fondée sur une propagation inverse à celle des deux méthodes précédentes et est illustrée par le diagramme d'activité UML de la figure 3.19 ci-contre. Ainsi, le marquage initial débute avec les 3 trois labels correspondant aux trois profils d'attaquant et perd l'un de ses labels dès que le FMS atteint un marquage non traversé par le profil associé à ce label (étapes b et c). Pour rappel, $P_o(\mathcal{P})$ désigne la projection d'un profil $\mathcal{P}$ sur l'ensemble des événements observables. Lorsqu'un marquage n'est plus labellisé par aucun label d'un profil d'attaquant, il se voit assigner le label $O_{\mathcal{P}}$ (étapes d et e). Par ailleurs, dans cette étape de labellisation, la première contrainte du diagnostiqueur partiel est appliquée à la sous-étape a) lorsqu'un état M_e précédé par un marquage M_1 associé à aucun profil, i.e. $\delta_1 = O_{\mathcal{P}}$, n'est pas exploré par l'algorithme. Enfin, notons que dans cet algorithme de construction de $Diag_{\mathcal{P}}$, les profils d'attaquant sont calculés préliminairement et considérés comme des entrées sous la forme de séquences d'évènements $\mathcal{P}$.

Parmi les trois profils d'attaquant, seul le profil 3 est diagnosticable puisqu'il est le seul à cibler les états de blocage partiel de $\mathcal{M}_{LL}$. Les profils 1 et 2 ne sont pas théoriquement diagnosticables l'un par rapport à l'autre puisqu'ils peuvent conduire tous les deux à un même

état de blocage dans $\mathcal{M}_{c_1} \subset \mathcal{M}_{c_2}$. Cependant, dans le cas où l'état ciblé est identique entre les deux profils, ils peuvent en pratique être diagnosticables l'un de l'autre si $\mathcal{P}_2 \not< \mathcal{P}_1$.

Diagnostic commun

A partir des deux diagnostiqueurs $Diag_a$ et $Diag_{\mathcal{P}}$, un diagnostiqueur commun, noté $Diag_c$, peut être construit (étape 5 de la figure 3.17). La composition de deux diagnostiqueurs construits à partir du même modèle de RdP, ici N_α, est détaillée mathématiquement dans l'annexe 6.10 et est employée à la construction de $Diag_c$. Cette opération de composition, notée $\circ$, consiste à fusionner les états de $Diag_a$ et de $Diag_{\mathcal{P}}$ atteints après une même séquence d'événements observables depuis un état initial x_0^c, lui aussi obtenu par la fusion de x_0^a et $x_0^{\mathcal{P}}$. L'opération de fusion de deux états x_1 et x_2, notée $\otimes$, se traduit par l'agglomération des labels associés au même marquage M dans x_1 et x_2. Ainsi, puisque le marquage initial est le même dans $Diag_a$ et $Diag_c$, soit M_{init}, l'état initial x_0^c consiste à la paire (M_{init}, Δ_0^c) avec Δ_0^c la fusion des labels de x_0^a et de $x_0^{\mathcal{P}}$.

La méthode de construction d'un diagnostiqueur par composition peut être appliquée à la construction de $Diag_c$ pour un marquage initial $M_{init} \in \mathcal{R}(N_\alpha, M_0^\alpha)$ et à partir des deux diagnostiqueurs $Diag_a(N_\alpha, M_{init}^\alpha)$ et $Diag_{\mathcal{P}}(N_\alpha, M_{init}^\alpha)$. Cependant, le diagnostiqueur $Diag_c(N_\alpha, M_{init}^\alpha)$ ne peut pas être construit de manière complètement hors-ligne par une simple composition de $Diag_a$ et $Diag_{\mathcal{P}}$. Trois problèmes émergent :

- (i) Premièrement, la construction de $Diag_c$ est dépendante de l'ordonnancement σ_f à travers $Diag_a$ et le calcul des profils d'attaquant. Or σ_f est recalculée en ligne par la méthode d'ordonnancement lorsque la condition Cn_d est vérifiée (3.3.1). Ainsi, dès que Cn_d est vraie, $Diag_c$ doit aussi être reconstruit sur l'horizon du nouvel ordonnancement. **On note σ_f^i, $Diag_a^i$ et $Diag_c^i$ les $i^{ième}$ ordonnancement et diagnostiqueurs obtenus depuis l'initialisation du module de diagnostic**.
- (ii) Deuxièmement, un diagnostiqeur $Diag_{\mathcal{P}}(N_\alpha, M')$ doit pour sa part être reconstruit dès qu'un nouvel état (M', R') est atteint par l'ordonnancement car les profils d'attaquant évoluent lorsqu'ils sont débutés depuis deux états différents du FMS. Par conséquent, pour chaque marquage M' traversé par N_α lors de l'ordonnancement σ_f^i, un diagnostiqueur $Diag_{\mathcal{P}}(N_\alpha, M')$ doit être construit et intégré à $Diag_c$ (deuxième contrainte de construction d'un diagnostiqueur partiel).
- (iii) Troisièmement, lorsqu'un ré-ordonnancement a lieu, le diagnostiqueur $diag_c^{i-1}$ est supprimé au profit d'un nouveau diagnostiqueur $diag_c^i$. Toutefois, au sein de x_c^{i-1} l'état de $diag_c^{i-1}$ lors du ré-ordonnancement, certains labels $\delta_{\mathcal{P}}$ peuvent encore y appartenir, signifiant que des profils d'attaquant sont encore en cours de diagnostic. Le diagnostic de ces profils doit être conservé depuis $Diag_c^{i-1}$ vers $Diag_c^i$ afin de ne perdre aucune information de diagnostic lors d'une reconfiguration en ligne du module de diagnostic.

Ainsi, $Diag_c$ est un diagnostiqueur dynamique dont la validité est limitée par σ_f et devant être reconstruit dès que la condition de ré-ordonnancement Cn_d est vérifiée. La méthode de construction d'une itération de ce diagnostiqueur commun $Diag_c^i$ est présentée dans le diagramme d'activité UML 3.20 et son pseudo-code est détaillé dans l'annexe 6.11.

Dans ce diagramme, une première composition $Diag_c^i = Diag_a \circ Diag_c^{i-1}|_{\Delta_{\mathcal{P}}}$ est réalisée lors de l'étape 1 grâce à l'appel à l'algorithme 3.18. La notation $Diag_c^{i-1}|_{\Delta_{\mathcal{P}}}$ désigne la restriction des

états du diagnostiqueur précédent $Diag_c^{i-1}$ aux pairs contenant des labels de profils d'attaquant. Cette restriction est implémentée dans l'algorithme 3.20 puisque seules ces paires sont requises pour le diagnostic dans $Diag_c^i$ de profils d'attaquants hérités de $Diag_c^{i-1}$. Puis pour chaque état (M_k, R_k) traversé par l'ordonnancement σ_f^i dans N_α, l'état x_k correspondant à M_k dans $Diag_c^i$, i.e. $M_{init}[\sigma > M_k, l(\sigma) = \lambda$ et $f(x_0^{c_i}, \lambda) = x_k$ avec $x_0^{c_i}$ l'état initial de $Diag_c^i$, ainsi que les profils $\mathcal{P}_1(M_k, R_k)$, $\mathcal{P}_2(M_k, R_k)$ et $\mathcal{P}_3(M_k, R_k)$ sont calculés dans les étapes 3 et 4. Enfin, à partir de x_k et des trois profils, la composition $Diag_c^i = Diag_c^i \circ Diag_{\mathcal{P}}(N_\alpha, \chi_{c,\mathcal{P}_k}(x_k))$ est réalisée à l'étape 5 où la fonction $\chi_{c,\mathcal{P}_k}$ définie dans l'annexe 6.11 permet de calculer l'état initial de $Diag_{\mathcal{P}_k}$ à partir de x_k. Cette boucle explorant tous les états (M_k, R_k) de l'ordonnancement σ_f^i permet le respect de la deuxième contrainte de construction d'un diagnostiqueur partiel, à savoir que seuls les états traversés par l'ordonnancement sont considérés pour le calcul des profils d'attaquant et la construction des $Diag_{\mathcal{P}_k}$. Une fois cette boucle terminée, l'algorithme retourne le diagnostiqueur $Diag_c^i$.

Notons que, pour chaque état (M_k, R_k) traversé par l'ordonnancement σ_f^i, les labels du diagnostiqueur des profils d'attaquant $Diag_{\mathcal{P}_k}$ sont $\Delta_{\mathcal{P}_k} = \{O_{M_k}, \delta_{\mathcal{P}_1}^{M_k}, \delta_{\mathcal{P}_2}^{M_k}, \delta_{\mathcal{P}_3}^{M_k}\}$, et l'ensemble de tous les labels des différents $Diag_{\mathcal{P}_k}, k \in [1, |\sigma_f^i|]$ est noté $\Delta_{\mathcal{P}} = \bigcup \Delta_{\mathcal{P}_k}$.

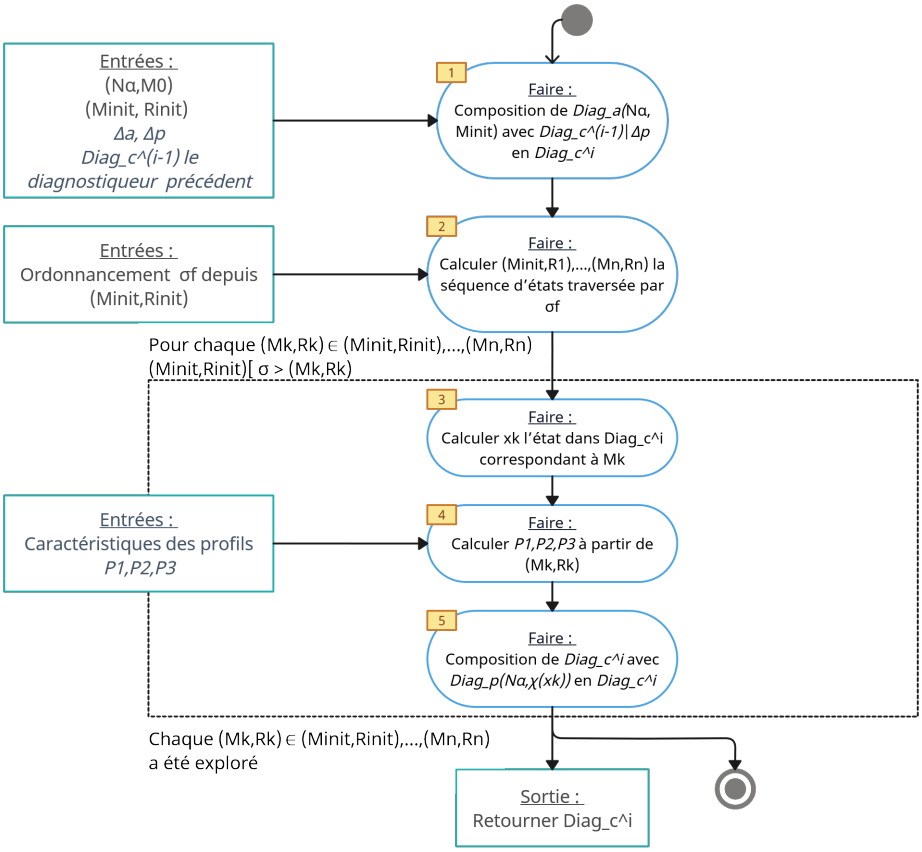


FIGURE 3.20 – Diagramme d'activités UML de l'algorithme de construction de $Diag_c^i$

Soit $Diag_c^i(N_\alpha, x_0^{c_i}) = (X_d^{c_i}, E_d^{c_i}, f_d^{c_i}, x_0^{c_i})$ le diagnostiqueur obtenu par la $i^{ème}$ exécution de l'algorithme ci-dessus lors de l'étape 5 du fonctionnement global du module de diagnostic (3.17). Ce diagnostiqueur est utilisé comme référentiel pour la détection et le diagnostic des attaques

de blocage contre le FMS et des profils d'attaquant tant qu'aucun nouveau ré-ordonnancement n'a lieu. Par conséquent, les différents états $x_d^c \in X_d^{c_i}$ de $Diag_c^i$, s'ils sont atteints en ligne par le FMS, peuvent être interprétés selon les différents cas présentés ci-contre :

a) Si les labels δ_{A1} ou δ_{A2} appartiennent à toutes les paires $(M', \Delta') \in x_d^c$, alors une attaque de blocage est diagnostiquée et un état de la DZ a été déjà atteint dans le second cas.

b) Si la proposition précédente est vraie et $\Delta' \cap \Delta_{\mathcal{P}} = \delta_i^{M_{k1}}, \delta_i^{M_{k2}}, ...$ avec $i \in \{1, 2, 3\}$ et $k1, k2, ... \in [1, |\sigma_f|]$, alors le profil d'attaquant i est diagnostiqué et tenu responsable de l'attaque détectée précédemment. Autrement dit, si un unique profil peut-être associé à toutes les paires d'un état x_d^c attaqué du diagnostiqueur, alors l'origine de l'attaque de blocage est associée à ce profil. De surcroît, si un unique label $\delta_i^{M_k}$ est associé à x_d^c, on sait avec certitude que le profil a été débuté au marquage M_k de l'ordonnancement.

c) A l'inverse, si la première proposition est vraie mais que $\Delta' \cap \Delta_{\mathcal{P}} = N_{M_{k1}}, N_{M_{k2}}, ...$, aucun label de profils d'attaquant ne peut être associé à l'état x_d^c, alors l'attaque reste détectée mais aucune origine ne peut lui être associée. Du point de vue du défenseur, ce cas peut paraître inattendu puisque cela signifie que l'attaquant n'est pas suffisamment préparé ou n'a pas d'objectif clair.

d) Si la première proposition est toujours vraie mais que les deux suivantes ne le sont pas, alors plusieurs profils d'attaquant peuvent être associés à x_d^c. Il n'est pas possible de conclure sur l'origine de l'attaque de blocage détectée.

e) Enfin, si la première proposition n'est pas vraie, cela signifie qu'aucune attaque de blocage n'est détectée même si x_d^c contient des labels des profils d'attaquant.

Dans la prochaine partie, le fonctionnement global du module de diagnostic est illustrée à travers un exemple complet reprenant toutes les contributions du chapitre 3.

3.3.3 Exemple et discussion

Exemple

Le processus de fonctionnement global du module de diagnostic (3.17) est illustré par un cas d'application sur l'exemple étudié précédemment (3.2.3). Les simulations de cet exemple sont réalisées sur le logiciel Matlab par un processeur quatre cœurs *Intel(R)Core(TM)i5-9400H*. Les conditions initiales de l'exemple précédent sont conservées et l'état initial (M_{init}, R_{init}) projeté aux états de $P^{G\alpha}$ est illustré dans le modèle $CS - net$ ci-contre (figure 3.21). Dans cette figure, R_{init} est représenté par les valeurs en bleu et M_{init} par les marquages des différentes places dont les places de contrôle. Au sein de N_α, les places de $P^{S\alpha}$ seraient marquées identiquement aux place de $P^{G\alpha}$ et incluraient les places de contrôle de P_c.

Dans cet exemple, on construit le diagnostiqueur $Diag_c^i = (X_d^{c_i}, F_d^{c_i}, f_d^{c_i}, x_0^{c_i})$ à partir de (M_{init}, R_{init}) et suivant l'ordonnancement réduit $\sigma_f = t_{15}t_{18}t_{20}t_{17}t_{19}t_{16}t_9t_{18}t_7t_{10}$. En simulant l'algorithme 3.20 à partir de ces conditions initiales et sur la fenêtre de σ_f on obtient un diagnostiqueur à 192 états dans $X_d^{c_i}$, à 192 arcs dans $F_d^{c_i}$ et à 47 labels correspondant aux trois labels de $Diag_a$ (O, δ_{A_1}, δ_{A_2}) et aux $Diag_{\mathcal{P}_k}$ des 11 marquages différents parcourus par le FMS lors de l'application de σ_f (quatre labels, un normal et trois profils, par état parcouru). Les premiers états de ce diagnostiqueur sont représentés dans la figure 3.22 ci-dessous. Dans cette figure, les états entourés en vert sont les états optimaux, ceux en noir les états non optimaux et

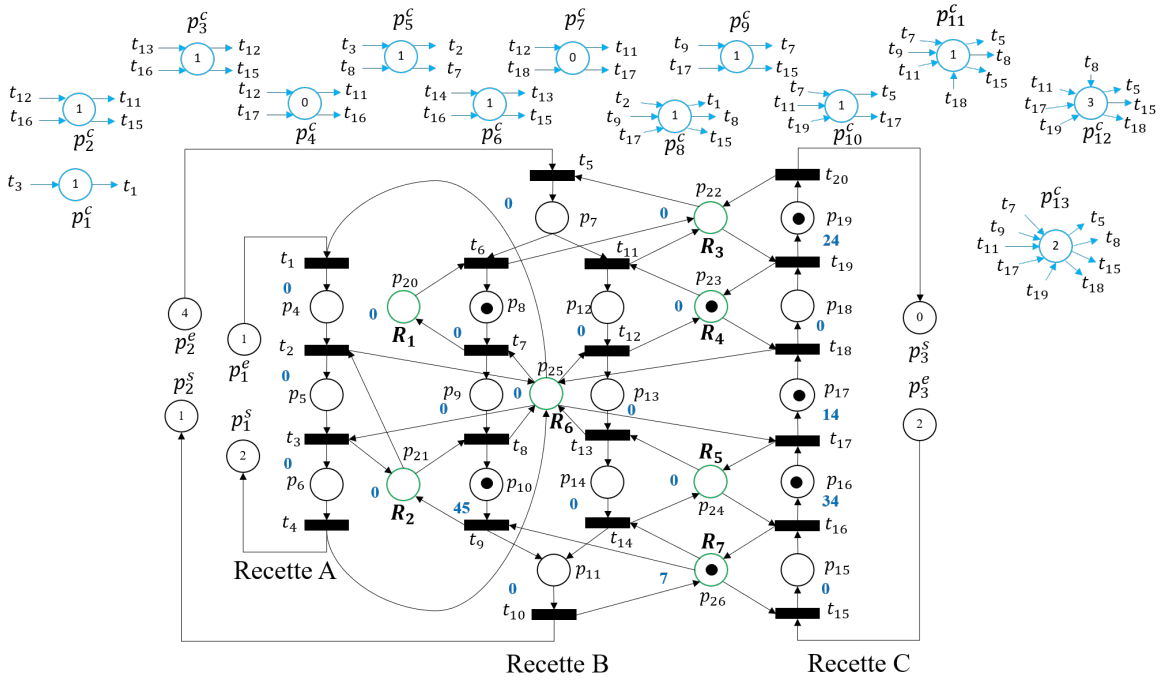


FIGURE 3.21 – État (M, R) pour la construction des diagnostiqueurs

non critiques, et ceux en rouge les états critiques appartenant à la DZ. Les nombres en gras autour des états sont les indices de ces états dans le diagnostiqueur $Diag_c^i$. Dans un état de $X_d^{c_i}$, le nombre à gauche désigne l'indice de l'état correspondant dans $\mathcal{R}(N, M_0)|_{(P_A \cup P_R)}$ par projection dans $P^{G\alpha} \setminus P^0$ et varie donc dans notre exemple entre 1 et 1650. Un indice peut se répéter dans plusieurs états du diagnostiqueur si la séquence pour y parvenir et les labels associés diffèrent. C'est par exemple le cas pour l'indice 1622 présent dans les états 15 et 17 de $Diag_c^i$.

Les arcs du diagnostiqueur sont labellisés par les transitions franchies dans N_α afin de visualiser plus rapidement les correspondances entre l'ordonnancement, les profils d'attaquant et le diagnostiqueur construit à partir de ces séquences de transitions. Néanmoins, seules les transitions appartenant à T sont représentées en raison de l'observabilité sur G du diagnostiqueur. Ainsi, à droite de la figure, le profil $\mathcal{P}_1(M_{init}, R_{init}) = t_{35}t_{58}t_{38}t_{41}t_{20}t_{45}t_{55}$ est modélisé à travers les états de $Diag_c^i$ contenant le label $\delta_1^{M_1}$ et la séquence de transitions $t_{18}t_1t_{20}t_5$. Cette séquence diffère de $\mathcal{P}_1(M_{init}, R_{init})$ puisque les transitions appartenant à T_-^α ne sont pas considérées (t_{21} à t_{40}) car non observables par le module de diagnostic et les transitions appartenant à T_+^α sont associées à leurs transitions dans T selon la fonction τ_α (t_{58} devient t_{18}). En supprimant t_{35} et t_{38} de $\mathcal{P}_1(M_{init}, R_{init})$ et en réalisant la conversion par τ_α, on obtient les même séquences $t_{18}t_1t_{20}t_5$ tandis que t_{15}, la transition nécessaire pour atteindre l'état de blocage ciblé par le profil, est omise de la figure pour des raisons de place. A gauche de cette figure, la trajectoire optimale du FMS est modélisée par le diagnostiqueur jusqu'à l'état (M_4, R_4) associé à l'état **20** de $Diag_c^i$. Enfin, on observe dans la figure que l'exploration du diagnostiqueur s'arrête comme souhaitée lorsqu'un état n'apportant plus d'information de diagnostic est atteint. Les états 7 à 11 de $Diag_c^i$ en font partie puisqu'ils correspondent à des états attaqués car non-optimaux (δ_{A1}) et à aucun profil d'attaquant (N_{M_1}).

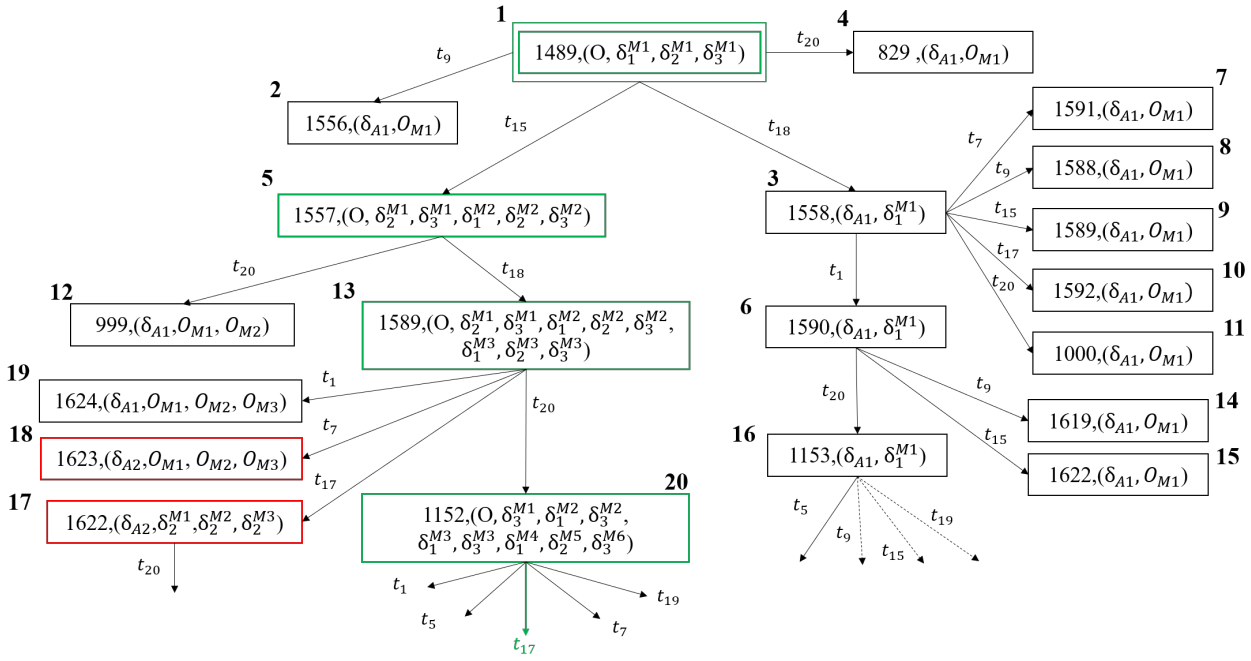


FIGURE 3.22 – Premiers états du diagnostiqueur $Diag_c^i$ depuis l'état (M_{init}, R_{ini})

On suppose ensuite que la séquence σ_f est correctement exécutée par le FMS et qu'un état (M_{11}, R_{11}) est atteint avec dans le $SC - net$ de la figure 3.21, $M_{11} = p_1^e + 2p_1^s + 4p_2^e + 2p_2^s + p_3^e + p_3^s + p_9 + p_{16} + p_{18} + p_{19} + p_{20} + p_{21} + p_{26}$ et $R(p_9) = 24$, $R(p_{16}) = 9$, $R(p_{18}) = 4$ et $R(p_{26}) = 7$. Dans cet état, de nouveaux produits arrivent en entrée des différentes recettes du FMS, à savoir 3 pour la recette A, 2 pour la recette B et 4 pour la recette C. On obtient alors le marquage réel M_{rl}^{11} tel que $M_{rl}^{11}(p_1^e) = 4$, $M_{rl}^{11}(p_2^e) = 6$ et $M_{rl}^{11}(p_3^e) = 5$. Afin de déterminer si un ré-ordonancement doit avoir lieu, la condition Cn_d est alors vérifiée selon ses trois sous-conditions.

- Premièrement, Cn_d^1 n'est pas vérifiée car l'ordonnancement n'est pas terminé au regard des produits en cours d'opération dans M_{11}.
- Deuxièmement, pour Cn_d^2 et les recettes B et C, on obtient $M_{rl}^{11}(p_3^e)/M_{rl}^{11}(p_2^e) = 6/5 = 1.2$, $M_{init}(p_3^e)/M_{init}(p_2^e) = 2/4 = 0.5$ et $M_{rl}^{11}(p_3^e)/M_{11}(p_3^e) = 5/1 = 5$. En choisissant $K_{cn_d}^1 = 2$ et $K_{cn_d}^2 = 1$, la condition Cn_d^2 est vérifiée puisque $1.2 \geq K_{cn_d}^1 \times 0.5$ et $5 \geq 1 + K_{cn_d}^2$.
- Troisièmement, pour Cn_d^3, on a $Mx(C) = 5$ et en choisissant $K_{cn_d}^3 = 2$ et $K_{cn_d}^4 = 3$, la condition est validée car $M_{11}(p_C^e) = 1 \leq K_{cn_d}^3 \times Mx(C)$ et $M_{rl}^{11}(p_C^e) = 5 \geq K_{cn_d}^4 \times M_{11}(p_C^e) = 3 \times 1$.

Les conditions 2 et 3 sont validées et reflètent d'une part une plus grande augmentation relative des produits en entrée de C par rapport à ceux en entrée de B et d'autre part un nombre réel de produits en entrée de C ($M_{rl}^{11}(p_C^e)$) plus élevé que le nombre estimé par simulation ($(M_{11}(p_C^e))$).

Par conséquent, Cn_d est vérifié et un ré-ordonnancement a lieu. En simulant l'algorithme $A^\star BT$ avec $(M_{rl}^{11}, R_{rl}^{11})$ et $K_{max} = 20$ en entrée d'après l'équation 3.2, on obtient un nouvel ordonnancement égal sur un horizon réduit à $\sigma_f^{i+1} = t_{15}t_{20}t_8t_{17}t_{16}t_5t_6t_{19}t_{18}t_{17}$. A partir de σ_f^{i+1} et de $(M_{rl}^{11}, R_{rl}^{11})$, un nouveau diagnostiqueur $Diag_c^{i+1}$ est construit sur l'horizon de σ_f^{i+1}. Ce diagnostiqueur possède 182 états et 181 arcs. Cependant, $Diag_c^{i+1}$ diffère de $Diag_c^i$ par le nombre

de labels qu'il contient puisqu'il intègre en son sain les profils d'attaquant de $Diag_c^i$ n'ayant pas été diagnostiqués lorsque le ré-ordonancement a lieu en (M_{11}, R_{11}) d'après l'étape 1 du diagramme 3.20.

Les premiers états de $Diag_c^{i+1}$ sont illustrés dans la figure 3.23. Dans cette figure, des labels de profils d'attaquant de $Diag_c^i$ sont conservés, à savoir $\delta_2^{M_7}, \delta_2^{M_8}, \delta_1^{M_9}, \delta_2^{M_9}, \delta_1^{M_{10}}, \delta_2^{M_{10}}, \delta_3^{M_{10}}$. Ces labels sont propagés vers l'avant dans les états de $Diag_c^{i+1}$ jusqu'à ce que les labels $N_{M_7}, N_{M_8}, N_{M_9}, N_{M_{10}}$ soient assignés à ces états ou que les états ciblés par les profils de ces labels soient atteints. Par exemple, pour le profil 2 débuté depuis les états M_7, M_8, M_9 et M_{10}, le diagnostic propage les labels associés $\delta_2^{M_7}, \delta_2^{M_8}, \delta_2^{M_9}, \delta_2^{M_{10}}$ à travers les états 1,2,7,19 de $Diag_c^{i+1}$ jusqu'à atteindre l'état de pré-blocage 1565. La conservation des profils d'attaquant non diagnostiqués par $Diag_c^i$ lorsque le ré-ordonnancement a lieu sont donc bien conservés dans $Diag_c^{i+1}$. Les nouveaux profils calculés à partir des états traversés par σ_f^{i+1} $(M_{11}, M_{12}, M_{13}, ...)$ sont ajoutés pour leur part dans chaque état du diagnostiqueur modélisant un état optimal du superviseur, encadrés en vert dans la figure. De surcroît, dans la figure 3.23, la différence entre les ordonnancements σ_f^i et σ_f^{i+1} est représentée à travers les labels O^i et O^{i+1} assignés à des états différents de $Diag_c^{i+1}$, soit les états (1,2,7,20) et (1,4,15,33) du diagnostiqueur.

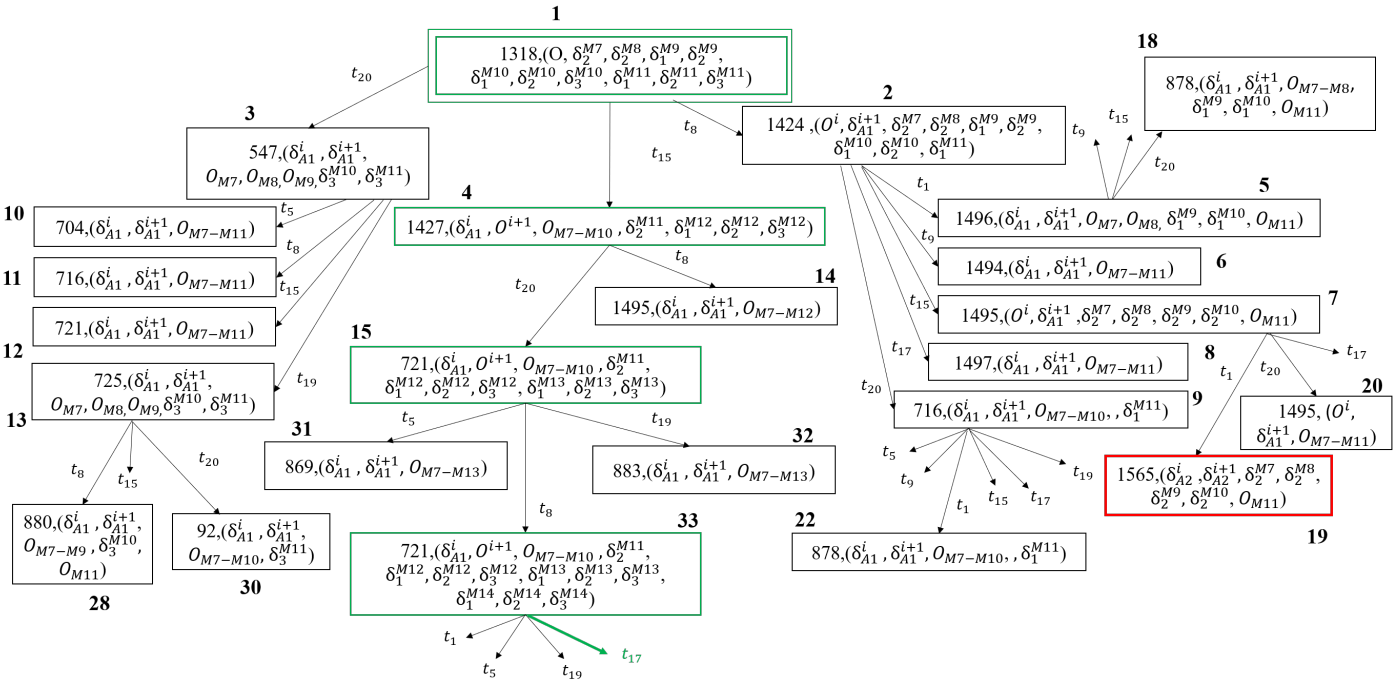


FIGURE 3.23 – Premiers états du diagnostiqueur $Diag_c^{i+1}$ depuis l'état (M_{11}, R_{11})

Dans $Diag_c^i$ et $Diag_c^{i+1}$, on étudie la diagnosabilité des différents profils d'attaquant selon le point 2 de l'interprétation des états de $Diag_c^i$, à savoir pour un profil $\mathcal{P}_i(M_{k_1}, R_{k_1})$ et son label associé $\delta_i^{M_{k_1}}$, $\mathcal{P}_i(M_{k_1}, R_{k_1})$ est diagnosticable s'il existe un état $x_c \in X_d^{c_i}$ tel que les seuls labels de profils assignés à x_c sont ceux du profil i, i.e. $\{\delta_i^{M_{k_1}}, \delta_i^{M_{k_2}}, ...\}$. D'après les résultats de simulation, tous les profils d'attaquant de $Diag_c^i$ et $Diag_c^{i+1}$ sont diagnosticables. Les labels d'attaques δ_{A1}^i et $\delta i + 1_{A2}$ sont eux par définition diagnosticables. Dans cet exemple, toutes les attaques de blocage et les profils d'attaquant associés sont détectés et diagnostiqués correctement

par le module de diagnostic proposé dans ce chapitre.

Discussion

Dans cet exemple, notre méthode de diagnostic des attaques de blocage et des profils d'attaquant et le fonctionnement en ligne du module de diagnostic ont été appliqués. En particulier, le rôle de la condition de ré-ordonnancement et la construction du diagnostiqueur commun lors de l'occurrence d'un ré-ordonnancement ont été exposés. Cet exemple a ainsi permis de valider certains objectifs attendus de la méthode de diagnostic :

- Les attaques de blocage sont toutes correctement détectées lorsqu'un état hors de l'ordonnancement est atteint (états noirs et rouges) ;
- Le diagnostic des profils d'attaquant permet d'enrichir le diagnostic d'une attaque de blocage. Dans la figure 3.22, les états 17 et 18 du diagnostiqueur conduisent tous deux à un état de la DZ et correspondent donc à des attaques δ_{A2}. Néanmoins, seul l'état 17 est labellisé par un profil d'attaquant et a par conséquent plus de chance de correspondre à une attaque par le profil 2 ;
- Le temps de calcul d'un ré-ordonnancement est égal à 1,1s pour $K_{max} = 20$. Il est donc inférieur à 10s (durée minimum d'une opération du FMS) et respecte la contrainte temporelle désirée.

Cependant, cet exemple met aussi en lumière un ensemble de limites de notre méthode qui devront être adressées dans de futurs travaux.

- Le temps de construction du diagnostiqueur de la figure 3.22 est égal à 1137s et de celui de la figure 3.23 à 174s. Ces temps sont élevés par rapport aux temps des opérations du FMS et résultent principalement du calcul des profils d'attaquant. Une des perspectives de nos travaux sera la réduction de ces temps en restreignant par exemple le diagnostiqueur à un ordonnancement partiel ou en améliorant les heuristiques pour le calcul des profils d'attaquant ;
- Le fonctionnement du module de diagnostic n'a pas été simulé dans un environnement dynamique au sein duquel des événements de ré-ordonnancement et des attaques ont lieu de manière aléatoire ;
- Les événements d'indisponibilité des ressources et des attaques contre ces événements n'ont pas été pris en compte. Un attaquant peut manipuler ces événements pour atteindre de nouveaux états de blocage ou maintenir sa sournoiserie. Cette limite fait l'objet du chapitre 4.

Conclusion

Dans ce chapitre, un module de diagnostic des attaques de blocage et des différents profils d'attaquant a été développé selon le diagramme d'activité présenté en introduction (3.1). Dans un premier temps, les modèles et méthodes nécessaires au fonctionnement de ce module ont été choisis et construits. Une méthode de prévention (3.2.1) et une méthode d'ordonnancement (3.2.2) de la littérature ont été choisies et appliquées afin de permettre la synchronisation de la supervision et du calcul de l'ordonnancement entre le module de supervision et notre

module de diagnostic. A partir de ces méthodes, de la définition d'un modèle des attaques de blocage (3.1.2,3.1.3) et de la sélection de trois profils d'attaquant (3.1.1), nous avons proposé un algorithme original de calcul des profils d'attaquant fondé sur la recherche $A^\star$ (3.2.3). Dans un second temps, la fonction de diagnostic de notre module est implémentée à partir de trois diagnostiqueurs originaux : un diagnostiqueur des attaques de blocage, un diagnostiqueur des profils d'attaquant et un diagnostiqueur commun fusionnant les deux précédents (3.3.2). Nous avons rendu le fonctionnement en ligne de ce module possible grâce à l'exploitation des données bas-niveau échangées entre les ressources et leurs contrôleurs locaux, à sa synchronisation avec l'état réel du FMS et à la prise en compte du ré-ordonnancement de celui-ci (3.3.1), induisant un re-développement des diagnostiqueurs. Enfin, notre méthode de diagnostic a été simulée et analysée par son application sur un exemple (3.3.3).

Cependant, toutes les étapes de développement du module de diagnostic présentées dans ce chapitre sont définies dans un contexte certain. Or, il a été exposé dans la partie 1.2.2 que les FMSs évoluent dans un environnement incertain au sein duquel les ressources peuvent devenir indisponibles en raison d'une défaillance, d'une opération de maintenance ou d'un acte malveillant. Dans le chapitre 4, le module de diagnostic sera étendu à ce contexte incertain des FMSs à l'aide de nouveaux modèles des attaques de blocage et des comportements optimaux et critiques du FMS.

Chapitre 4

Approche de diagnostic des attaques de blocage dans un contexte incertain

Introduction

Le chapitre 4 introduit la problématique de l'indisponibilité des ressources du FMS à la méthode de diagnostic des attaques de blocage et des profils d'attaquant développée dans le chapitre 3. L'environnement incertain des FMSs, que nous limitons volontairement à l'indisponibilité de ses ressources, a été défini dans le chapitre 1 (sous-partie 1.2.2) et a mis en exergue deux catégories d'indisponibilités, soit les défaillances de ressources et les opérations de maintenance préventive. Ces deux catégories d'indisponibilités résultent en différents événements observables et contrôlables communiqués entre le superviseur et les ressources. Néanmoins, certains de ces événements, selon leur origine, ne sont pas directement observables par le module de diagnostic (par exemple, une commande de mise à l'arrêt pour une opération de maintenance envoyée depuis le superviseur à un contrôleur local). De surcroît, en raison de leur nature contrôlable ou observable, ces événements d'indisponibilité sont vulnérables aux attaques d'insertion et de suppression par définition d'un attaquant expert (définition 2.2.9). Ainsi, dans un contexte incertain, deux nouvelles problématiques sont ouvertes au module de diagnostic ; (i) le diagnostic d'un changement de mode lorsqu'une ressource devient indisponible ou de nouveau disponible, et (ii) le diagnostic de l'origine, naturelle ou malveillante, d'un changement de mode.

Dans ce chapitre, ces deux problématiques sont étudiées et intégrées à l'objectif global de diagnostic des attaques de blocage et des profils d'attaquant dans un contexte incertain. Le fonctionnement global du module de diagnostic dans un contexte incertain est illustré par le diagramme d'activité 4.1. Premièrement, dans la partie 4.1, les conséquences d'un changement de mode de disponibilités sur l'allocation des ressources sont analysées (sous-partie 4.1.1) afin d'étendre les méthodes de supervision (sous-partie 4.1.2) et d'ordonnancement (sous-partie 4.1.3) du FMS au contexte incertain. Deuxièmement, dans la partie 4.2, les attaques contre les événements relatifs aux changements de mode sont étudiées afin de mettre en exergue les vulnérabilités du FMS et du module de diagnostic face à ces attaques (sous-partie 4.2.1). Le modèle des attaques de blocage est alors étendu pour inclure ces nouvelles attaques (sous-partie 4.2.2) et l'algorithme de calcul des profils d'attaquant est modifié pour les intégrer (sous-partie 4.2.3). Troisièmement, dans la partie 4.3, le module de diagnostic est amélioré afin de répondre

aux deux problématiques introduites dans les deux premières parties. Deux diagnostiqueurs sont ainsi proposés, le premier en charge de l'identification des changements de mode (sous-partie 4.3.1) et le second de la détection des attaques de blocage et des profils d'attaquant dans un contexte incertain (sous-partie 4.3.2). Enfin, des indicateurs sont développés pour enrichir le diagnostic et pour évaluer l'origine naturelle ou malveillante d'un changement de mode (sous-partie 4.3.3).

Ainsi, en comparaison au fonctionnement global du module de diagnostic dans un contexte certain (diagramme 3.1), ce chapitre propose cinq nouvelles contributions :

- Des méthodes d'ordonnancement et de supervision adaptées aux changements de mode (en haut de la figure 4.1) ;
- Un nouveau modèle original des attaques de blocage incluant les attaques contre les événements d'indisponibilité. Ce nouveau modèle entraine des modifications à l'algorithme de calcul des profils d'attaquant (à droite de la figure 4.1) ;
- Un diagnostiqueur commun des attaques de blocage, des profils d'attaquant et des changements de mode (au milieu de la figure 4.1) ;
- Un nouveau fonctionnement en ligne du module : une nouvelle condition de changement de mode déclenche un ré-ordonnancement (en bas à gauche de la figure 4.1) ;
- Des indicateurs sur l'origine d'un changement de mode (en bas à gauche de la figure 4.1).

4.1 Analyse d'un changement de mode

L'indisponibilité d'une ressource et les changements de mode de fonctionnement ont des conséquences sur l'ordonnancement, sur les états accessibles et sur la DZ du FMS (c.f. partie 1.2.2). Dans cette partie, ces conséquences et une méthode de supervision et d'ordonnancement adaptée à celles-ci sont présentées. Dans une première sous-partie 4.1.1, les conséquences sur l'allocation des ressources d'un changement de mode sont répertoriées et analysées. Face à ces conséquences, des méthodes de supervision et d'ordonnancement de la littérature sont appliquées et leurs limites au regard de notre positionnement sont identifiées. Dans une deuxième sous-partie 4.1.2, une méthode générale de supervision et d'ordonnancement répondant à nos problématiques et reprenant les méthodes développées dans le chapitre 3 est proposée. Enfin, dans la dernière sous-partie (4.1.3), un exemple d'application de cette méthode générale de supervision et d'ordonnancement d'un FMS dans un contexte incertain est donné.

4.1.1 Conséquences d'un changement de mode

Dans le fonctionnement global du module de diagnostic (diagramme 4.1), les conséquences d'un changement de mode sont requises pour la définition des méthodes de supervision et d'ordonnancement dans un contexte incertain. L'objectif de cette sous-partie est d'introduire ces conséquences.

Définition des conséquences

Soit un changement de mode entre $Mode_{\mathcal{R}_i}$ et $Mode_{\mathcal{R}_j}$ où la ressource r_k devient indisponible, i.e. $\mathcal{R}_j \setminus \mathcal{R}_i = r_k$. Au sein du FMS, l'indisponibilité de r_k affecte d'une part

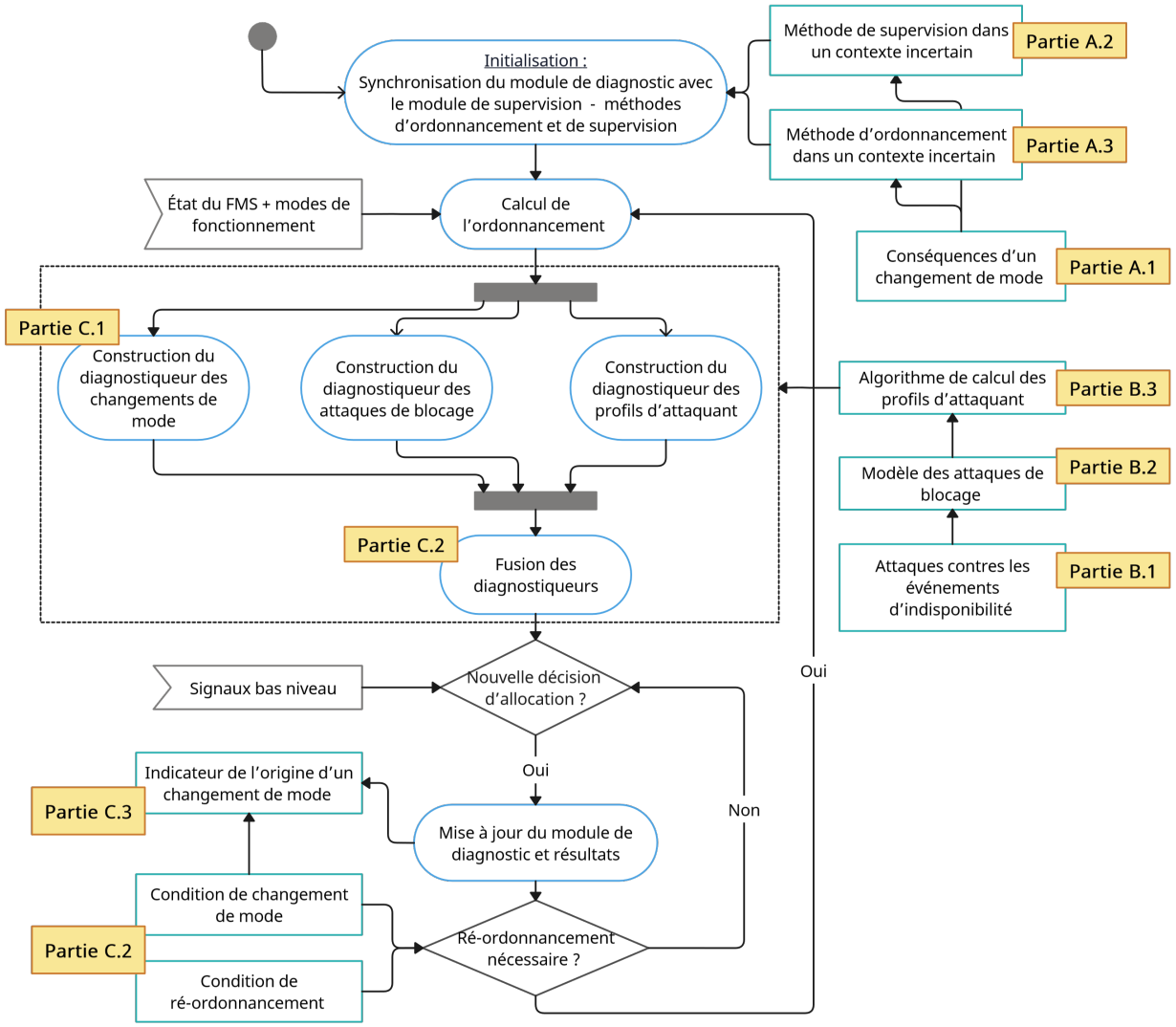


FIGURE 4.1 – Diagramme d'activité UML global du module de diagnostic dans un contexte incertain

l'ordonnancement, le FMS ne pouvant plus allouer r_k à des opérations la requérant, et modifie d'autre part l'ensemble des états d'allocations atteignables par le FMS et les séquences de décisions les reliant puisque les transitions appartenant à $p^\bullet_{r_k}$ deviennent infranchissables. Par définition, ces deux entités affectées par le changement de mode, à savoir l'ordonnancement et l'ensemble des marquages accessibles, sont dépendantes de l'état initial à partir duquel elle sont calculées. Soit M cet état initial, on note $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$ le changement de mode ayant lieu à partir de M. Dans les prochains paragraphes, les conséquences d'un changement de mode seront définies à partir d'un modèle du FMS résultant d'un changement de mode.

Soit $N = (P_A \cup P_R \cup P^0, T, F)$ un modèle S^{3}PR, M_0 un marquage initial acceptable de N et $M \in \mathcal{R}(N, M_0)$ le marquage de N lorsque le changement de mode a lieu, i.e. lorsque r_k, modélisé par $p_k \in P_R$ devient indisponible. Soit $\mathcal{C}_A = \{C_1, C_2, ..., C_{n_A}\}$ l'ensemble des circuits opérations transitions élémentaires appartenant à N, i.e. $\forall C \in \mathcal{C}_A, C \cap P_R = \emptyset$. Ces circuits

représentent les différents chemins que peuvent emprunter les produits à travers le FMS et on peut ainsi définir pour une recette A_1 l'ensemble de ses circuits par $\mathcal{C}^{A_1}$ où $\forall C \in \mathcal{C}^{A_1}, p^0_{A_1} \in C$ avec $p^0_{A_1} \in P^0$ la place d'entrée de la recette A_1. Par conséquent, $\mathcal{C}_A = \mathcal{C}^{A_1} \cup \mathcal{C}^{A_2} \cup ... \cup \mathcal{C}^{A_{|P^0|}}$. A partir de $\mathcal{C}_A$ et de $H_{r_k} = p^{\bullet\bullet}_{r_k} \cap P_A$, nous définissons l'ensemble des places activité de N dont les produits qu'elles opèrent requièrent systématiquement r_k pour être terminés par :

$$P^{pre}_{r_k} = \{p \in P_A | \exists p' \in H_{r_k} \ t.q. \ \exists p^0 \in P^0, C_1 \in \mathcal{C}_A, p, p', p^0 \in C_1, p^0 \notin pt_1p_1...p_mt_mp' \subset C_1$$

$$\text{et } \nexists C_2 \in \mathcal{C}_A, C_2 \neq C_1, p^0, p \in C_2, (pt_1...p_ft_fp^0 \subset C_2) \cap H_{r_k} = \emptyset\}$$

Dans $P^{pre}_{r_k}$, $pt_1p_1...p_mt_mp' \subset C_1$ représente le chemin reliant p à p' dans C_1, la condition relative à C_1 explicitant l'antécédence de p par rapport à p' dans la recette qu'il partage. Pour sa part, la condition relative à C_2 permet d'assurer qu'il n'existe pas de circuit parallèle à C_1 permettant de finir un produit en p (modéliser par le chemin $pt_1...p_ft_fp^0$ de p à p^0 dans C_2) sans traverser une place activité requérant r_k, sinon p n'appartient pas à $P^{pre}_{r_k}$ car non bloquée par l'indisponibilité de r_k. En prenant un chemin vide, on a intuitivement $H_{r_k} \subset P^{pre}_{r_k}$.

En complément, notons $P^0_{r_k} = \{p^0 \in P^0 | \forall C \in \mathcal{C}_A, p^0 \in C, \exists p \in H_{r_k} \ t.q. \ p \in C\}$ l'ensemble des places d'entrée bloquées en cas d'indisponibilité de r_k. Si r_k devient indisponible, tous les produits présents dans des places $p \in P^{pre}_{r_k}$ sont donc bloqués dans le FMS sans que les recettes de fabrication ne puissent être achevées et aucune opération sur un produit présent dans une place d'entrée de $p^0 \in P^0_{r_k}$ ne peut être débutée.

En s'inspirant des travaux de [131], les ensembles $P^{pre}_{r_k}$ et $P^0_{r_k}$ permettent de construire à partir de N deux sous-RdPs, le premier composé des places activité et d'entrée bloquées de $P^{pre}_{r_k} \cup P^0_{r_k}$, le second des places activité et d'entrée non-bloquées car non affectées par l'indisponibilité de r_k. Soit $N_B = (P_B, T_B, F_B)$ le sous-RdP bloqué et $N_{NB} = (P_{NB}, T_{NB}, F_{NB})$ le sous-RdP non bloqué. On définit N_B pour une ressource r_k non disponible par :

1. $P_B = P^B_A \cup P^B_R \cup P^0_B$ avec $P^B_A = P^{pre}_{r_k}$, $P^0_B = P^0_{r_k}$ et $P^B_R = {}^{\bullet\bullet}(P^B_A) \cap P_R$; ple en conclusion de cette partie
2. $T_B = \{t \in T | {}^\bullet t \cap P_A \in P^B_A\}$;
3. et $F_B = \{(X, Y) \in F | X, Y \in (T_B \cup P_B)\}$.

On définit alors P_{NB} à partir de N_B de la manière suivante :

1. $P_{NB} = P^{NB}_A \cup P^{NB}_R \cup P^0_{NB}$ avec $P^{NB}_A = P_A \setminus P^B_A$, $P^0_{NB} = P^0 \setminus P^0_B$ et $P^{NB}_R = {}^{\bullet\bullet}(P^{NB}_A) \cap P_R$;
2. $T_{NB} = \{t \in T | {}^\bullet t \cap P_A \in P^{NB}_A\}$;
3. $F_{NB} = \{(X, Y) \in F | X, Y \in (T_{NB} \cup P_{NB})\}$.
4. et $\forall t \in T_{NB} | t^\bullet = \{\emptyset\}$, on ajoute des places de sortie $p^{NB}_s \in P_{NB}$ et des arcs $(t, p^{NB}_s) \in F_{NB}$ modélisant la place de fin des produits appartenant à des circuits bloqués mais ne requérant pas les places bloquantes de H_{r_k}.

Au sein de ces deux sous-modèles, les places ressources $p_r \in P^B_R \cap P^{NB}_R$ sont partagées puisqu'elles sont requises pour des opérations bloquées et des opérations non-bloquées. On note alors l'ensemble des places activité de N_B requérant une ressource partagée avec une place activité de N_{NB} par :

$$P^B_{ptg} = \{p \in P^B_A | p_r = {}^{\bullet\bullet}p \cap P^B_R \in P^{NB}_R\}$$

et l'ensemble des places activité de N_{NB} appartenant uniquement à des circuits bloqués et succédant donc des places de $P^{pre}_{r_k}$ par :

$$P^{post}_{r_k} = \{p \in P^{NB}_A | \forall C \in \mathcal{C}, p \in C, C \cap P^{pre}_{r_k} \neq \emptyset\}$$

Notons que la définition des modèles N_B et N_{NB} peut être étendue à un ensemble de ressources indisponibles $\mathcal{R}_k$ à partir des ensembles $H_{\mathcal{R}_k} = \bigcup H_{r_k}$, $P^{pre}_{\mathcal{R}_k} = \bigcup P^{pre}_{r_k}$ et $p^0_{\mathcal{R}_k} = \bigcup P^0_{r_k}$ avec $r_k \in \mathcal{R}_k$. L'ensemble $P^{post}_{r_k}$ peut aussi être étendu à $P^{post}_{\mathcal{R}_k}$ Une illustration de ces modèles N_B et N_{NB} est proposée dans l'exemple en conclusion de cette partie (4.4).

En s'appuyant sur N_B, N_{NB} et P^B_{ptg}, les conséquences du changement de modes $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$ peuvent être répertoriées en étudiant les effets du marquage M sur ces deux sous-RdPs N_B et N_{NB} considérés séparément. Soient $M|_{P_B}$ et $M|_{P_{NB}}$ les projections de M sur les places de P_B et de P_{NB} respectivement. Les deux conséquences suivantes sont envisagées :

La **première** conséquence, appelée "cascade d'indisponibilité", est définie par :

Définition 4.1.1 (Cascade d'indisponibilité)**.** Une conséquence de cascade d'indisponibilité est définie par le marquage d'une place activité $p \in P^B_{ptg}$ et a pour conséquence le blocage de cette ressource partagée et son indisponibilité dans N_{NB}. ◇

Des places activité non bloquantes peuvent être atteignables depuis une place $p \in P^B_{ptg}$, à savoir des places activité requérant des ressources non bloquantes $p_r \in P^B_R \setminus P^{NB}_R$. Toutefois, il se peut que le nombre de produits présents dans des places bloquées de $M|_{P_B}$ et les places activité non bloquantes atteignables par ces produits ne permettent pas d'éviter cette conséquence formulée par $\nexists M' \in \mathcal{R}(N_B, M|_{P_B}) | \forall p \in P^B_{ptg}, M'(p) = 0$. Ce cas entraine un nouveau calcul de N_B et N_{NB} à partir de $P^{pre}_{r_k} \cup P^{pre}_{r_l}$ et $P^0_{r_k} \cup P^0_{r_l}$ avec r_l la ressource devenue indisponible via le marquage d'une place $p \in P^B_{ptg}$. Cependant, ces nouveaux modèles N_B et N_{NB} peuvent aussi être à l'origine de nouvelles indisponibilités induites de ressources et ce phénomène se réitère jusqu'à atteindre un état du FMS où plus aucune ressource partagée entre N_B et N_{NB} n'est détenue par une place activité de N_B. Une nouvelle conséquence de **cascade d'indisponibilités** a alors lieu.

Deuxièmement, dans N_{NB}, son graphe des marquages accessibles $\mathcal{RG}(N_{NB}, M|_{P_{NB}})$ représente le nouvel ensemble des états atteignables depuis M par le FMS lors de son fonctionnement dans le $Mode_{\mathcal{R}_j}$. La construction de $\mathcal{RG}(N_{NB}, M|_{P_{NB}})$ a pour conséquences la redéfinition de la LZ et de la DZ de N_{NB} par rapport à celles du mode précédent $Mode_{\mathcal{R}_i}$. Par exemple, en supposant $M \notin \mathcal{M}_{DZ}$, un état $M|_{P_{NB}}$ peut devenir un état de blocage dans N_{NB} si $L(N, M)|_1 \subset p^{\bullet}_k$, autrement dit si toutes les transitions franchissables depuis M requièrent la ressource r_k devenue indisponible dans N_{NB}.

Proposition 4.1.1. *Les conséquences du changement de modes $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$ (induisant l'indisponibilité de r_k) sur le fonctionnement du FMS se définissent uniquement par la cascade d'indisponibilité (4.1.1) et par un nouveau graphe des marquages accessibles.*

Preuve 4.1.1. Dans N_B, puisque les conséquences du marquage des places activités $p \in P^B_{ptg}$ ont déjà été considérées, seul le marquage des places $p \in P^B_A \setminus P^B_{ptg}$ reste à étudier. Or, le marquage de ces places est sans effet pour le fonctionnement du FMS dans N_B puisqu'elle requiert des ressources non utilisées par le FMS au sein du $Mode_{\mathcal{R}_j}$. Dans N_{NB}, l'étude de $\mathcal{RG}(N_{NB}, M|_{NB})$ permet d'être exhaustif sur les conséquences d'un changement de mode puisque tous les états atteignables du FMS sont étudiés. Ainsi, les conséquences d'un changement de mode sur le fonctionnement d'un FMS se résument à la cascade d'indisponibilité et à un nouveau graphe des marquages accessibles calculé à partir de N_{NB} et $M|_{P_{NB}}$. ⌟

Remarque 4.1.1. Le changement de mode inverse de $Mode_{\mathcal{R}_j}$ vers $Mode_{\mathcal{R}_i}$, pour lequel r_k redevient disponible, conduit aux conséquences opposées, soit la libération des ressources devenues indisponibles par la cascade d'indisponibilité et à un nouveau graphe des marquages accessibles dans lequel les circuits bloqués peuvent de nouveau être empruntés. ⌟

Ces deux conséquences d'un changement de mode doivent être prises en compte par les méthodes de supervision et d'ordonnancement du FMS choisies dans un contexte incertain. Dans la prochaine sous-partie, différentes méthodes de la littérature sont analysées selon leurs capacités à faire face à ces conséquences tout en garantissant les propriétés propres aux modules de supervision et de diagnostic introduites précédemment.

Application des méthodes de supervision et d'ordonnancement au contexte incertain

Dans cette partie, une application et une analyse critique des travaux de la littérature est menée dans le but de développer un module de supervision adapté aux changements de mode et au diagnostic des attaques de blocage dans un contexte incertain (sous-partie 4.1.2). A partir de ces analyses, une méthode de gestion des états de blocage et une méthode d'ordonnancement seront choisies.

Premièrement, parmi les méthodes de gestion des états de blocage et en se fondant sur les mêmes justifications que dans le contexte certain (sous-partie 3.2.1), les méthodes de prévention sont préférées aux méthodes d'évitement et de détection et reprise des états de blocage dans un contexte incertain. Néanmoins, en cas de changement de modes, deux problématiques héritées des conséquences d'un changement de mode (sous-partie 4.1.1) se posent : la construction d'une politique de contrôle capable d'une part d'empêcher les cascades d'indisponibilités et d'autre part de prévenir le FMS d'atteindre les états des DZs des différents modes.

Deuxièmement, pour le choix d'une méthode d'ordonnancement, les propriétés de celle-ci dans le contexte sans indisponibilité doivent être conservées : déterminisme, taille partielle ou totale, sous-optimalité de l'ordonnancement et temps de calcul faible (sous-partie 3.2.2). Ces propriétés sont nécessaires à la synchronisation entre le module de supervision et le module de diagnostic. Toutefois, les changements de mode entraînent un ré-ordonnancement plus fréquent de l'allocation des ressources. En effet, lorsque le module de supervision reçoit un événement de défaillance depuis une ressource ou un ordre d'opération de maintenance préventive d'un opérateur, l'ordonnancement est recalculé avec l'objectif de terminer les produits appartenant à des recettes non-bloquées du FMS. Par conséquent, une méthode d'ordonnancement ne permet

pas de minimiser les conséquences d'un changement de mode mais elle doit être capable de maintenir ses propriétés et performances en présence de ces conséquences. Dans nos travaux, la méthode de recherche $A^\star BT$ est conservée dans le contexte incertain puisque qu'elle garantie les propriétés attendues. Toutefois, sa méthode d'exécution sera discutée à partir des catégories d'approches d'ordonnancement présentées dans la partie 1.1.3 : préventives, réactives, préventives réactives et pro-actives réactives.

Dans cette sous-partie, des méthodes de prévention sont appliquées pour faire face aux deux conséquences de cascade d'indisponibilités et d'évolution de la DZ. Pour chaque méthode, un calcul de l'ordonnancement par la méthode $A^\star BT$ est réalisé afin de mettre en lumière les performances de la méthode dans un FMS possédant plus d'états contraints par la prévention des conséquences des changements de mode.

Cascade d'indisponibilités

Le phénomène de **cascade d'indisponibilités** a lieu au sein d'un changement de modes $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$ conduisant à la construction des RdPs N_B et N_{NB} lorsque $\nexists M' \in \mathcal{R}(N_B, M|_B) | \forall p \in P^B_{ptg}, M'(p) = 0$ - empêcher ce phénomène consiste alors à contraindre les états accessibles M du $Mode_{\mathcal{R}_i}$ afin de garantir l'existence d'un tel marquage M' pour chaque $Mode_{\mathcal{R}_j}$ atteignable depuis $Mode_{\mathcal{R}_i}$.

Cette contrainte peut être appliquée de la manière suivante : pour chaque $Mode_{\mathcal{R}_i}$ conduisant à la construction des modèles N_B et N_{NB} à partir de l'ensemble $\mathcal{R}_i$, $n_1 \in \mathbb{N}$ places activités de P^B_{ptg} ne pouvant atteindre qu'un ensemble de $n_2 \in \mathbb{N}$ places activités non marquées de $P^B_A \setminus (P^B_{ptg} \cup H_{\mathcal{R}_i})$ avec $n_2 < n_1$ ne peuvent pas être marquées simultanément. En effet, si ces n_1 places activités sont marquées simultanément, cela signifie qu'au moins un produit au sein de ces n_1 places ne pourra pas atteindre une place activité parmi les n_2 places activités n'appartenant pas à P^B_{ptg}. Ce produit occupera une place activité de P^B_{ptg} et créera alors une conséquence de cascade d'indisponibilité.

Cependant, dans nos travaux, l'application de cette contrainte est trop restrictive en raison de l'indisponibilité des ressources hors opération. En effet, cette contrainte implique que pour chaque ressource r_{k1}, une place activité p précédent une place $p' \in H_{r_{k1}}$ requérant r_{k1} ne peut être marquée que si elle requiert une ressource r_{k2} tel que $H_{r_{k2}} \cap P^B_{ptg} = \emptyset$. En d'autres termes, aucune place activité requérant r_{k2} ne doit appartenir à N_{NB}, sinon le marquage de p, en bloquant la ressource r_{k2}, créera une cascade d'indisponibilité. Dans l'exemple de la figure 1.6, cette condition est inapplicable et a pour conséquence l'interdiction du marquage de p_9 et p_4 dans le cas où R_2 devient indisponible hors-opération, et du marquage de p_{15} et p_{13} lorsque R_4 devient indisponible. Ces interdictions entraînent le blocage de toutes les recettes du FMS et prouvent l'inapplicabilité de la contrainte.

Toutefois, dans le but de prévenir partiellement les conséquences de cascades d'indisponibilités, la contrainte est appliquée uniquement aux cas où les ressources deviendraient indisponibles en opération. Ainsi, pour chaque $Mode_{\mathcal{R}_i}$ ne bloquant pas toutes les recettes (le cas échéant, une conséquence de cascade d'indisponibilité n'aurait aucun effet sur les circuits déjà bloqués), et pour chaque marquage accessible $M \in \mathcal{R}(N, M_0)$ dans le mode sans indisponibilité $Mode_0$, les marquages ne respectant pas la contrainte précédente auquel s'ajoute la condition $\forall r_k \in \mathcal{R}_i, \exists p \in H_{r_i} | M(p) = 1$ d'indisponibilité des ressources en opération sont intégrés à l'ensemble des marquages à interdire $\mathcal{M}_{cas}$. Puis, la méthode d'Uzam [50] est appliquée au

modèle S^{3}PR N en définissant les "First Bad Markings" égaux aux états de la DZ ou appartenant à $\mathcal{M}_{cas}$.

Cette contrainte "partielle" est simulée sur l'exemple de la figure 1.6 afin de créer des moniteurs capables de prévenir les cascades d'indisponibilités dans le cas des indisponibilités des ressources en opération. Les résultats de simulation conduisent à l'ajout de 44 places moniteur contre 13 dans le cas de la prévention des états de la DZ du $Mode_0$. Le nouveau modèle ne peut plus atteindre que 44 états d'allocations permettant la réalisation de toutes les recettes. En appliquant l'algorithme de recherche $A^\star BT$ pour calculer un ordonnancement ne traversant pas les places de $\mathcal{M}_{cas}$, les résultats suivants sont obtenus : pour $K_{max} = 100$, l'ordonnancement a un coût en temps de 1974s pour réaliser 4 produits de la recette A, 8 de la recette B et 6 de la recettes C (2ème état initial considéré dans le tableau 3.1) contre un coût de 896s dans le cas sans restriction des états de $\mathcal{M}_{cas}$. Outre la dégradation des performances d'ordonnancement par un facteur 2, la décision de mettre en place de tels moniteurs doit aussi prendre en compte la résolution de la seconde conséquence d'un changement de mode : la prévention des états des DZs des différents modes de fonctionnement.

Évolution de la DZ entre les modes

La **prévention** des états des DZs des différents modes de fonctionnement a été proposée dans plusieurs travaux de la littérature [131], [134], [142] au sein desquels la réalisation des circuits ne requérant par les ressources indisponibles est maintenue de manière vivante (sans nouveau blocage), i.e. le modèle N_{NB} est vivant pour n'importe quel marquage M du mode précédent. Ces méthodes ont par ailleurs l'objectif de maintenir le système vivant lorsque les ressources redeviennent disponibles et certains circuits de N redeviennent non bloqués. La majorité de ces méthodes [131], [134], [142] sont fondées sur des ressources dont les capacités sont supérieures ou égales à 1, leur offrant une plus grande flexibilité face aux indisponibilités. Ces méthodes peuvent tout de même être appliquées à des FMSs dont les ressources ont des capacités unitaires. Cependant, elles ne sont pas retenues pour le développement des modules de supervision et de diagnostic dans un contexte incertain car ces dernières sont fondées sur une analyse structurelle du FMS et non sur l'étude du graphe des marquages accessibles, leur conférant une permissivité plus restreinte.

Une autre méthode de la littérature [150] fondée sur l'étude du graphe des marquages accessibles $\mathcal{RG}(N_{\mathcal{R}_{ind}}, M_0^{\mathcal{R}_{ind}})$ est adaptée et appliquée à nos travaux. Cette méthode s'appuie sur l'identification des composants fortement connexes (CFC) du graphe orienté $(V, F) = \mathcal{RG}(N_{\mathcal{R}_{ind}}, M_0^{\mathcal{R}_{ind}})$ (voir annexe 6.1), soit les sous-ensembles d'états d'un graphe tel que tous les états d'un CFC sont atteignables depuis n'importe quel état de ce même CFC. Parmi les CFC identifiés, seuls ceux au sein desquels toutes les transitions de T sont franchissables sont conservés afin d'assurer la vivacité de tous les circuits de N et forment l'ensemble des états autorisés du système. Les états à interdire sont donc l'ensemble complémentaire dans $\mathcal{RG}(N_{\mathcal{R}_{ind}}, M_0^{\mathcal{R}_{ind}})$ des états autorisés. Ainsi, des places de contrôle sont construites à partir des états à interdire assimilables aux "First Bad Markings" considérés par [50]. Néanmoins, cette méthode doit être adaptée aux hypothèses de nos travaux.

Premièrement, similairement à la conséquence de cascade d'indisponibilités, seules les indisponibilités des ressources en opération sont considérées. Par conséquent, le modèle $N_{\mathcal{R}_{ind}}$ utilisé est restreint aux sous-réseaux de reprise connectés aux places opérations de N (1.2.2).

Deuxièmement, les états à autoriser ne sont plus uniquement restreints aux CFCs au sein desquels toutes les transitions de T sont franchissables. En cas d'indisponibilités de ressources, l'ensemble des états autorisés est étendu aux CFCs représentant la réalisation des recettes ne requérant pas ces ressources. Le reste des CFCs correspondant aux blocages partiels ou incluant des cascades d'indisponibilités bloquant des circuits ne requérant pas les ressources indisponibles sont quant à eux ajoutés à l'ensemble des états interdits.

Cette adaptation de la méthode de [150] appliquée à l'exemple précédent permet d'obtenir les résultats suivants. En supposant toutes les ressources susceptibles de devenir indisponibles, le superviseur ρ ainsi construit par l'ajout de places de contrôle à N n'autorise plus que 101 états dans $\mathcal{RG}(N, M_0)$ (contre 998 dans le contexte certain), assure l'absence d'états de blocage, pré-blocage ou blocage partiel dans $\mathcal{RG}(N_{\mathcal{R}_{ind}}, M_0^{\mathcal{R}_{ind}})|_\rho$ malgré l'occurrence possible d'indisponibilités en opération et permet d'obtenir un ordonnancement selon l'état initial $M_0(p_e^A) = 4, M_0(p_e^B) = 8, M_0(p_e^C) = 6$, et $K_{max} = 100$ dont le coût est égal à 1686s. En fusionnant les places de contrôle développées dans ce paragraphe avec celles construites pour faire face aux cascades d'indisponibilités, le FMS n'est plus capable d'atteindre que 27 états et l'ordonnancement au sein de ce système supervisé nécessite un coût temporel de 2044s. A l'instar de la méthode de prévention des effets de cascade d'indisponibilités, les résultats d'ordonnancement ont des coûts au minimum deux fois plus élevés que dans le contexte certain.

Ainsi, la méthode de prévention des états de blocage dans un contexte incertain adaptée des travaux de [150] permet de maintenir la production des produits ne requérant pas les ressources indisponibles et de prévenir tous les états des DZs des différents modes de fonctionnement. En omettant les indisponibilités hors-opération, la méthode de diagnostic présentée dans le chapitre 3 (3.3) pourrait par conséquent être appliquée au FMS muni des moniteurs construits dans les deux dernière sous-parties. Les attaques de blocage seraient alors systématiquement détectées dès qu'un état interdit serait atteint dans n'importe quel mode de fonctionnement.

Analyse et critiques

Le déploiement d'un superviseur, fusionnant les places moniteur développées dans les deux précédentes sous-parties pour la prévention partielle des conséquences de cascades d'indisponibilités et des états des DZs des différents modes de fonctionnement, n'est cependant pas retenu dans nos travaux pour les quatre raisons suivantes.

- Les indisponibilités des ressources hors-opérations ne sont pas considérées par ce superviseur bien qu'elles fassent partie de nos hypothèses de travail initiales (sous-partie 1.2.2)
- Sa faible permissivité impose une chute élevée des performances temporelles de l'ordonnancement. De surcroît, le déploiement d'une telle méthode à une seule fin de diagnostic des attaques de blocage contredit la justification première du choix d'une approche de détection pour faire face à ces attaques, à savoir le faible interventionnisme de cette méthode sur le fonctionnement nominal du FMS.
- Une telle méthode de prévention n'apporte pas de solution à la problématique de diagnostic de l'origine, malveillante ou naturelle, d'un changement de mode. En effet, malgré la détection des attaques de blocage lorsqu'un état de la DZ du mode actuel est atteint, il n'est pas possible pour le module de diagnostic de conclure si les indisponibilités en présence ont été provoquées par l'attaquant ou si celui-ci a attendu que ces dernières aient lieu avant de lancer son attaque.

- Un tel superviseur engendre une perte d'informations et d'observations exploitables pour le diagnostic des profils d'attaquant et des attaques de changement de mode. Par exemple, en considérant que l'attaquant ne cherche pas à éviter les états hors de la DZ du mode actuel interdits par la méthode de supervision, nous supposons que deux profils distincts $\mathcal{P}_1 \neq \mathcal{P}_2$ possèdent le même suffixe conduisant à un état hors DZ interdit, i.e. $\mathcal{P}_1 = \sigma\sigma_1, \mathcal{P}_2 = \sigma\sigma_2$, $M[\sigma > M'$ avec $M, M' \in \mathcal{R}(N, M_0), M' \notin \mathcal{M}_{DZ}$ mais $\rho(M)[\sigma] = 0$ à savoir que la séquence σ est interdite depuis M, mais ces deux profils ne possèdent par les même préfixes $\sigma_1 \neq \sigma_2$. Si la méthode de diagnostic et/ou le FMS sont arrêtés au premier état d'attaque diagnostiqué, à savoir M', la différenciation entre $\mathcal{P}_1$ et $\mathcal{P}_2$ n'est plus réalisable car seule la séquence $l^\alpha(\sigma)$ est observée et pas les séquences $l^\alpha(\sigma_1)$ et $l^\alpha(\sigma_2)$.

Les quatre arguments exposés ci-dessus nous conduisent à rejeter le superviseur inspiré de la littérature présenté dans cette sous-partie 4.1.1. Une méthode de prévention et d'ordonnancement originale répondant à nos hypothèses est proposée dans la prochaine sous-partie 4.1.2.

4.1.2 Méthode de prévention et d'ordonnancement dans un contexte incertain

Au sein du fonctionnement du module de diagnostic dans un contexte incertain (diagramme 4.1), une méthode de prévention et d'ordonnancement synchronisée entre les modules de supervision et de diagnostic est requise. Dans cette partie, la méthode de prévention et d'ordonnancement choisie et implémentée au sein du module de supervision est introduite et illustrée par le diagramme d'activité de la figure 4.2 ci-contre. Cette méthode repose sur le fonctionnement par étapes ci-dessous :

- Le superviseur est muni des moniteurs construits pour le $mode_0$ sans indisponibilité (sous-partie 3.2.2) ;
- L'ordonnancement est calculé par la recherche $A^\star BT$ selon l'état du FMS, son mode de fonctionnement et la condition de ré-ordonnancement présentée dans le chapitre 3 (sous-partie 3.3.1).
- Lors d'un changement de mode $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$ identifié par le module de supervision à partir d'un événement de défaillance ou un ordre de maintenance préventive :
  1. Les moniteurs du $mode_0$ sont conservés et continuellement mis à jour selon le franchissement des transitions connectées aux places de contrôle ;
  2. En entrant dans le nouveau mode, selon l'état M du FMS, les conséquences de cascade d'indisponibilité pouvant avoir lieu sont calculées selon les marquages des places de $P^{pre}_{\mathcal{R}_k}$ et avec $\mathcal{R}_k$ l'ensemble des ressources indisponibles. On a $\mathcal{R}_j \subset \mathcal{R}_k$;
  3. Parmi toutes les conséquences de cascades d'indisponibilités, un marquage des places de $P^{pre}_{\mathcal{R}_k}$ est choisi afin de bloquer le moins de circuits et de places possibles ;
  4. Les modèles N_B et N_{NB} sont construits selon les ressources indisponibles $\mathcal{R}_k$ et le marquage des places de $P^{pre}_{\mathcal{R}_k}$;
  5. Des moniteurs temporaires sont déployés à partir de la méthode de [50] appliquée aux circuits vivants de N_{NB}. En cas de retour au $mode_0$ ou d'un nouveau changement de mode, les moniteurs temporaires sont supprimés par la méthode de prévention ;
  6. L'ordonnancement est désormais calculé par la recherche $A^\star BT$ à partir de N_{NB}, de ses moniteurs et de son état $(M|_{P_{NB}}, R|_{P_{NB}})$. La condition de ré-ordonnancement s'applique aussi uniquement à N_{NB} et $M|_{P_{NB}}$.

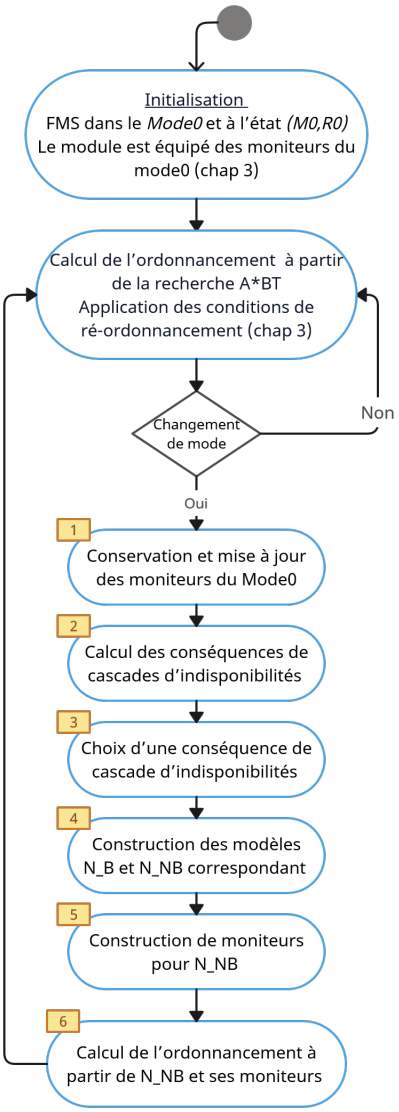


FIGURE 4.2 – Diagramme d'activité UML de la méthode de prévention et d'ordonnancement du module de supervision dans un contexte incertain

Les étapes 1 à 6 de la configuration de la méthode en cas de changement de mode $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$ sont introduites et développées dans les prochaines sous-parties.

Étape 1

Au sein de l'étape 1, la construction des places de contrôle du $mode_0$ est détaillée dans la section 3.2.1. Leur conservation entre les changements de mode est rendue possible par l'intersection vide entre les transitions de T_B et celles de T_{NB}. Ainsi, dès qu'une transition est franchie dans N_B ou N_{NB}, elle n'est franchie qu'une seule fois et permet la mise à jour des moniteurs de N. Une illustration des modèles N_B et N_{NB} est proposée dans l'exemple en conclusion de cette partie (figures 4.4b et 4.4a). Cette étape 1 permet de garantir qu'aucun état de blocage du $mode_0$ ne soit atteint au sein des autres modes de fonctionnement et par conséquent que le FMS reste vivant lorsque la reprise de toutes les ressources indisponibles de $\mathcal{R}_j$ a lieu.

Étape 2

A partir d'un changement de modes $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$, l'étape 2 requiert un algorithme capable de calculer toutes les conséquences de cascade d'indisponibilité pouvant avoir lieu selon le marquage choisi des places de $P^{pre}_{\mathcal{R}_j}$. Une conséquence peut ainsi être modélisée par un ensemble $\mathcal{R}'_j = \mathcal{R}_j \cup \mathcal{R}^{csc}_j$ et un marquage M_{rc} atteignable depuis M dans le $mode_{\mathcal{R}_j}$ où $\mathcal{R}^{csc}_j$ désigne l'ensemble des ressources rendues indisponibles par cascades d'indisponibilité et M_{rc} traduit le marquage choisit des places de $P^{pre}_{\mathcal{R}'_j}$. Le calcul de l'ensemble de ces conséquences est rendu possible par l'algorithme dont le pseudo-code est présenté dans l'annexe 6.12.

Cet algorithme, inspiré des travaux de Liu [150], s'appuie sur la construction de toutes les composantes fortement connexes (CFC) de $(V_r, F_r) = \mathcal{RG}(N_{\mathcal{R}_j}, M|_{P_{\mathcal{R}_j}})|_{\rho'}$ avec $N_{\mathcal{R}_j}$ le modèle incluant les réseaux de reprise pour les ressources indisponibles de $\mathcal{R}_j$, $M|_{P_{\mathcal{R}_j}}$ la projection de M dans le modèle $N_{\mathcal{R}_j}$ (voir annexe 6.12) et ρ' la politique de contrôle interdisant le franchissement des transitions de reprise dans $N_{\mathcal{R}_j}$(transitions rendant les ressources de $\mathcal{R}_k$ de nouveau disponibles). La projection de M dans le modèle $N_{\mathcal{R}_j}$ consiste au marquage des places d'indisponibilité d'une ressource de $\mathcal{R}_j$ selon si la ressource est en opération ou hors-opération. Dans cet algorithme, une CFC correspond à un ensemble d'états dont l'accessibilité des uns avec les autres est obtenue depuis un sous-RdP vivant de $N_{\mathcal{R}_j}|_{\rho'}$ au sein duquel les ressources de $\mathcal{R}_j$ ne peuvent être utilisées. Dans un S^{3}PR, ce sous-RdP vivant se compose d'un ensemble de circuits non-bloqués et exploitables pour la réalisation de recettes.

Chaque CFC est initié par un marquage racine M_{rc} accessible depuis $M|_{P_{\mathcal{R}_j}}$ à partir duquel les circuits non-bloqués du CFC sont parcourus. Parmi les places bloquées du CFC, soient celles dont le marquage est fixe dans tous les états du CFC, sont identifiées les places activités bloquées à cause d'une ressource indisponible appartenant à $\mathcal{R}_j$ ou devenue indisponible par cascade d'indisponibilité ($\mathcal{R}^{csc}_j$) et les autres places activité bloquées. La première catégorie de places bloquées permet de définir l'ensemble des ressources indisponibles du CFC, soit $\mathcal{R}'_j = \mathcal{R}_j \cup \mathcal{R}^{csc}_j$. Les places de la seconde catégorie sont rassemblées au sein de l'ensemble P'_{blck} et correspondent aux places activité non marquées de $P^{post}_{\mathcal{R}'_j}$ ainsi qu'aux places activité bloquées par l'apparition d'un blocage partiel au sein du modèle N_{NB} construit depuis $\mathcal{R}'_j$. On note $P'_{blck} = P^{post}_{\mathcal{R}'_j} \cup P'_{DZ}$.

Pour chaque CFC identifiée, les conséquences de cascade d'indisponibilité associées au CFC sont alors définies par le triplet $(\mathcal{R}'_k, M_{rc}, P'_{DZ})$ et regroupées dans l'ensemble Csc. Ainsi, pour un changement de mode $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$, **l'algorithme est capable de calculer la totalité des conséquences de cascades d'indisponibilités pouvant avoir lieu et ne bloquant par l'entièreté des recettes**. En effet, toutes les CFCs de $\mathcal{RG}(N_{\mathcal{R}_j}, M|_{P_{\mathcal{R}_j}})|_{\rho'}$ sont étudiées dans cet algorithme et elles représentent tous les comportements vivants que peut avoir un FMS à la suite de ce changement de mode. A l'inverse, si aucune CFC n'est identifiée, le FMS est entièrement bloqué suite au changement de mode. Deux avantages de cette méthode de calcul des conséquences de cascades d'indisponibilités sont à mettre en lumière.

Le **premier avantage** est sa capacité à déplacer des produits d'une place activité ne requérant pas de ressources indisponibles vers une place activité atteignable appartenant à $P^{pre}_{\mathcal{R}_j}$, si l'appartenance du produit à la première place activité citée créer des blocages supplémentaires. La places activité de $P^{pre}_{\mathcal{R}_j}$ sert donc de zone de stockage temporaire à un produit pouvant être bloquant.

Le **second avantage** est le vidage systématique, lorsque possible, des places de $P^{post}_{\mathcal{R}_j}$ par les conséquences explorées les moins bloquantes. En effet, lorsqu'elles sont marquées, ces places détiennent une ressource et la bloquent ; vider ces places devient donc prioritaire pour les conséquences les moins bloquantes. Au sein des états racines M_{rc} de ces conséquences, les places de $P^{post}_{\mathcal{R}_j}$ initialement marquées ne le sont plus, hors inclusion de celles-ci au sein d'un blocage partiel inévitable, par définition d'un CFC.

Étape 3

L'**étape 3** consiste à choisir une conséquence de cascades d'indisponibilités $(\mathcal{R}'_j, M_{rc}, P'_{blck})$ au sein de Csc. Dans ce travail, le choix d'une conséquence s'appuie successivement sur :

1. La maximisation des recettes pouvant être réalisées malgré les blocages ;
2. La minimisation du nombre d'états précédents bloqués soit $min(|P^{pre}_{\mathcal{R}'_j}|)$;
3. La minimisation des états bloqués par des blocages partiels, à savoir $min(|P'_{blck} \setminus P^{post}_{\mathcal{R}'_j}|)$;
4. La minimisation du nombre de transitions entre $M|_{P\mathcal{R}_j}$ et M_{rc}.

Dans le cas où plusieurs conséquences ne bloquant pas les mêmes circuits et/ou les mêmes places activité peuvent être choisies après application de ces critères, un dernier critère de sélection doit permettre l'unicité du choix final d'une conséquence. Aucun critère d'unicité n'étant pertinent au regard de l'optimisation de la vivacité du FMS après le changement de mode, un critère (5) de priorité relative des circuits est finalement choisi.

Ainsi, chaque circuit de $\mathcal{C}$ se voit assigné une priorité unique dans $\mathbb{N}$ et la conséquence finale sélectionnée est celle avec le circuit vivant non commun à d'autres conséquences éligibles avec la priorité la plus élevée. En d'autres termes, pour deux conséquences Csc_1 et $Csc_2 \in Csc$ ne pouvant être départagées selon les trois premiers critères et partageant les circuits vivants $\{C_1, C_2, ..., C_k\}$ avec les priorités les plus élevées au sein des circuits vivants de Csc_1 et de Csc_2, la conséquence finale choisie est celle possédant le circuit n'appartenant pas à $\{C_1, C_2, ..., C_k\}$ ayant une priorité plus élevée que les circuits non commun de la seconde conséquence.

L'unicité de la solution choisie par ce critère est prouvée pour chaque paire de conséquences ne possédant pas les mêmes circuits vivants puisque le critère (5) permet de les départager à

partir des circuits vivants non communs. Cependant, si deux conséquences ne pouvant être départagées par les trois premiers critères présentent les mêmes ensembles de circuits vivants $\mathcal{C}_1^v = \mathcal{C}_2^v = \{C_1, C_2, ..., C_k\}$, le critère (5) ne peut toujours pas les départager. Par conséquent, ce critère est associé à une seconde application des critères (2) et (3) aux circuits bloqués de Csc_1 et de Csc_2, à savoir $\mathcal{C}_1 \setminus (\{C_1, C_2, ..., C_k\}) = \mathcal{C}_1^b = \mathcal{C}_2 \setminus (\{C_1, C_2, ..., C_k\}) = \mathcal{C}_2^b$, selon leurs priorité décroissantes.

Ce dernier critère (6) départage les deux conséquences à partir du circuit bloqué commun C_b avec la plus grande priorité et n'ayant pas le même nombre de places précédentes $|P^{pre}_{\mathcal{R}'_j}|$ ou de places suivantes bloquées $|P'_{blck} \cap P^{post}_{\mathcal{R}'_j}|$ au sein de Csc_1 et Csc_2. Ce circuit existe systématiquement puisque le cas contraire cela signifierait que $Csc_1 = Csc_2$ et rendrait alors caduque le choix entre ces deux conséquences. L'existence systématique d'un tel circuit prouve l'unicité du choix final entre Csc_1 et Csc_2 à partir de ce dernier critère. Cette preuve peut être étendue à l'ensemble des conséquences Csc et prouve ainsi l'unicité de la conséquence finale choisie à partir de ces six critères.

Dans nos travaux, **toutes les conséquences de cascades d'indisponibilités sont calculées hors-ligne** pour chaque marquage $M \in \mathcal{R}(N, M_0)$ et pour chaque mode de fonctionnement $Mode_{\mathcal{R}_i}$ ne bloquant pas l'entièreté du FMS. Ce calcul hors-ligne est possible en raison du caractère fini de $\mathcal{R}(N, M_0)$ et de l'ensemble des modes de fonctionnement. De surcroît, le choix d'une conséquence est aussi réalisé hors-ligne pour chaque paire $(M, Mode_{\mathcal{R}_i})$. Enfin, pour la conséquence choisie, on note σ_c la séquence de transitions reliant M à M_{rc}.

Étape 4

L'**étape 4** reprend les définitions de N_B et N_{NB} pour les construire à partir des ressources indisponibles $\mathcal{R}'_j$ de la conséquence choisie précédemment. Soit C_{NB}, l'ensemble des circuits de places activité et d'entrée de N_{NB}. Un nouveau modèle, noté N^C_{NB} est défini en conservant uniquement les circuits de C_{NB} et les ressources nécessaires aux places activité de ces circuits. En d'autres termes les places de $P_{NB} \cap P^{post}_{\mathcal{R}'_j}$, les arcs en entrée et en sortie de ces places et les places ressources requises uniquement par ces places sont ôtés de N_{NB} pour obtenir N^C_{NB}. Le modèle N^C_{NB} est un S^{3}PR. Un modèle SC-net, noté N^{C-t}_{NB}, est alors construit à partir du S^{3}PR N^C_{NB} selon la méthode présentée dans le chapitre 3 (sous-partie 3.2.2).

Étape 5

L'étape 5 consiste en l'application de la méthode de construction des places de contrôle de [50] fondée sur l'étude du graphe des marquages accessibles du modèle N^C_{NB} obtenu après application des étapes 2, 3 et 4. Cette méthode peut sans modification être appliquée au modèle N^C_{NB} à partir du marquage $M|_{P_{NB}}$ et de l'ensemble des marquages accessibles $\mathcal{R}(N^C_{NB}, M|_{P_{NB}})$. Les places de contrôle ainsi construites sont définies dans P^{NB}_c et par les arcs F^{NB}_c. Dans cette étape, les moniteurs du $Mode_0$ sont conservés et mis à jour tandis que les moniteurs temporaires construits par cette même étape pour le mode précédent, si il est différent de $Mode_0$, sont supprimés de (P^{NB}_c, F^{NB}_c).

Cette méthode de prévention choisie pour faire face au contexte incertain des FMSs permet de maintenir la permissivité complète du $Mode_0$ sans indisponibilité aux dépens de la vivacité du FMS au sein de modes de fonctionnement présentant une ou plusieurs ressources indisponibles.

En effet, contrairement aux travaux de la littérature cités précédemment et appliqués à la prévention des états de la DZ des différents modes de fonctionnement [150], notre méthode de prévention ne permet pas de prévenir les états de blocage, pré-blocage et blocage partiel lorsqu'une ressource devient indisponible. Le choix lors de l'étape 4 d'une conséquence de cascade d'indisponibilité permet toutefois de maximiser le nombre de circuits et transitions toujours vivants du FMS à la suite du changement de mode.

De la même manière que les conséquences de cascades d'indisponibilités, **les modèles N_B, N_{NB}, N_{NB}^C et N_{NB}^{C-t} sont construits hors-ligne** à partir de chaque paire $(M, Mode_{\mathcal{R}_j})$. Les places de contrôle de (P_c^{NB}, F_c^{NB}) calculées à partir de N_{NB}^C et $M|_{P_{NB}^C}$ le sont aussi.

Étape 6

Enfin, à partir de N_{NB}^{C-t} et du marquage $M_{rc}|_{P_{NB}^{C-t}}$ de la conséquence choisie, l'algorithme de recherche $A^\star BT$ (sous-partie 3.2.2)calcule l'ordonnancement du FMS au sein du $mode_{\mathcal{R}_j}$. Pour le calcul de l'ordonnancement dans ce nouveau mode, on suppose que $R'|_{P_{NB}^{C-t}}$ est un vecteur nul, le FMS attendant que toutes les opérations en cours soit terminées avant de reprendre son activité. Cette hypothèse sur R' est prise car le temps de calcul des différentes étapes de la méthode de prévention et d'ordonnancement peut-être significatif et réduire d'une valeur inconnue les éléments non nuls de R'. On note alors $\sigma(M, mode_{\mathcal{R}_j}) = \sigma_c\sigma_f$ l'ordonnancement complet calculé depuis M dans le $mode_{\mathcal{R}_j}$ avec σ_c la séquence reliant M à M_{rc} et σ_f l'ordonnancement calculé par la recherche $A^\star BT$ depuis $M_{rc}|_{P_{NB}^{C-t}}$ dans N_{NB}^{C-t}. Dans $\sigma(M, mode_{\mathcal{R}_j})$, R est omis puisque nul en cas de changement de mode.

Les étapes de 1 à 6 présentées dans cette sous-partie sont réitérées par le module de supervision dès qu'un nouveau changement de mode est observé. Au sein du mode actuel $Mode_j$, l'ordonnancement est calculé selon la méthode présentée dans le chapitre 3 (sous-partie 3.2.2) en appliquant la condition de ré-ordonnancement (déf. 3.3.2) au modèle N_{NB}^{C-t} construit lors de l'étape 4.

A l'instar des conséquences de cascades d'indisponibilités, des modèles N_B, N_{NB}, N_{NB}^C, N_{NB}^{C-t}, et des places de contrôle de (P_c^{NB}, F_c^{NB}), **tous les ordonnancements ayant lieu après un changement de mode sont calculés hors-ligne**. Ce calcul hors-ligne est réalisable en raison d'une part, du vecteur R devenant nul lorsqu'un changement de mode a lieu et d'autre part, du nombre fini de marquages des places d'entrée du FMS (P_0^e) grâce à l'équation définissant le marquage de ces places (éq. 3.2). Il existe donc "$|\mathcal{M}_{LZ}|\times(Nombre\ de\ modes\ non\ totalement\ bloquant)\times(Nombre\ de\ marquages\ des\ places\ d'entrée)$" ordonnancements à calculer. Ce calcul hors-ligne a pour principal intérêt l'accélération du calcul des profils d'attaquant dans un contexte incertain (voir sous-partie 4.2.3). Notre méthode d'ordonnancement dans un contexte incertain est donc une méthode dite "préventive réactive" (sous-partie 1.3.3) puisque tous les ordonnancements possibles sont calculés préventivement hors-ligne puis sont sélectionnés réactivement en-ligne lorsque des changements de mode ont lieu ou lorsque de nouveaux produits arrivent en entrée du FMS (condition de ré-ordonnancement, déf. 3.3.2). Toutes les conséquences, modèles et ordonnancements calculés hors-ligne sont intégrés directement dans le module de diagnostic et permettront la construction en-ligne des différents diagnostiqueurs.

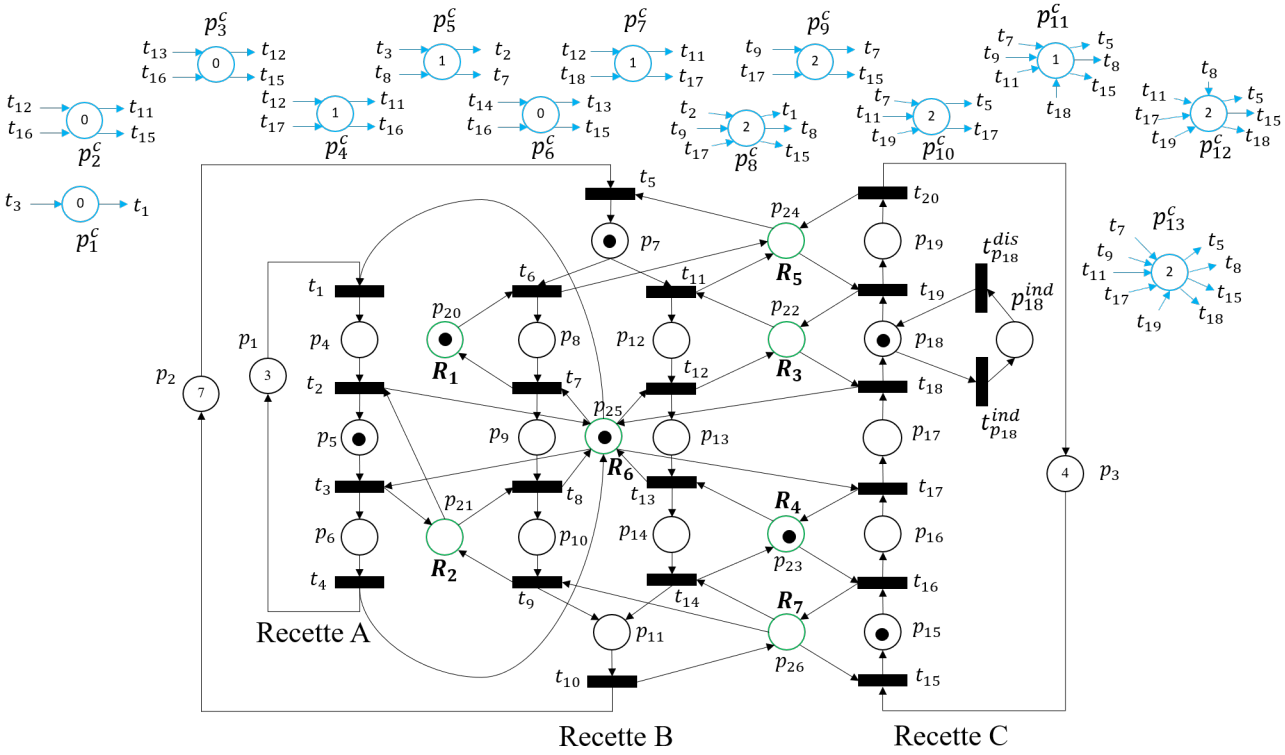


FIGURE 4.3 – Exemple du $Mode_0$ d'un FMS où la ressource R_3 peut devenir indisponible

4.1.3 Exemple

Soit l'exemple de la figure 4.3 représentant un FMS contrôlé fonctionnant dans le $Mode_0$ et un état $M = 3p_1^e + 7p_2^e + 4p_3^e + p_5 + p_7 + p_{15} + p_{18} + p_{20} + p_{23} + p_{25} + p_4^c + p_5^c + p_7^c + 2p_8^c + 2p_9^c + 2p_{10}^c + p_{11}^c + 2p_{12}^c + 2p_{13}^c$. A partir de cet état M, la ressource R_3 devient indisponible pendant l'opération de la place p_{18}, la transition $t_{p_{18}}^{ind}$ est franchie et la place p_{18}^{ind} devient marquée. Le changement de mode de $mode_0$ à $mode_{R_3}$ a lieu et est pris en compte par la méthode de prévention et d'ordonnancement en contexte incertain du module de supervision. Dans les prochains paragraphes, chaque étape de la re-configuration de cette méthode lors du changement de mode est présentée.

Premièrement, les moniteurs du $Mode_0$ de N sont conservés à l'identique selon les marquages illustrés dans la figure 4.3.

Deuxièmement, l'ensemble des conséquence de cascades d'indisponibilités sont calculées par l'algorithme présenté dans la sous-partie 4.1.2. Le modèle N_{R_3} est intégré à l'exemple 4.3 par la place p_{18}^{ind} avec $M|_{P_{R_3}}(p_{18}^{ind}) = 1$, $M|_{P_{R_3}}(p_{18}) = 0$ et $M|_{P_{R_3}}(p) = M(p)$ pour toute autre place p de N. Dans cet exemple, on a $H_{R_3} = \{p_{12}, p_{18}\}$, $P_{R_3}^{pre} = \{p_{12}, p_{15}, p_{16}, p_{17}, p_{18}\}$ et $p_{R_3}^0 = p_3$. L'algorithme de la sous-partie 4.1.2 permet d'obtenir sept composantes fortement connexes de $\mathcal{RG}(N_{R_3}, M|_{P_{R_3}}) = (V_r, F_r)$ permettant de distinguer sept conséquences de cascades d'indisponibilités différentes. Ces sept conséquences sont synthétisées dans le tableau 4.1. La première colonne représente les ressources bloquées, par indisponibilité ou cascades d'indisponibilités, la deuxième colonne le marquage racine de la composante fortement connexe associée à la conséquence, la troisième colonne les places activité bloquées par blocage partiel (P'_{DZ}), la quatrième colonne les recettes "vivantes" du FMS et enfin la cinquième colonne la taille des

TABLEAU 4.1 – Conséquences de cascades d'indisponibilité pour l'exemple 4.3

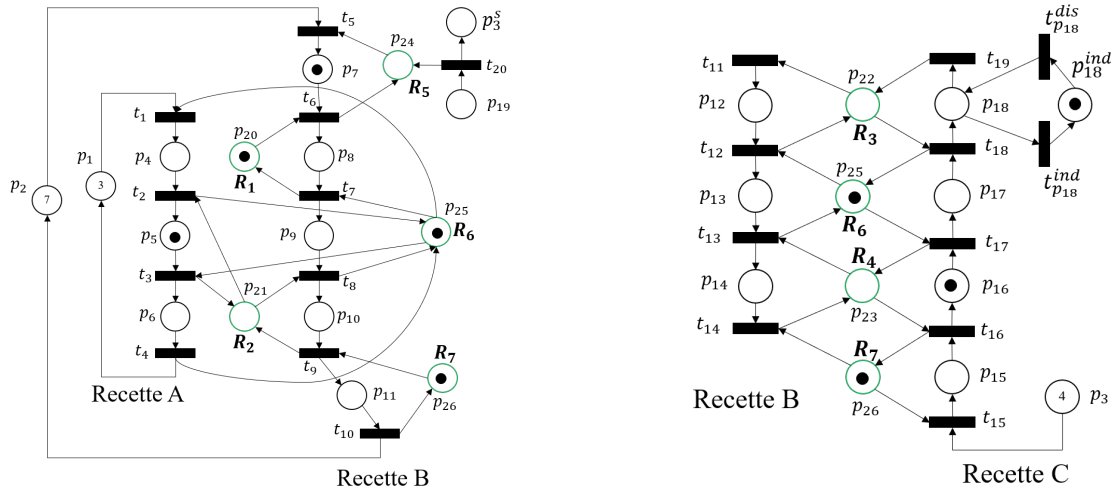
$\mathcal{R}'_k$	M_{rc}	P_{DZ}	C_{NB}	$\lvert\sigma\rvert$
[22,26,24]	$3p_1+7p_2+4p_3+p_5+p_7+p_{15}+p_{20}+p_{23}+p_{25}+p_4^c+p_5^c$ $+p_7^c+2p_8^c+2p_9^c+2p_{10}^c+p_{11}^c+2p_{12}^c+2p_{13}^c+p_{18}^{ind}$	/	1	0
[22,23,26,24]	$3p_1+7p_2+3p_3+p_5+p_7+p_{15}+p_{16}+p_{20}+p_{25}+p_4^c$ $+p_5^c+p_7^c+2p_8^c+2p_9^c+2p_{10}^c+p_{11}^c+2p_{12}^c+2p_{13}^c+p_{18}^{ind}$	/	1	2
[22,23,26,20]	$3p_1+7p_2+3p_3+p_5+p_8+p_{15}+p_{16}+p_{24}+p_{25}$ $+p_5^c+p_7^c+2p_8^c+2p_9^c+2p_{10}^c+p_{12}^c+p_{13}^c+p_{18}^{ind}$	/	1	3
[22,26,20]	$3p_1+7p_2+4p_3+p_4+p_8+p_{15}+p_{21}+p_{23}+p_{24}+p_4^c+2p_5^c$ $+p_7^c+p_8^c+2p_9^c+2p_{10}^c+p_{11}^c+2p_{12}^c+2p_{13}^c+p_{18}^{ind}$	/	1	1
[22,26,20,24]	$4p_1+6p_2+4p_3+p_7+p_8+p_{15}+p_{21}+p_{23}+p_{25}+p_1^c+p_4^c$ $+2p_5^c+p_7^c+2p_8^c+2p_9^c+p_{10}^c+p_{12}^c+p_{13}^c+p_{18}^{ind}$	/	1	2
[22,23]	$3p_1+6p_2+4p_3+p_6+p_7+p_8+p_{16}+p_{21}+p_{26}+p_1^c+p_2^c$ $+p_3^c+2p_5^c+p_6^c+p_7^c+2p_8^c+2p_9^c+p_{10}^c+p_{12}^c+p_{13}^c+p_{18}^{ind}$	/	[1 ;2]	1
[22,23,26]	$3p_1+8p_2+3p_3+p_6+p_{15}+p_{16}+p_{20}+p_{21}+p_{24}+p_1^c$ $+2p_5^c+p_7^c+p_8^c+p_9^c+3p_{10}^c+p_{11}^c+2p_{12}^c+2p_{13}^c+p_{18}^{ind}$	/	1	9

séquences de transitions minimum conduisant l'état $M|_{P_{R_3}}$ à l'état racine M_{rc}.

Troisièmement, en appliquant les consignes de choix d'une conséquence, seule la cinquième possède le maximum de recettes vivantes et est donc retenue. Pour cette conséquence, on a $\mathcal{R}'_k = \{p_{22}, p_{23}\}$, $H_{\mathcal{R}'_k} = \{p_{12}, p_{18}, p_{14}, p_{16}\}$, $P^{pre}_{\mathcal{R}'_k} = p_{15}, p_{16}, p_{17}, p_{18}, p_{12}, p_{13}, p_{14}$, $P^{post}_{\mathcal{R}'_k} = p_{19}$, $P'_{DZ} = \emptyset$, et le marquage des places précédentes est égal à $M|_{P^{pre}_{\mathcal{R}'_k}} = p_{16}$. Enfin, la première partie de l'ordonnancement du FMS dans ce nouveau mode consiste en la séquence de transitions $\sigma_c = t_{16}$ telle que $M|_{P_{R_3}}[\sigma_c > M_{rc}$.

Quatrièmement, les modèles N_B, N_{NB}, et N^C_{NB} et N^{C-t}_{NB} sont construits selon la conséquence choisie. Dans la figure 4.4, les modèles N_B, N_{NB} sont représentés. Dans cette figure, les places activités de $H_{\mathcal{R}'_k} \cup P^{pre}_{\mathcal{R}'_k}$ appartiennent à P_B tandis que le reste des places activités non bloquées appartiennent à P_{NB}. Une place de sortie p^s_3 a été ajoutée à N_{NB} à la suite des places de $P^{post}_{\mathcal{R}'_k} = p_{19}$ connectées à des transitions sans arc de sortie. Les places de contrôle de P_c sont conservées à l'identique au sein de ces deux modèles. Les modèles N^C_{NB} et N^{C-t}_{NB} sont construits en retirant de N_{NB} les places de $P^{post}_{\mathcal{R}'_k}$, les places de sortie p^s ainsi que les transitions et arcs associés à ces places. Dans l'exemple, les places p_{19}, p^s_3, la transition t_{20} et les arcs $\{(p_{19}, t_{20}), (t_{20}, p^s_3), (t_{20}, p_{24})\}$ sont ôtés de N_{NB} pour construire N^C_{NB}. Le modèle N^C_{NB} est un S^{3}PR car chacun de ses circuits vivants connecté aux ressources qu'il requiert peut être associé à un modèle S^{2}PR. Ainsi, la construction de N^{C-t}_{NB} suit la méthode de construction d'un modèle *SCnet* à partir d'un modèle S^{3}PR proposé dans la partie 3.2.2. Le modèle N^{C-t}_{NB} est illustré dans la figure 4.5.

Cinquièmement, à partir de N^C_{NB}, deux nouvelles places de contrôle sont construites par la méthode de prévention et sont illustrées dans la figure 4.5. Ces deux places de contrôle sont redondantes avec celles de N puisque la première correspond à p^c_1 et la seconde à p^c_5 dans 4.3. Une perspective à ces travaux pourrait avoir comme objectif l'optimisation du nombre de places de contrôle nécessaires à la méthode de prévention en supprimant les places redondantes.



(A) Modèle N_{NB} (B) Modèle N_B

FIGURE 4.4

Sixièmement, le modèle N_{NB}^{C-t} équipé des places de contrôle permet pour sa part le calcul de l'ordonnancement pour terminer la production des produits des recettes A et B. En étendant M_{rc} à $M_{rc}(p_1^e) = 3$ et $M_{rc}(p_2^e) = 7$ et en sachant que $M_{rc}(p_5) = 1$ et $M_{rc}(p_7) = 1$, le FMS doit donc terminer 4 produits de la recette A et 8 produits de la recette B. L'ordonnancement σ_f obtenu a une taille $|\sigma_f| = 61$ et un coût de 922s avec $K_{max} = 100$. L'ordonnancement complet généré lorsque le changement de mode a lieu est alors égal $\sigma(M, Mode_{R_3}) = \sigma_c\sigma_f$ avec $\sigma_c = t_{16}$ tel que $M|_{P_{R_3}}[\sigma_c > M_{rc}$.

4.2 Modèle des attaques de blocage et calcul des profils d'attaquant dans un contexte incertain

Dans cette deuxième partie, le modèle des attaques de blocage et le calcul des profils d'attaquant sont étendus au contexte incertain. Premièrement, les vulnérabilités des événements de changement de mode face aux attaques d'insertion et de suppression sont étudiées de manière exhaustive. Parmi toutes les attaques d'insertion et de suppression de ces événements, seules certaines sont retenues en raison de leur détectabilité non immédiate par les modules de supervision et de diagnostic. Dans une deuxième sous-partie, les attaques sélectionnées sont intégrées au modèle des attaques de blocage. Enfin, dans une troisième sous-partie, les profils d'attaquant sont calculés à partir de ce nouveau modèle, engendrant des trajectoires d'attaques différentes de celles calculées dans un contexte certain.

Les différents travaux présentés dans cette partie sont des contributions originales. Ils sont illustrés par la re-mobilisation (blocs grisés de la figure 4.6 ci-dessous) du diagramme d'activité UML du fonctionnement global du module de diagnostic proposé en introduction de ce chapitre (figure 4.1).

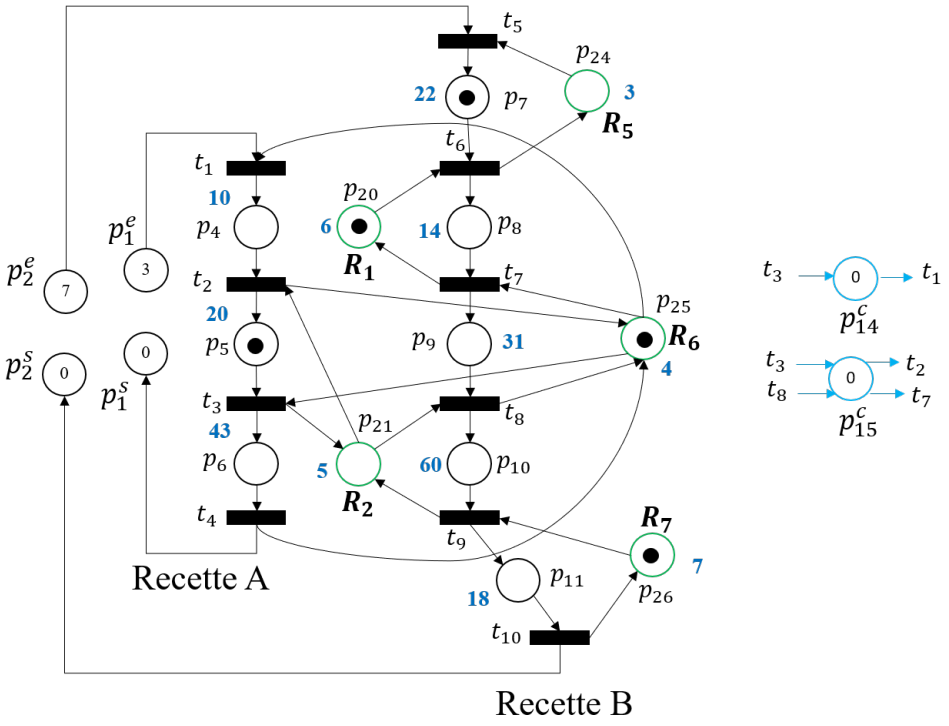


FIGURE 4.5 – Modèle N_{NB}^{C-t} de l'exemple de la figure 4.3

4.2.1 Vulnérabilités d'un changement de mode et attaques

Au sein du fonctionnement global du module de diagnostic dans un contexte incertain (figure 4.6), l'analyse des vulnérabilités d'un changement de mode et l'identification des attaques contre les événements d'indisponibilité sont requises afin d'étendre le modèle des attaques de blocage à ces événements. Cette analyse est menée dans cette sous-partie.

Un changement de mode entre $Mode_{\mathcal{R}_i}$ et $Mode_{\mathcal{R}_j}$ peut avoir lieu en raison de la défaillance d'une ressource ou d'une opération de maintenance préventive. Ces deux origines possibles à l'indisponibilité de r_k résultent en des événements observables et contrôlables échangés entre le module de supervision, les contrôleurs locaux et les ressources. Ces événements, présentés dans la partie 1.2.2 et la figure 1.8, sont vulnérables aux attaques d'insertion et de suppression lorsqu'ils transitent à travers des composants vulnérables du FMS. Par conséquent, un attaquant peut faire croire librement à G et S à la disponibilité ou à la non-disponibilité d'une ressource. Néanmoins, l'existence des modules de supervision et de diagnostic peuvent rendre inefficaces certaines de ces attaques lorsque ces dernières génèrent une incohérence détectable immédiatement par l'un de ces deux modules. Dans le but d'identifier et de sélectionner les attaques contre les événements de changement de mode pertinentes pour l'attaquant, et donc sournoises pour le module de diagnostic, le tableau 4.7 ci-dessous est proposé.

Dans ce tableau, les colonnes représentent l'état de disponibilité réel de la ressource r_k pour G et pour S et les lignes désignent l'état de disponibilité que souhaite faire croire l'attaquant à G et à S. En détail, l'état de disponibilité d'une ressource correspond pour S à "en maintenance" ou non et pour G à "défaillante" ou non. Dans chaque case, les effets de l'attaque correspondante sont illustrés par un schéma de communication entre la ressource r_k et S au milieu duquel

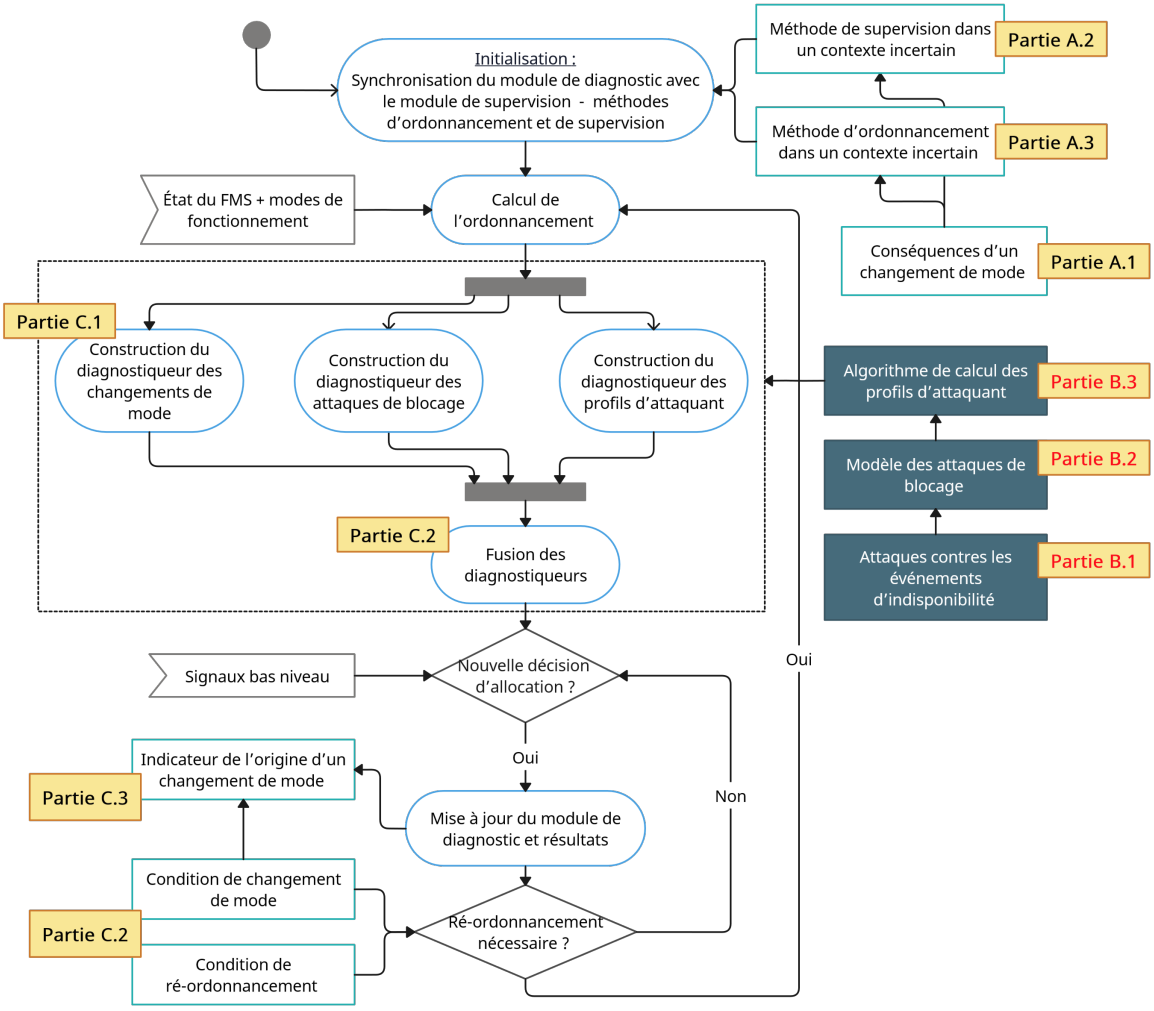


FIGURE 4.6 – Diagramme d'activité UML global du module de diagnostic

se glisse le bloc de l'attaquant modifiant les événements échangés entre S et r_k sur l'état de disponibilité de la ressource.

Par exemple, la case ligne 3, colonne 1 représente le cas où la ressource est disponible pour S (colonne 1), où aucune opération de maintenance préventive n'est prévue, mais où l'attaquant souhaite faire croire à sa cible, S (ligne 3), que l'état de r_k pour S (colonne 1) est indisponible (ligne 3 et 4). Ainsi, dans cette case, un *emoticon* vert est positionné à côté de S et une flèche verte est émise de S pour symboliser la disponibilité de r_k pour S, tandis qu'une flèche rouge est émise du bloc attaquant vers S pour représenter l'attaque et sa cible. Néanmoins, cette attaque n'est pas pertinente pour l'attaquant car détectable immédiatement par le module de supervision en raison de l'incohérence entre l'état de maintenance émis et l'état de maintenance manipulé reçu.

A l'instar de cet exemple, toutes **les cases en rouge** dans le tableau illustrent des attaques non pertinentes pour les mêmes raisons. En particulier, les cellules (ligne 4, colonne 2) et (ligne 2, colonne 4) sont rendues non pertinentes pour l'attaquant en raison de la présence du module de diagnostic dont le positionnement dans le schéma de communication entre S, r_k et l'attaquant est

illustré au centre du tableau. En effet, en supposant le module de diagnostic capable d'observer l'état réel des ressources (défaillantes ou non) grâce à son observabilité des événements échangés entre la ressource et son contrôleur local, l'attaquant ne peut pas utiliser la ressource lorsqu'elle est défaillante (ligne 2, colonne 4) ou ne pas l'utiliser pour défaillance alors qu'elle est disponible pour G (ligne 4, colonne 2) sans être détecté immédiatement par le module de diagnostic. Parmi les autres cellules du tableau, celles en **gris** ne sont pas étudiées car elles correspondent aux cas où l'attaquant ne manipule pas l'état réel de r_k pour G et pour S.

Pour leur part, **les cases vertes** restantes sont les plus pertinentes pour l'attaquant et sont étudiées en détail dans les points ci-dessous.

1. La cellule (ligne 4, colonne 1) modélise le cas où l'attaquant fait croire à G à l'indisponibilité de la ressource r_k pour des raisons de maintenance en ne communiquant plus d'événements de commande de $CTRL_k$ à r_k. Cette attaque n'est pas détectable par le module de diagnostic puisqu'une opération de maintenance est légitime pour ce dernier. Pour le module de supervision, l'attaquant le trompe en insérant les événements observables normalement émis par la ressource rendue non disponible par l'attaquant.
2. La cellule (ligne 3, colonne 2) représente l'attaque faisant croire au superviseur S à l'indisponibilité de la ressource r_k pour G en insérant les événements de défaillance de r_k. L'idée de cette attaque est de faire atteindre par G un état de la DZ du mode $Mode_{\mathcal{R}_j}$ dans lequel r_k est indisponible.
3. La cellule (ligne 2, colonne 3) décrit l'attaque qui en supprimant un ordre de maintenance continue d'utiliser la ressource r_k pour atteindre un état de la DZ. L'attaquant doit supprimer en conséquence les événements observables émis depuis r_k.

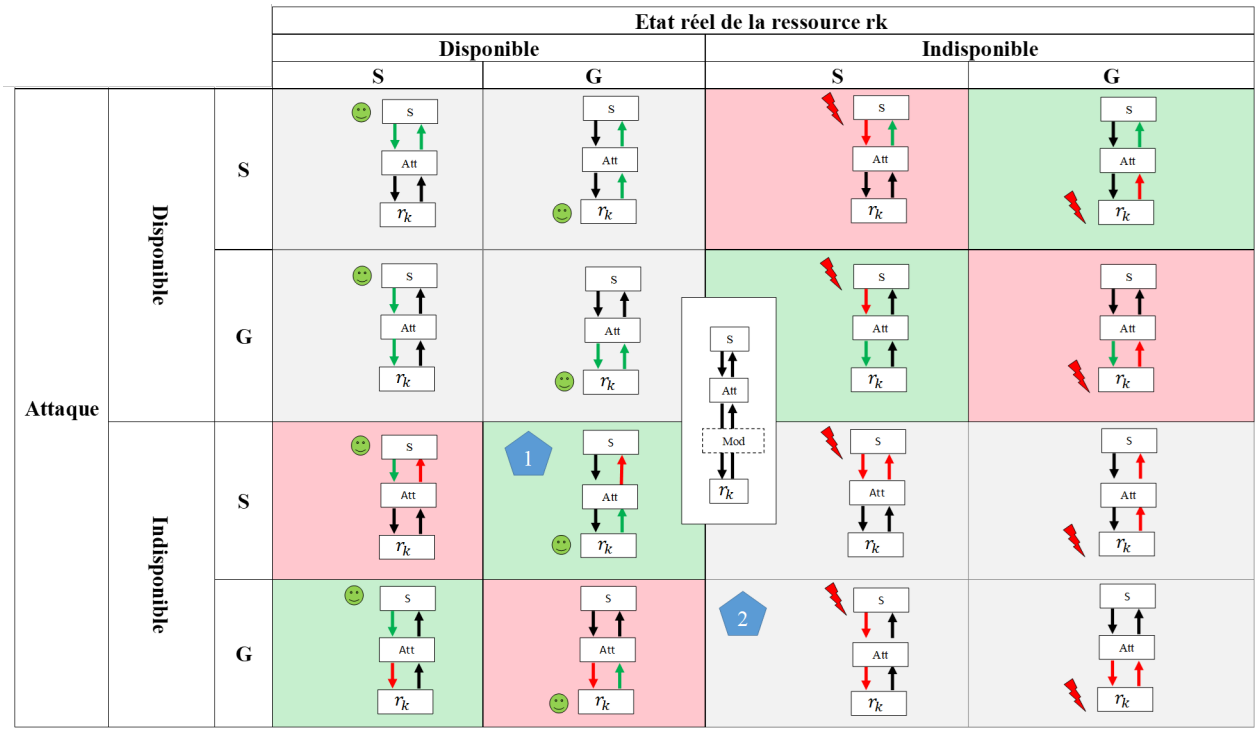


FIGURE 4.7 – Attaques contres les événements d'indisponibilité d'une ressource r_k dans un FMS

4. La cellule (ligne 1, colonne 4) illustre l'attaque de suppression de l'événement d'observation symbolisant la défaillance de r_k. Le module de supervision continue d'envoyer des événements de contrôle à r_k malgré son indisponibilité.

Parmi ces attaques, celles présentées dans les cellules (ligne 3, colonne 2), (ligne 1, colonne 4) et (ligne 4, colonne 1) ne sont pas retenues pour les attaques de blocage dans un contexte incertain. La première (ligne 3, colonne 2) est jugée non pertinente car, d'une part, elle serait détectée par les opérateurs programmant l'opération de maintenance et observant une non mise à l'arrêt de r_k et, d'autre part, elle ne permet pas à l'attaquant de modifier l'ensemble des états d'allocation atteignables dans G, altération pouvant lui permettre d'optimiser le coût de sa trajectoire vers un état de la DZ d'un nouveau mode $Mode_{\mathcal{R}_j}$. Similairement à la cellule (ligne 2, colonne 4), la deuxième attaque (ligne 1, colonne 4) n'a pas d'intérêt dans ces travaux puisqu'elle serait détectée par le module de diagnostic dès qu'un ordre serait envoyé de S à la ressource défaillante r_k. Enfin, la troisième attaque (ligne 4, colonne 1) n'est pas conservée car elle créée un état de blocage uniquement temporaire en supprimant de manière systématique les événements de contrôle envoyés depuis S à r_k et en insérant les événements d'observation attendus par S. Cette attaque est assimilable à une attaque de déni de service envers une ressource et ne créée pas de changement de mode stable permettant un blocage durable puisque si l'attaque s'arrête, la ressource est de nouveau disponible et le blocage de G n'est plus.

Ainsi, seule l'attaque de la cellule (ligne 3, colonne 2) est conservée. Cette dernière, combinée à l'attaque de la cellule (ligne 4, colonne 3) et indiquées toutes deux dans le tableau par les pentagones annotés des chiffres 1 et 2, a pour effet de faire changer de mode de fonctionnement le module de supervision en lui faisant croire à la défaillance de r_k, tout en ne permettant pas au module de diagnostic de détecter cette attaque puisque l'identification de ce changement de mode peut être assimilé par ce dernier à une opération de maintenance préventive. Cette attaque, exploitant la non communication entre les deux modules de supervision et de diagnostic, empêche l'un et l'autre de détecter l'attaque d'insertion de l'événement de défaillance de r_k.

Remarque 4.2.1. Dans le tableau 4.7, toutes les capacités d'un attaquant à manipuler les événements relatifs aux changements de mode ont été répertoriées. En effet, tous les cas ont été considérés en distinguant les états disponibles et indisponibles d'une ressource r_k pour G et pour S. ⌟

Dans la prochaine sous-partie (4.2.2), l'attaque d'insertion des événements d'observation de la défaillance d'une ressource est intégrée au modèle des attaques de blocage N_α.

4.2.2 Modèle des attaques de blocage dans un contexte incertain

L'objectif de cette sous-partie est d'étendre le modèle des attaques de blocage N_α au contexte incertain des FMSs. Au sein du fonctionnement global du module de diagnostic (figure 4.6), ce modèle d'attaque est requis par l'algorithme de calcul des profils d'attaquant.

Dans la précédente sous-partie 4.2.1, parmi les attaques contre les événements relatifs à l'indisponibilité des ressources, l'attaquant cible les événements d'observation d'une défaillance de ressource envoyés d'un contrôleur local vers le superviseur. Afin de modéliser ces attaques, les

transitions et places d'indisponibilité doivent dans un premier temps être intégrées à N_{GS} afin de créer un nouveau modèle $N^{GS}_{\mathcal{R}_{ind}}$. Dans un second temps, les transitions modélisant le type d'attaque retenu sont ajoutées à $N^{GS}_{\mathcal{R}_{ind}}$ afin d'obtenir le modèle d'attaque $N^{a}_{\mathcal{R}_{ind}}$. Ce modèle est finalement fusionné avec N_α pour obtenir un modèle complet des attaques de blocage noté $N^{\alpha}_{\mathcal{R}_{ind}}$. Ces différents modèles sont construits à partir de N, $N_{\mathcal{R}_{ind}}$ et N_α directement sans avoir recours aux extensions incluant tous les événements propres à l'allocation des ressources.

La construction de $N^{GS}_{\mathcal{R}_{ind}} = (P^{GS}_{\mathcal{R}_{ind}}, T^{GS}_{\mathcal{R}_{ind}}, F^{GS}_{\mathcal{R}_{ind}}, E^{GS}_{\mathcal{R}_{ind}}, l^{GS}_{\mathcal{R}_{ind}})$ repose sur les étapes suivantes :

- Ajout des sous-réseaux (N_{r_k}, M_{r_k0}) et (N_{p_j}, M_{p_j0}) aux modèles N_G et N_S. Ces sous-réseaux sont ajoutés $\forall r_j \in \mathcal{R}_{ind}$ et $\forall p_j \in H_{\mathcal{R}_{ind}}$. Ces deux nouveaux modèles sont notés $N^{G}_{\mathcal{R}_{ind}}$ et $N^{S}_{\mathcal{R}_{ind}}$;
- Construction de $N^{GS}_{\mathcal{R}_{ind}}$ en fusionnant $N^{G}_{\mathcal{R}_{ind}}$ et $N^{S}_{\mathcal{R}_{ind}}$ (annexe 6.2). Les transitions sont mises en commun tandis que les places et arcs propres sont conservés. Ainsi, on obtient pour chaque place d'indisponibilité deux places p_r^{indG}, p_r^{indS} ou p_j^{indG}, p_j^{indS} pour G et S ;
- Initialement le marquage initial des places d'indisponibilité est nul, i.e. $M_0^{\mathcal{R}_{ind}}(p_r^{indG}) = M_0^{\mathcal{R}_{ind}}(p_r^{indS}) = M_0^{\mathcal{R}_{ind}}(p_j^{indG}) = M_0^{\mathcal{R}_{ind}}(p_j^{indS}) = 0$.
- Chaque transition $t_p^{ind} \in T^{GS}_{\mathcal{R}_{ind}}$ (resp. $t_p^{dis} \in T^{GS}_{\mathcal{R}_{ind}}$) avec $p \in (p_{r_k} \cup H_{r_k})$ est labellisée par un événement $e^{ind}(r_k) \in E^{GS}_{\mathcal{R}_{ind}}$ (resp. $e^{dis}(r_k) \in E^{GS}_{\mathcal{R}_{ind}}$).

Dans le modèle $N^{GS}_{\mathcal{R}_{ind}}$, on considère donc que les événements de défaillance ayant lieu dans G sont transmis correctement à S et réciproquement. Les événements relatifs aux opérations de maintenance et aux défaillances ne sont pas différenciés car ils conduisent tous deux à l'indisponibilité des mêmes ressources.

Les attaques contre les événements observables de défaillance d'une ressource ciblent l'ensemble des ressources de $\mathcal{R}_{ind}$. D'après le tableau 4.7, seule l'insertion de ces événements est considérée afin de faire croire au superviseur à la défaillance d'une ressource. La création des transitions modélisant ce type d'attaque est donc similaire à celle des transitions modélisant l'insertion malveillante d'un événement observable présentée dans la partie 3.1.1. Ces nouvelles transitions sont définies dans $N^{a}_{\mathcal{R}_{ind}} = (P^{a}_{\mathcal{R}_{ind}}, T^{a}_{\mathcal{R}_{ind}}, F^{a}_{\mathcal{R}_{ind}}, E^{a}_{\mathcal{R}_{ind}}, l^{a}_{\mathcal{R}_{ind}})$ par :

- $\forall t_p^{ind} \in T^{GS}_{\mathcal{R}_{ind}}$, une transition $t_{p+}^{ind} \in T^{a}_{\mathcal{R}_{ind}}$ est créée. On note T_+^{ind} l'ensemble de ces transitions ;
- $\forall t_p^{dis} \in T^{GS}_{\mathcal{R}_{ind}}$, une transition $t_{p+}^{dis} \in T^{a}_{\mathcal{R}_{ind}}$ est créée. On note T_+^{dis} l'ensemble de ces transitions ;
- Ces deux transitions sont connectées aux places de $P^{GS}_{\mathcal{R}_{ind}}$ par les arcs suivants : (p^S, t_{p+}^{ind}), (t_{p+}^{ind}, p^{indS}), (p^{indS}, t_{p+}^{dis}), (t_{p+}^{dis}, p^S). Ici, p^S est la place p dans N_S.
- Une transition t_{p+}^{ind} (resp.t_{p+}^{dis}) est labellisée par $e^{ind}_{DEF}(r_k)+ \in E^{a}_{\mathcal{R}_{ind}}$ (resp. $e^{dis}_{DEF}(r_k)+ \in E^{a}_{\mathcal{R}_{ind}}$). L'indicateur DEF est choisi pour désigner les événements d'indisponibilité de r_k relatifs à une défaillance simulée de la ressource ;
- Similairement à N_α et aux SC-nets (voir partie 3.2.2), les places d'attente de $N^{a}_{\mathcal{R}_{ind}}$ sont décomposées en places d'entrée et en places de sortie pour G et S, soit les ensembles P_E^G, P_S^G, P_E^S et P_S^S.

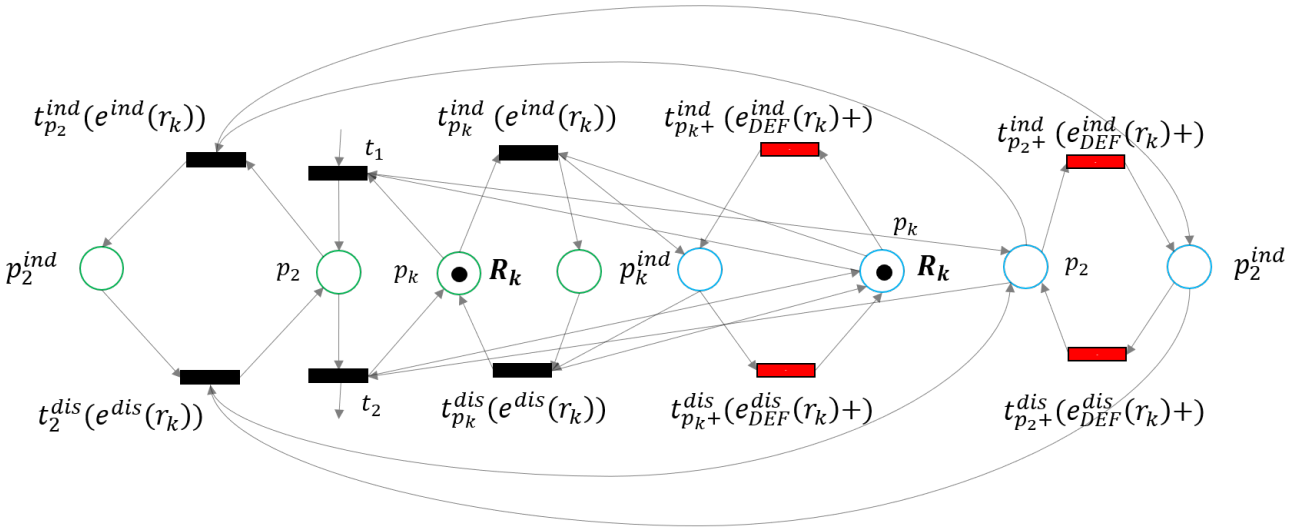


FIGURE 4.8 – Extrait simplifié du modèle $N^a_{\mathcal{R}_{ind}}$

L'ajout des transitions t^{ind}_{p+} et t^{dis}_{p+} est illustré dans la figure 4.8 ci-contre. Ce modèle permet d'illustrer les conséquences des attaques $e^{ind}_{DEF}(r_k)+$ sur l'allocation des ressources par le FMS. Ainsi, en insérant l'événement $e^{ind}_{DEF}(r_k)+$, l'attaquant vide les places p^S_k et p^S_2 et fait croire au superviseur à l'indisponibilité par défaillance de r_k en marquant les places p^{indS}_k et p^{indS}_2. Conséquemment, la transition t_2 n'est plus franchissable, condamnant dans G l'évolution des produits présents dans p^G_2 ou dans un place antérieure.

Enfin, le modèle N_α ayant toutes ses places incluses dans le modèle $N^a_{\mathcal{R}_{ind}}$ ($P_\alpha = P_{GS} \subset P^a_{\mathcal{R}_{ind}}$), ces deux modèles peuvent être fusionnés au sein d'un modèle $N^\alpha_{\mathcal{R}_{ind}} = (P^\alpha_{\mathcal{R}_{ind}}, T^\alpha_{\mathcal{R}_{ind}}, F^\alpha_{\mathcal{R}_{ind}}, E^\alpha_{\mathcal{R}_{ind}}, l^\alpha_{\mathcal{R}_{ind}}, Ct^\alpha_{\mathcal{R}_{ind}}, D^\alpha_{\mathcal{R}_{ind}})$ de la manière suivante :

- $P^\alpha_{\mathcal{R}_{ind}} = P^a_{\mathcal{R}_{ind}}$;
- $T^\alpha_{\mathcal{R}_{ind}} = T^a_{\mathcal{R}_{ind}} \cup T^\alpha_+ \cup T^\alpha_-$;
- $F^\alpha_{\mathcal{R}_{ind}} = F^a_{\mathcal{R}_{ind}} \cup F^\alpha_+ \cup F^\alpha_-$;
- $E^\alpha_{\mathcal{R}_{ind}} = E^a_{\mathcal{R}_{ind}} \cup E_\alpha$;
- $l^\alpha_{\mathcal{R}_{ind}} : T^\alpha_{\mathcal{R}_{ind}} \to E^\alpha_{\mathcal{R}_{ind}}$ est définie par $l^\alpha_{\mathcal{R}_{ind}}(t) = l_\alpha(t)$ si $t \in T_\alpha = T \cup T^\alpha_+ \cup T^\alpha_-$ et $l^\alpha_{\mathcal{R}_{ind}}(t) = l^a_{\mathcal{R}_{ind}}(t)$ si $t \in T^a_{\mathcal{R}_{ind}} \setminus T$;
- La fonction coût $Ct^\alpha_{\mathcal{R}_{ind}}$ est définie dans le prochain paragraphe ;
- $D^\alpha_{\mathcal{R}_{ind}} = D_\alpha$;
- Un marquage initial $M_0^{\mathcal{R}_{ind}\alpha}$ de $N^\alpha_{\mathcal{R}_{ind}}$ est acceptable si $M_0^{\mathcal{R}_{ind}\alpha}|_{P_\alpha}$ et $M_0^{\mathcal{R}_{ind}\alpha}|_{P^a_{\mathcal{R}_{ind}}}$ sont acceptables pour leurs RdPs respectifs.

Les transitions t^{ind}_{p+} et t^{dis}_{p+} intégrées au modèle $N^\alpha_{\mathcal{R}_{ind}}$ possèdent des coûts de réalisation pour un attaquant. La fonction coût Ct^α est ainsi étendue à ces nouvelles transitions par la fonction $Ct^\alpha_{\mathcal{R}_{ind}} : E^\alpha_{\mathcal{R}_{ind}} \to \mathbb{N}$. Cette fonction coût unitaire doit respecter les mêmes contraintes présentées dans la partie 3.1.2. Au sein des contraintes 1 et 2, les événements $e^{ind}_{DEF}(r_k)+$ et

$e^{dis}_{DEF}(r_k)+$ sont associés à la ressource r_k. Ainsi, pour deux ressources r_1 et r_2 avec r_1 plus vulnérables que r_2, on a $Ct(e^{ind}_{DEF}(r_1)+) < Ct(e^{ind}_{DEF}(r_2)+)$ et $Ct(e^{dis}_{DEF}(r_1)+) < Ct(e^{dis}_{DEF}(r_2)+)$. La fonction $CT^{\alpha}_{\mathcal{R}_{ind}} : E^{\alpha\star}_{\mathcal{R}_{ind}} \to \mathbb{N}$ est elle aussi définie pour évaluer sous la forme d'une somme de coûts unitaires le coût total d'un séquence d'événements ayant lieu dans $N^{\alpha}_{\mathcal{R}_{ind}}$. Ces nouvelles fonctions coût sont sollicitées dans la prochaine partie lors du calcul des profils d'attaquant dans un contexte incertain.

4.2.3 Profils d'attaquant dans un contexte incertain

L'objectif de cette sous-partie est d'adapter la définition des profils d'attaquant et l'algorithme de calcul des profils au contexte incertain des FMSs. Au sein du fonctionnement global du module de diagnostic (figure 4.6), cet algorithme est requis pour la construction du diagnostiqueur des profils d'attaquant.

Les différents profils d'attaquant présentés dans la partie 3.1.1 et simulés dans la partie 3.2.2 obtiennent dans un contexte incertain la capacité d'insérer les événements $e^{ind}_{DEF}(r_k)+$ et $e^{ind}_{DEF}(r_k)+$. En utilisant ces nouvelles attaques conjointement à celles introduites dans le chapitre 3, les profils d'attaquant peuvent atteindre un état critique ciblé en respectant leurs contraintes respectives de manière plus optimale que dans un contexte certain. Néanmoins, afin de calculer les trois profils d'attaquant à partir de $N^{\alpha}_{\mathcal{R}_{ind}}$, des modifications à l'algorithme $A^{\star}\mathcal{P}$ (sous-partie 3.2.3, annexe 6.6) et aux différentes fonctions heuristiques sont nécessaires. Ces modifications sont présentées et illustrées par un exemple dans les trois prochaines sous-parties.

Modifications de l'algorithme de calcul des profils d'attaquant

Dans un contexte incertain, cinq modifications sont apportées à l'algorithme $A^{\star}\mathcal{P}$ (sous-partie 3.2.3, annexe 6.6). L'algorithme complet incluant ces modifications est présenté et détaillé dans l'annexe 6.13.

Premièrement, le modèle utilisé en entrée de l'algorithme et pendant l'exploration des trajectoires d'attaque devient $N^{\alpha}_{\mathcal{R}_{ind}}$.

Deuxièmement, le mode dans lequel le FMS se trouve est pris en compte pour le calcul des profils par le module de diagnostic. Le mode actuel est intégré à l'algorithme par le marquage en entrée M_{init} des places p^{indG}_r et p^{indG}_j d'indisponibilité des ressources et par l'ordonnancement σ calculé depuis ce mode actuel par la méthode d'ordonnancement présentée dans la partie précédente.

Troisièmement, l'ensemble T_{fr} des transitions franchissables par l'attaquant à partir d'un marquage $M \in \mathcal{R}(N^{\alpha}_{\mathcal{R}_{ind}}, M_{init})$ est étendu par l'ajout des transitions $(T^{ind}_{+} \cup T^{dis}_{+})$ des attaques d'insertion des événements d'observation de défaillance et de reprise d'une ressource. A l'inverse, on suppose que les transitions d'indisponibilités naturelles, $T_{\mathcal{R}^{GS}_{ind}} \setminus T$, ne sont pas franchissables au sein d'un profil et n'appartiennent pas à T_{fr}. En d'autres termes, **l'attaquant suppose qu'aucune indisponibilité par défaillance d'une ressource ou par ordre de maintenance ne peut avoir lieu lors de son attaque**. L'ensemble T_{fr} est d'autre part calculé selon les deux hypothèses suivantes :

- Si une ou plusieurs décisions d'allocation de l'ordonnancement sont instantanées depuis un état (M, R) $(\Gamma = 0)$, ces dernières et leurs suppressions par l'attaquant sont prioritaires

sur les attaques $e^{ind}_{DEF}(r_k)+$ et $e^{dis}_{DEF}(r_k)+$. Par conséquent, selon les ressources requises et libérées par ces décisions d'allocation supprimées, les transitions d'attaque franchissables de $T_{fr} \cap (T^{ind}_{+} \cup T^{dis}_{+})$ doivent être mises à jour ;

- L'objectif premier des attaques de blocage, à savoir d'atteindre un état de la DZ à travers la création d'une condition d'attente circulaire entre des produits, exclue les blocages malveillants créés par un mode correspondant à l'indisponibilité d'un ensemble de ressources requises par toutes les recettes du FMS. Par conséquent, les transitions de T^{ind}_{+} permettant d'atteindre un tel mode depuis un état $M \in \mathcal{R}(N^{\alpha}_{\mathcal{R}^{\alpha}_{ind}}, M_{init})$ ne sont pas franchissables par l'attaquant dans T_{fr}.

Quatrièmement, si un attaquant décide de réaliser une attaque de changement de mode, l'état M' atteint depuis un état M de $OPEN$ en franchissant une transition t^{ind}_{p+} ou t^{dis}_{p+} génère un nouvel ordonnancement par le module de supervision. Ce dernier est calculé à partir de M' et du modèle N^{C-t}_{NB} construit selon la méthode présentée dans la partie précédente (4.1.2). Ce nouvel ordonnancement est nécessaire au calcul des profils d'attaquant lorsque, ultérieurement à M', des attaques de suppression de décisions d'allocation $e^{\alpha}(O_i, R_k)-$ doivent être réalisées par l'attaquant pour rester sournois. Pour rappel, en raison du changement de mode, R' est défini par le vecteur nul $0_{1\times|P^{\alpha}_{\mathcal{R}_{ind}}|}$.

Cinquièmement, les ensembles $\mathcal{M}_c$ de chaque profil doivent inclure les états critiques relatifs au mode dans lequel l'état M appartient. Ce mode peut être déterminé à partir du marquage des places de $P^{S}_{\mathcal{R}_{ind}}$ dans M. A partir de ce mode et du marquage $M|_{P^G}$ des places de G, le modèle N^{C}_{NB} est construit selon la méthode introduite dans la partie précédente (4.2). Les ensembles $\mathcal{M}_c$ sont alors définis à partir de N^{C}_{NB} par :

- $\mathcal{M}_{c_1} = \{M \in \mathcal{R}(N^{C}_{NB}, M^{C}_{0,NB}) | \nexists t \in T^{C}_{NB}, M' \in \mathcal{R}(N^{C}_{NB}, M^{C}_{0,NB})\ t.q.\ M[t > M'\}$;
- $\mathcal{M}_{c_2} = \{M \in \mathcal{R}(N^{C}_{NB}, M^{C}_{0,NB}) | \nexists t \in T^{C}_{NB}, M' \in \mathcal{R}(N^{C}_{NB}, M^{C}_{0,NB})\ t.q.\ M[t > M'\} \cup \{M \in \mathcal{R}(N^{C}_{NB}, M^{C}_{0,NB}) | \forall M' \in \mathcal{R}(N^{C}_{NB}, M^{C}_{0,NB}), \sigma \in T^{C\star}_{NB}\ t.q. M[\sigma > M' \Rightarrow M' \in \mathcal{M}_{c_1} \cup \mathcal{M}_{c_2}\}$;
- et $\mathcal{M}_{c_3} = \mathcal{M}_{DZ} \setminus \mathcal{M}_{c_2}$.

Nouvelles heuristiques

Parmi les heuristiques introduites pour le calcul des trois profils d'attaquant dans un contexte certain, les heuristiques $h1$, h^a_2, $h^b_2 = h^a_3$ et h^b_3 doivent être modifiées. Les heuristiques $h^b_2 = h^a_3$ doivent être mises à jour avec la nouvelle fonction coût unitaire $Ct^{\alpha}_{\mathcal{R}_{ind}}$ et les nouvelles attaques considérées dans $T^{ind}_{+} \cup T^{dis}_{+}$.

Le mode de fonctionnement dans lequel un marquage M se trouve influe sur les fonctions coût et heuristique du profil 1 au regard de la maximisation de la détention des ressources dans M (caractéristique 3.b) et à la minimisation des états de pré-blocage et de blocage partiel traversés par le profil (caractéristique 3.a). Précisément, dans cette version modifiée de l'algorithme $A^{\star}\mathcal{P}$, l'ensemble U_{max} est défini par $U_{max} = \min_{M_{c_1} \in \mathcal{M}_{c_1}}(M_{c_1}(P_R \cap P^{C}_{NB}))$ et l'ensemble $\mathcal{M}_{c_2} \setminus \mathcal{M}_{c_1}$ à éviter est calculé à partir du modèle N^{C}_{NB} et de $M|_{N^{C}_{NB}}$.

Les heuristiques h_1, h^a_2 et h^b_3 sont pour leur part altérées par l'existence dans les différents modes de fonctionnement d'ensembles distincts de siphons stricts minimaux que l'attaquant cherche à vider. Pour un marquage $M \in \mathcal{R}(N^{\alpha}_{\mathcal{R}_{ind}}, M_0^{R_{ind}{}^{\alpha}})$ donné, ces trois heuristiques doivent prendre en compte le mode de fonctionnement dans lequel se trouve le FMS. Ainsi, l'ensemble Π_{SMS} doit être calculé à partir du modèle N^{C}_{NB} représentant le sous S^3PR vivant du FMS

obtenu depuis l'état M modélisant en son sein la disponibilité des différentes ressources du FMS et après l'application des conséquences de cascade d'indisponibilité depuis M (voir partie 4.1.2). De la même manière, les ensembles $\mathcal{M}_{c_1} \cup \mathcal{M}_{c_2}$ requis dans h_2^a ainsi que l'ensemble $\mathcal{M}_{c_3}$ utilisé pour le calcul de K_{LL} dans h_3^b doivent être construits depuis ce modèle N_{NB}^C et $M|_{N_{NB}^C}$.

A partir des modifications apportées à l'algorithme $A^\star P$ et aux différentes heuristiques, le calcul des trois profils d'attaquant dans un contexte incertain est illustré par un exemple dans la prochaine sous-partie.

Exemple

Le calcul des profils d'attaquant dans un contexte incertain est réalisé à partir de l'exemple conducteur du chapitre 4 dont l'état M est illustré dans la figure 4.3. Toutes les ressources peuvent devenir indisponibles, i.e. $\mathcal{R}_{ind} = \{R_1, R_2, R_3, R_4, R_5, R_6, R_7\}$. Par conséquent, l'attaquant est capable de faire croire à l'indisponibilité par défaillance ($e_{DEF}^{ind}(r_k)+$) ou à la reprise ($e_{DEF}^{dis}(r_k)+$) de n'importe quelle ressource r_k du FMS hormis la ressource R_6, cette dernière bloquant tout le FMS en cas d'indisponibilité. Préalablement au calcul des profils, les coûts de chaque attaque $e_{DEF}^{ind}(r_k)+$ et $e_{DEF}^{dis}(r_k)+$ doivent être définis. Ces coûts sont présentés dans le tableau 4.2. Les symboles "" désignent les événements $e_{DEF}^{ind}(r_k)+$ et $e_{DEF}^{dis}(r_k)+$ définis pour chaque ressource $r_k \in \mathcal{R}_{ind}$.

Tableau 4.2 – Coût des attaques d'indisponibilité et de disponibilité des ressources

Ressources	R_1		R_2		R_3		R_4		R_5		R_6		R_7	
Evenement	$e_{DEF}^{dis}(R_1)+$	$e_{DEF}^{ind}(R_1)+$	$e_{DEF}^{dis}(R_2)+$	$e_{DEF}^{ind}(R_2)+$	""	""	""	""	""	""	""	""	""	""
Coût	35	37	45	47	40	43	40	43	42	46	48	51	37	41

Les conditions initiales au calcul des profils d'attaquants sont : $K_{max} = 100$, $M = 3p_1^e + 7p_2^e + 4p_3^e + p_5 + p_7 + p_{15} + p_{18} + p_{20} + p_{23} + p_{25} + p_4^c + p_5^c + p_7^c + 2p_8^c + 2p_9^c + 2p_{10}^c + p_{11}^c + 2p_{12}^c + 2p_{13}^c$, $R(p_5) = 8$, $R(p_7) = 12$, $R(p_{15}) = 4$ et R nul pour toutes les autres places. L'ordonnancement partiel obtenu à partir de cet état est $\sigma = t_3 t_6 t_{16} t_5 t_{15} t_4$. Les profils d'attaquant $\mathcal{P}_1, \mathcal{P}_2$ et $\mathcal{P}_3$ calculés dans un contexte incertain à partir de l'algorithme $A^\star\mathcal{P}$ modifié (sous-partie 4.2.3) en prenant ces conditions initiales en entrée sont :

$$\sigma_{\mathcal{P}_1(M,R)} = t_{p_{20}+}^{ind} t_{1+}$$

$$\sigma_{\mathcal{P}_2(M,R)} = t_{p_{20}+}^{ind} t_{p_5+}^{ind}$$

$$\sigma_{\mathcal{P}_3(M,R)} = t_3 t_6 t_{16} t_{p_8+}^{ind} t_{5+}$$

Ces profils ont des coûts respectifs $c_1 = 1$, $c_2 = 72$ et $c_3 = 1960$. Les profils 2 et 3 respectent les contraintes de coût de l'attaque puisque $CT_{\mathcal{R}_{ind}}^{\alpha}(\mathcal{P}_2(M,R)) = 318 < C_{max}^2 = 400$ et $CT_{\mathcal{R}_{ind}}^{\alpha}(\mathcal{P}_3(M,R)) = 122 < C_{max}^3 = 800$. En comparaison, les $\mathcal{P}_1, \mathcal{P}_2$ et $\mathcal{P}_3$ générés dans un contexte certain à partir de M sont :

$$\sigma_{\mathcal{P}_1(M,R)} = t_{3-} t_{6-} t_{16-} t6 + t_5 t_{15-} t_{16+} t_{15+} t_{1+}$$

$$\sigma_{\mathcal{P}_2(M,R)} = t_{1+}$$

$$\sigma_{\mathcal{P}_3(M,R)} = t_3 t_6 t_{16} t_{5-} t_{15-} t_{19+} t_4 t_7 t_{20+} t_{6-} t_{5+} t_{19-} t_{11+}$$

Ces trois profils calculés dans un contexte sans indisponibilité ont des coûts respectifs $c_1 = 5$, $c_2 = 97$ et $c_3 = 60605$. Ces coûts sont tous supérieurs aux coûts des profils calculés dans le contexte incertain montrant l'intérêt pour l'attaquant de réaliser des attaques d'indisponibilité des ressources. Cette observation peut être expliquée par le fait que l'indisponibilité d'une ressource réduit l'ensemble des circuits vivants d'un FMS, dégradant sa flexibilité et créant par conséquent de nouveaux états de blocage, de pré-blocage et de blocage partiel. Par exemple, au sein des profils 1 et 2, l'attaque d'indisponibilité de la ressource R_1 (t^{ind}_{p20+}) permet de créer immédiatement un état de blocage entre les recettes B et C puisque les produits en p_7 et p_{18} requièrent mutuellement la ressource détenue par l'autre. En effectuant une attaque d'indisponibilité, l'attaquant est ainsi capable de bloquer trois circuits du FMS.

4.3 Diagnostic des attaques de blocage dans un contexte incertain

Dans cette dernière partie, le diagnostiqueur des attaques de blocage dans un contexte incertain est construit à partir des méthodes, modèles et algorithmes présentés dans les deux premières parties de ce chapitre. Premièrement, le diagnostic des changements de mode par le module de diagnostic est défini et un diagnostiqueur est proposé (sous-partie 4.3.1). Deuxièmement, un diagnostiqueur global des attaques de blocage et des profils d'attaquant dans un contexte incertain est développé et intègre le diagnostic des changements de mode (sous-partie 4.3.2). Enfin, face à l'incapacité du diagnostiqueur à distinguer l'origine naturel ou malveillante d'un changement de mode, des indicateurs de malveillance d'un changement de mode sont proposés (sous-partie 4.3.3).

Les différents travaux présentés dans cette partie sont des contributions originales. Ils sont illustrés par la re-mobilisation (blocs grisés de la figure 4.9 ci-dessous) du diagramme d'activité UML du fonctionnement global du module de diagnostic proposé en introduction de ce chapitre (figure 4.1).

4.3.1 Diagnostic d'un changement de mode

L'objectif de cette sous-partie est le développement d'un diagnostiqueur des changements de mode. Au sein du fonctionnement global du module de diagnostic (figure 4.9), ce diagnostiqueur est requis pour la construction du diagnostiqueur commun des attaques de blocage, des profils d'attaquant et des changements de mode dans un contexte incertain.

Présentation du Diagnostiqueur

Dans un contexte incertain, la prévention des états de la DZ et l'ordonnancement diffèrent selon le mode dans lequel le FMS se trouve (partie 4.1). Or, il a été montré dans le chapitre 3

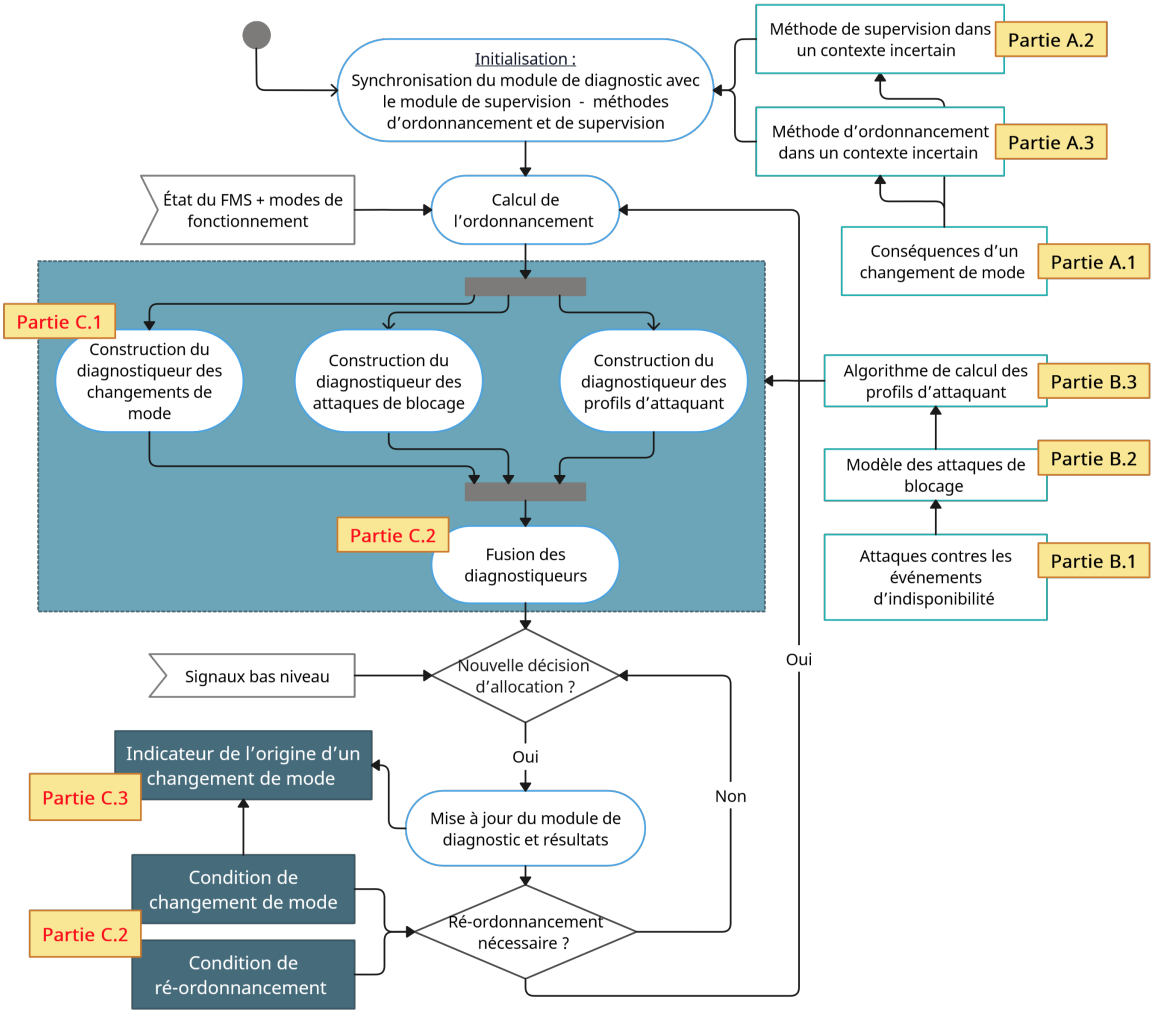


FIGURE 4.9 – Diagramme d'activité UML global du module de diagnostic

que le module de diagnostic requiert la connaissance des méthodes de prévention et d'ordonnancement du module de supervision afin de pouvoir diagnostiquer les attaques de blocage et les différents profils d'attaquant. Par conséquent, dans un contexte incertain, le module de diagnostic doit être capable d'identifier le mode dans lequel se trouve le FMS afin de générer les états critiques et optimaux du mode nécessaires au diagnostic des attaques. Le diagnostic des changements de mode est ainsi ajouté à ses missions.

Un diagnostiqueur $Diag_m$ est développé pour répondre à cette mission. Ce dernier a pour objectif d'évaluer pour un état $M \in \mathcal{R}(N_t, M_0^t)$ (modèle SC-net) observable par le module de diagnostic les modes dans lequel cet état pourrait être. Préalablement à la présentation détaillée de $Diag_m$ et de l'algorithme dédié à sa construction, les quatre hypothèses suivantes sur les changements de modes sont prises. Ces quatre hypothèses se focalisent sur les changements de mode provoqués par des opérations de maintenance préventive puisque le module de diagnostic est capable par son positionnement de détecter immédiatement les indisponibilités par défaillance (sous-partie 4.2.1).

Premièrement, lors d'un changement de mode par opération de maintenance, une unique ressource devient indisponible ou de nouveau disponible. Cette hypothèse se fonde sur le postulat que deux ressources sont mises à l'arrêt ou redémarrées l'une après l'autre et non simultanément dans le but de s'assurer de la bonne application des ordres d'arrêt ou de reprise émis par les opérations de maintenance sur ces ressources.

Deuxièmement, lors d'un changement de mode par opération de maintenance, toutes les ressources rendues indisponibles ne peuvent pas bloquer toutes les recettes du FMS. Dans le cas contraire, un arrêt total du FMS est privilégié et le module de diagnostic sera alors réinitialisé après l'opération de maintenance.

Troisièmement, deux changements de mode successifs ne peuvent avoir lieu qu'après un séquence de décisions d'allocation suffisamment grande. En effet, on suppose que deux opérations de maintenance préventive successives sont suffisamment espacées afin de terminer et de valider, lors d'une phase de test, la première opération.

Quatrièmement, par extension de la troisième hypothèse, on suppose qu'un nouveau changement de mode ne peut avoir lieu que si le mode actuel du FMS est connu par le module de diagnostic ou, dans le cas où il existe plusieurs modes candidats pour le module, si ceux-ci génèrent le même ordonnancement depuis M.

Dans l'exemple de la figure 4.3, à partir de l'état M et le $Mode_0$, les modes atteignables en vertu de la première hypothèse sont ceux où une unique ressource est indisponible à savoir les modes : $Mode_{R_1}, Mode_{R_2}, Mode_{R_3}, Mode_{R_4}, Mode_{R_5}, Mode_{R_7}$. Le $Mode_{R_6}$ est omis de cette liste puisque l'indisponibilité de R_6 entraîne l'arrêt complet du FMS (hypothèse 2). En appliquant à partir de M, pour $K_{max} = 100$ et pour chacun de ces modes atteignables les étapes de construction du modèle N_{NB}^{C-t} introduites précédemment (sous-partie 4.1.2), les ordonnancements partiels $\sigma = \sigma_c\sigma_f$ présentés dans le tableau 4.3 peuvent être calculés. L'ordonnancement à partir du $Mode_0$ y est aussi référencé. Dans ce tableau, la première colonne représente le mode, la deuxième l'ordonnancement σ_c résultant de la conséquence de cascade d'indisponibilités, la troisième l'ordonnancement σ_f calculé depuis le modèle N_{NB}^{C-t} construit à partir de la paire $(M, Mode)$ et après l'exécution de σ_c et la quatrième les places activité de N bloquées par le changement de mode. A partir de cette dernière colonne, on observe pour le $mode_{R_1}$ un blocage partiel puisque des places activité n'appartenant pas à $P_{R_1}^{pre}$ sont bloquées. Ce blocage partiel résulte d'une condition d'attente circulaire créée entre les ressources R_5 et R_3 à travers les places activité marquées p_7 et p_{18}.

Parmi tous les ordonnancements $\sigma = \sigma_c\sigma_f$ du tableau, ceux générés par les paires de modes ($Mode_{R_1}$,$Mode_{R_5}$) et ($Mode_{R_4}$,$Mode_{R_7}$) sont identiques, rendant les deux modes de chaque paire a priori non diagnosticables l'un de l'autre par le module de diagnostic. A partir de cette observation de non diagnosabilité entre deux modes et en appliquant l'hypothèse 4, un nouveau changement de mode peut ainsi avoir lieu après trois décisions d'allocation effectives. Ce résultat correspond à la transition $t_6 \in \sigma(M, Mode_{R_3})$ permettant de distinguer le $Mode_{R_3}$ des modes $Mode_{R_1}$ et $Mode_{R_5}$.

TABLEAU 4.3 – Ordonnancements obtenus pour chaque mode atteignables depuis le $mode_0$ et l'état M

Mode	σ_c	σ_f	$(P \setminus P_{NB}^C) \cap P_A$
0	/	$t_3t_6t_{16}t_5t_{15}t_4t_7t_6t_{19}t_8t_{17}$	[]
R1	t_{16}	$t_3t_4t_1t_2t_3t_4t_1t_2t_3t_4$	[2,3,7,8,9,10,11,12,13,14,15,16,17,18,19]
R2	t_6	$t_{16}t_{17}t_{15}t_{19}t_{18}t_{20}t_{16}t_{19}t_{17}t_{15}$	[1,4,5,6,8,9,10]
R3	t_{16}	$t_3t_4t_6t_1t_2t_5t_3t_4t_7t_6$	[3,12,13,14,15,16,17,18,19]
R4	$t_6t_{19}t_{20}$	$t_3t_4t_1t_2t_3t_4t_1t_2t_3t_4$	[2,3,7,8, 9,10,11,12,13,14,15,16,17,18,19]
R5	t_{16}	$t_3t_4t_1t_2t_3t_4t_1t_2t_3t_4$	[2,3,7,8,9,10,11,12,13,14,15,16,17,18,19]
R7	$t_6t_{19}t_{20}$	$t_3t_4t_1t_2t_3t_4t_1t_2t_3t_4$	[2,3,7,8,9,10,11,12,13,14,15,16,17,18,19]

Algorithme

La construction de $Diag_m$ est fondée sur un état M et un $Mode_{\mathcal{R}_i}$ à partir duquel les changements de mode ont lieu et doivent être diagnostiqués. Ainsi, $Diag_m$ associe à chaque $mode_{\mathcal{R}_j}$ atteignable selon les hypothèses 1 et 2 un label $\delta_{\mathcal{R}_j}^M$. L'ensemble des modes atteignables depuis $(mode_{\mathcal{R}_i})$ est noté $\mathcal{RM}(mode_{\mathcal{R}_i})$. A l'inverse, l'absence de changement de mode est labellisée par $O_{\mathcal{R}_i}^M$. Pour sa part, le respect de l'ordonnancement actuel depuis M dans $mode_{\mathcal{R}_i}$ est labellisé au sein de $Diag_a$ (sous-partie 3.3.2) et sera combiné avec les labels de changement de mode lors de la construction du diagnostiqueur commun (sous-partie 4.3.2). L'algorithme de construction de $Diag_m(N_t, x_0^m) = (X_m, E_m, f_m, x_0^m)$ à partir de la paire $(M_{init}, mode_{\mathcal{R}_i})$ est présenté dans le diagramme d'activité 4.10 ci-contre et est détaillé dans l'annexe 6.14. Cet algorithme est fondé sur l'algorithme de construction d'un diagnostiqueur (sous-partie 3.3.2) auquel deux modifications majeures sont apportées.

Dans un premier temps, l'initialisation du diagnostiqueur (étapes 1 à 5) définit non seulement x_0^m (étape 2) dans lequel le marquage initial M_{init} est associé à tous les labels $\delta_{\mathcal{R}_j}^{M_{init}}$ des modes atteignables de $\mathcal{RM}(mode_{\mathcal{R}_i})$ mais il définit aussi un second état x_m^ε. Cet état x_m^ε représente le cas où un changement de mode vers $Mode_{\mathcal{R}_j}$ entraîne un blocage total du FMS. Ce blocage correspond à un ordonnancement $\sigma(M_{init}, mode_{\mathcal{R}_j})$ vide (étape 3). Pour rappel, cet ordonnancement est calculé depuis la paire $(M_{init}, mode_{\mathcal{R}_j})$ selon la méthode de prévention et d'ordonnancement proposée précédemment (sous-partie 4.2.2). Si un tel blocage a lieu, $(M_{init}, \delta_{\mathcal{R}_j}^{M_{init}})$ est composé à x_m^ε (étape 4). Les deux états définis lors de cette initialisation sont reliés par un arc $f_m(x_0^m, \varepsilon) = x_m^\varepsilon$ labellisé par ε, à savoir l'absence d'événement.

Dans un second temps, pour chaque marquage M à explorer appartenant à $\mathcal{M}_e$ (étape 9), leur labellisation repose sur l'ordonnancement propre à chaque mode $Mode_{\mathcal{R}_j} \in \mathcal{RM}(Mode_{\mathcal{R}_i})$. Ainsi, pour chaque $Mode_{\mathcal{R}_j}$ (étape 10), si M est atteint depuis M_{init} en suivant l'ordonnancement $\sigma(M_{init}, Mode_{\mathcal{R}_j})$ (étape 11), le label $\delta_{\mathcal{R}_j}^{M_{init}}$ est ajouté à Δ_e (étape 11) afin de labelliser M. A l'inverse, si aucun ordonnancement $\sigma(M_{init}, Mode_{\mathcal{R}_j})$ ne relie M_{init} à M (aucun changement de mode ne correspond à M), le label $O_{\mathcal{R}_i}^{M_{init}}$ est associé à M (étape 12). Notons que dans $Diag_m$, les marquages labellisés sont ceux du SC-net N_t et non de N dans le but de correctement distinguer les marquages successifs au sein des ordonnancements $\sigma(M, Mode_{\mathcal{R}_j})$. En effet, sans distinction du marquage des places d'entrée et de sortie de N_t, un marquage de N peut être traversé plusieurs fois au sein d'un même ordonnancement. Ce choix de conserver les marquages de N_t est aussi requis pour la fusion des états de $Diag_m$ avec les états des autres diagnostiqueurs

$Diag_a$ et $Diag_{\mathcal{P}}$. En effet, dans ceux-ci, les marquages labellisés ont été projetés aux places de $P^{G\alpha}$ incluant les places d'entrée et de sortie du modèle N_t (voir sous-partie 3.3.2).

Enfin, cet algorithme construit lui aussi un diagnostiqueur partiel puisqu'il arrête son exploration lorsqu'un état n'étant labellisé par aucun changement de mode est atteint (soit un état labellisé par $O_{\mathcal{R}_i}^{M_{init}}$) ou lorsqu'un ordonnancement $\sigma(M, mode_{\mathcal{R}_j})$ est terminé. Ces états sont exclus de l'exploration lors de la construction de $\mathcal{M}_e$. L'algorithme de construction de $Diag_m$ présenté dans cette sous-partie sera appelé par la méthode de construction du diagnostiqueur commun (sous-partie 4.3.2) dès que les changements de mode depuis un état M et un mode $Mode_{\mathcal{R}_i}$ devront être diagnostiqués. Dans les deux prochaines sous-parties, la diagnosabilité des changements de mode est analysée et la construction de $Diag_m$ est illustrée sur l'exemple du tableau 4.3.

Diagnosabilité des changements de mode

Proposition 4.3.1 (Non-Diagnosabilité des changements de mode)**.** *Les changements de mode d'un FMS modélisé par* $(N, M_0) = (P_A \cup P^0 \cup P_R, T, F, M_0)$ *ne sont pas diagnosticables si* $\exists r_1, r_2$ *avec* $p_{r_1}, p_{r2} \in P_R$ *tel que* $\mathcal{C}_{r_1} = \{C_i \in \mathcal{C} | H_{r_1} \cap C_i \neq \emptyset\}$ *est égal à* $\mathcal{C}_{r_2}$*. Autrement dit, ils ne sont pas diagnosticables si les opérations requérant les ressources* r_1 *et* r_2 *partagent les même circuits dans* N*.*

Preuve 4.3.1. La preuve de cette proposition peut être apportée en prenant l'état M_0. Dans cette état, nous avons $\forall p_a \in P_A$, $M_0(p_a) = 0$. Par conséquent aucune conséquence de cascade d'indisponibilité ou d'états de blocage dans un nouveau mode ne peut avoir lieu. Ainsi, lorsque r_1 ou r_2 devient indisponible, seules les places activité de H_{r_1} et H_{r_2} sont bloquées entraînant le blocage des circuits du FMS dans $\mathcal{C}_{r_1}$ et $\mathcal{C}_{r_2}$ auxquels ces places activité appartiennent. Les circuits vivants $\mathcal{C} \setminus \mathcal{C}_{r_1}$ et $\mathcal{C} \setminus \mathcal{C}_{r_2}$ sont donc identiques entre les modes $mode_{r_1}$ et $mode_{r_2}$, résultant en un unique modèle N_{NB}^{C-t} et un unique ordonnancement depuis M_0^t. L'unicité de l'ordonnancement calculé depuis ces deux modes rend alors leur diagnostic depuis l'état M_0^t impossible par $Diag_m$. ⌟

Notons que cette proposition est extensible à des ensembles de ressources indisponibles $\mathcal{R}_1 = \{r_1^1, r_2^1, ..., r_{n_1}^1\}$ et $\mathcal{R}_2 = \{r_1^2, r_2^2, ..., r_{n_2}^2\}$ où $H_{\mathcal{R}_1} = \bigcup H_{r_{k_1}^1}$ et $\mathcal{C}_{\mathcal{R}_1} = \bigcup \mathcal{C}_{r_{k_1}^1}$ avec $k_1 \in [1, ..., n_1]$. Dans l'exemple de la figure 4.3, les ressources R_3 et R_4 respectent la condition $\mathcal{C}_{R_3} = \mathcal{C}_{R_4} = \{C_B^2, C_C^1\}$ rendant les changements de modes de ce FMS non diagnosticables globalement. Afin d'illustrer cette non diagnosabilité, un exemple d'application de $Diag_m$ sur l'exemple de la figure 4.3 est présenté en conclusion de cette partie.

Exemple

Dans cet exemple, le FMS se trouve dans $mode_0$ et dans l'état M modélisé par la figure 4.3. L'ensemble des modes atteignables est égal à $\mathcal{RM}(Mode_0) = \{Mode_{R_1}, Mode_{R_2}, Mode_{R_3}, Mode_{R_4}, Mode_{R_5}, Mode_{R_7}\}$. La construction de $Diag_m$ depuis cet état initial est illustrée partiellement dans la figure 4.11. L'ensemble des marquages M_i labellisés dans cette figure sont détaillés dans le tableau 4.4.

Dans ce diagnostiqeur, aucun changement de mode ne conduit à un blocage total du FMS. Par conséquence, x_m^{ε} est vide. Depuis M_{init}, deux marquages atteignables M_9 et M_{10} ne correspondant à aucun changement de mode, sont labellisés par O_0^M et sont entourés en

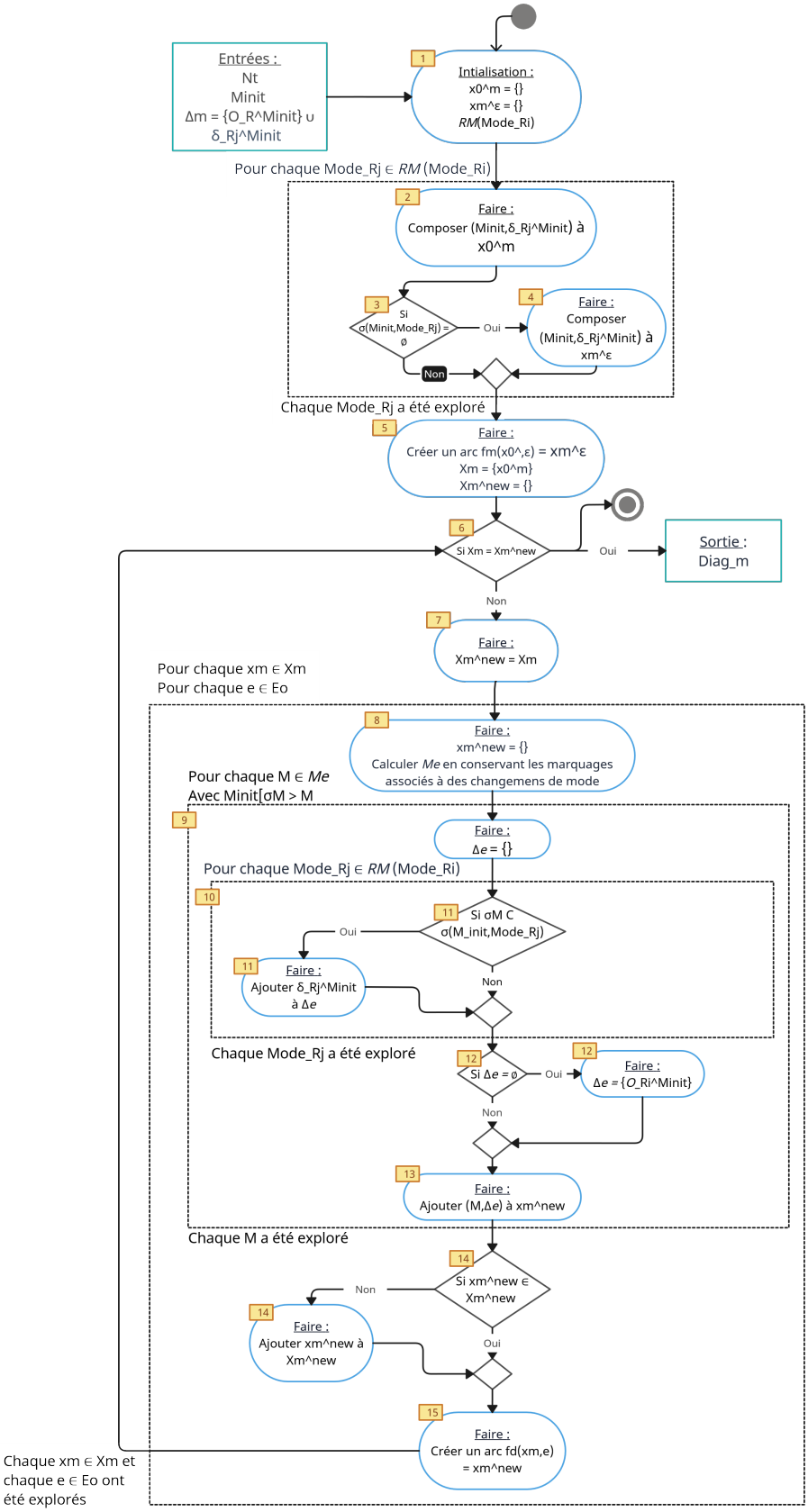


FIGURE 4.10 – Diagramme d'activité UML de l'algorithme de construction de $Diag_m$

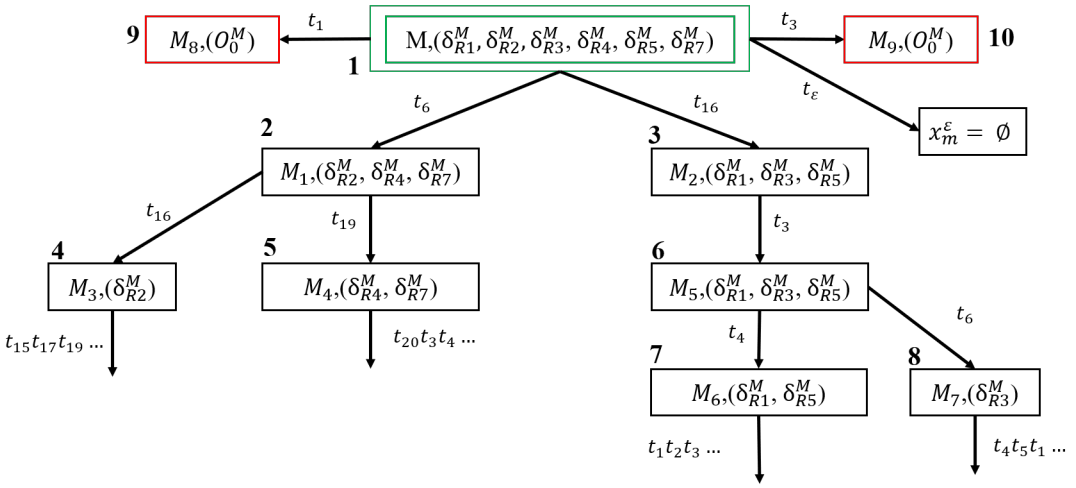


FIGURE 4.11 – Exemple de construction partielle de $Diag_m$ pour l'exemple de la figure 4.3

TABLEAU 4.4 – Tableau des marquages de la figure 4.11

Indice	Marquage	Labels
1	$M = 3p_1^e + 7p_2^e + 4p_3^e + p_5 + p_7 + p_{15} + p_{18} + p_{20} + p_{23} + p_{25}$	$\delta_{\mathcal{R}_1}^M \delta_{\mathcal{R}_2}^M \delta_{\mathcal{R}_3}^M \delta_{\mathcal{R}_4}^M \delta_{\mathcal{R}_5}^M \delta_{\mathcal{R}_7}^M$
2	$M_1 = 3p_1^e + 7p_2^e + 4p_3^e + p_5 + p_8 + p_{15} + p_{18} + p_{24} + p_{23} + p_{25}$	$\delta_{\mathcal{R}_2}^M \delta_{\mathcal{R}_4}^M \delta_{\mathcal{R}_7}$
3	$M_2 = 3p_1^e + 7p_2^e + 4p_3^e + p_5 + p_7 + p_{16} + p_{18} + p_{20} + p_{25} + p_{26}$	$\delta_{\mathcal{R}_1}^M \delta_{\mathcal{R}_3}^M \delta_{\mathcal{R}_5}^M$
4	$M_3 = 3p_1^e + 7p_2^e + 4p_3^e + p_5 + p_8 + p_{16} + p_{18} + p_{24} + p_{25} + p_{26}$	$\delta_{\mathcal{R}_2}^M$
5	$M_4 = 3p_1^e + 7p_2^e + 4p_3^e + p_5 + p_8 + p_{15} + p_{19} + p_{22} + p_{23} + p_{25}$	$\delta_{\mathcal{R}_4}^M \delta_{\mathcal{R}_7}^M$
6	$M_5 = 3p_1^e + 7p_2^e + 4p_3^e + p_6 + p_7 + p_{16} + p_{18} + p_{20} + p_{21} + p_{26}$	$\delta_{\mathcal{R}_1}^M \delta_{\mathcal{R}_3}^M \delta_{\mathcal{R}_5}^M$
7	$M_6 = 3p_1^e + 7p_2^e + 4p_3^e + p_1^s + p_7 + p_{16} + p_{18} + p_{20} + p_{21} + p_{25} + p_{26}$	$\delta_{\mathcal{R}_1}^M \delta_{\mathcal{R}_3}^M \delta_{\mathcal{R}_5}^M$
8	$M_7 = 3p_1^e + 7p_2^e + 4p_3^e + p_1^s + p_8 + p_{16} + p_{18} + p_{21} + p_{24} + p_{25} + p_{26}$	$\delta_{\mathcal{R}_3}^M$
9	$M_8 = 2p_1^e + 7p_2^e + 4p_3^e + p_1^s + p_4 + p_7 + p_{16} + p_{18} + p_{20} + p_{21} + p_{25} + p_{26}$	$\delta_{\mathcal{R}_1}^M \delta_{\mathcal{R}_5}^M$
10	$M_9 = 2p_1^e + 7p_2^e + 4p_3^e + p_4 + p_5 + p_7 + p_{15} + p_{18} + p_{20} + p_{23}$	O_0^M
11	$M_10 = 3p_1^e + 7p_2^e + 4p_3^e + p_6 + p_7 + p_{15} + p_{18} + p_{20} + p_{21} + p_{23}$	O_0^M

rouge dans la figure. En particulier, M_9 (indice 9) est un état de blocage au sein du FMS. D'autres transitions depuis des états de la figure 4.11 conduisent à des états ne correspondant à aucun changement de mode mais ne sont pas représentés pour des raisons de lisibilité. De la même manière, en sortie des états 4, 5, 8 et 9, les états du diagnostiqueur labellisés par des changements de mode ne sont pas représentés car ces derniers possèdent les mêmes labels que les états 4, 5, 8 et 9 en suivant les séquences de transitions annotées à côté des flèches de sortie.

Dans la figure 4.11, seuls les modes $Mode_{\mathcal{R}_2}$ et $Mode_{\mathcal{R}_3}$ sont diagnostiqués au sein des états 4 et 8 du diagnostiqueur. Les autres modes de $\mathcal{RM}(Mode_0)$ ne peuvent l'être puisque les modes des paires $(Mode_{\mathcal{R}_4}, Mode_{\mathcal{R}_7})$ et $(Mode_{\mathcal{R}_1}, Mode_{\mathcal{R}_5})$ ne sont pas diagnosticables l'un de l'autre au sein des états 5 et 9 de la figure. Toutefois, les ordonnancements en sortie des états 5 et 9 étant identiques indifféremment du mode, l'hypothèse 4 peut être appliquée et un nouveau changement de mode peut avoir lieu. Précisément, un nouveau changement de mode peut avoir lieu à partir des états 4, 5, 8 et 9 du diagnostiqueur.

L'algorithme de construction de $Diag_m$ illustré par l'exemple ci-dessus est requis pour la construction d'un diagnostiqueur commun des changements de mode, des attaques de blocage et des profils d'attaquant dans un contexte incertain. Ce diagnostiqueur est présenté dans la prochaine sous-partie.

4.3.2 Diagnostiqueur des attaques de blocage dans un contexte incertain

L'objectif de cette sous-partie est le développement du diagnostiqueur commun. Au sein du fonctionnement global du module de diagnostic (figure 4.9), ce diagnostiqueur commun est requis pour le diagnostic en ligne des attaques de blocage, des profils d'attaquant et des changements de mode dans un contexte incertain.

Présentation du diagnostiqueur

Le diagnostic des attaques de blocage dans un contexte incertain est fondé sur la construction d'un diagnostiqueur commun intégrant le diagnostiqueur des changements de mode $Diag_m$ (sous-partie 4.3.1), le diagnostiqueur des attaques de blocage $Diag_a$ (sous-partie 3.2.2) et le diagnostiqueur des profils d'attaquant calculés dans un contexte incertain $Diag_{\mathcal{P}}$ (sous-partie 4.2.3). Ce diagnostiqueur est une extension du diagnostiqueur $Diag_c$ présenté dans le chapitre 3 (sous-partie 3.3.2) et est noté $Diag_{cu}$. Identiquement à $Diag_c$, $Diag_{cu}$ est construit de manière itérative selon l'occurrence de différents événements au sein du FMS. La $i^{ème}$ itération de $Diag_{cu}$ est notée $Diag^i_{cu}$.

La construction itérative de $Diag^i_{cu}$ est fondée sur des conditions de déclenchement. Ces conditions démarrent la construction d'une nouvelle itération $Diag^{i+1}_{cu}$ à partir du diagnostiqueur actuel $Diag^i_{cu}$. Elles se décomposent en conditions de ré-ordonnancement (c.f. chapitre 3, sous-partie 3.3.2) et en conditions de changement de mode. Elles sont présentées dans les deux prochains paragraphes.

La condition de ré-ordonnancement Cn_d (définition 3.3.2) du chapitre 3 est conservée et appliquée sans modification. Ainsi, la constante $Mx(A)$ propre à la recette A reste inchangée

même en cas de circuits bloqués dans A. Toutefois, les recettes bloquées n'entraînent pas le déclenchement des sous-conditions Cn_d^1, Cn_d^2 et Cn_d^3 pour les raisons suivantes. Premièrement, Cn_d^1 est validée lors de la terminaison de l'ordonnancement et dépend donc uniquement des recettes non bloquées du FMS. Deuxièmement, pour Cn_d^2, on suppose que lorsqu'une recette A est bloquée aucun produit n'arrive à son entrée, i.e. $M_{rl}(p_A^e) = M(p_A^e)$. Par conséquent, le second terme de Cn_d^2 n'est jamais validé puisqu'il suppose une augmentation entre $M(p_A^e)$ et $M_{rl}(p_A^e)$. Troisièmement, Cn_d^3 n'est jamais validée par une recette bloquée car elle suppose aussi une augmentation entre $M(p_A^e)$ et $M_{rl}(p_A^e)$.

Ce choix de conserver Cn_d inchangée au sein du contexte incertain repose d'une part sur la non validation de Cn_d par des recettes bloquées (voir ci-dessus) et d'autre part sur la nécessité de définir une condition commune à tous les modes d'indisponibilité dans le cas où le module de diagnostic ne peut pas identifier le mode actuel. En effet, dans ce cas-ci, une condition de ré-ordonnancement propre à chaque mode entraînerait pour le module de diagnostic des calculs d'ordonnancement superfétatoires pour des modes n'étant *in fine* pas présents.

La condition de changement de mode, notée Cd_m, est activée dès qu'un état atteint par le FMS ne respecte par l'ordonnancement du ou des modes actuels mais respecte l'ordonnancement d'au moins un mode atteignable. Mathématiquement, cette condition est définie par :

Définition 4.3.1 (Condition de changement de mode Cm_d)**.**
Pour un $Mode_{\mathcal{R}_i}$, un état actuel M du FMS, et un ordonnancement σ depuis M_{init}, la condition de changement de mode est définie par :

$$Cm_d = Cm_d^1 \wedge Cm_d^2$$

avec

$$Cm_d^1 =< \nexists \sigma_M \in T^\star | M_{init}[\sigma_M > M, l(\sigma_M) < l(\sigma) >$$

et

$$Cm_d^2 =< \exists Mode_{\mathcal{R}_j} \in \mathcal{RM}(Mode_{\mathcal{R}_i}), \exists M' \in \mathcal{R}(N_t, M_{init}), \exists \sigma_{M'} \in T^\star, \exists \sigma_{M,M'} \in T^\star |$$

$$\mathrm{M}_{init}[\sigma_{M'} > M', l(\sigma M') < l(\sigma)\ et M'[\sigma_{M',M} > M, l(\sigma_{M',M}) < l(sigma(M', Mode_{\mathcal{R}_j}) >$$

Dans cette condition, Cm_d^1 vérifie que l'état M ne respecte pas l'ordonnancement actuel σ et Cm_d^2 qu'il existe un mode atteignable $Mode_{\mathcal{R}_j}$ depuis $Mode_{\mathcal{R}_i}$ et un état M' traversé par l'ordonnancement actuel tel que M respecte l'ordonnancement du changement de mode ayant lieu en $(M', Mode_{\mathcal{R}_j})$.

◇

Cette condition de changement de mode ne peut se déclencher que lorsque l'hypothèse 4 est respectée : un changement de mode ne peut avoir lieu que s'il existe un ordonnancement commun entre les différents modes actuels diagnostiqués par le module de diagnostic. Cette hypothèse permet ainsi de considérer comme anormal un état atteint ne respectant pas l'un des ordonnancements des modes éligibles entre (i) un changement de mode, déclenchant la condition et la génération des ordonnancements pour chaque mode atteignable, et (ii) le respect de l'hypothèse 4 une fois qu'un seul ordonnancement commun a été identifié.

Dans l'exemple de la figure 4.11, en rappelant que l'ordonnancement du $Mode_0$ est égal à $\sigma_f = t_3t_6t_{16}t_5t_{15}t_4t_7t_6t_{19}t_8t_{17}$, la condition de changement de mode se déclenche dès que les états M_1 ou M_2 sont atteints par les franchissements de t_6 ou t_{16} puisque ces états ne respectent pas σ_f et sont labellisés par des modes atteignables. A l'inverse, les états M_9 et M_{10} sont considérés comme anormaux et donc assimilables à des attaques. Puis, entre le changement de mode en M_1 et M_2, le prochain changement de mode ne peut avoir lieu qu'après M_3, M_4, M_7 et M_8 lorsque un ordonnancement commun est identifié (voir flèches de sortie). Par conséquent, si un état ne correspondant pas à M_3 ou à M_4 depuis M_1, à M_5 depuis M_2, à M_6 depuis M_5 et à M_7 ou M_8 depuis M_5 est atteint, il est alors associé à un état anormal par le module de diagnostic.

Algorithme

L'algorithme de construction de $Diag^i_{cu}$ est présenté dans le diagramme d'activité UML 4.12 et est détaillé par son pseudo-code dans l'annexe 6.15. Les différentes étapes de ce diagramme sont présentées dans cette sous-partie.

Dans ce diagramme, la phase d'initialisation (étapes 1 à 3) consiste à conserver du diagnostiqueur précédent $Diag^{i-1}_{cu}$ les états pertinents pour le diagnostic au sein de la nouvelle itération $Diag^i_{cu}$ du diagnostiqueur commun. Par conséquent, la phase d'initialisation cherche en premier lieu à identifier l'origine du déclenchement de la nouvelle itération (étape 1), entre ré-ordonnancement (étape 2) et changement de mode (étape 3). Dans le premier cas, il existe un label d'état optimal $O^{\mathcal{R}_i}_a/O^{M_k,\mathcal{R}_i}_a$ au sein de l'état actuel x^{i-1}_{cu} du diagnostiqueur précédent. Dans le second cas, il n'existe pas de tels labels au sein de x^{i-1}_{cu} puisque M_{init} est un état ne respectant pas l'ordonnancement précédemment calculé.

Une fois l'origine du déclenchement identifiée, la phase d'initialisation définit le nouveau diagnostiqueur à partir d'une réduction du précédent. Dans le premier cas (étape 2), les états atteignables depuis x^{i-1}_{cu} et labellisés par des profils d'attaquant($\delta^{M_k,\mathcal{R}_i}_{\mathcal{P}}$) ou des changements de mode ($\delta^{M_k}_{\mathcal{R}_i}$) sont conservés de $Diag^{i-1}_{cu}$. La notation $|_{(\Delta_m\cup\Delta_{\mathcal{P}})}$ désigne cette restriction à ces labels (c.f. annexe 6.11). Les labels des attaques de blocage ne se propagent pas entre les deux diagnostiqueurs puisque le FMS se trouve dans un état optimal lors du ré-ordonnancement. Puis, tous les modes dans lesquels peut se trouver le FMS sont identifiés à l'aide des labels de trajectoire optimale ($O^{\mathcal{R}_i}_a/O^{M_k,\mathcal{R}_i}_a$) de x^{i-1}_{cu} et sont rassemblés dans $LABEL(x^{cu_i}_0)$. Dans le second cas (étape 3), les états labellisés par les attaques de blocage ($\delta^{\mathcal{R}_i}_{A1}$,$\delta^{\mathcal{R}_i}_{A2}$,$\delta^{M_k,\mathcal{R}_i}_{A1}$,$\delta^{M_k,\mathcal{R}_i}_{A2}$) et les profils d'attaquant ($\delta^{M_k,\mathcal{R}_i}_{\mathcal{P}}$) sont conservés de $Diag^{i-1}_{cu}$. La notation $|_{(\Delta_a\cup\Delta_{\mathcal{P}})}$ désigne cette restriction à ces labels (c.f. 6.11). Dans $Diag^{i-1}_{cu}$, x^{i-1}_{cu} est labellisé par un label d'attaque de blocage puisqu'il correspond à un état ne respectant pas l'ordonnancement attendu. Les labels de changement de mode et les états labellisés uniquement par ceux-ci ne sont pas conservés dans $Diag^i_{cu}$ car un deuxième changement de mode ne peut avoir lieu immédiatement (hypothèse 4). Puis, tous les modes dans lesquels peut se trouver le FMS sont identifiés à l'aide des labels de changement de mode de x^{i-1}_{cu} et sont rassemblés dans $LABEL(x^{cu_i}_0)$.

La construction complète de $Diag^i_{cu}$ est réalisée par les étapes 4 à 12 du diagramme. Dans $Diag^i_{cu}$, chaque ordonnancement éligible depuis M_{init} représente une trajectoire que peut suivre le FMS. Ces ordonnancements correspondent aux labels identifiés précédemment dans $LABEL(x^{cu_i}_0)$, soit un ordonnancement par mode actuel dans x^{i-1}_{cu} dans le cas d'un ré-ordonnancement et un ordonnancement par changement de mode possible dans x^{i-1}_{cu} dans le cas d'un changement de mode. Dans le second cas, il est à noter qu'un même nouveau

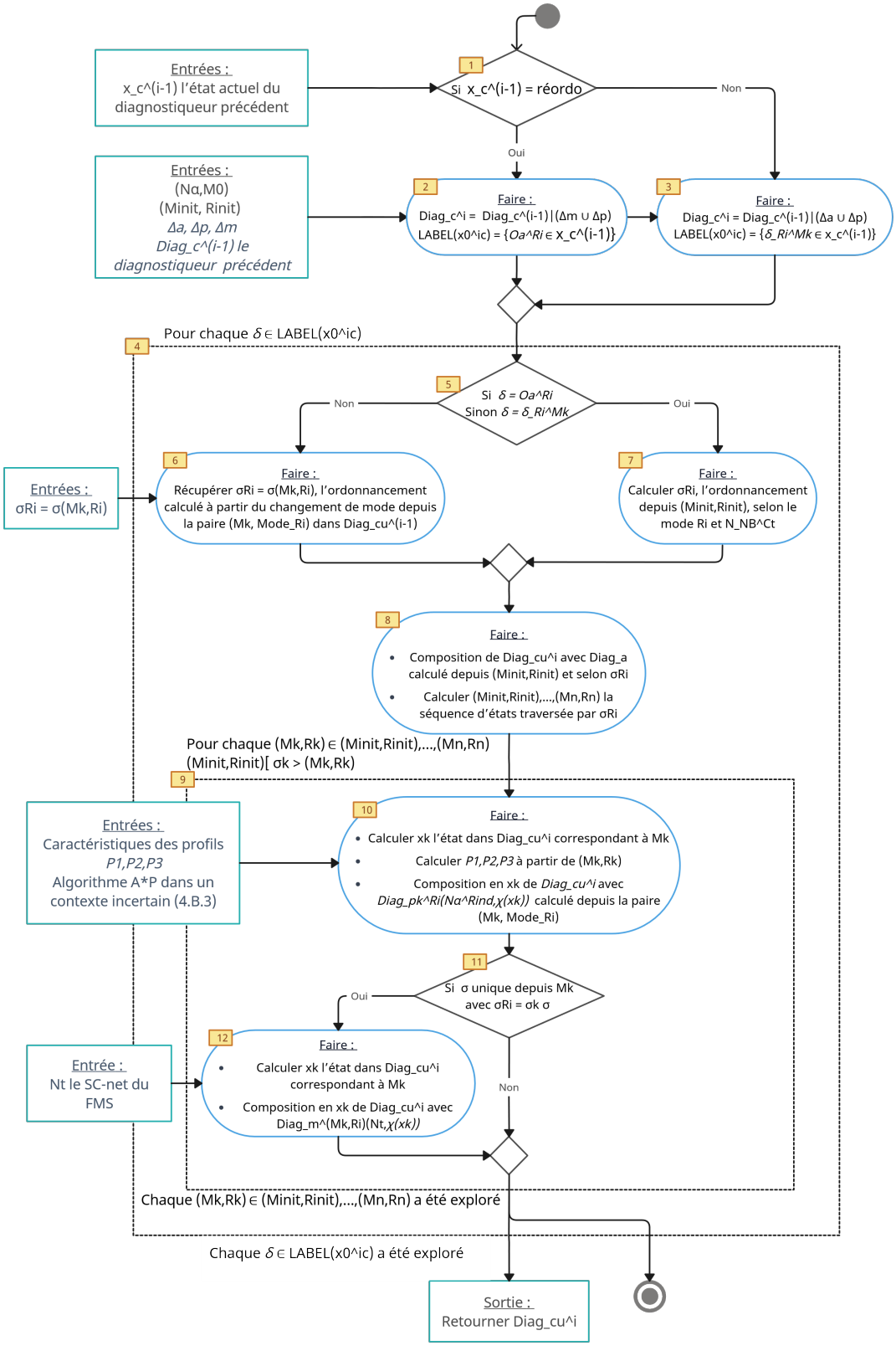


FIGURE 4.12 – Diagramme d'activité UML de l'algorithme de construction du diagnostiqueur commun $Diag^i_{cu}$

mode $Mode_{\mathcal{R}_j}$ peut générer plusieurs ordonnancements si son changement de mode peut avoir lieu depuis différents états M_k précédant M_{init}. Ainsi pour chaque label de $LABEL(x_0^{cu_i})$ (étape 4), l'ordonnancement correspondant $\sigma_{\mathcal{R}_i}$ depuis M_{init} est déterminé. Dans le cas d'un ré-ordonnancement (étape 7), $\sigma_{\mathcal{R}_i}$ est calculé à partir de (M_{init}, R_{init}), du mode $Mode_{\mathcal{R}_i}$ et du modèle N_{NB}^{C-t} construit à partir de cet état initial. Dans le cas d'un changement de mode labellisé par $\delta_{\mathcal{R}_i}^{M_k}$ (étape 8), $\sigma_{\mathcal{R}_i}$ a déjà été calculé pour la construction de $Diag_m$ depuis $(M_k, Mode_{\mathcal{R}_i})$ (ordonnancement $\sigma(M_k, Mode_{\mathcal{R}_i})$) dans $Diag_{cu}^{i-1}$ et peut donc directement être récupéré depuis ce dernier. A partir de cet ordonnancement $\sigma_{\mathcal{R}_i}$, le diagnostiqueur des attaques de blocage $Diag_a^{\sigma_{\mathcal{R}_i}}$ est composé à $Diag_{cu}^i$ (étape 8) et représente la trajectoire optimale et les attaques de blocage contre un FMS suivant cet ordonnancement. Les labels associés à ce diagnostiqueur sont notés $\Delta_a^{\mathcal{R}_i} = \{O_a^{\mathcal{R}_i}, \delta_{A1}^{\mathcal{R}_i}, \delta_{A2}^{\mathcal{R}_i}\}$ ou $\Delta_a^{\mathcal{R}_i} = \{O_a^{M_k,\mathcal{R}_i}, \delta_{A1}^{M_k,,\mathcal{R}_i}, \delta_{A2}^{M_k,\mathcal{R}_i}\}$ dans le cas d'un label de changement de mode $\delta_{\mathcal{R}_i}^{M_k}$.

Puis, pour chaque état (M_k, R_k) traversé par l'ordonnancement $\sigma_{\mathcal{R}_i}$ depuis $(M_{init}, R_{init}$ (étapes 8 et 9), les diagnostiqueurs de profils d'attaquant (étape 10) et de changements de mode (étape 12) sont construits et composés à $Diag_{cu}^i$. Les labels du diagnostiqueur des profils d'attaquant sont notés $\Delta_{\mathcal{P}}^{\mathcal{R}_i,k} = \{O_{\mathcal{P}}^{\mathcal{R}_i,k}, \delta_{\mathcal{P}_1}^{M_k,\mathcal{R}_i}, \delta_{\mathcal{P}_2}^{M_k,\mathcal{R}_i}, \delta_{\mathcal{P}_3}^{M_k,\mathcal{R}_i}\}$ tandis que ceux du diagnostiqueur des changement de mode sont notés $\delta_{\mathcal{R}_i}^{M_k}$. Dans le cas des diagnostiqueurs de changement de mode, ils sont construits et composés à $Diag_{cu}^i$ uniquement si l'hypothèse 4 est respectée au sein de l'état (M_k, R_k), en d'autres termes s'il existe un ordonnancement unique depuis (M_k, R_k) pour tous les modes atteignables de $\mathcal{RM}(Mode_{\mathcal{R}_i})$ (étape 11). Pour leur part, les diagnostiqueurs des profils d'attaquant sont construits à partir des profils calculés dans la partie 4.2.3 et projetés aux événements observables par le module de diagnostic par l'application de f_a^1 (effets sur G du profil). Rappelons ainsi que les attaques d'insertion d'un événement observable de défaillance ou de reprise ne sont pas observables par le module puisque réalisées entre les contrôleurs locaux et le module de supervision. Par conséquent, certains profils peuvent atteindre un état ciblé grâce à des changements de mode sans qu'aucun événement ne soit observé par le module de diagnostic.

Remarque 4.3.1. Rappelons que la fusion des diagnostiqueurs $Diag_m$, $Diag_a$ et $Diag_{\mathcal{P}}$ pour la construction de $Diag_{cu}^i$ a été rendue possible par les marquages labellisés communs à tous ces diagnsotiqueurs, soit les marquages des places de $P^{G\alpha}$ (ou places du SC-net N_t) observables par le module de diagnostic. ⌟

Enfin, le diagnostiqueur $Diag_{cu}^i$ ainsi construit est partiel, puisque les diagnostiqueurs qui le composent le sont et car celui-ci ne considère que les ordonnancements éligibles pouvant avoir lieu depuis (M_{init}, R_{init}) selon x_{cu}^{i-1} et les modes qui le caractérisent. Dans la prochaine sous-partie, la construction de $Diag_{cu}^i$ par l'algorithme présenté ci-dessus est illustrée à partir de l'exemple conducteur de ce chapitre.

Exemple

A partir de l'exemple de la figure 4.3, une première itération de $Diag_{cu}^i$ est construite. L'état initial dans N_t muni des place moniteurs du $Mode_0$ (c.f. figure 4.3 et section 3.2.1) est $M = 3p_1^e+7p_2^e+4p_3^e+p_5+p_7+p_{15}+p_{18}+p_{20}+p_{23}+p_{25}+p_4^c+p_5^c+p_7^c+2p_8^c+2p_9^c+2p_{10}^c+p_{11}^c+2p_{12}^c+2p_{13}^c$, $R(p_5) = 8$, $R(p_7) = 12$, $R(p_{15}) = 4$, puis R nul pour toutes les autres places. Le mode actuel est le $Mode_0$ et on définit $K_{max} = 100$. On définit le diagnostiqueur précédent par l'unique état actuel $x_0^{cu_{i-1}} = x_{cu}^{i-1} = x_0^{cu_i} = \{(M, O_a^0)\}$ avec O_a^0 le label associé dans $Diag_a$ à une trajectoire optimale pour le $Mode_0$. Ainsi, un unique label est considéré dans la construction de $Diag_{cu}^i$ et correspond

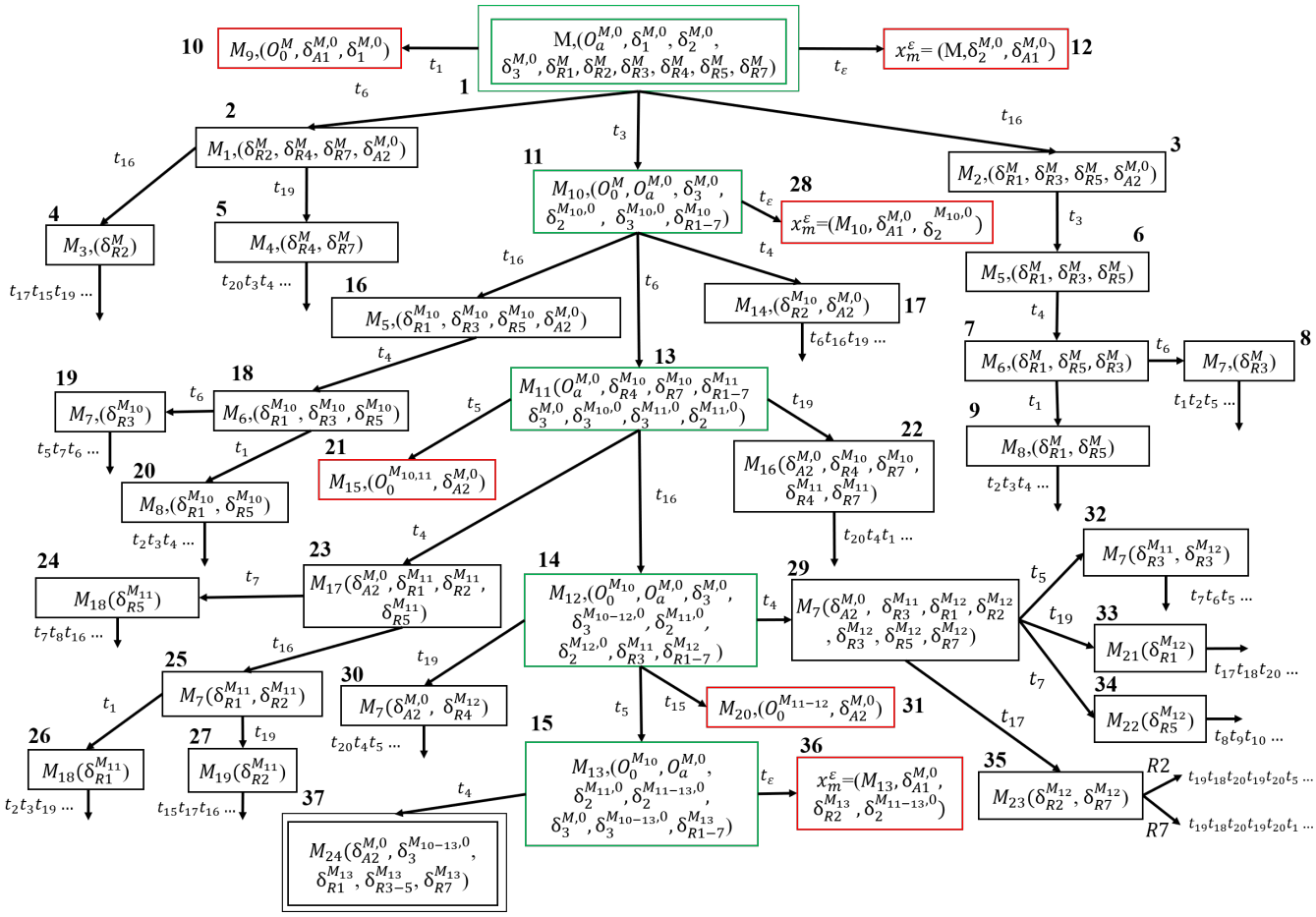


FIGURE 4.13 – Exemple d'application de la construction de $Diag^{i}_{cu}$ à partir de l'exemple de la figure 4.3 *(vert = états optimaux, rouge = états d'attaque, noir = états incertains)*

à l'ordonnancement partiel $\sigma = t_3t_6t_{16}t_5t_{15}t_4t_7t_6t_{19}t_8t_{17}$ (tableau 4.3). L'ordonnancement réel est composé de 92 décisions d'allocations et de 93 états $(M_k, R_k) \in \mathcal{R}(N_t, (M, R)$ parcourus.

Pour chaque état (M_k, R_k) traversé par l'ordonnancement σ, un diagnostiqueur des profils d'attaquant dans un contexte incertain $Diag_{\mathcal{P}}^{M_k,0}$ et un diagnostiqueur des changements de mode $Diag_{m}^{M_k,0}$ sont construits. Ces différents diagnostiqueurs sont composés avec le diagnostiqueur des attaques de blocage $Diag_{a}^{0}$ construit à partir de $x_0^{cu_i}$ et suivant l'ordonnancement σ. Cette composition forme le diagnostiqueur $Diag^{i}_{cu}$ illustré partiellement dans la figure 4.13. Cette figure est une extension de la figure 4.11. Dans cette figure 4.13, les labels d'absence de profils d'attaquant $O_{\mathcal{P}}^{M_i,0}$ ne sont pas représentés pour des raisons de clarté. Seuls les états traversés par un profil sont labellisés par celui-ci. Pour la même raison, le label d'attaque $\delta_{A2}^{M,0}$ n'est pas propagé au-delà du premier état atteint par non respect de l'ordonnancement.

Remarque 4.3.2. Dans les diagnostiqueurs $Diag_{cu}$ construits pour cet exemple et illustrés dans les figures 4.13 et 4.14, le profil d'attaquant $\mathcal{P}_1$ n'est représenté que depuis l'état M en raison d'un temps de convergence très long de l'algorithme $A^{\star}\mathcal{P}$ depuis les autres états de l'ordonnancement. Ce temps de convergence long s'explique par l'absence ou un nombre faible d'états hors-DZ capables d'atteindre un état de blocage total sans traverser des états de pré-blocage ou de blocage partiel. Par exemple, il existe seulement 11 de ces états hors-DZ dans le $Mode_0$. ⌟

On observe dans la figure 4.13 différents cas de labellisation pouvant être analysés. Premièrement, tous les états en verts et labellisés par $O_a^{M,0}$ représentent les états optimaux identifiés par le diagnostiqueur $Diag_a^0$. A l'inverse, tous les autres états labellisés par $\delta_{A1}^{M,0}$ ou $\delta_{A2}^{M,0}$ représentent des attaques pour $Diag_a^0$ avec $A1$ le label lorsqu'un état de la DZ est atteint et $A2$ lorsque l'ordonnancement n'est pas respecté. S'ils sont en rouge, ces états sont diagnostiqués comme relatifs à une attaque.

Deuxièmement, les états en noir représentent les états ne respectant pas l'ordonnancement labellisé par $\delta_{A2}^{M,0}$ mais correspondant par ailleurs à des changements de mode pour les différents $Diag_m$ construits depuis les états traversés par l'ordonnancement. Ainsi, un état en noir correspond soit à un changement de mode, soit à une attaque et déclenche la construction d'un nouveau diagnostiqueur $Diag_{cu}^{i+1}$ à partir de son marquage M et de chaque label de changement de mode qui lui est attribué.

Troisièmement, les états en noir depuis lesquels une flèche noire de sortie est tirée représentent les états succédant un changement de mode pour lesquels un ordonnancement commun a émergé. Par conséquent, un nouveau changement de mode légal peut avoir lieu depuis ces états. Parmi ces états, l'état 35 possède deux flèches de sortie représentant deux ordonnancements différents pour les changements de mode $\delta_{R_2}^{M_{12}}$ et $\delta_{R_7}^{M_{12}}$. Cet état illustre le cas où un grand nombre de décisions d'allocation est nécessaire pour distinguer deux modes différents.

Quatrièmement, les labels de profils d'attaquant conduisent à des états critiques soit en atteignant un état de la DZ du mode actuel (état 10), soit en rendant plusieurs ressources indisponibles simultanément induisant une absence d'occurrence d'événements t_ε (états 12 et 28), soit en réalisant un seul changement de mode conduisant à un état de la DZ du nouveau mode (état 36), soit en se confondant avec d'autres changements de mode non critique (état 37). Ce dernier état est choisi comme déclencheur de la génération d'un nouveau diagnostiqueur $Diag_{cu}^{i+1}$.

Une seconde itération de $Diag_{cu}$, notée $Diag_{cu}^{i+1}$, est construite lorsque l'état 37 est atteint par $Diag_{cu}^{i}$. Cet état correspond à une condition de changement de mode. Lors de la construction de $Diag_{cu}^{i+1}$, les labels d'attaque $\delta_{A2}^{M,0}$ et de profil $\delta_3^{M_{10},0}, \delta_3^{M_{11},0}, \delta_3^{M_{12},0}, \delta_3^{M_{13},0}$ sont conservés dans $x_0^{cu_{i+1}} = (M_5, \{\delta_{A2}^{M,0}, \delta_3^{M_{10},0}, \delta_3^{M_{11},0}, \delta_3^{M_{12},0}, \delta_3^{M_{13},0}\})$ et les états labellisés par $\delta_3^{M_{10},0}, \delta_3^{M_{11},0}, \delta_3^{M_{12},0}, \delta_3^{M_{13},0}$ sont intégrés au diagnostiqueur précédent $Diag_{cu}^{i}|_{(\Delta_a \cup \Delta_{\mathcal{P}})}$. Lors du changement de mode à l'état 37, 5 labels de changement de mode sont considérés soit $LABEL(x_0^{cu_{i+1}}) = \{\delta_{R_1}^{M_{13}}, \delta_{R_3}^{M_{13}}, \delta_{R_4}^{M_{13}}, \delta_{R_5}^{M_{13}}, \delta_{R_7}^{M_{13}}\}$. Depuis ces labels, quatre ordonnancements différents sont parcourus puisque $\sigma_{R_1} = \sigma_{R_7}$ en raison du blocage partiel obtenu si la ressource R_1 devient indisponible. Cette nouvelle itération du diagnostiqueur $Diag_{cu}$ est représentée partiellement dans la figure 4.14. Dans cette figure, les labels de profils d'attaquant calculés depuis les états parcourus par les ordonnancements de $Diag_{cu}^{i+1}$ ne sont pas représentés pour des raisons de clarté. L'objectif de cette figure est en premier lieu de montrer la construction de $Diag_{cu}^{i+1}$ lorsque plusieurs labels de changement de mode sont assignés à $x_0^{cu_{i+1}}$.

Dans $Diag_{cu}^{i+1}$, les observations suivantes sur ses états peuvent être faites. Premièrement, les états en vert et les labels $O_a^{M_{25},R_1}, O_a^{M_{25},R_2}, O_a^{M_{25},R_4}, O_a^{M_{25},R_5}$ et $O_a^{M_{25},R_7}$ représentent les états parcourus par les différents ordonnancements générés depuis les différents labels de $LABEL(x_0^{cu_{i+1}})$. On remarque ainsi que par rapport $Diag_{cu}^{i}$ plus d'état sont considérés comme optimaux par le superviseur. Par conséquent, moins d'états non optimaux sont accessibles par $Diag_{cu}^{i+1}$ rendant alors la détection d'attaque de blocage moins efficace.

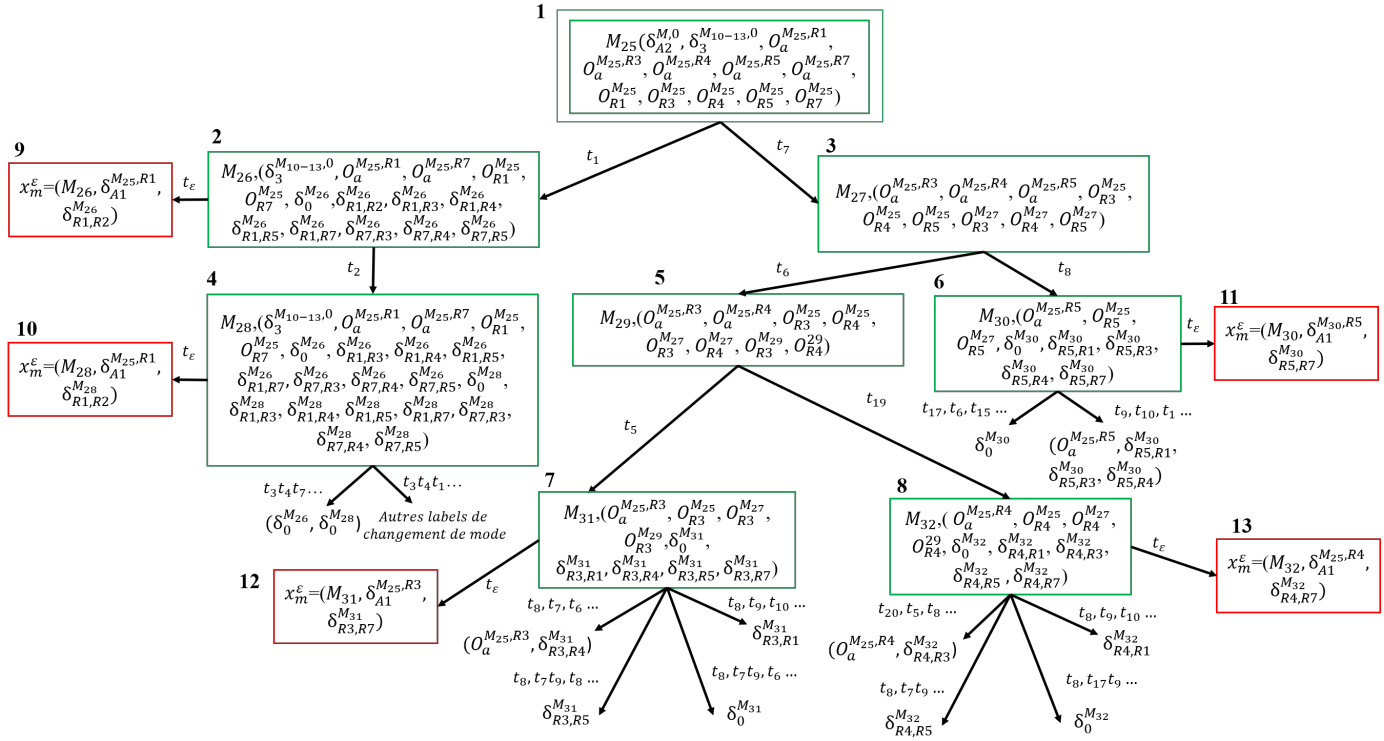


FIGURE 4.14 – Exemple d'application de la construction de $Diag_{cu}^{i+1}$ à partir de l'état 37 de la figure 4.13 *(vert = états optimaux, rouge = états d'attaque, noir = états incertains)*

Deuxièmement, parmi les états optimaux, l'hypothèse 4 est vérifiée pour les états 4, 6, 7 et 8 lorsque un ordonnancement commun est identifié. Au sein de ces états, les labels de changement de mode $\Delta_{\mathcal{R}_i}^{M}$ apparaissent. Au sein des états 2 et 4, ces nouveaux changements de mode entraînent peu de distinctions entre eux puisque seul le retour au $Mode_0$ génère un ordonnancement différent de tous les autres changements de mode. De surcroît, l'impossibilité de distinguer au sein des états 2 et 4 entre l'indisponibilité de R_1 et celle de R_7 augmente fortement le nombre de changements de mode pouvant avoir lieu. Dans les états 6, 7 et 8, l'existence d'une unique ressource indisponible réduit le nombre de nouveaux changements de mode possibles. De plus, dans les états 7 et 8, les changements de mode génèrent globalement des ordonnancements différents permettant leurs identifications.

Troisièmement, les états en rouge correspondent à des états de blocage atteints après un changement de mode au sein duquel les ressources indisponibles ne sont pas requises par toutes les recettes du FMS. En effet, ce cas de blocage de toutes les recettes par les ressources indisponibles n'est pas recherché ni par le superviseur (hypothèse 2), ni par l'attaquant (4.2.3). Ainsi, les états 9 et 10 sont des états de blocage en raison d'un blocage partiel provoqué par l'indisponibilité de R_1 en M_{25}, tandis que les états 11, 12 et 13 sont bloqués à la suite d'une conséquence de cascade d'indisponibilité provoquée par l'indisponibilité de R_7 sur le produit dans les places p_9 ou p_10 requérant les ressources R_2 ou R_6.

Quatrièmement, le profil d'attaquant labellisé par $\delta_3^{M_{10}}, \delta_3^{M_{11}}, \delta_3^{M_{12}}$ et $\delta_3^{M_{13}}$ et hérité du diagnostiqueur précédent n'est jamais diagnostiqué dans $Diag_{cu}^{i+1}$. En effet, ce dernier a atteint son objectif en créant un blocage partiel grâce à l'indisponibilité malveillante de R_1 en M_{25}, puis reste sournois en étant non différentiable du changement de mode causant l'indisponibilité de

R_7. Ces deux changements de mode génèrent le même ordonnancement entre eux et avec tous les nouveaux changements de mode pouvant avoir lieu dans les états 2 et 4, complexifiant alors le diagnostic du profil. Dans la prochaine sous-partie, les résultats obtenus dans cet exemple par la construction de $Diag_{cu}^{i}$ et $Diag_{cu}^{i+1}$ permettent une analyse critique de notre méthode.

4.3.3 Diagnosabilité, indicateurs de l'origine d'un changement de mode et discussion

Dans cette dernière sous-partie du chapitre 4, les caractéristiques et résultats du diagnostiqueur $Diag_{cu}$ sont analysés. Premièrement, la diagnosabilité des attaques de blocage dans $Diag_{cu}$ est étudiée. Puis, des indicateurs de malveillance d'un changement de mode sont identifiés. Enfin, une discussion critique de notre méthode de diagnostic des attaques de blocage dans un contexte incertain est proposée.

Diagnosabilité

L'exemple présenté dans la sous-partie précédente met en exergue la non diagnosabilité de certaines attaques de blocage dans un contexte incertain en raison de l'incertitude de diagnostic entre un changement de mode et une attaque de blocage. Parmi les différentes attaques de blocage, certaines sont diagnosticables, d'autres non.

Premièrement, les attaques conduisant à un état de blocage ou de pré-blocage sont systématiquement diagnosticables. En effet, ces dernières conduisent *in fine* le FMS à un état à partir duquel aucun événement d'allocation de ressource n'a lieu et n'est observé par le module de diagnostic, état atteint dans le diagnostiqueur par la transition t_ε. En supposant que les opérations de maintenance s'interdisent à conduire volontairement le système dans un de ces états de blocage total (hypothèse 2), le diagnostic d'une attaque est donc immédiat. Cependant, une fois l'un de ces états atteint, la trajectoire de l'attaque, à savoir son état d'origine et les changements de mode qu'elle insère, ne sont pas systématiquement diagnosticables comme en témoigne l'état 27 de l'exemple 4.13. Dans cet état, le profil $\delta_2^{M_{10}}$ correspond à deux changements de mode successifs (indisponibilité de R_1 et R_2) bloquant le FMS. Toutefois, ces deux changements de mode ne peuvent théoriquement pas avoir lieu selon les hypothèse 3 et 4 présentées en introduction de cette partie et sont par conséquent omis des labels de l'état 28.

Deuxièmement, les attaques ciblant un état de blocage partiel du FMS ne sont pas toujours diagnosticables par rapport à un changement de mode rendant indisponible un ensemble de ressources bloquant les mêmes circuits que le blocage partiel. C'est le cas dans l'exemple 4.14 du profil d'attaquant labellisé par $\delta_3^{M_{10-13}}$ dans les états 2 et 4 du diagnostiqueur. Ce dernier ne peut pas être différencié du changement de mode rendant la ressource R_7 indisponible.

A contrario, les attaques de blocage partiel sont systématiquement diagnosticables par rapport à un changement de mode rendant une ressource de nouveau disponible puisque des circuits sont libérés et rendus vivants par ce changement tandis que l'attaque cherche à bloquer certains circuits de $\mathcal{C}$. De plus, pour rappel, la méthode de supervision dans un contexte incertain (4.1.2) intégrée au sein du module de supervision a été prouvée apte à prévenir les états de blocage, pré-blocage et blocage partiel lorsqu'un changement de mode rend de nouveau disponible une ressource.

Troisièmement, les attaques sortant le FMS de sa trajectoire optimale en atteignant un état labellisé $\delta_{A2}^{M_k}$ ne sont plus diagnosticables dans un contexte incertain lorsqu'il existe des changements de mode dont l'ordonnancement ne correspond pas aux ordonnancements des modes actuels. Dans l'exemple de la figure 4.13, ce cas est présent à de nombreuses reprises, par exemple dans les états 2, 16 ou 23. On observe ainsi une grande perte d'information pour le diagnostic des attaques de blocage par rapport au contexte certain présenté dans le chapitre 3. Face à cette perte d'information, différents indicateurs évaluant la malveillance d'un changement de mode sont proposés.

Indicateurs de malveillance d'un changement de mode

Les indicateurs d'évaluation de la malveillance d'un changement de mode sont construits à partir des différentes hypothèses de travail prises dans ce manuscrit et de la combinaison des résultats de diagnostic des différents diagnostiqueurs composés dans $Diag_{cu}$. Ainsi, quatre indicateurs sont identifiés :

- Si un changement de mode a lieu alors que l'hypothèse 4 n'est pas respectée, alors ce changement de mode est malveillant. Dans l'exemple 4.14, c'est le cas de changements mode qui auraient eu lieu depuis les états 1, 3 et 5 et ne respecteraient pas les ordonnancements (états en vert) ;
- Si un changement de mode conduit à un état de blocage et que l'hypothèse 2 n'est pas vérifiée (les ressources indisponibles bloquent toutes les recettes), alors ce changement de mode est malveillant. Cet indicateur est fondé sur le rejet par la maintenance préventive de tels états où des blocages partiels (états 9, 10 de l'exemple 4.14) ou des conséquences de cascades d'indisponibilités (états 11, 12, 13 de l'exemple 4.14) apparaissent ;
- Si un changement de mode correspond à un ou plusieurs profils d'attaquant, alors ce changement de mode est suspicieux. Il n'est pas identifié comme malveillant mais révèle une potentielle attaque. Il s'agit de l'état 37 dans l'exemple 4.13 entraînant la construction de $Diag_{cu}^{i+1}$;
- A l'inverse, un changement de mode ne correspondant à aucun profil d'attaquant est considéré moins suspicieux.

Cependant, ces indicateurs ne sont pas suffisants pour qualifier systématiquement de malveillant ou non un changement de mode. Dans l'exemple 4.13, tous les changements de mode ayant lieu au sein des états en noir ne peuvent pas être identifiés comme malveillants tant qu'un état labellisé uniquement par un label d'attaque n'est pas atteint. De plus, aucune conclusion ne peut être faite si l'attaque atteignant cet état est lancée après un changement de mode d'origine naturelle. Dans la figure 4.14, cette incapacité d'identification de la malveillance d'un changement de mode est illustrée par les états 2 et 4 où un profil d'attaquant labellisé par $\delta_3^{M_{10-13}}$ créant un blocage partiel n'est pas différentiable d'un second changement (indisponibilité de R_7) ne pouvant être qualifié de malveillant selon les critères présentés ci-dessus. Ces limites sont discutées dans la prochaine partie et des perspectives d'amélioration sont proposées.

Discussion

A partir de l'exemple des figures 4.13 et 4.14 et des indicateurs de malveillance, les quatre limites suivantes de notre méthode de diagnostic des attaques de blocage dans un contexte incertain peuvent être identifiées :

- La majorité des changements de mode ne peut être qualifiée de malveillante ou de suspicieuse. On observe ainsi une grande perte d'information pour le diagnostic des attaques de blocage par rapport au contexte certain puisque ces changements de mode correspondent à un non respect de l'ordonnancement labellisé par δ_{A2} ;
- Les attaques ciblant un blocage partiel du FMS ne sont pas systématiquement diagnosticables (c.f. sous-partie 4.3.3)et peuvent être confondues avec des changements de mode non identifiés comme malveillants ;
- La prise en compte lors de la construction d'une nouvelle itération de $Diag_{cu}$ de tous les labels relatifs aux modes actuels a pour conséquence de créer de nombreux états optimaux au détriment d'états suspicieux (figure 4.14) ;
- Les hypothèses 3 et 4 peuvent engendrer une explosion des changements de modes possibles lorsque ces derniers ne sont pas distinguables les uns des autres et génèrent le même ordonnancement. Il s'agit dans la figure 4.14 des états 2 et 4 où seul le $Mode_0$ est diagnosticable par rapport aux autres modes. Cette explosion s'explique par le fait que les nouvelles ressources devenant indisponibles dans les états 2 et 4 ne sont pas requises par les opérations de la recette A.

Face à ces limites, deux perspectives principales inspirées des travaux sur la planification des opérations de maintenance préventive [370]-[372] sont envisagées. Premièrement, des règles plus restrictives de planification des opérations de maintenance préventive pourraient permettre de réduire considérablement le nombre de changements de mode pouvant avoir lieu (limites 1, 3 et 4) et d'interdire les changements de mode lorsqu'une confusion avec une attaque de blocage partiel pourrait avoir lieu (limite 2). Une méthode de développement automatisée de ces règles à partir des différents modes et marquages atteignables par le FMS pourrait être envisagée.

Deuxièmement, une composante stochastique pourrait être associée aux changements de mode afin d'évaluer leur probabilité d'occurrence. Cette composante pourrait permettre au diagnostiqueur d'estimer si un changement de mode a une forte probabilité de correspondre à une opération de maintenance ou non. Les indicateurs de fiabilité et de maintenabilité des ressources pourraient alors être utilisés pour construire cette composante stochastique. Ainsi, plus la probabilité d'une opération de maintenance sur une ressource du FMS est faible, plus le changement de mode causant l'indisponibilité de cette ressource a des chances d'être malveillant.

Conclusion

Dans ce chapitre 4, le module de diagnostic des attaques de blocage a été étendu au contexte incertain des FMSs. Tout au long de ce chapitre, le fonctionnement global du module de diagnostic dans un contexte incertain a été illustré par plusieurs diagrammes d'activité UML (figures 4.1, 4.6 et 4.9) et a été présenté en trois parties. Premièrement, les méthodes de

prévention et d'ordonnancement introduite dans le chapitre 3 (partie 3.2) ont été adaptées à ce contexte (4.1). Puis, un nouveau modèle original d'attaque incluant les attaques contre les événements d'indisponibilité a été proposé (4.2). A partir de ce modèle, l'algorithme de calcul des trois profils d'attaquant a été modifié pour inclure les attaques de changement de mode. Enfin, un algorithme itératif de construction d'un diagnostiqueur commun des changements de mode, des attaques de blocage et des profils d'attaquant a été proposé (4.3). Ce diagnostiqueur itératif est une extension du diagnostiqueur commun introduit dans le chapitre 3 auquel est ajouté le diagnostic des changements de mode. Chaque nouvelle itération est alors construite si un ré-ordonnancement ou un changement de mode a lieu.

En conclusion de ce chapitre 4, des limites de notre méthode de diagnostic ont été exposées, puis des perspectives d'amélioration ont été proposées. Dans le dernier chapitre de ce manuscrit, une implémentation du module de diagnostic sur une plateforme expérimentale est menée afin de mettre en lumière la pertinence et les limites de l'application de notre méthode à un système réel.

Chapitre 5

Application

Introduction

Dans ce dernier chapitre, notre module de diagnostic est appliqué sur une plateforme manufacturière expérimentale. L'objectif de cette application est d'illustrer par un exemple réel les capacités et les résultats de notre module de diagnostic proposé dans les chapitres 3 et 4.

Dans une première partie (5.1), la plateforme est introduite au regard de ses composants physiques et de son réseau industriel de communication. Un fonctionnement FMS de la plateforme est proposé et conduit au développement et à l'implémentation complète d'un module de supervision adapté. Dans la deuxième partie de ce chapitre (5.2), les attaques de blocage sont implémentées sur la plateforme. Unitairement, chaque attaque d'insertion et de suppression d'une décision d'allocation est programmée, puis, ces attaques unitaires sont combinées afin de développer des scénarios d'attaque complexes. Dans la troisième partie de ce chapitre (5.3), trois scénarios d'attaque correspondant à trois profils d'attaquant distincts sont expérimentés sur la plateforme. Les données expérimentales de ces scénarios sont finalement extraites et analysées hors-ligne afin d'illustrer les performance de détection de notre module de diagnostic sur un cas réel.

5.1 Présentation de la plateforme et implémentation du module de supervision

Dans cette première partie, la plateforme manufacturière utilisée pour l'expérimentation de notre méthode de diagnostic des attaques de blocage dans un contexte incertain est introduite et un module de supervision adapté au fonctionnement FMS de la plateforme est développé.

5.1.1 La plateforme "transfert-libre"

Présentation générale de la plateforme

La plateforme manufacturière choisie pour nos expérimentations, dénommée plateforme "transfert-libre", est mise à disposition par le pôle *S.mart* (Systems Manufacturing Academics Ressources Technologies) Rhône-Alpes Ouest, une structure hébergée au sein de l'INSA de

Lyon et mettant à disposition pour les enseignants-chercheurs, doctorants et étudiants des technologies industriels récentes à vocation d'enseignement et de recherche. Dans le cadre de ces travaux expérimentaux, nous souhaitons en premier lieu remercier Rémi Mouquet et Ludovic Pouliquen pour leur accompagnement technique.

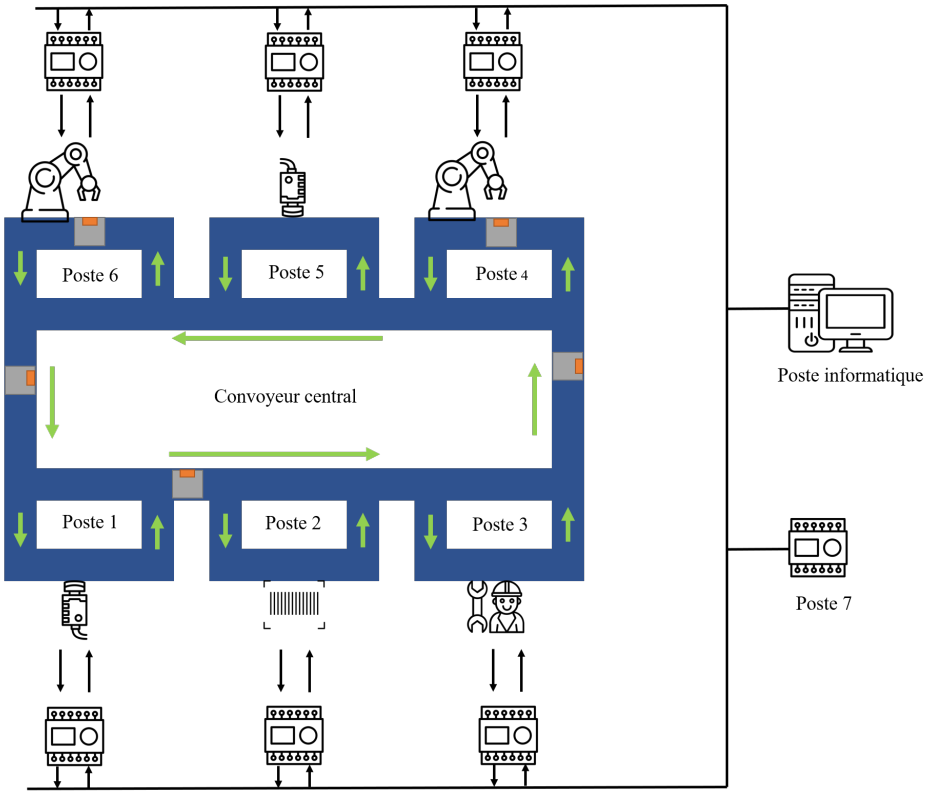


FIGURE 5.1 – Schéma de la plateforme "transfert-libre" et de son architecture réseau

La plateforme transfert-libre, schématisée dans la figure 5.1 et illustrée par la photo 5.2 ci-dessous, est constituée de 6 postes de travail distincts (poste 1 à 6) reliés entre eux par un convoyeur central, et d'un poste de supervision (poste 7). Les postes 1 à 6 réalisent des opérations sur des palettes (carrés gris) équipées d'étiquettes RFID (rectangles oranges). Ces palettes sont transportées au sein des postes par des convoyeurs locaux et d'un poste à l'autre par le convoyeur central selon les flèches vertes de la figure 5.1. Chaque poste de travail est contrôlé par un automate Schneider Modicon M251 programmé à l'aide du logiciel SoMachine depuis un poste informatique du réseau local de la plateforme. Le poste de supervision est lui aussi hébergé sur un automate Schneider Modicon M251. L'architecture réseau de la plateforme est illustrée à droite de la figure 5.1 par de traits noirs connectant les différents automates et poste informatique. Les automates et le poste informatique sont connectés par des câbles Ethernet reliés à un switch et les automates communiquent entre eux via le protocole Modbus TCP/IP.

Présentation d'un poste de travail

Au sein de la plateforme, chaque poste de travail (Poste 1 à 6) est construit selon la même architecture physique et avec les même composants opérationnels (capteurs, actionneurs et

FIGURE 5.2 – Illustration de la plateforme "transfert-libre"

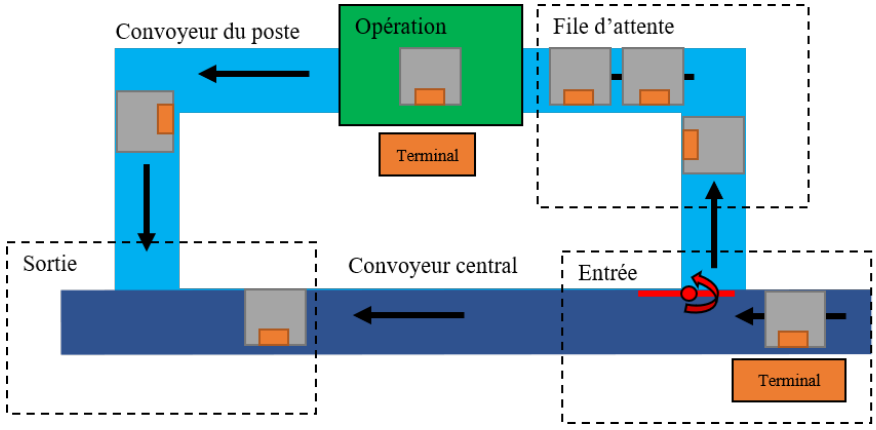


FIGURE 5.3 – Schéma d'un poste de travail

transitique). Cette structure commune à chaque poste de travail est illustrée à travers le schéma 5.3 ci-dessus. L'organisation d'un poste de travail se décompose selon 4 zones : un poste d'entrée, une file d'attente, un poste d'opération et un poste de sortie. Le poste d'entrée a pour mission d'orienter un produit vers le poste d'opération si ce dernier la requiert. Précisement, le produit est bloqué en arrivant au poste d'entrée, son étiquette RFID est lue et son interprétation permet au poste d'entrée de diriger le produit à l'aide d'un rail d'aiguillage sur le convoyeur du poste ou de le laisser sur le convoyeur central en direction du poste suivant. Lorsqu'une nouvelle palette est acceptée dans le poste d'opération, elle est tout d'abord bloquée par une butée puis indexée afin d'être totalement immobilisée lors de l'opération. Le terminal RFID permet alors de lire sur l'étiquette du produit la recette à réaliser ou attendue. Lorsque l'opération est terminée, l'indexeur et la butée sont baissés afin de libérer la palette. Enfin, le poste de sortie gère la réintroduction de la palette opérée sur le convoyeur centrale afin d'éviter toute collision avec d'autres palettes circulant sur le convoyeur central.

L'ensemble des capteurs et actionneurs d'un poste de travail permettant la réalisation des actions des quatre zones sont reliés aux entrées/sorties de l'automate du poste par des câbles électriques 0-24V/4-20mA. **Les signaux de commande et d'observation échangés entre l'automate et les capteurs/actionneurs via ces câbles sont les signaux bas-niveau de**

la plateforme, considérés fiables dans nos travaux pour le diagnostic d'attaques.

5.1.2 Configuration FMS de la plateforme

La configuration FMS de la plateforme "transfert-libre" que nous avons déployée s'appuie sur l'architecture et le fonctionnement des FMSs présentés dans le premier chapitre de ce manuscrit (1). Cette configuration est introduite dans cette sous-partie selon les cinq étapes suivantes : l'identification des ressources, la définition des recettes, le pilotage d'une décision d'allocation par le poste 7, l'indisponibilité des ressources et la définition de la méthode de prévention et d'ordonnancement.

Identification des ressources

Au sein de la plateforme, les postes de travail 1, 3, 4, 5 et 6 sont des ressources opérationnelles tandis que le poste 2 joue le rôle des places d'entrée-sortie des différentes recettes. Le convoyeur central n'est pas considéré comme une ressource de transitique du FMS. Dans cette configuration, les 5 ressources opérationnelles ne réalisent plus les opérations pour lesquelles elles ont été conçues. Elles conservent le fonctionnement standard présenté à travers le schéma 5.3 (déviation de la palette par le poste d'entrée, blocage au poste d'opération, libération et sortie) tandis que l'opération n'est que simulée par un délai temporel entre l'arrivée et le blocage de la palette au poste d'opération et sa libération. Chacune de ces ressources possède une capacité unitaire. Ainsi, au sein d'un poste opérationnel, un unique produit peut être envoyé vers ce poste jusqu'à la fin de l'opération.

Définition des recettes

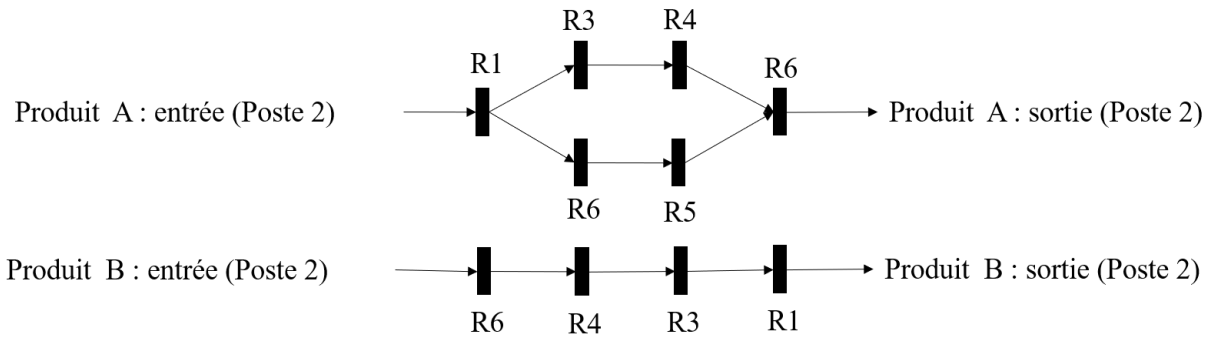


FIGURE 5.4 – Recettes A et B réalisés sur la plateforme "transfert-libre"

Deux recettes A et B requérant ces 5 ressources sont définies et sont illustrées par la figure 5.4. Dans cette figure, les différents postes ou ressources sont notés $R1, R2, R3, R4, R5, R6$. La recette A suit le flux suivant : $R1$, $R3$, $R4$, $R6$ ou $R1$, $R6$, $R5$, $R6$; la recette B le flux : $R6$, $R4$, $R3$ et $R1$. Ces deux recettes partagent des ressources ($R1$, $R3$, $R4$ et $R6$) et un parallélisme d'opération est présent au sein de la recette A. Notons que dans nos travaux expérimentaux, les termes ressources et postes de travail désignent la même entité. Ainsi, la ressource $R6$ est aussi le poste $R6$.

A partir des ressources et des recettes A et B présentées ci-dessus, le modèle S^3PR de la figure 5.5 modélise le fonctionnement FMS souhaité de la plateforme transfert-libre. Les deux

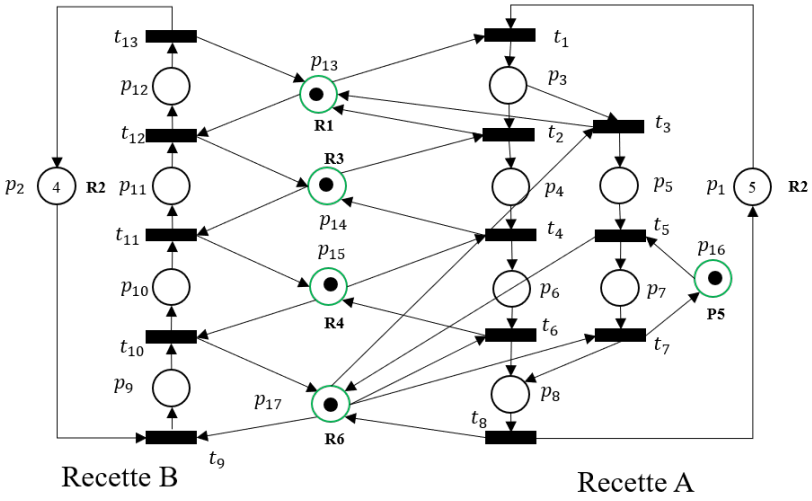


FIGURE 5.5 – Modèle S^{3}PR de la configuration FMS de la plateforme "transfert-libre"

recettes A et B du modèle S^{3}PR suivent les séquences d'opérations présentées dans la figure 5.4 et les capacités des différentes ressources sont bien unitaires.

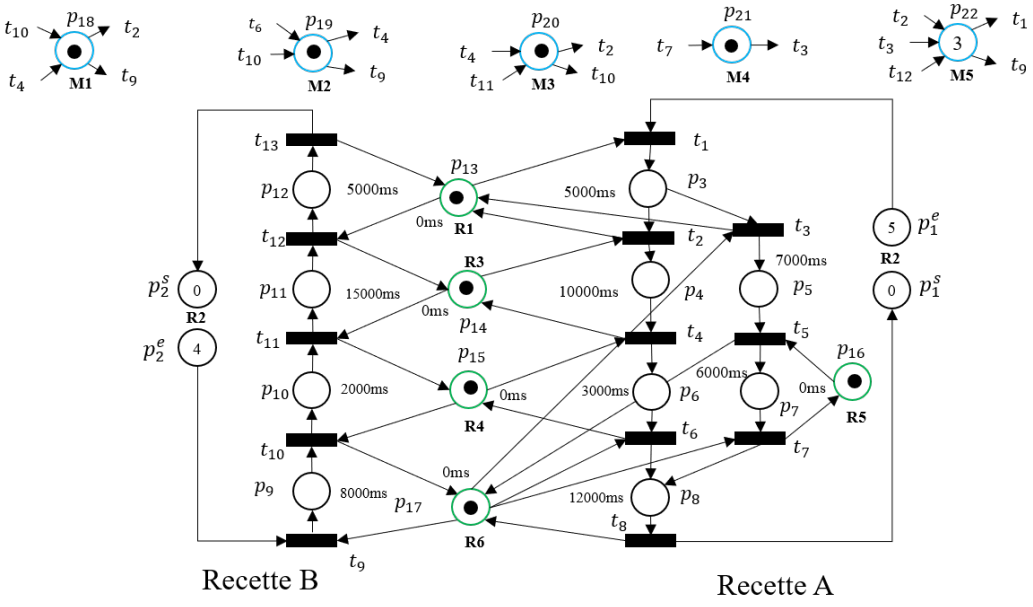


FIGURE 5.6 – Modèle $SC-net$ de la configuration FMS de la plateforme "transfert-libre"

Ce modèle de la figure 5.5, noté N, est étendu à un modèle $SC-net$ suivant la méthode présentée dans la partie 3.2. Ce modèle $SC-net$ est noté N_t et est illustré dans la figure 5.6 où des places d'entrées et de sorties remplacent les places d'attente de N. Les places activité de ce modèle sont temporisées et les temps de chaque opération sont annotés à coté des places correspondantes. Notons qu'au sein de la plateforme "transfert-libre" les temps de préparation des ressources sont nuls ; la ressource est disponible dès que la palette a été libérée par le poste d'opération. Ces temps de préparation sont modélisés par des "*0ms*" au sein de la figure 5.6 de N_t. La programmation des recettes et du modèle N_t est présentée dans la sous-partie 5.1.4.

Pilotage d'une décision d'allocation

Le pilotage de la plateforme est contrôlé par l'automate du poste 7 défini comme le module de supervision du FMS. Le module de supervision du poste 7 est chargé de l'allocation des 5 ressources aux produits en cours de fabrication des deux recettes présentées ci-dessus. Une décision d'allocation des ressources prise par le poste 7 respecte la capacité unitaire des 5 ressources. Un produit ne peut donc être envoyé d'un poste à un autre poste que si le second est libre. Le cas contraire, le produit reste bloqué au sein du poste d'opération du premier poste maintenant son statut de ressource non disponible.

Par conséquent, en prenant l'exemple de la première opération de la recette B, le poste 7 ne peut piloter la libération du produit au poste $R6$ que lorsque l'opération en $R6$ est terminée et lorsque le poste $R4$ requis pour l'opération suivante est libre et n'a pas déjà été alloué. La décision d'allocation ainsi prise suit six étapes :

1. Le poste 7 reçoit l'événement indiquant que l'opération de $R6$ est terminée ;
2. Le poste 7 vérifie que la ressource $R4$ requise par l'opération suivante est disponible ;
3. Au sein du poste 7, le poste $R4$ est réservé pour l'opération suivante du produit en $R6$;
4. Via le terminal RFID du poste d'opération de $R6$, la prochaine destination de la palette est marquée sur son étiquette RFID, à savoir le poste $R4$. Lors de cette étape, le temps de délai simulé de la prochaine opération par $R4$ est aussi inscrit sur l'étiquette RFID de la palette ;
5. La palette est libérée du poste d'opération de $R6$;
6. Une fois la palette libérée, la ressource $R6$ devient disponible pour le poste 7.

La libération de la palette par $R6$ correspond donc au début de la nouvelle opération réalisée par $R4$. Le transport de la palette par le convoyeur central de $R6$ à $R4$ est donc inclus dans la nouvelle opération. En résumé, pour le module de supervision de $R7$, l'opération par $R4$ commence lorsque $R4$ est réservée (étape 3) et se termine à la fin de l'opération de $R4$ (étape 1). Par ailleurs, le temps de délai inscrit dans l'étiquette RFID de la palette lors de l'étape 4 est lu par le poste $R4$ lorsque la palette est correctement bloquée au sein de son poste d'opération puis est exécuté afin de simuler l'opération. La programmation d'une décision d'allocation est présentée dans la sous-partie 5.1.3.

Indisponibilité des ressources

Au sein de la plateforme "transfert-libre", toutes les ressources correspondant aux postes $R1, R3, R4, R5, R6$ peuvent devenir indisponibles. Un modèle incluant les réseaux de reprise de chaque place activité et ressource de N est construit à partir de la méthode présentée dans la partie 1.2. Ce modèle est noté $N_{\mathcal{R}_{ind}}$ avec $\mathcal{R}_{ind} = \{R1, R3, R4, R5, R6\}$. La programmation des indisponibilités est présentée dans la sous-partie 5.1.5.

Méthode de prévention et d'ordonnancement

Le module de supervision du poste 7 est équipé de la méthode de prévention et d'ordonnancement développée dans nos travaux. Pour notre expérimentation et selon les scénarios d'attaque choisis, seules les places de contrôle du $Mode_0$ (sans indisponibilité) de la plateforme sont

programmées. Ces places de contrôle sont illustrées en haut de la figure 5.6. La programmation de ces places de contrôle est présenté dans la sous-partie 5.1.6.

La méthode d'ordonnancement n'est pas implémentée au sein de l'automate du poste 7. Seul un ordonnancement fixe calculé hors-ligne à partir du modèle 5.6 est exécuté depuis l'état initial de la plateforme : aucune opération n'est en cours et toutes les palettes sont en attente en entrée du poste 2. Cet ordonnancement partiel est le suivant : $\sigma_f = t_1t_9t_{10}t_3t_5t_9t_{11}t_{10}t_7t_{12}t_{13}t_{11}t_1t_8t_3t_5t_9t_{10}t_7t_{12}$. La programmation de cet ordonnancement et de son exécution est présentée dans la sous-partie 5.1.6.

Dans les prochaines sous-parties, notre programmation au sein des automates 1 à 7 du fonctionnement FMS de la plateforme est présentée selon les quatre étapes d'implémentation suivantes : implémentation d'une décision d'allocation, implémentation des recettes, implémentation des indisponibilités et implémentation de la méthode de prévention et d'ordonnancement.

5.1.3 Implémentation d'une décision d'allocation et d'une opération

La programmation des recettes A et B que nous souhaitons produire sur la plateforme "transfert-libre" se traduit par l'implémentation du modèle SC-net de la figure 5.6 au sein de l'automate du poste 7. Ce modèle se décompose en une succession de transitions et de places représentant des décisions d'allocation et les opérations débutées par ces décisions. L'objectif de cette sous-partie est de présenter la programmation d'une décision d'allocation et de l'opération qu'elle débute au sein de l'automate du poste 7 et des automates des postes de travail concernés par cette décision. A partir de la programmation d'une décision d'allocation (une transition de N_t) et de son opération (une place de N_t), nous pourrons programmer l'ensemble des recette A et B introduites précédemment.

Dans cette sous-partie, la décision de débuter la deuxième opération de la recette B, requérant le poste $R4$ et libérant le poste $R6$, est utilisée comme exemple directeur. Premièrement, tous les événements de contrôle commande associés à cette décision et échangés entre l'automate du superviseur (poste 7) et les automates des ressources $R6$ et $R4$ sont identifiés. Deuxièmement, un modèle RdP de la loi de commande du poste d'opération de $R6$ est proposé et intègre les événements identifiés préalablement. Troisièmement, la transition et les places associées à la décision d'allocation (t_{10}) et aux opérations avant (p_9) et après (p_{10}) la décision dans le SC-net N_t sont étendues afin d'intégrer elles aussi ces événements. Quatrièmement, le modèle de la loi de commande et le modèle étendu de la décision d'allocation sont fusionnés afin de représenter la communication des événements entre l'automate du poste 7 et ceux des postes $R6$ et $R4$. Enfin, notre programmation au sein du logiciel SoMachine de ces différents modèles est exposée.

Identification des événements pour le pilotage d'une décision d'allocation

Les événements partagés entre les ressources $R6$ et $R4$ et le superviseur (poste 7) peuvent être identifiés à partir des étapes d'une décision d'allocation présentés précédemment (sous-partie 5.1.2). Ces étapes sont complétées par des événements nécessaires à la robustesse de la communication entre le superviseur et l es automates des postes $R6$ et $R4$. Les événements identifiés triés par occurrence sont les suivants :

1. L'opération 0_1^B au poste $R6$ est terminée. L'événement d'observation $e_{fin}(0_1^B)$ est envoyé au superviseur $R7$ depuis $R6$;
2. Une décision d'allocation est prise par le superviseur ; l'opération O_2^B est débutée. L'événement de commande $e_{deb}(O_2^B)$ est envoyé de $R7$ à $R6$. Cet événement contient le prochain poste $R4$ requis par la palette et le temps de délai de O_2^B ;
3. L'automate de $R6$ confirme la bonne réception de la décision d'allocation et des informations associées. L'événement d'observation $e_{rec}(O_2^B)$ est envoyé de $R6$ au superviseur ;
4. Le superviseur ordonne l'écriture de ces nouvelles informations sur l'étiquette RFID de la palette. L'événement de commande $e_{ecr}(O_2^B)$ est envoyé de $R7$ à $R6$;
5. L'automate de $R6$ confirme la bonne écriture des informations sur l'étiquette. L'événement d'observation $e_{fecr}(O_2^B)$ est communiqué de $R6$ à $R7$;
6. La palette est libérée de $R6$. Une commande de libération $e_{lib}(R6)$ est envoyée de $R7$ à $R6$;
7. La palette a correctement été libéré par $R6$. Un événement d'observation $e_{dis}(R6)$ attestant cette libération est transmis de $R6$ à $R7$;
8. Le superviseur confirme à $R6$ la bonne réception de $e_{dis}(R6)$ afin de réinitialiser la loi de commande du poste d'opération de $R6$. L'événement de commande $e_{rec}(R6)$ est envoyé de $R7$ à l'automate de $R6$.

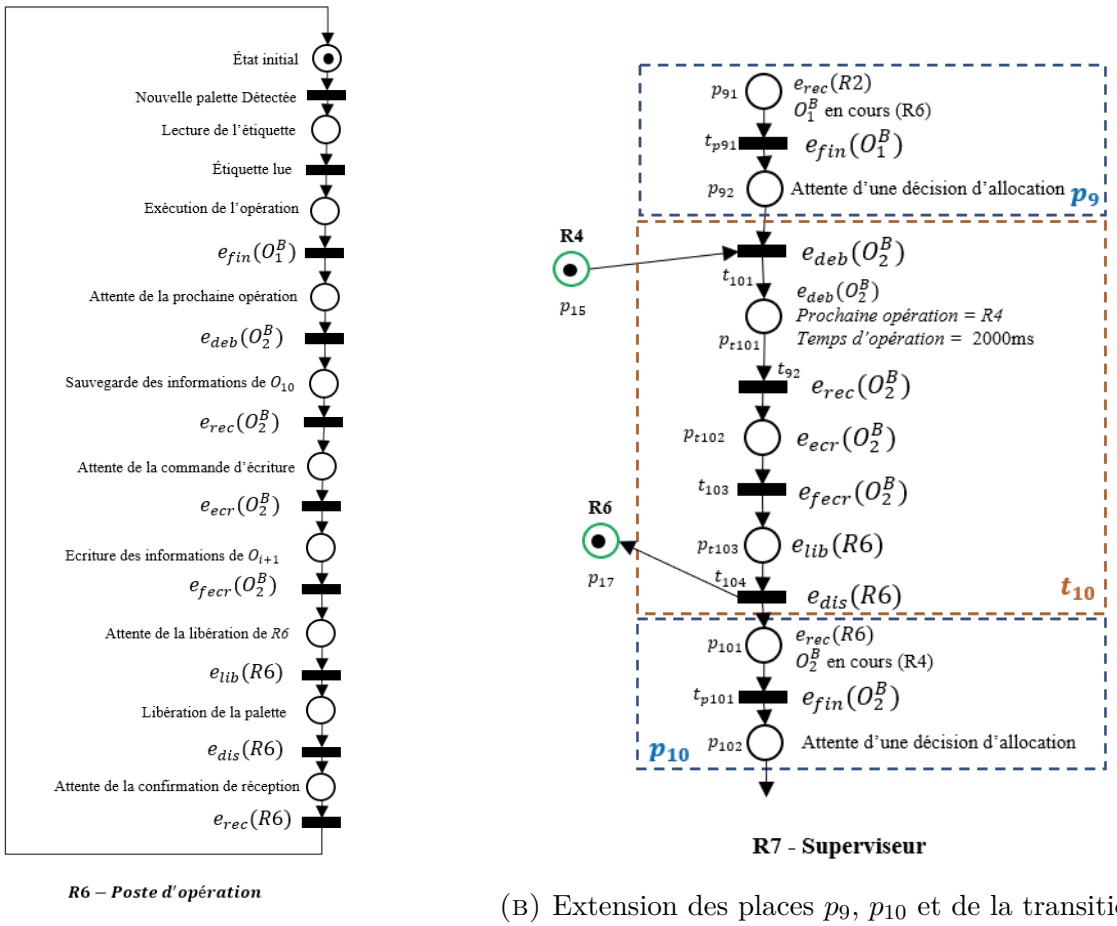
A partir de l'étape 8, l'opération O_2^B débute et correspond à la deuxième opération de la recette B. Celle-ci est ensuite exécutée sans requérir l'intervention du superviseur puisque le transport de la palette sur le convoyeur central et son opération par $R4$ ne nécessite que les informations écrites sur son étiquette, soit son poste de destination et son temps de délai simulé. L'opération O_2^B se termine alors pour le superviseur lorsqu'il reçoit l'événement de fin d'opération $e_{fin}(O_2^B)$ depuis $R4$ (étape 1).

La robustesse de la communication entre le superviseur et les ressources est ici garantie par l'alternance systématique entre un événement d'observation et un événement de commande. Le superviseur ne génère un événement de commande qu'à partir de la réception depuis une ressource d'un événement d'observation. Cette robustesse permet d'assurer la synchronisation entre la loi de pilotage du superviseur et les lois de commande des ressources.Dans notre exemple, l'automate du poste $R6$ pourrait succéder la réception de la décision d'allocation (étape 2) et l'écriture des informations contenues dans cette décision au sein de l'étiquette de la palette (étape 5) sans envoyer au superviseur $R7$ une confirmation de bonne réception de la décision. Cependant, $R7$ doit être certain que les informations écrites sont les dernières reçues et non celles de la dernière décision d'allocation exécutée par $R4$. L'étape 3 permet ainsi de confirmer à $R7$ la bonne réception de la nouvelle décision et la mise à jour des informations à écrire par $R6$ avant d'envoyer à cette dernière la commande d'écriture.

Remarque 5.1.1. Au sein du poste $R4$, le poste d'entrée n'est pas piloté par le superviseur. En effet, tel que détaillée dans la partie précédente, sa tâche consiste à bloquer toute nouvelle palette, à lire sur son étiquette RFID sa prochaine ressource requise et à dévier la palette sur le convoyeur du poste si la prochaine ressource requise correspond au poste. Le pilotage a ici lieu au sein du poste d'opération de la précédente ressource détenue par la palette ($R6$ ici) lors de l'écriture de la prochaine ressource requise sur l'étiquette de cette palette (étape (4) d'une décision d'allocation). Identiquement, le poste de sortie de $R4$ n'est pas piloté par le

superviseur car sa mission est d'éviter les collisions entre les palettes sortant du poste avec celles circulant sur le convoyeur central à l'aide uniquement des capteurs et actionneurs contrôlés par l'automate de $R4$. Par conséquent, les postes d'entrée et de sortie d'une ressource de la plateforme ne partagent pas d'événements avec le superviseur et sont contrôlés uniquement par l'automate propre à la ressource.

Modèle RdP de la loi de commande d'un poste de travail



(A) Modèle RdP de la loi de commande du poste de travail de $R6$

(B) Extension des places p_9, p_{10} et de la transition t_{10} de N_t aux événements de la décision d'allocation de débuter O_2^B

FIGURE 5.7 – Modèles RdP pour le pilotage FMS de la plateforme "transfert-libre"

Au sein de l'automate du poste $R6$, la loi de commande du poste d'opération de $R6$ intègre les événements échangés avec le superviseur. Cette loi de commande séquentielle est divisée selon les neufs étapes suivantes :

1. État initial : le poste d'opération de $R6$ est dans l'attente d'une nouvelle palette ;
2. Une nouvelle palette est détectée au poste d'opération et est bloquée. L'étiquette de la palette est lue afin d'extraire le temps simulé d'opération ;
3. Le poste $R6$ réalise l'opération ;
4. L'opération est terminée par $R6$ et la prochaine décision d'allocation est attendue $(e_{fin}(O_1^B))$;

5. La nouvelle décision d'allocation est reçue par $R6$ ($e_{deb}(O_2^B)$). $R6$ en notifie le superviseur $R7$ ($e_{rec}(O_2^B)$) ;
6. La prochaine opération (numéro du poste, délai d'opération) est écrite sur l'étiquette de la palette ($e_{ecr}(O_2^B)$). L'écriture a correctement été réalisée par le poste d'opération de $R6$ ($e_{fecr}(O_2^B)$) et $R7$ en est informé ;
7. La palette est libérée par le poste d'opération ($e_{lib}(R6)$). La palette a correctement été libérée par $R6$ ($e_{dis}(R6)$) et $R7$ en est informé ;
8. L'automate du poste $R6$ reçoit la confirmation de la réception par $R7$ de son événement de bonne libération ($e_{rec}(R6)$), retour à l'état initial.

Cette loi de commande du poste d'opération de $R6$ peut être modélisée par le RdP séquentiel illustré à gauche de la figure 5.7 et représentant par ses transitions les 8 étapes introduites précédemment. Dans ce RdP, les transitions modélisent tous les événements de commande et d'observations, échangés ou non avec S, nécessaires à la loi de commande du poste d'opération. A droite de la figure, un RdP alternatif représentant la loi de commande du poste d'opération de $R2$ (poste d'entrée/sortie du FMS) est représenté. Cette loi de commande diffère de celle des autres postes par l'absence des étapes de lecture de l'étiquette et d'opération. En effet, au poste 2, les palettes n'ont pas encore été associées à une recette. Le poste $R2$ informe S de la disponibilité de ces palettes neutres en entrée du FMS à travers l'événement $e_{pr}(R2)$, i.e. une palette est "prête" en $R2$ à débuter une recette.

Extension du modèle N_t aux événements d'une décision d'allocation

La loi de pilotage de $R7$ est elle modélisée par une extension du modèle N_t. Cette extension illustrée à droite de la figure 5.7, étend les places activité p_9 et p_{10} de N_t en un sous-réseau composé de deux places et une transition et la transition t_{10} de N_t en un sous-réseau incluant 3 places et 4 transitions. Le sous-réseau de la place p_{10} modélise l'exécution de l'opération O_2^B, à savoir de la libération de la palette par $R6$ à la fin de l'opération et à l'attente de la prochaine décision d'allocation par $R4$. Le second sous-réseau, celui de la transition t_{10} de N_t, modélise la décision d'allocation de débuter l'opération O_2^B par$R4$. Ce sous sous-réseau s'étend de la prise et l'envoie de la décision à la ressource $R6$ jusqu'à la réception de l'observation de libération de la palette par cette même ressource $R6$. Dans chaque sous-réseau, les transitions sont labellisées et franchies par les événements d'observation reçus depuis les différents postes d'opération tandis que les places sont labellisées par les événements de commande envoyés de S aux ressources.

Fusion des modèles

Les deux modèles présentés dans les sous-parties précédentes, soit le RdP de la loi de commande du poste d'opération de $R6$ et l'extension des places et transitions p_9,p_{10} et t_{10} de N_t, sont fusionnés afin de modéliser la communication des évenements entre $R6$ et $R7$. Cette fusion est illustrée dans la figure 5.8 et le modèle de la loi de commande de $R2$ ainsi que l'extension de la transition t_9 sont modélisées. Dans cette figure, la communication des événements partagés entre la loi de commande de $R6$ et la loi de pilotage de $R7$ est modélisée par l'ajout de places reliant les transitions des deux modèles présentés ci-dessus. Une place communication, annotée C dans la figure 5.8, devient marquée lorsque l'événement qu'elle communique a lieu et est vidée lorsque cet événement a correctement été reçu par son destinataire et a permis de faire avancer sa loi de commande ou de pilotage. Dans le cas des événements de commande labellisant les

places du modèle étendu de N_t, ils sont émis lors du franchissement des transitions les précédent en supposant que dès lors que la transition est franchie, la commande est envoyée. Néanmoins, la commande $e_{deb}(O_2^B)$ est elle modélisée par une transition puisque une décision d'allocation n'est pas systématiquement prise dès la fin d'une opération dont l'événement labellise la précédente transition.

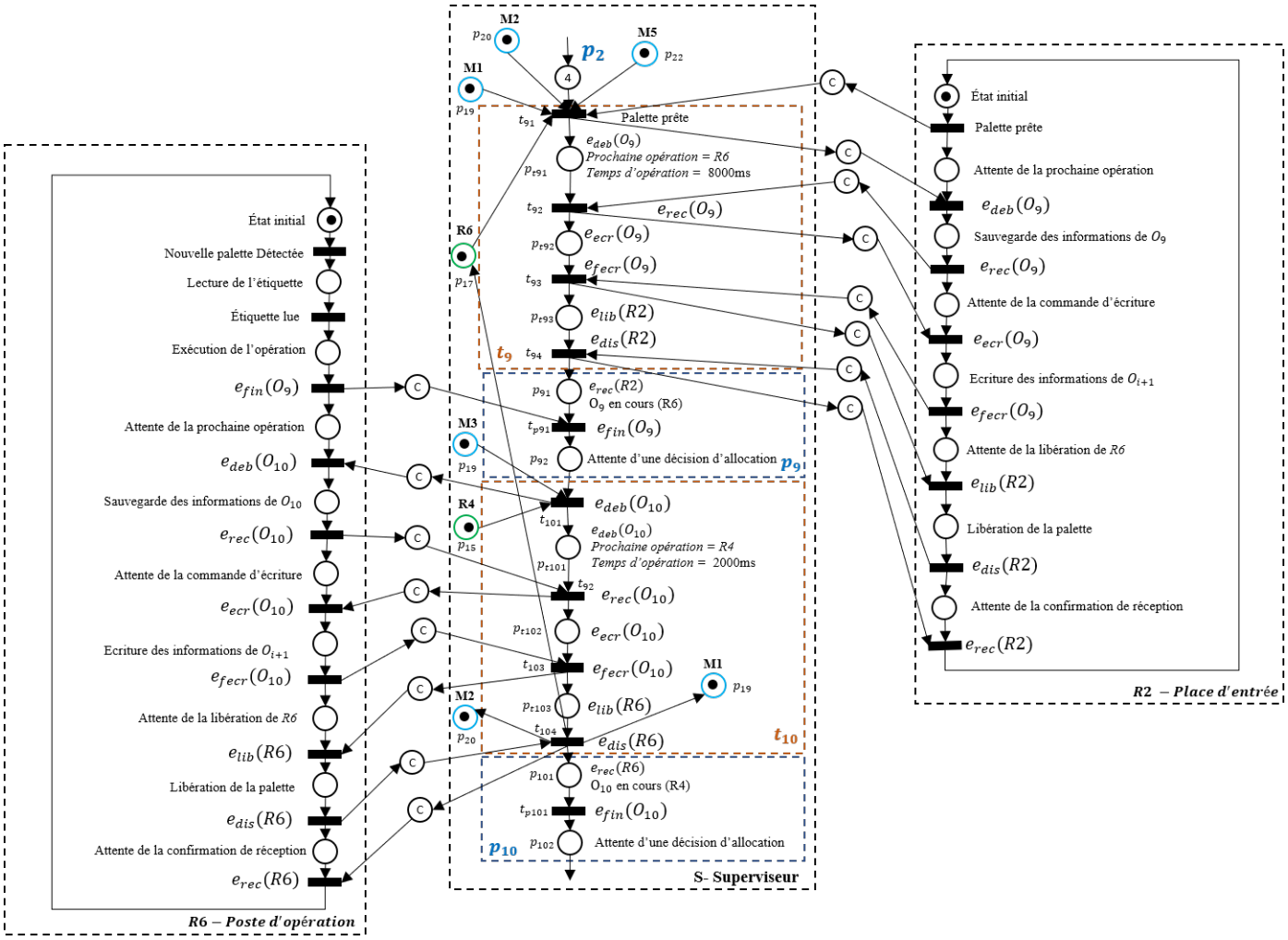


FIGURE 5.8 – Modèle RdP étendu du pilotage FMS de la plateforme par S

Programmation sous SoMachine

La programmation du modèle global illustré partiellement dans la figure 5.8 est réalisée en trois temps.

Premièrement, la programmation du modèle de pilotage au sein de l'automate 7 repose sur la conversion des RdPs en langage Ladder proposée dans les travaux de [373]. Dans cette conversion, chaque place p est une variable interne à l'automate dont la valeur indique le marquage de la place. Dans un premier programme Ladder (à droite de la figure 5.9), chaque transition t est une ligne de Ladder où les entrées (ou contacts) sont les variables des places précédent la transition ($p \in {}^\bullet t$) ainsi que les variables des événements requis pour le franchissement de la transition tandis que les sorties de cette ligne (ou bobines) sont le marquage des places de sortie

de la transition ($p \in t^\bullet$) et le vidage de ses places d'entrées ($p \in^\bullet t$). Dans un nouveau programme Ladder (à droite de la figure 5.9), les actions ou commandes engendrées par le marquage d'une place sont transcrites sous la forme de lignes Ladder où les entrées sont les variables des places dont le marquage engendre l'action et les sorties sont les variables de commande devant être modifiées.

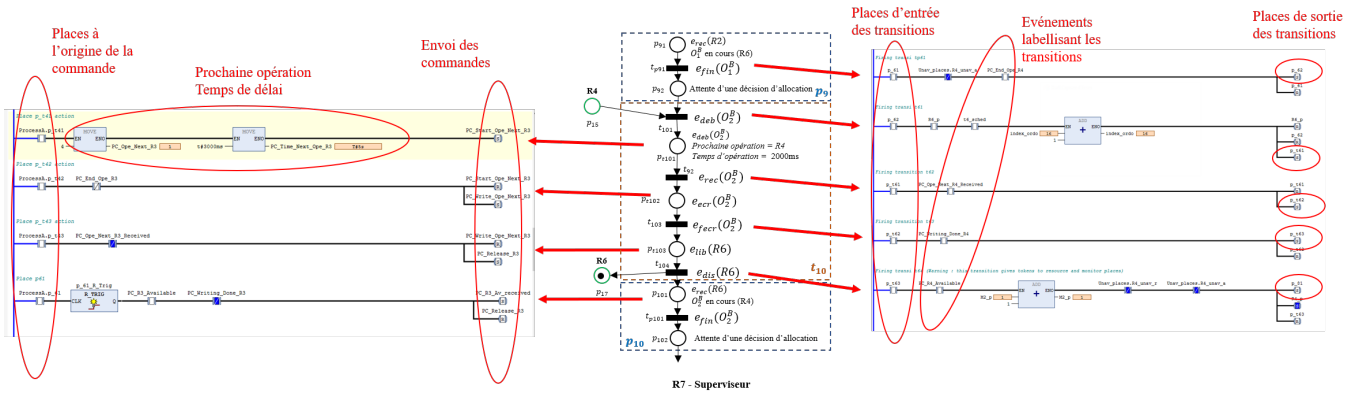


FIGURE 5.9 – Programmation sous SoMachine du RdP d'une décision d'allocation

Deuxièmement, la programmation de la loi de commande du poste d'opération de $R6$ est achevée via l'outil de modélisation des Grafcets ou SFC (Sequential Function Chart Language) conjointement au langage Ladder. Le premier outils permet de programmer la séquence d'étapes de la loi de commande tandis que le second les actions à réaliser lorsqu'une nouvelle étape est atteinte après le franchissement d'une transition. Les programmes de ces lois de commande sont illustrés dans la figure 5.10.

Troisièmement, la communication des événements partagées est programmée à l'aide d'une option du logiciel SoMachine permettant d'avoir recours à des variables partagées entre deux automates Schneider. Ces communications sont configurées pour représenter avec exactitude la communication d'un événement discret, à savoir qu'une trame est envoyée d'un automate à un autre uniquement lorsque la variable partagée change de valeur dans l'automate d'origine, i.e. lorsque l'événement discret a lieu. Au sein de l'automate 7, des variables internes à celui-ci sont définies afin de modéliser les places annotées C dans 5.8 et viennent prendre les valeurs des variables partagées (observation) ou sont copiées dans ces variables partagées (commande) afin de modéliser la communication d'un événement entre S et les automates des différents postes. Ces variables de communication seront requises pour la modélisation des attaques. Ainsi, les variables d'observation et de commande communiquées entre $R1$ et $R7$ sont illustrées par les captures d'écran 5.12 et 5.13 tandis que la configuration de la communication de ces variables entre $R6$ et $R7$ est présenté dans l'image 5.11.

5.1.4 Implémentation des recettes

Notre programmation des recettes au sein de l'automate du poste 7 s'appuie sur les travaux présentés dans la partie précédente (5.1.3). Ainsi, chaque chaque place et transition du modèle $SC - net$ (figure 5.6) est étendue suivant la méthode utilisée pour les places et transitions p_9, p_{10} et t_{10} et illustrée dans la figure 5.9. En parallèle, les lois de commande des postes d'opération des autres ressources $R1, R2, R3, R4$ et $R5$ sont programmées dans leurs automates respectifs.

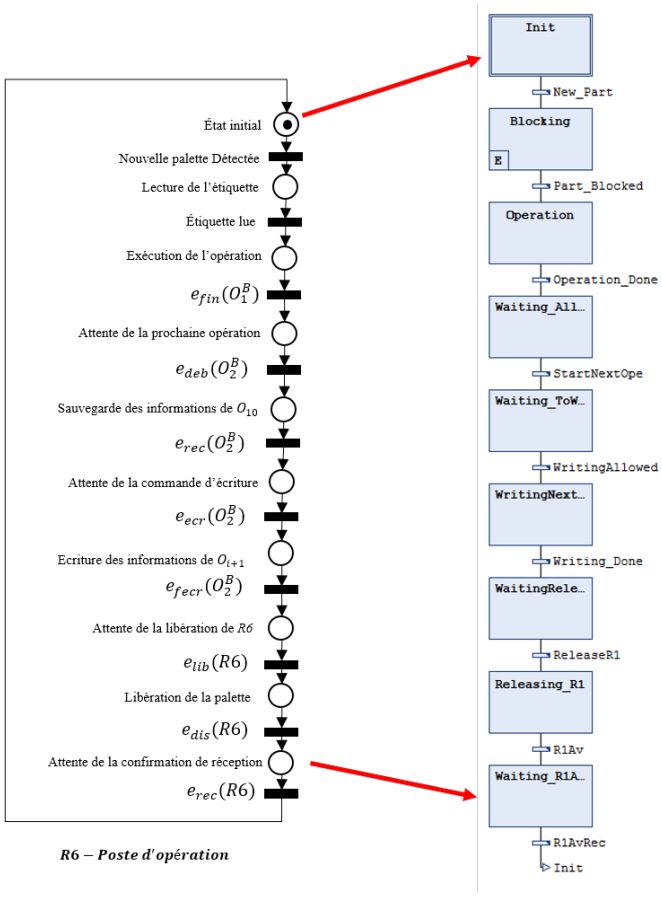


FIGURE 5.10 – Programmation sous SoMachine des commandes envoyées par une décision d'allocation

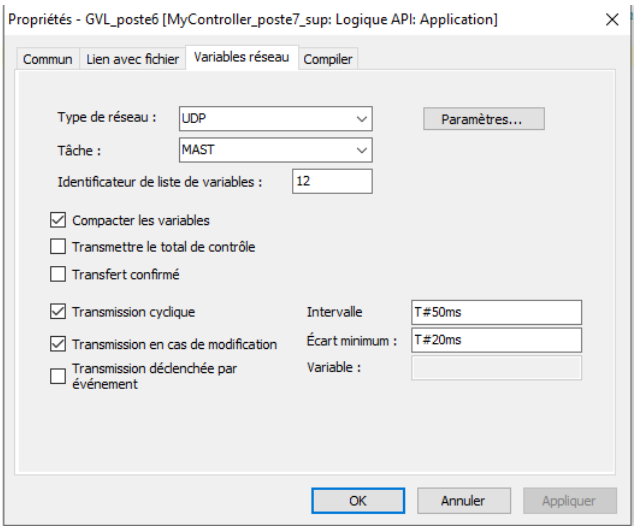


FIGURE 5.11 – Configuration sous SoMachine de la communication de variables entre $R6$ et $R7$

Ces lois de commande sont implémentées identiquement à celle du poste $R6$ illustrée dans la figure 5.10. Enfin, les événements partagés entre le superviseur $R7$ et les différentes ressources sont programmées selon la méthode exposée dans les figures 5.12 et 5.13.

```
// PLC variable for the ressource allocation loop (shared with the supervisor - PLC7)
    Start_Ope_R1 : BOOL ;              // PLC1 to Sup // = start of an operation on R1
    End_Ope_R1 : BOOL ;                 // PLC1 to Sup // = 1 : observation meaning the operation at station 1 has terminated
    Ope_Next_R1_Received  : BOOL ;          // PLC1 to Sup // = 1  : observation meaning the command for new operation to start has been recieved (+ Next_Ope_R1_Time /
    Writing_Done_R1 : BOOL ;        // PLC1 to Sup // = 1 : observation meaning the writting on the part lavel of "Next_Ope_R1_Time" & "Ope_Next_R1" has been executed co
    R1_Available : BOOL ;              // PLC1 to Sup // = 1 :  Station R1 has been freed properly and is free, = 0 : Station R1 is busy with a part
```

FIGURE 5.12 – Événements d'observation communiqués de $R1$ à $R7$

```
// Station 1
Ope_Next_R1  : INT ;                // Sup to PLC 1 // Next Operation station for the part waiting in Working station 1
Time_Next_Ope_R1 : TIME ;             // Sup to PLC 1 // Next Operation time for the part waiting in Working station 1
Start_Ope_Next_R1 : BOOL ;          // Sup to PLC 1 // "Command" Boolean indicating to the Working Station 1 that an allocation decision has been taken => the next o
Write_Ope_Next_R1 : BOOL ;          // Sup to PLC 1 // Command validating the writting of "Ope_Next_R1" and "Time_Ope_R1" on the part label (part in station 1)
Release_R1 : BOOL ;                 // Sup to PLC 1 // = 1 . Command from the Sup to release the part in Working Station 1
R1_Av_received : BOOL ;             // Sup to PLC 1 // = 1 . Indicate to station 1 that the supervisor has recieved correctly the information that R1 is now free
```

FIGURE 5.13 – Événements de commande communiqués de $R7$ à $R1$

Cependant, la programmation du poste de travail $R2$, dont le rôle est le lancement de nouveaux produits et la réception de produits finis, diffère des autres postes. Sa loi de commande et son pilotage associé dans $R7$ sont illustrés à droite de la figure 5.8. Au sein de la programmationn des recettes dans l'automate du superviseur $R7$, deux variables de décompte des palettes à produire et des palettes produite sont crées et son illustrées dans le langage Ladder de la capture d'écran 5.14 ci-dessous.

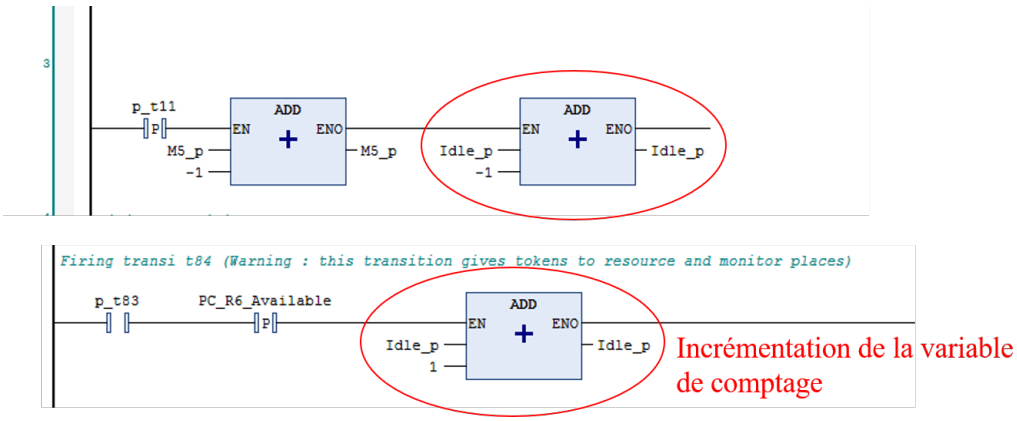


FIGURE 5.14 – Programmation sous SoMachine de l'incrémentation et de la décrémentation des variables de comptage de palettes

A total, notre programmation des recettes A et B dans le logiciel SoMachine (figure 5.9) représente 142 lignes de langage Ladder (74 lignes pour la programmation du SC-net et 68 lignes pour la programmation des commandes) et 173 variables internes (108 variables modélisant les places du SC-net et 65 variables les événements d'observation et de commande). Pour sa part, la programmation des lois de commande est une duplication entre toutes les ressources d'un programme générique contenant 9 places et 9 transitions dans son modèle Grafcet, 14 lignes Ladder pour la programmation de ses actions et 10 variables modélisant les événements de commande et d'observation échangées avec $R7$. Enfin, la programmation des événements communiqués entre $R7$ et les ressources (variables des places annotées C) représente 60 variables partagées entre les différents automates.

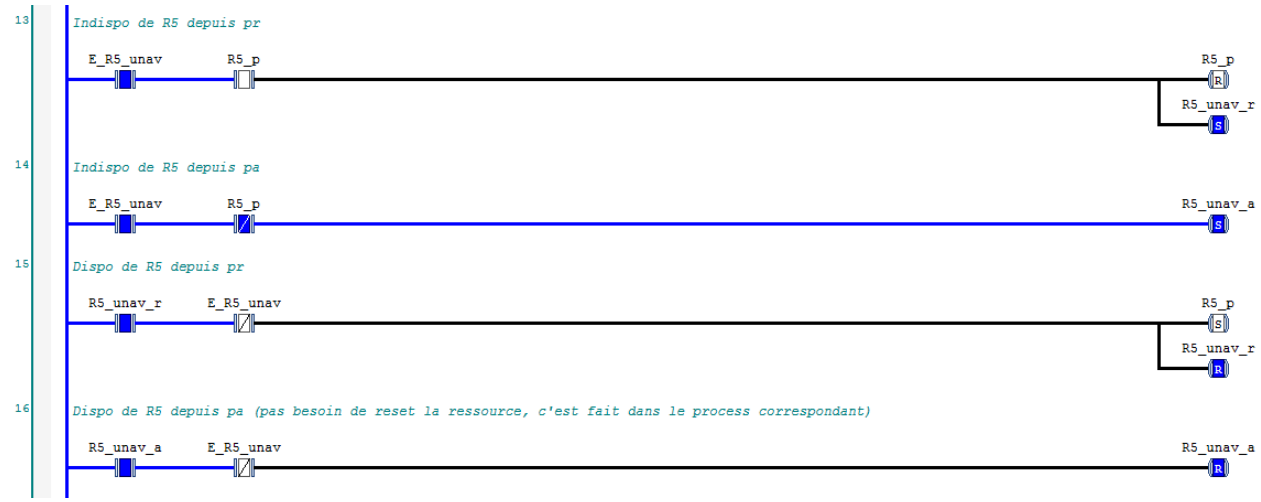


FIGURE 5.15 – Programmation sous SoMachine des variables d'indisponibilité des ressources

5.1.5 Implémentation des indisponibilités

Nous avons implémenté l'indisponibilité des ressources directement dans le programme Ladder de l'automate 7 à travers cinq variables booléenne internes, notées E_R6_unav pour la ressource $R6$, représentant la disponibilité des cinq postes de travail hors $R2$. Une valeur $E\_R6\_unav = 1$ signifie la disponibilité de la ressource $R6$, une valeur de 0 son indisponibilité. Par conséquent, la simulation d'une défaillance du poste $R6$ correspond au forçage à 0 de la variable interne E_R6_unav au sein de l'automate 7. La programmation de ces variables est illustrée dans la figure 5.15 et correspond à 20 lignes Ladder et 15 nouvelles variables. Dans ce programme, on distingue pour chaque ressource si elle est devenue indisponible en opération ($R5\_unav\_a$) ou hors-opération ($R5\_unav\_r$).

Par définition de l'indisponibilité suite à la défaillance d'une ressource, une décision d'allocation ne peut pas être prise par $R7$ si (1) la ressource $R6$ requise pour l'opération suivante est indisponible (indisponibilité d'une ressource à l'arrêt) et/ou (2) si la ressource $R4$ détenue par le produit avant la décision d'allocation est indisponible (indisponibilité d'une ressource en cours d'opération). Dans les figures 5.7 et 5.8, le franchissement de la transition de début d'opération (t_{101} labellisée par $e_{deb}(O_2^B)$) correspond à la prise de cette décision. D'après les points (1) et (2), cette transition ne peut être franchie que si le poste détenu $R6$ et celui requis $R4$ sont non défaillants. Dans le programme Ladder, cela correspond à l'ajout dans la ligne du franchissement de la transition t_{91} représentant la fin de l'opération par $R6$ de l'entrée booléenne $\overline{R6\_unav\_a}$ (le poste $R6$ n'est pas devenue indisponible en cours d'opération) et de l'ajout dans la ligne de franchissement de t_{101} modélisant le début de l'opération par $R4$ de l'entrée booléenne $R4\_p$ (le poste $R4$ est disponible pour la prochaine opération). Ainsi, si l'une de ces deux variables est nulle, i.e. une des deux ressources est indisponible, alors les sorties de la ligne de t_{101} ne sont pas exécutées et l'opération par $R4$ n'est pas débutée. La ligne Ladder correspondant à l'ajoute de ces entrées pour $R4$ libérée et $R6$ requise est illustrée dans la capture d'écran 5.16.

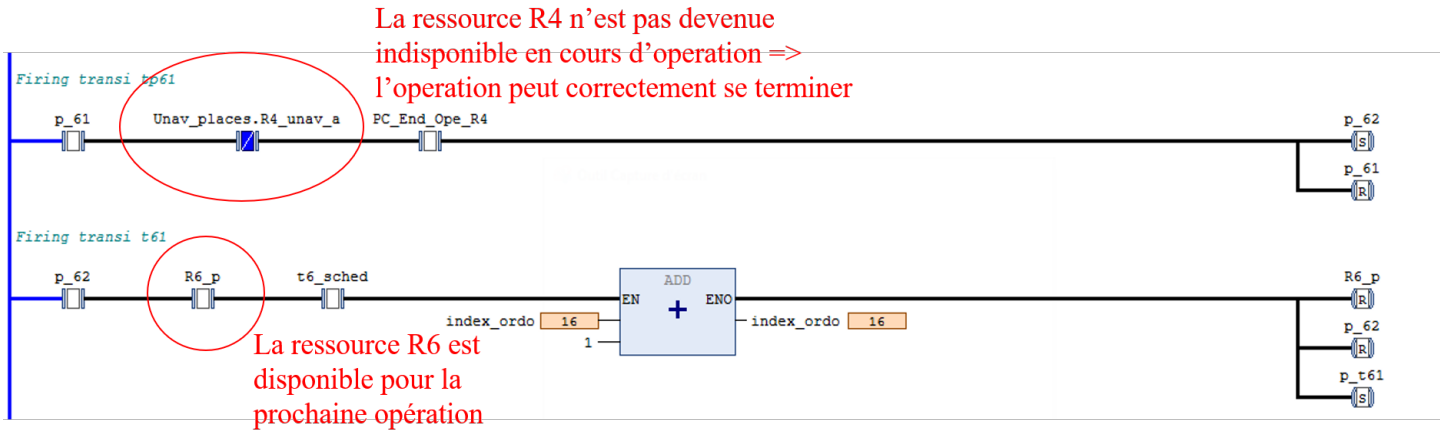


FIGURE 5.16 – Programmation sous SoMachine de la condition de disponibilité des ressources libérées et requises par une décision d'allocation

5.1.6 Implémentation de la méthode de prévention et d'ordonnancement

Dans cette dernière sous-partie, nous avons programmé au sein de l'automate 7 les places de contrôle du $Mode_0$ et l'ordonnancement fixe calculé hors-ligne depuis l'état initial de la plateforme et le $Mode_0$.

Programmation des places de contrôle

Pour chaque place de contrôle du $Mode_0$, nous avons défini une variable dans le programme de l'automate 7. Ainsi, cinq variables internes $M1\_p, M2\_p, M3\_p, M4\_p, M5\_p$ sont crées pour les 5 moniteurs du $Mode_0$ (figure 5.6). Ces variables sont illustrées dans la figure 5.17.

```
75  // Monitor 1
76  M1_p : INT ;
77  // Monitor 2
78  M2_p : INT ;
79  // Monitor 3
80  M3_p : INT ;
81  // Monitor 4
82  M4_p : INT ;
83  // Monitor 5
84  M5_p : INT ;
```

FIGURE 5.17 – Programmation sous SoMachine des variables des places moniteurs du module de supervision de la plateforme

Dans le modèle étendu (5.8), ces différentes variables représentant le marquage de chaque place de contrôle (en bleu dans la figure) sont décrémentées lors du franchissement de transitions symbolisant le début d'une opération (transition t_{101} par exemple) et sont incrémentées lors du franchissement de transitions symbolisant la libération d'une palette (transition t_{104} par exemple). Dans le logiciel Somachine, l'incrémentation et la décrémentation des ces variables est donc directement programmé dans les lignes Ladder des transitions associées. La figure 5.18 ci-dessous illustre la décrémentation du moniteur $M2$ (place p_{20}) et l'incrémentation du moniteur $M1$ (place p_{19}).

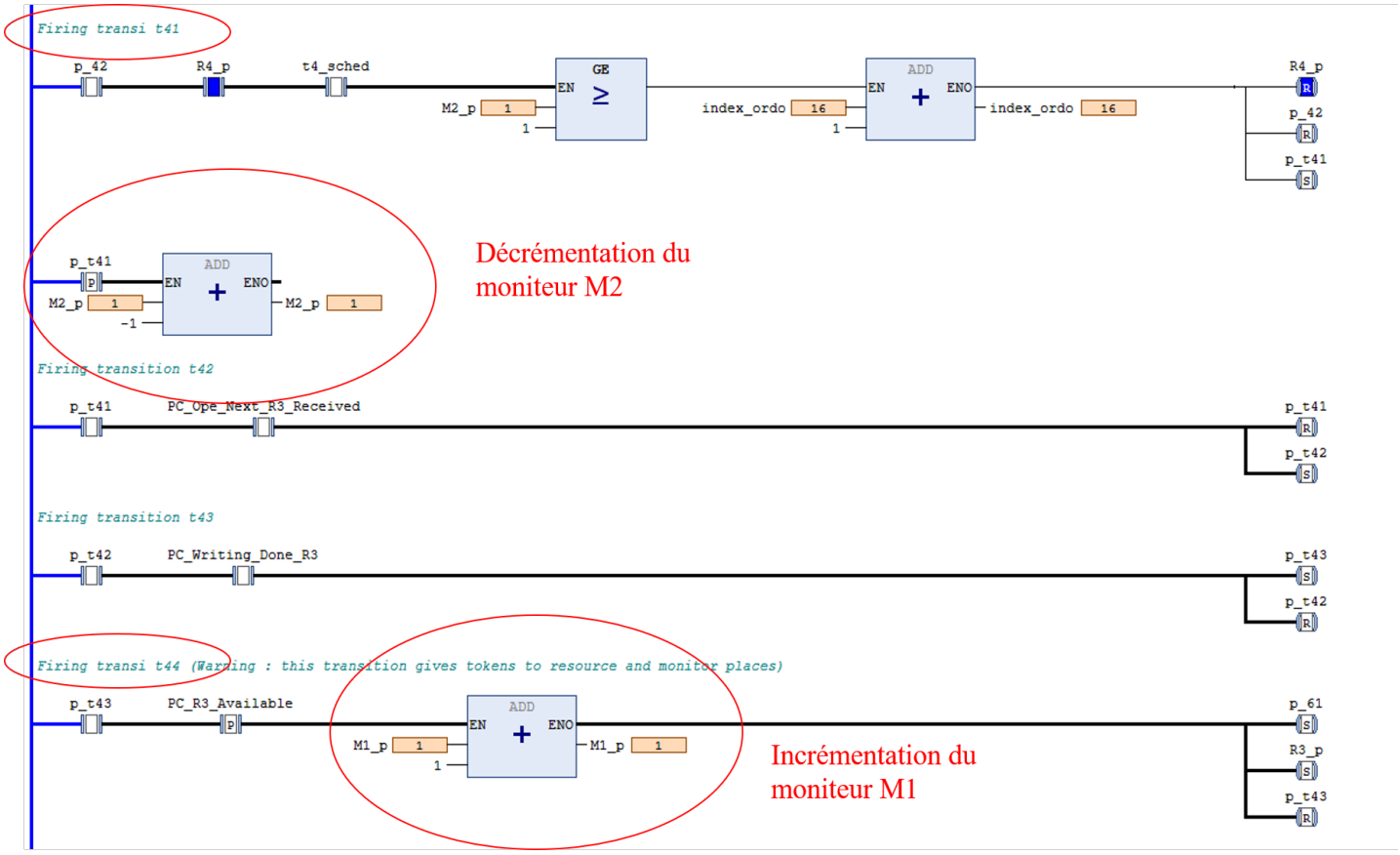


FIGURE 5.18 – Programmation sous SoMachine de la décrémentation et de l'incrémentation des places de contrôle

Programmation de l'ordonnancement

Dans le cadre de notre expérimentation, nous avons programmé l'ordonnancement de manière statique : un seul ordonnancement, celui depuis l'état initial de la plateforme et le $Mode_0$, est calculé hors-ligne et appliqué en ligne depuis l'automate du poste $R7$. Ainsi, l'ordonnancement est programmé sous la forme d'un tableau statique dans la mémoire de l'automate de $R7$. Ce tableau est illustré dans la capture d'écran 5.19.

```
// Scheduling
Ordo : ARRAY[1..50] OF INT := [1,9,10,3,5,9,11,10,7,12,13,11,1,8,3,5,9,10,7,12,11,8,9,13,1,10,3,5,12,13,11,7,8,1,3,12,13,1,5,2,1,7,4,2,8,6,4,8,6,8] ;
index_ordo : INT := 0 ;
```

FIGURE 5.19 – Programmation sous SoMachine de l'ordonnancement statique

L'exécution de l'ordonnancement est programmé grâce à une variable *index_ordo* que nous incrémentons dès qu'une décision d'allocation de l'ordonnancement est prise. Ainsi, cette variable permet au poste $R7$ de savoir l'état précis de l'ordonnancement et de connaitre en temps réel la prochaine décision à exécuter. Dans le logiciel *SoMachine*, une variable *t_sched* est créée pour chaque décision d'allocation des recettes A et B (pour chaque transition du modèle 5.6). Une variable *t_sched* est alors égale à 1 si elle correspond à la prochaine décision d'allocation de l'ordonnancement, et à 0 sinon. Par exemple, pour la décision associée à la transition t_6 de la recette A, nous avons $t_6\_sched = 1$ si $Ordo(index\_ordo) = 6$. La mise à jour des variables

t_sched selon l'état de l'ordonnancement est illustrée dans la figure 5.20 ci-contre et est exécuté par un programme "texte structuré" de 85 lignes.

```
    //t1_sched
IF Ordo[index_ordo] = 1 THEN
    t1_sched := 1 ;
ELSIF Ordo[index_ordo] <> 1 THEN
    t1_sched := 0 ;
END_IF
    //t2_sched
IF Ordo[index_ordo] = 2 THEN
    t2_sched := 1 ;
ELSIF Ordo[index_ordo] <> 2 THEN
    t2_sched := 0 ;
END_IF
```

FIGURE 5.20 – Programmation sous SoMachine de la mise à jour de l'état de l'ordonnancement

Finalement, les différentes variables *t_sched* sont programmées comme des entrées des lignes Ladder correspondant au franchissement des transitions t. Cette programmation Ladder permet de restreindre les décisions d'allocation prises par $R7$ à l'ordonnancement choisi et est ilustrée dans la capture d'écran 5.21 pour la transition t_6. Dans cette figure, l'incrémentation de la variable *index_ordo* est elle aussi programmeé et a lieu dès que la décision associée à t_6 est prise.

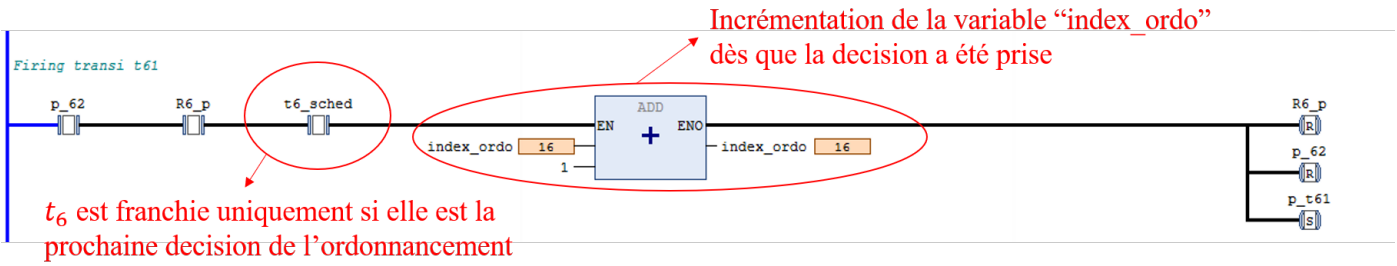


FIGURE 5.21 – Programmation sous SoMachine de la décrémentation et de l'incrémentation des places de contrôle

5.1.7 Conclusion

Dans cette première partie du chapitre 5, notre programmation du module de supervision au sein des automates de la plateforme "transfert-libre" a été présentée. Cette programmation a permis d'obtenir un fonctionnement "FMS" de la plateforme pour la production de deux recettes A et B. Puis, notre méthode de prévention appliquée au mode sans indisponibilité de la plateforme et un ordonnancement statique pour ce mode et depuis l'état initial de la plateforme ont été implémentés dans l'automate du poste $R7$ afin de piloter la production de ces recettes. Toutefois, nous envisageons d'intégrer à l'automate du superviseur $R7$ tous les ordonnancements calculés hors-ligne pour chaque mode de fonctionnement de la plateforme, pour chaque état de l'allocation des postes aux recettes A et B et pour chaque combinaison de produits en entrée de ces recettes, afin que $R7$ puisse sélectionner dynamiquement un ordonnancement. Enfin, nous avons simulé la défaillance des différents poste de travail $R1, R3, R4, R5, R6$ de la plateforme au sein du programme du superviseur $R7$ en contraignant les décisions d'allocation prises par $R7$ à la disponibilité des ressources libérées et requises par chaque décision.

Dans la prochaine partie (5.2), les attaques d'insertion et de suppression de tous les événements d'observation, de commande et de défaillance sont implémentées au sein de la plateforme. A partir de ces attaques d'insertion et de suppression, une méthode de mise en œuvre de scénarios d'attaques de blocage est introduite.

5.2 Implémentation des attaques de blocage

Dans cette partie, les attaques de blocage contre la plateforme "transfert-libre" sont implémentées. Premièrement (sous-partie 5.2.1), le modèle des attaques de blocage contre la plateforme est construit à partir de son modèle SC-net (figure 5.6). Deuxièmement (sous-partie 5.2.2), les attaques d'insertion et de suppression des événements échangés entre $R7$ et les autres ressources pour l'allocation des ressources (voir partie 5.1.3) sont implémentées au sein de la plateforme. Les attaques contre les événements de défaillance (voir partie 5.1.5) sont elles aussi programmées. Enfin, dans la sous-partie 5.2.3, la programmation de scénarios d'attaque de blocage combinant les attaques unitaire de la partie précédente est exposée de manière globale.

5.2.1 Modélisation des attaques

L'attaquant ciblant la plateforme transfert libre est un attaquant expert capable d'insérer ou de supprimer n'importe quelle décision d'allocation des ressources $R1, R3, R4, R5, R6$ aux opérations des recettes A et B. De surcroit, l'attaquant est capable de faire croire à la défaillance ou à la reprise d'une ressource de la plateforme au superviseur programmé dans le poste 7 via la manipulation des événements d'observation relatifs aux défaillances et reprises. A partir du modèle SC-net N_t et du modèle des indisponibilité $N_{\mathcal{R}_{ind}}$ de la plateforme "transfert-libre" (voir partie 5.1.2), le modèle des attaques de blocage dans un contexte incertain $N^{\alpha}_{\mathcal{R}_{ind}}$ de la plateforme est construit selon les méthodes présentées dans les parties 3.1 et 4.2.2. Les coûts associés aux attaques modélisés dans ce nouveau modèle sont résumés dans le tableau 5.1 ci-contre. Les contraintes de coût des attaques présentées dans le chapitre 3 (3.1) sont respectées dans ce modèle en prenant $K = 3$ (moyenne = 31,14, écart-type = 5.79, borne inférieure = 13.75, borne supérieure = 48.5).

5.2.2 Implémentation des attaques

Une attaque de blocage est une séquence malveillante d'insertions et de suppressions d'événements d'observation et de commande relatifs à des décisions d'allocation ou à la disponibilité des postes de la plateforme. Les deux prochaines sous-parties présentent l'implémentation des attaques d'insertion et de suppression de chaque événement partagé entre $R7$ et les autres postes ainsi que l'intégration des attaques d'indisponibilités.

Dans nos travaux, nous avons implémenté et expérimenté toutes les attaques au sein de l'automate de $R7$ en raison de restrictions d'accès au réseau industriel Ethernet de la plateforme. Toutefois, cette configuration permet de faciliter la synchronisation entre les scénarios d'attaques de blocage et le pilotage de la plateforme par l'automate de $R7$. Cette attaque est caractéristique d'une injection de code malveillante.

TABLEAU 5.1 – Coûts des attaques du modèle $N^{\alpha}_{\mathcal{R}_{ind}}$ de la plateforme expérimentale

Transition ou Ressource	Attaque 1	Coût total attaque 1	Attaque 2	Coût total attaque 2
T_1	T_-	51	T_+	54
T_2	T_-	112	T_+	116
T_3	T_-	120	T_+	121
T_4	T_-	119	T_+	121
T_5	T_-	110	T_+	116
T_6	T_-	124	T_+	127
T_7	T_-	116	T_+	115
T_8	T_-	68	T_+	68
T_9	T_-	70	T_+	72
T_{10}	T_-	125	T_+	128
T_{11}	T_-	116	T_+	122
T_{12}	T_-	113	T_+	115
T_{13}	T_-	50	T_+	49
$R1$	T_{ind+}	39	T_{dis+}	37
$R3$	T_{ind+}	46	T_{dis+}	45
$R4$	T_{ind+}	43	T_{dis+}	40
$R5$	T_{ind+}	37	T_{dis+}	35
$R6$	T_{ind+}	48	T_{dis+}	48

Attaques contre les décisions d'allocation

Au sein du fonctionnement FMS de la plateforme que nous avons programmé (figure 5.8), une attaque d'insertion (resp. de suppression) d'une décision d'allocation correspond à l'insertion (resp. la suppression) de tous les événements de commande (resp. d'observation) et la suppression de tous les événements d'observation (resp. de commande) relatifs à cette décision et à l'opération qu'elle débute. Par exemple, l'insertion malveillante de la décision de débuter la deuxième opération de la recette B, soit O_2^B, modélisée par t_{10} dans la figure 5.8 correspond à :

- L'insertion des événements de commande $e_{deb}(O_2^B), e_{ecr}(O_2^B), e_{lib}(R_6), e_{rec}(R_6)$ et des informations *"prochaine opération = R4"* et *"temps d'opération = 2000ms"* ;
- Et à la suppression des événements d'observation $e_{rec}(O_2^B), e_{fecr}(O_2^B), e_{dis}(R_6), e_{fin}(O_2^B)$ selon l'ordre d'occurrence de chacun de ces événements.

Dans la configuration de la plateforme choisie, les attaques d'insertion et de suppression des événements partagés sont réalisées lors de leur communication entre $R7$ et les autres postes. Nous avons donc décidé de modéliser ces attaques par une extension des places de communication C du modèle de fusion 5.8. Par conséquent, l'objectif de cette partie est de présenter cette extension des places C et de la programmer au sein de l'automate de $R7$ pour chaque événement partagé. Pour rappel, dans le programme de $R7$, les places C de communications de chaque événement partagé ont été modélisés par des variables internes à l'automate de $R7$.

Ainsi, l'extension d'une place C est un sous-réseau de Petri composé de 10 places et 6 transitions. Ce sous-réseau est illustré dans la figure 5.22. Soit un événement partagé e émis

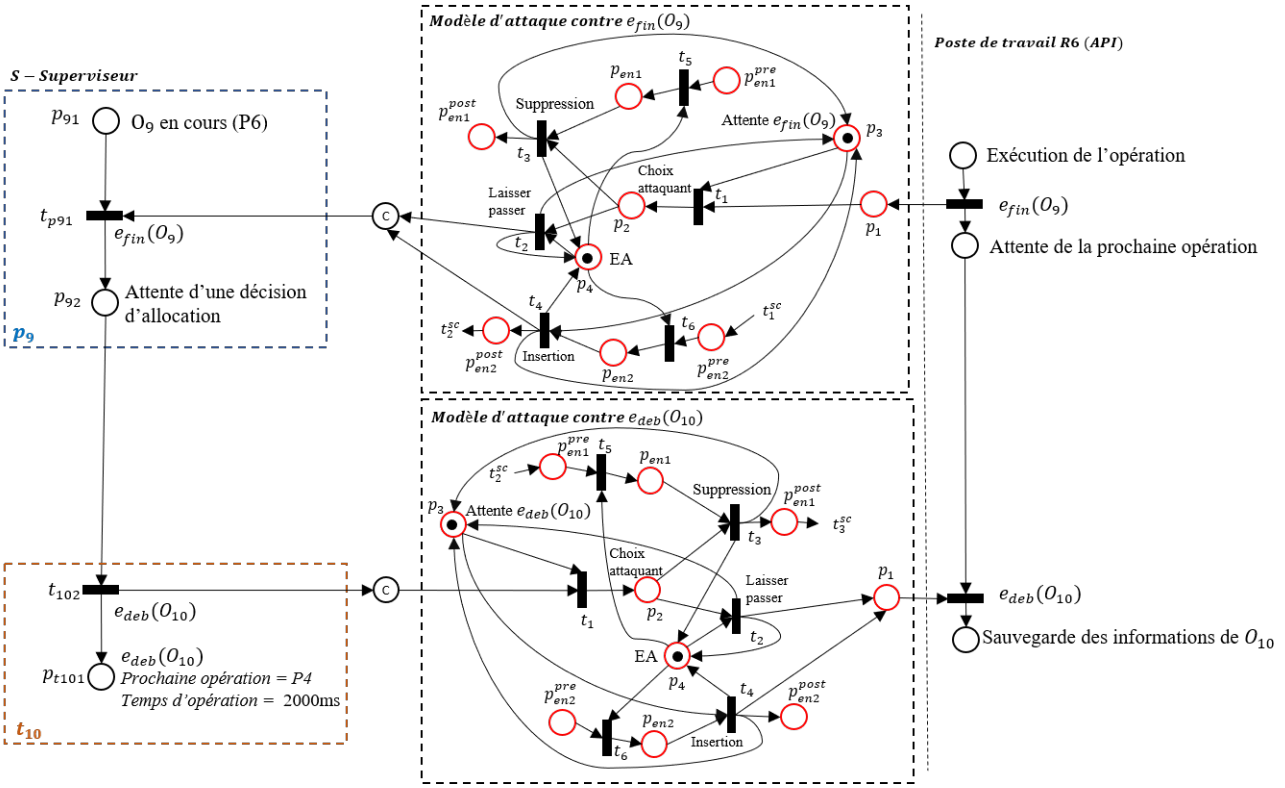


FIGURE 5.22 – Exemple du sous-RdP des attaques d'insertion et de suppression

depuis $R7$ (sous-réseau en bas de la figure) ou depuis le poste de travail $R6$ (sous-réseau en haut de la figure) ; le sous-réseau modélisant les attaques d'insertion et de suppression contre cet événement est construit selon les étapes suivantes.

1. Le sous-réseau attend l'occurrence de e à savoir lorsque la place p_1 pour une observation ou la place C pour une commande est marquée. Dans ces sous-réseaux, la place p_1 correspond à la variable réellement échangée entre S et les ressources et la place C à la variable interne à l'automate 7 ;
2. La transition t_1 est franchie, l'événement a eu lieu et est communiqué, et un choix est laissé à l'attaquant au sein de la place p_2. L'attaquant peut soit laisser passer l'événement, la transition t_2 est alors franchie et l'événement est correctement communiqué à son destinataire, soit supprimer e. Ce second choix se traduit par le franchissement de la transition t_3.
3. Dans le cas d'une insertion de e, l'occurrence de e (marquage de p_1 ou de la place C) n'est pas requis. Le franchissement de la transition t_4 permet alors d'insérer e et d'ajouter un jeton à la place C ou p_1 correspondante.

Pour ces trois étapes, la place p_3 labellisée *attente e* permet de ne considérer dans le RdP qu'une unique occurrence de e et la génération d'un unique jeton. Elle restreint ainsi le franchissement de t_1 (étape 2 - occurrence naturelle de e) et celui de t_4 (étape 3 - insertion de e).

Néanmoins, les attaques d'insertion et de suppression doivent pouvoir être correctement pilotées par l'attaquant dans les sous-RdPs présentés précédemment. Dans cette optique, quatre types de places sont ajoutées à l'extension de la place C (figure 5.22) :

- Une place EA est ajoutée et contraint le franchissement des transitions t_5, t_2 et t_4 à un unique franchissement afin de n'avoir simultanément qu'une attaque d'insertion, une attaque de suppression ou une décision de l'attaquant de laisser passer e.
- Les places p_{en1} et p_{en2} permettent de préparer une attaque et la rendre prioritaire sur toute autre action. Par exemple, le marquage de p_{en1} signifie que lors de la prochaine occurrence de e, l'événement sera supprimé au détriment de son insertion ou du laisser passer.
- Les places p_{en1}^{pre} et p_{en2}^{pre} précédent les places p_{en1} et p_{en2} sont quant à elles reliées au modèle d'un scénario d'attaque (sous-partie 5.2.3) et sont remplies par ce dernier afin de lancer l'attaque correspondante. Le franchissement des transitions $t_5 = p_{en1}^{pre\bullet}$ ou $t_6 \in p_{en2}^{pre\bullet}$ permet alors d'exécuter l'attaque.
- Enfin, les places p_{en1}^{post} et p_{en2}^{post} ont pour leur part la mission de valider au scénario la bonne

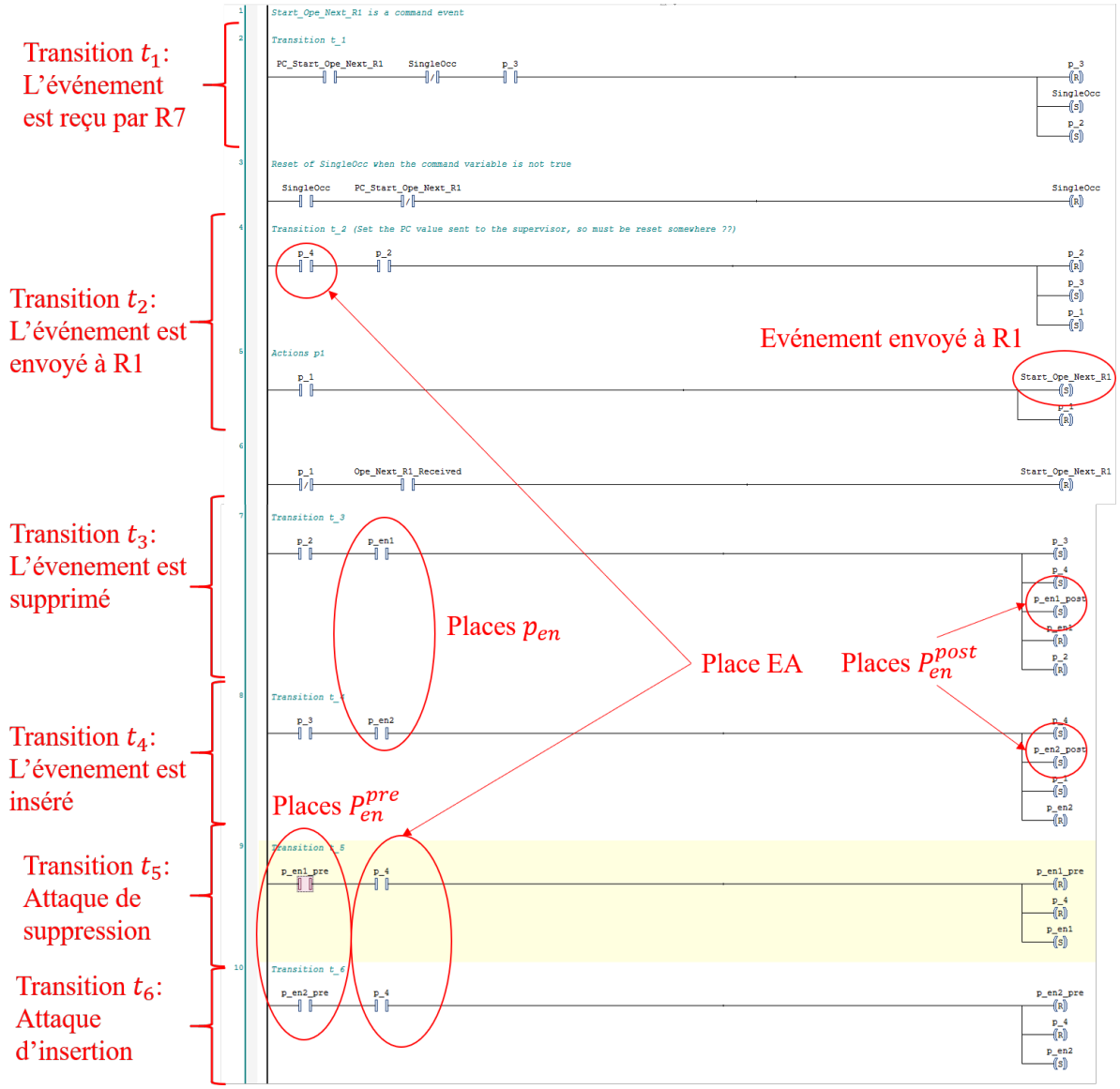


FIGURE 5.23 – Programmation sous SoMachine de l'attaque d'insertion de la commande de début d'opération par $R1$

réalisation de l'attaque souhaitée.

Au sein de l'automate du superviseur $R7$, nous avons programmé le sous-RdP de la figure 5.22 pour chaque variable interne à l'automate représentant un événement partagé de la plateforme. Le type d'événement, entre observation et commande, a été pris en compte dans le RdP que nous implémentons pour chaque événement. Ces sous-RdP ont été programmés en langage Ladder de la même manière que les RdPs de la première partie [373]. Une illustration de ce programme pour l'attaque d'insertion de l'événement de début d'opération pour la ressource $R1$ est proposée dans la figure 5.23. Dans cette figure, le temps de l'opération n'est pas précisé et le sera par le scénario d'attaque. Ainsi, le programme de la figure 5.23, permet de débuter de manière malveillante n'importe quelle opération réalisée par le poste $R1$. Au total, nous avons implémenté 48 programmes Ladder identiques à celui de la figure 5.23 dans l'automate du superviseur $R7$.

Attaques contre la disponibilité des ressources

Une attaque contre la disponibilité d'un poste de travail, par exemple la ressource $R5$, est réalisée par la modification de la variable E_R5_unav. Ainsi, afin de simuler la défaillance de la ressource $R6$, l'attaquant force la variable E_R5_unav à 0. Selon l'état de la ressource $R5$, en opération ou hors opération, les variables $R5\_unav\_a$ ou $R5\_unav\_r$ sont mises à 1 (sous-partie 5.1.5). Les attaques contre la disponibilité sont directement programmées au sein des actions des scénarios d'attaque tel qu'illustré dans la figure 5.24.

FIGURE 5.24 – Programmation sous SoMachine de l'attaque d'indisponibilité de la ressource $R5$

L'intégration des attaques d'insertion et de suppression des événements partagés ainsi que les attaques d'indisponibilité dans les scénario d'attaque sont présentées dans la prochaine partie.

5.2.3 Les scénarios d'attaque de blocage

Un scénario d'attaque de blocage est une séquence d'attaques d'insertion et de suppression de décisions d'allocation et d'attaques d'indisponibilité conduisant à un état de la DZ de la platerforme. Tous les scénarios d'attaque que nous testons sont programmés directement dans l'automate de $R7$. Leur programmation est une transcription d'un RdP séquentiel modélisant les étapes de l'attaque en un modèle Grafcet propre au logiciel SoMachine. La capture d'écran de la figure 5.25 illustre ces deux modèles RdP et Grafcet appliqués à la suppression de la décision d'allocation de la transition t_3 (début d'une opération de la recette A). Une présentation

générale des scénarios d'attaque est proposée dans cette sous-partie.

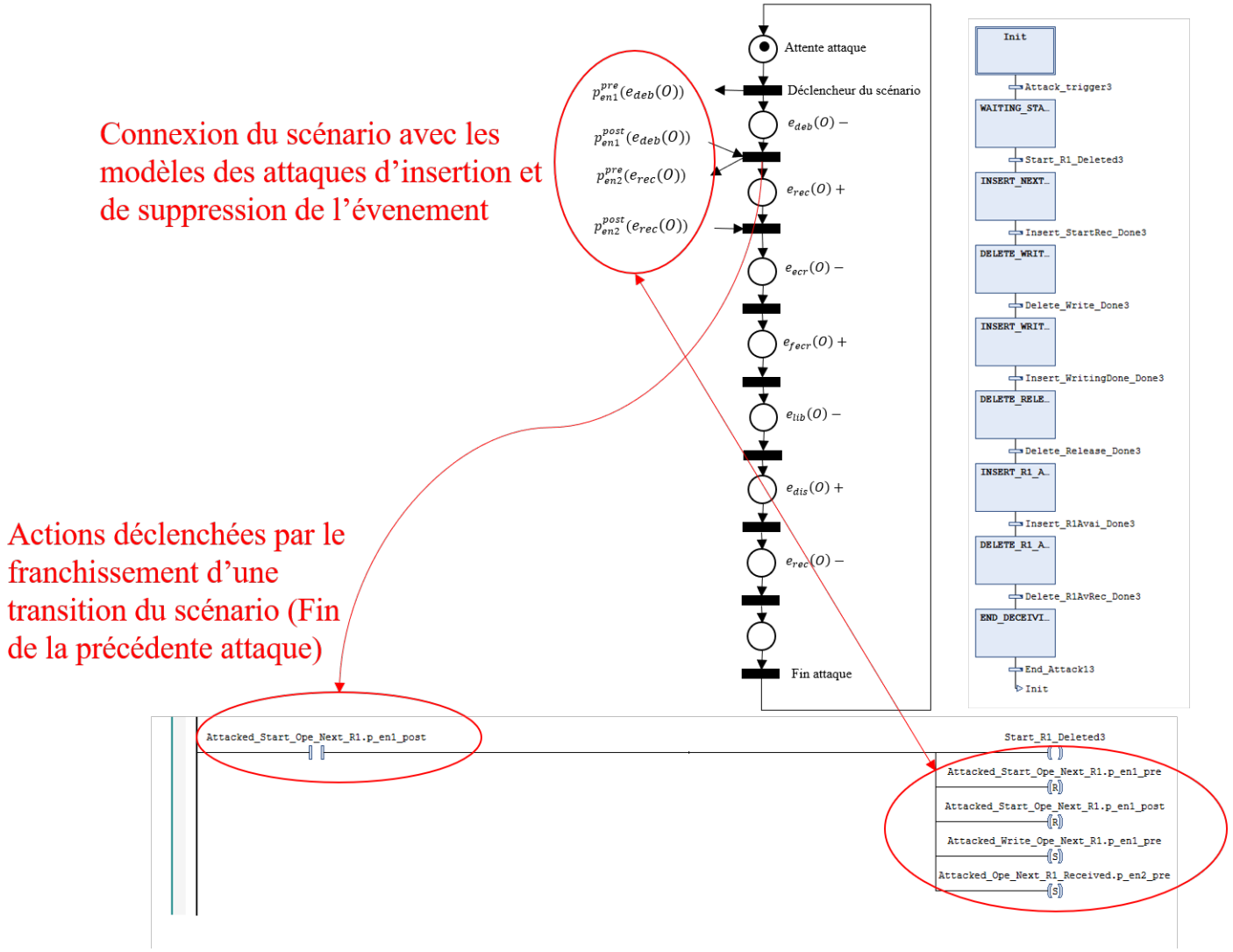


FIGURE 5.25 – Programmation sous SoMachine du scénario d'attaque de suppression d'une décision d'allocation

Ainsi, un scénario d'attaque possède un état d'attente (premier état du Grafcet) marqué initialement et connectée vers une transition dont le franchissement signifie le début du scénario. Cette transition est franchie lorsqu'un événement spécifique a lieu ou lorsque le FMS atteint un état d'allocation précis. Cette première transition est succédée par un enchainement séquentiel de places et de transitions représentant les étapes du scénario. Chaque étape d'un scénario correspond au lancement d'une attaque unitaire (insertion ou suppression d'un événement partagé, attaque d'indisponibilité) et le scénario passe à l'étape suivante, i.e. franchie une transition, lorsque l'attaque unitaire précédente a correctement été réalisée.

La réalisation d'une attaque unitaire d'insertion ou de suppression d'un événement partagé est pilotée grâce aux arcs entre les transitions du modèle RdP du scénario avec les places p_{en}^{pre} et p_{en}^{post} du modèle RdP de cette attaque unitaire (sous-partie 5.2.2 et figure 5.22). Ainsi, pour

une attaque unitaire débutée par la place p d'un scénario d'attaque, un arc $({}^\bullet p, p_{en}^{pre})$ permet de débuter l'attaque dans le modèle de l'attaque unitaire tandis qu'un arc $(p_{en}^{post}, p^\bullet)$ informe de la bonne réalisation de l'attaque et permet de débuter la prochaine. Ces arcs sont illustrés dans le modèle RdP en haut à gauche de la figure 5.25 tandis que le lancement d'une attaque unitaire et l'information de la bonne réalisation d'une attaque unitaire sont illustrés dans le Ladder en bas de la figure 5.25. Si la place p est associé à une attaque d'indisponibilité, par exemple de la ressource $R5$, la variable E_unav_R5 ciblée est modifiée en conséquence dans un programme Ladder (voir capture d'écran 5.24).

Remarque 5.2.1. Dans le programme de l'automate de $R7$, nous avons décidé de programmer séparément les attaques de suppression d'une décision d'allocation et les scénarios d'attaque contenant toutes les attaques d'insertion de décisions et d'indisponibilité. Ainsi, au cours d'un scénario, son programme Grafcet vérifie la bonne réalisation des attaques de suppression par des transitions dédiées.

⌟

A partir de l'implémentation du module de supervision pour le pilotage des recettes A et B de la plateforme (partie 5.1) et de la programmation des scénarios d'attaque de blocage (partie 5.2), nous définissons dans la prochaine partie un environnement de test au sein duquel l'expérimentation de trois scénarios d'attaque et l'analyse de leur diagnostic par notre méthode seront menées.

5.3 Expérimentation, résultats et discussion

Dans cette dernière partie du chapitre 5, notre méthode de diagnostic est appliquée sur la plateforme transfert-libre et les résultats expérimentaux obtenus sont analysés. Dans la première sous-partie 5.3.1, l'environnement de test des scénarios d'attaque et de notre méthode de diagnostic est présenté et introduit le cadre expérimental des résultats analysés dans cette partie. Puis, dans la deuxième sous-partie 5.3.2, trois scénarios d'attaques sont choisis, expérimentés et les résultats de diagnostic de ces scénarios sont présentés. Enfin, dans la dernière sous-partie 5.3.3, les différents résultats de diagnostic obtenus pour les trois scénarios sont discutées, des limites de nos travaux expérimentaux sont identifiées et des perspectives à ceux-ci sont proposées.

5.3.1 Environnement de test

Pour mener à bien l'expérimentation de scénarios d'attaque et leur diagnostic, nous avons mis en œuvre l'environnement de test suivant :

- Les scénarios d'attaque de blocage sont calculés hors-ligne par l'algorithme de calcul des profils d'attaquant. Ils sont ensuite programmés dans l'automate du superviseur $R7$ (sous-partie 5.2.3) et expérimentés ;
- Le module de diagnostic des attaques de blocage dans un contexte incertain est implémenté hors-ligne selon les étapes suivantes :
  1. Les événements bas-niveau sont communiqués depuis les automates des différents postes à l'automate du poste $R7$. Ces événements sont le début et la fin de chaque opération par les postes de travail $R1, R3, R4, R5, R6$ et le début d'une recette par le poste $R2$;

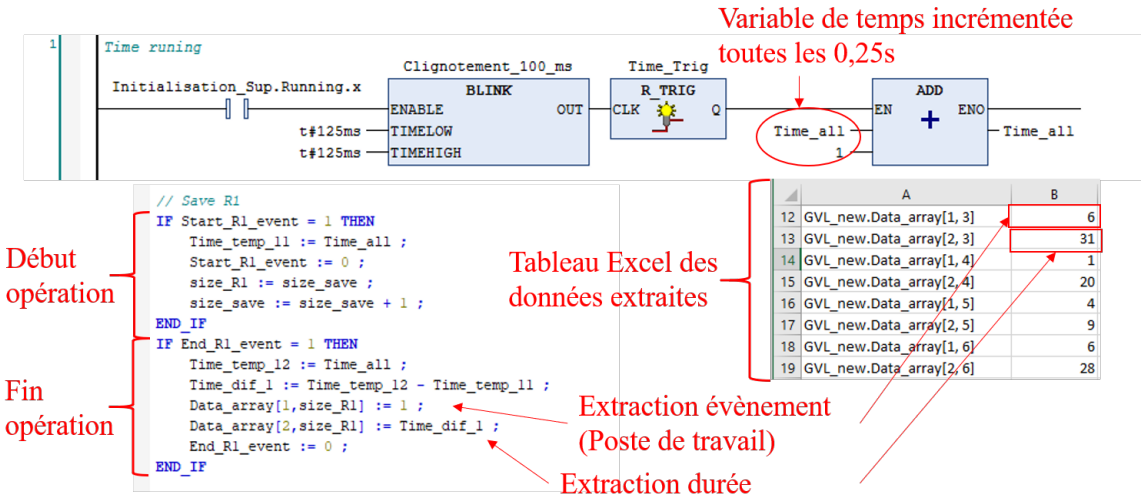


FIGURE 5.26 – Programmation sous SoMachine de l'extraction des données événementielles et temporelles des opérations

2. Le temps s'écoulant entre les événements de début et de fin d'opération d'un poste de travail est calculé au sein de l'automate de $R7$. Ainsi, puisque les durées de chaque opération de la plateforme (figure 5.6) sont différentes, toutes les opérations peuvent être identifiées les unes des autres.
3. Chaque événement représentant l'occurrence d'une opération et la durée associée à cette opération sont sauvegardés dans l'automate 7 puis extraits sous la forme d'une séquence vers un fichier Excel. La programmation de cette extraction est illustrée dans la figure 5.26.
4. Le diagnostiqueur des attaques de blocage dans un contexte incertain est construit hors-ligne sous Matlab.
5. La séquence événementielle extraite depuis $R7$ est analysée et comparée avec les états du diagnostiqueur. Une détection a lieu si un état d'attaque est atteint.

Cet environnement de test présenté ci-dessous est suivi lors de l'expérimentation de trois scénarios d'attaque. Cette expérimentation est présentée dans la prochaine partie.

Remarque 5.3.1. L'extraction des signaux bas-niveaux (signaux échangés entre l'automate d'un poste de travail et les capteurs/actionneurs du poste, voir partie 5.1.1) pour le diagnostic des attaques de blocage n'a pas été expérimentée car nous ne possédons pas les technologies nécessaires. Une perspective à nos travaux expérimentaux pourrait portée sur le référencement et l'installation de ces technologies sur le plateforme "transfert-libre". ⌟

5.3.2 Choix des scénarios d'attaque, expérimentation et diagnostic

L'expérimentation des trois scénarios d'attaque de blocage débute à un état initial de la plateforme illustré par la figure 5.6 et dans le mode sans indisponibilité ($Mode_0$). Aucun produit n'est en cours de fabrication et la plateforme a pour objectif la réalisation de 6 produits de la recette A et 4 de la recette B. Depuis le $Mode_0$, les modes atteignables sont $\mathcal{RM} = \{R3, R4, R5\}$, l'indisponibilité des ressources $R1$ et $R6$ bloquant l'entièreté de la plateforme. Pour rappel,

l'ordonnancement partiel $\sigma_f = t_1t_9t_{10}t_3t_5t_9t_{11}t_{10}t_7t_{12}t_{13}t_{11}t_1t_8t_3t_5t_9t_{10}t_7t_{12}$ a été calculé depuis cet état initial (sous-partie 5.1.2). La programmation sous SoMachine des Grafcets de ces trois scénarios d'attaque est illustrée dans l'annexe 6.16 de ce manuscrit.

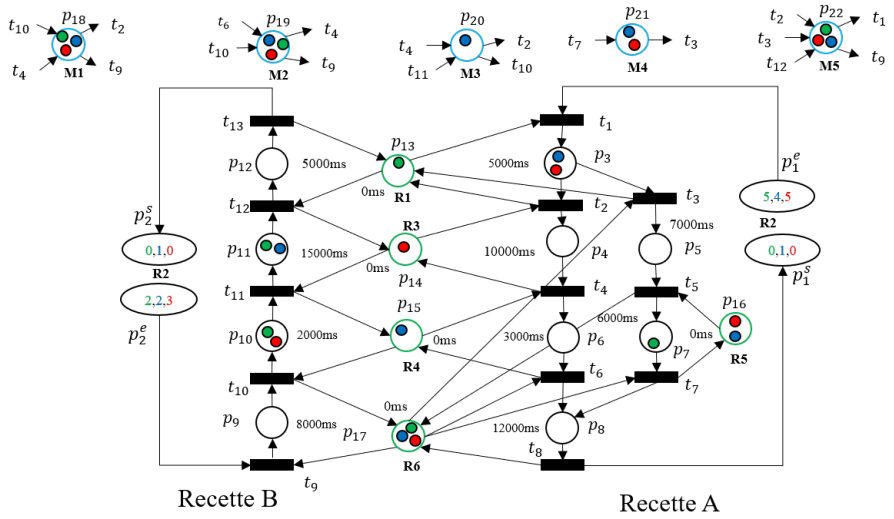


FIGURE 5.27 – Marquages de début des scénario 1 (vert), 2 (bleu) et 3 (rouge)

Dans les trois prochaines sous-parties, les trois scénarios d'attaque expérimentés dans nos travaux sont présentés. Afin de suivre les étapes de ces trois scénarios, les trois états à partir desquels ils sont déclenchés sont illustrés dans la figure 5.27. Le marquage du profil 1 est en vert, celui du profil 2 en bleu et enfin celui du profil 3 en rouge. De plus, pour chaque scénario, la séquence extraite depuis l'automate de $R7$ des opérations et de leurs durées ayant lieu sur la plateforme depuis l'état initial (tableaux 5.2, 5.3 et 5.4), le diagnostiqueur partiel construit à partir de l'état de déclenchement du scénario (figures 5.28, 5.30 et 5.32) ainsi que l'état de la plateforme observé par le module de supervision (captures d'écran 5.29, 5.31 et 5.33) sont donnés. Notons que dans les tableaux 5.2, 5.3 et 5.4, la correspondance entre ressource requise par l'opération, durée de l'opération et transition de décision d'allocation correspondante a été réalisée manuellement. Enfin, les transitions en rouge dans les tableaux correspondent aux observations des événements du scénario d'attaque par le module.

Scénario d'attaque n°1

Poste de travail	2	2	6	1	4	6	2	6	5	3	2	2	4	6	1
Durée (seconds)	0	0	7.75	5	2.25	6.75	0	7.75	6	14	0	0	2.25	7.75	5
Transition	/	/	t_9	t_1	t_{10}	t_3	/	t_9	t_5	t_{11}	/	/	t_{10}	t_9	t_1

TABLEAU 5.2 – Observations du module de diagnostic lors du scénario d'attaque 1

Le **premier scénario** est égal au profil d'attaquant 1 calculé dans un contexte certain depuis le $9^{ème}$ état traversé par l'ordonnancement soit $M_9 = 5p_1^e + 2p_2^e + p_7 + p_{10} + p_{11} + p_{13} + p_{17} + p_{18} + p_{19} + p_{22}$ (figure 5.27). Dans cet état, les postes $R3, R4$ et $R5$ sont occupés par les opérations des places p_7 (recette A), p_{10} et p_{11} (recette B). En sortie de l'algorithme de calcul des profils d'attaquant, ce scénario 1 est égal à la séquence de transitions $t_{7-}t_{1+}t_{9+}$ dans le modèle

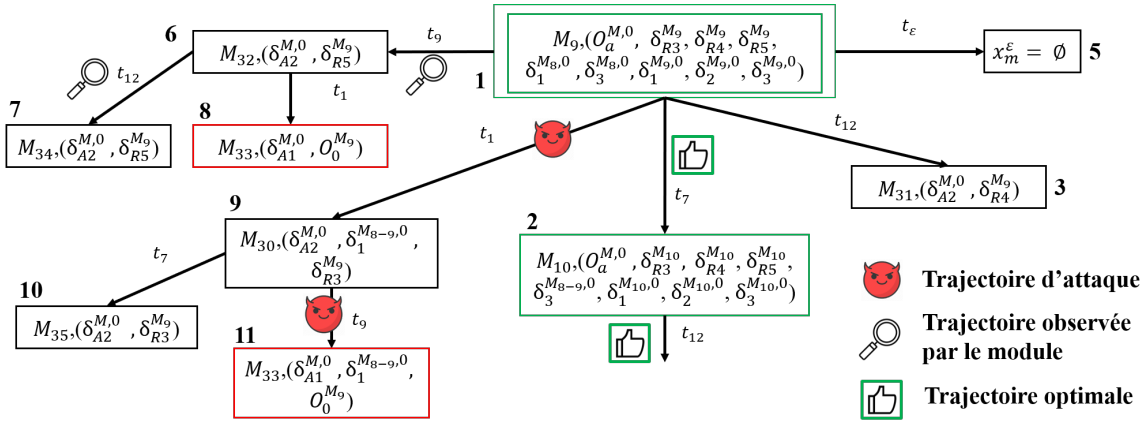


FIGURE 5.28 – Diagnostiqueur $Diag^1_{cu}$ restreint localement à l'état M_9

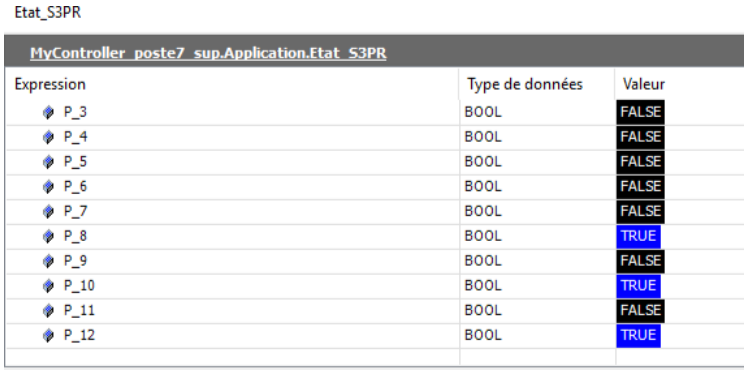


FIGURE 5.29 – État de la plateforme observé par le module de supervision après l'attaque 1

d'attaque $N^{\alpha}_{\mathcal{R}_{ind}}$. En d'autres termes, l'objectif de ce scénario est de débuter un produit de la recette A (transition t_{1+}) afin d'occuper le poste $R1$ et empêcher le produit en p_{11} (ressource $R3$) d'être opéré par $R1$, puis de débuter un produit de la recette B (transition t_{9+}) afin de détenir la ressource $R6$ et d'empêcher le nouveau produit de la recette A d'être opéré soit par $R3$ (détenue par la place p_4 et bloquée par la première attaque) soit par $R6$ (détenue par la place p_9). La condition d'attente circulaire entre les ressources atteinte par le scénario 1 est donc la suivante : $R6$ requiert $R4$, $R4$ requiert $R3$, $R3$ requiert $R1$, et $R1$ requiert $R6$ ou $R3$ qui sont déjà détenues. En parallèle, le scénario d'attaque 1 supprime la décision d'allocation t_7 (t_{7-}) afin de rester sournois.

Scénario d'attaque n°2

2	2	6	1	4	6	2	6	5	3	6	4	1	2	3	1	ε
0	0	7.75	5	2.25	6.75	0	7.75	6	14.25	11.25	2.25	5	0	14.25	5	/
/	/	t_9	t_1	t_{10}	t_3	/	t_9	t_5	t_{11}	t_7, t_8	t_{10}	t_{12}, t_{13}	/	t_{11}	t_1	t_ε

TABLEAU 5.3 – Observations du module de diagnostic lors du scénario d'attaque 2

Le **deuxième scénario** est une application du profil d'attaquant 2 dans un contexte incertain. Il débute à partir de l'état $M_{15} = 4p^e_1 + p^S_1 + 2p^e_2 + p^s_2 + p_3 + p_{11} + p_{15} + p_{16} + p_{17} + p_{18} + p_{19} + p_{20} + p_{21} + p_{22}$ et est égal à $t_{3-}t^{ind}_{p_{16}+}$ (voir figure 5.27). Dans cet état, les postes

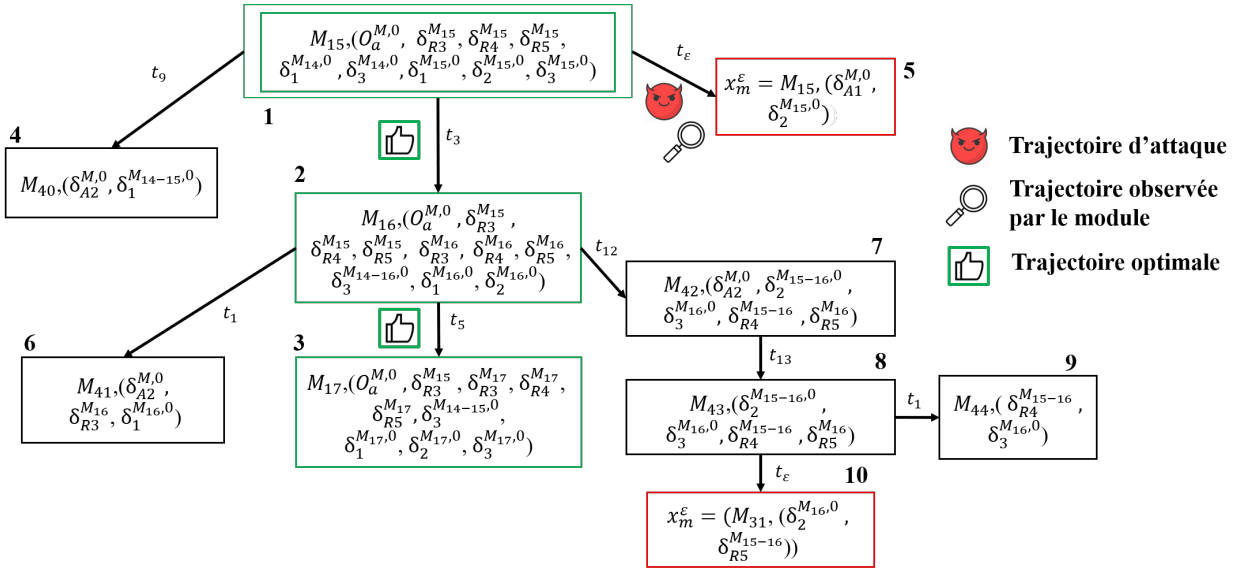


FIGURE 5.30 – Diagnostiqueur $Diag^1_{cu}$ restreint localement à l'état M_{15}

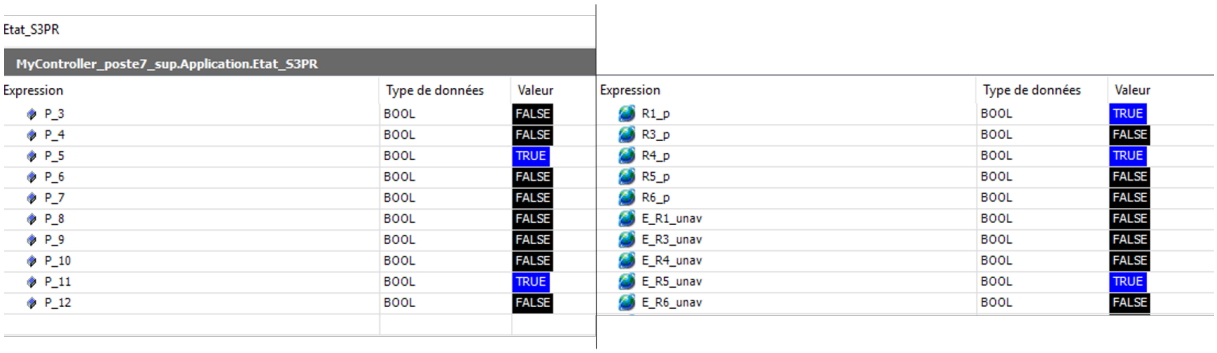


FIGURE 5.31 – État de la plateforme observé par le module de supervision après l'attaque 2

$R1, R3$ et $R6$ sont occupés par les opérations des places p_3 (recette A), p_{11} et p_9 (recette B). Cette attaque consiste à bloquer les produits dans les places p_{11} (ressource $R3$) et p_3 (ressource $R1$) en rendant indisponible la ressource $R5$ (t^{ind}_{p16+}) condamnant ainsi la branche de la recette A requérant $R5$. Nous obtenons ainsi que le produit détenant $R1$ requiert $R3$ et inversement. Par ailleurs, le produit en p_3 ne peut pas être déplacé vers la place p_5 (recette A) puisqu'il détiendrait et bloquerait par cascade d'indisponibilité la ressource $R6$ requise par les deux recettes du FMS. En effet, ce produit en p_5 requerrait alors la ressource $R5$ devenue indisponible suite à à l'attaque. En parallèle, le scénario d'attaque 2 supprime la décision d'allocation t_3 (t_{3-}) afin de rester sournois.

Scénario d'attaque n°3

Poste de travail	2	2	6	1	3	4
Durée (seconds)	0	0	7.5	5	9.5	2.25
Transition	/	/	t9	t1	t2	t10

TABLEAU 5.4 – Observations du module de diagnostic lors du scénario d'attaque 3

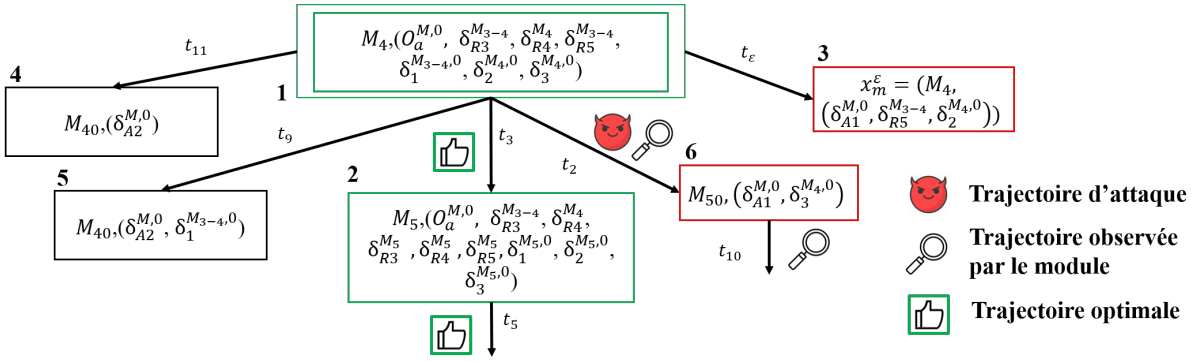


FIGURE 5.32 – Diagnostiqueur $Diag_{cu}^1$ restreint localement à l'état M_4

Etat_S3PR

MyController_poste7_sup.Application.Etat_S3PR

Expression	Type de données	Valeur
P_3	BOOL	FALSE
P_4	BOOL	FALSE
P_5	BOOL	TRUE
P_6	BOOL	FALSE
P_7	BOOL	FALSE
P_8	BOOL	FALSE
P_9	BOOL	FALSE
P_10	BOOL	TRUE
P_11	BOOL	FALSE
P_12	BOOL	FALSE

FIGURE 5.33 – État de la plateforme observé par le module de supervision après l'attaque 3

Le **troisième scénario** correspond au profil d'attaquant 3 généré dans un contexte incertain depuis l'état $M_4 = 5p_1^e + 3p_2^e + p_3 + p_10 + p_14 + p_{16} + p_{17} + p_{18} + p_{19}p_{21} + p_{22}$. Dans cet état, les postes $R6$ et $R3$ sont occupés par les opérations des places p_7 (recette A), et p_{10} (recette B). Cette attaque est égale à $t_{3\_}t_{2+}$. Elle cherche à bloquer la recette B et le circuit de la recette A requérant les ressources $R3$ et $R4$ (passant par p_4p_6) en bloquant mutuellement le produit de la recette A détenant la ressource $R3$ (place p_4) avec celui de la recette B détenant la ressource $R4$ (place p_{10}). En parallèle, le scénario d'attaque 2 supprime la décision d'allocation t_3 ($t_{3\_}$) afin de rester sournois.

Remarque 5.3.2.
Dans les tableaux 5.2, 5.3 et 5.4, trois observations singulières peuvent être faites.

- Lorsque la ressource $R2$ est identifiée au sein de la première ligne, un produit débute depuis les recettes A ou B mais aucune transition du modèle de la plateforme (figure 5.27) ne peut lui être associée (ligne 3) car la première opération de la recette débutée n'est pas encore connue ;
- Certaines transitions précédant le déclenchement des scénarios d'attaque ne respectent pas l'ordonnancement. Par exemple, les transitions t_1 et t_9 débutant l'ordonnancement sont inversées dans le tableau 5.2. Cette différence s'explique par le temps de trajet des palettes entre les postes faussant le temps réel d'une opération. En effet, une opération réelle débute lors de la libération de la palette par le poste précédent et non lors de l'arrivée de la palette au poste suivant ;

- Dans le tableau 5.3, les transitions t_7 et t_8 puis t_{12} et t_{13} sont associées à la même opération de fin de recette puisque les transitions d'évacuation des palettes (t_8 et t_{13}) sont systématiquement franchies lorsque les dernières opérations des recettes se terminent.

⌟

Dans la prochaine sous-partie, les résultats obtenus pour chaque scénario d'attaque à travers les tableaux des séquences de décisions observées par le module (tableaux 5.2, 5.3, 5.4), les diagnostiqueurs locaux (figures 5.28, 5.30, 5.32) et les états observés par le superviseur $R7$ (captures d'écran 5.29, 5.31, 5.33) sont analysés puis discutés au regard des limites et perspectives de nos travaux expérimentaux.

5.3.3 Analyse des résultats de diagnostic et discussion

Dans cette dernière partie du chapitre 5, les résultats des trois scénarios d'attaque présentés précédemment sont analysés les uns après les autres. A partir de ces analyses, plusieurs limites à nos travaux son identifiées et différentes perspectives sont proposées pour y répondre.

Scénario 1

Le **scénario d'attaque 1** a pour objectif la suppression de t_7 et l'insertion de t_1 et t_9. La suppression a bien lieu puisque l'état de la plateforme selon le module de supervision (figure 5.29) considère que t_7 a été franchie et que la place p_8 est marquée. Toutefois, cette dernière a correctement été supprimée par l'attaquant car elle n'est pas observée par le module de diagnostic (tableau 5.2). Au sein des observations réalisées par le module, les transitions t_1 et t_9 ont bien lieu et ont donc été correctement insérées par l'attaquant. Cependant, elles sont observées dans le mauvaise ordre, t_9 avant t_1. Au regard de ces deux transitions, le module de diagnostic réalise les interprétations suivantes à partir du diagnostiqueur $Diag^1_{cu}$ de la figure 5.28.

Premièrement, l'inversion entre les deux transitions provoque une mauvaise interprétation de la part du diagnostiqueur (figure 5.28) : ses états 6 et 8 traversés par la séquence t_9t_1 ne sont pas labellisés par le profil d'attaquant 1. A l'inverse, les états 9 et 11 le sont. Cette perte d'information n'est cependant pas préjudiciable car l'attaque est détectée lorsque l'état de blocage M_{33} est atteint dans les états 8 et 11 du diagnostiqueur.

Deuxièmement, le franchissement de la transition t_9 observé par le module de diagnotic ne peut pas être directement interprété comme une attaque. En effet, cette transition correspond aussi à l'ordonnancement généré par le module de supervision en cas d'indisponibilité de $R5$. L'attaque ne peut donc pas être détectée avant que l'état de blocage ciblée ne soit atteint après le franchissement de t_1. Cette confusion entre indisponibilité et attaque est aussi présente dans l'état 9 si la transition t_1 est observée avant t_9. Dans cette état, le profil 1 peut se confondre avec l'indisponibilité de la ressource $R3$.

Scénario 2

Le **scénario d'attaque 2** a pour objectif la suppression de t_3 et l'insertion de l'événement d'observation de défaillance de la ressource $R5$. La suppression est réalisée correctement car la place p_5 est marquée dans la figure 5.31 mais la transition t_3 n'est pas observée dans le tableau

5.3. Pour sa part, l'attaque d'indisponibilité atteint aussi son objectif puisque la place $R5\_p$ à droite de la figure 5.31 n'est pas marquée (la ressource n'est pas disponible) tandis que la place de l'événement de défaillance E_R5_unav l'est. Dans le diagnostiqueur de la figure 5.30, l'attaque est détectée par l'absence d'événements lorsque l'état M_{15} est atteint par le FMS (transition t_ε et état 5).

Dans le diagnostiqueur 5.30, un second scénario proche de l'attaque 2 est illustré par les états 2, 7, 8 et 10. Ce scénario consiste à rendre $R5$ indisponible sans supprimer t_3. La plateforme devient alors bloquée par cascade d'indisponibilité, le produit en p_5 bloquant la ressource $R6$ requise par les deux recettes A et B. Conséquemment à t_3, le module de supervision cherche à vider la place p_{11} de son produit en franchissant t_{12} et t_{13}. Dans ce second scénario, l'attaque est confondue avec l'indisponibilité naturelle de la ressource $R5$ à travers le label $\delta_{R5}^{M_{16}}$ ainsi qu'avec le profil $\delta_3^{M16,0}$ et l'indisponibilité de $R4$ jusqu'à l'état 8. Cependant, l'indisponibilité de $R5$ ne peut pas résulter d'une opération de maintenance car elle est à l'origine d'une conséquence de cascade d'indisponibilité. Par conséquent, cette variante du scénario 2 est détectée lorsque l'état 10 du diagnostiqueur 5.30 est atteint.

Scénario 3

Le **scénario d'attaque 3** a pour objectif la suppression de t_3 et l'insertion de la transition t_2 afin de créer un blocage partiel entre les produits en p_4 et p_{10}. L'attaque de suppression est effective lors de l'expérimentation puisque la place p_5 est marquée dans la figure 5.33 tandis que la transition t_3 n'est pas observée par le module de diagnostic (tableau 5.4). L'insertion de la transition t_2 est pour sa part observée par le module de diagnostic (tableau 5.4). Dans le diagnostiqueur de la figure 5.32, l'attaque est détectée lors de l'observation du franchissement de cette transition t_2.

Dans ce diagnostiqueur 5.32, le scénario d'attaque 3 ne peut pas être confondu avec un changement de mode ou un autre profil d'attaquant. L'état atteint est pour sa part un état de la DZ du $Mode_0$ à travers le label $\delta_{A1}^{M,0}$. La détection est donc immédiate dans cette troisième expérimentation.

Limites et perspectives expérimentales

A partir de l'analyse des trois scénarios d'attaque, trois limites et perspectives de notre méthode de diagnostic et de son expérimentation ont été identifiées.

Premièrement, l'expérimentation doit être améliorée pour représenter plus fidèlement le fonctionnement FMS de la plateforme et l'occurrence d'attaques. Ainsi, les temps de trajets doivent être inclus aux temps d'opération du modèle 5.6 dans le but d'obtenir des séquences d'observation (tableaux 5.2, 5.3, 5.4) fidèles à l'ordonnancement et aux scénarios d'attaque. Une seconde perspective expérimentale est le lancement des attaques directement depuis le réseau industriel et non par simulation dans l'automate de supervision. Par exemple, un module Arduino pourrait être connecté au réseau de la plateforme afin de réaliser ces attaques. Enfin, le module de diagnostic doit à terme pouvoir extraire les événements de début et de fin d'opération depuis les signaux bas-niveau et le réseau de terrain transportant ces signaux.

Deuxièmement, l'expérimentation a permis de mettre en lumière les limites de différenciation entre attaques et indisponibilités présentées dans le chapitre 4 (4.3). Ainsi, tous les scénarios

expérimentaux sont diagnosticables mais les scénarios 1 et 2 ne sont pas différentiables de changements de mode tant que l'état critique ciblé n'est pas atteint. Face à la non diagnosabilité entre attaques et changements de mode, les perspectives de travail proposées dans le chapitre précédent, soient la construction d'une loi de planification des opérations de maintenance plus restrictive et l'ajout d'une composante stochastique au diagnostiqueur, peuvent être appliquées.

Troisièmement, aucun des trois scénarios d'attaque proposés dans ce chapitre n'illustre la non diagnosabilité de certaines attaques ciblant des blocages partiels avec des changements de mode. Pour répondre à cette dernière limite, un nouveau fonctionnement FMS de la plateforme possédant plus de recettes différentes est nécessaire. La programmation d'un tel fonctionnement est l'une de nos perspectives de travail. De surcroit, une condition suffisante à l'existence de telles attaques dans un modèle S^3PR donné pourra être développée.

Conclusion

Dans ce dernier chapitre, notre méthode de diagnostic des attaques de blocage dans un contexte incertain a été évaluée sur une application expérimentale. Dans une première partie (5.1), la plateforme expérimentale choisie a été présentée et le pilotage FMS de cette plateforme que nous avons implémenté a été détaillé. Dans une deuxième partie (5.2), la programmation des attaques de blocage contre la plateforme a été exposée. Enfin, dans la dernière partie (5.3), nous avons expérimenté trois scénarios d'attaque sur la plateforme. Ces scénarios d'attaque ont été observés en ligne puis comparés hors-ligne avec notre diagnostiqueur des attaques de blocage. L'analyse conjointe des différents scénarios et du diagnostiqueur nous a alors permis d'identifier des limites à nos travaux expérimentaux et de proposer plusieurs perspectives pour y répondre.

Conclusion générale

Les travaux que nous avons présenté dans ce manuscrit répondent à la problématique du diagnostic des attaques de blocage ciblant les systèmes manufacturiers flexibles dans un contexte incertain.

Ainsi, en réponse à cette problématique, un module de diagnostic exploitant les données bas-niveau de l'architecture de pilotage des FMSs a été développé. Ce module, initialement construit pour le cas sans indisponibilité des ressources puis étendu au cas avec indisponibilité, a pour mission principale le diagnostic conjoint des attaques de blocage, des profils d'attaquant, des changements de mode d'indisponibilité et de l'origine, naturelle ou malveillante, d'un changement de mode. Pour chacun de ces diagnostics, nous avons sélectionné et développé un ensemble de modèles et de méthodes, contributions originales de nos travaux.

- Pour le diagnostic des attaques de blocage, un modèle RdP de ces attaques a été proposé à partir des travaux de [222]. Puis, les attaques contre les événements d'indisponibilité des ressources ont été intégrées à ce modèle dans le contexte incertain. Deux méthodes d'ordonnancement et de prévention des états de blocage de la littérature ont été choisies et adaptées au contexte incertain dans le but de permettre au module de diagnostic, d'une part, de connaître en ligne les états critiques et optimaux du FMS et de les diagnostiquer, et d'autre part, de synchroniser son état avec celui du module de supervision du FMS et d'éviter par conséquent les erreurs de détection. Outre les attaques de blocage partiel dans un contexte incertain, toutes les attaques de blocage sont diagnosticables par notre module ;
- Pour le diagnostic des profils d'attaquant, trois profils ont été sélectionnés pour leur représentativité de la diversité des attaquants et des différents objectifs d'une attaque de blocage. Parmi ces objectifs, l'état ciblé, la taille, le coût et la durée de l'attaque ont été considérés. Le dessein de ce deuxième diagnostic est d'enrichir le diagnostic d'une attaque de blocage par des profils pouvant être à l'origine de celle-ci. Un algorithme original de calcul des profils d'attaquant fondé sur la recherche $A^\star$ et garantissant la sournoiserie de l'attaque a ainsi été développé dans nos travaux ;
- Pour le diagnostic des changements de mode d'indisponibilités, tous les ordonnancements pouvant être appliqués par le module de supervision pour chaque mode et chaque état du FMS ont été calculés hors-ligne et intégrés au module de diagnostic ;
- Pour le diagnostic de l'origine d'un changement de mode, quatre indicateurs ont été identifiés et permettent de détecter avec certitude la malveillance de certains changements de mode.

Pour chacune de ces quatre catégories de diagnostic, nous avons développé un algorithme de construction du diagnostiqueur correspondant. En ligne, notre module de diagnostic compose ces quatre diagnostiqueurs en un diagnostiqueur commun dès l'occurrence d'un réordonnancement du FMS ou d'un changement de mode. Le module opère alors la comparaison entre l'état du diagnostiqueur et l'état réel du FMS afin de pouvoir diagnostiquer une attaque de blocage, une suspicion de profil d'attaquant ou un changement de mode et son origine.

En conclusion de ce manuscrit, nous avons réalisé sur une plateforme manufacturière expérimentale l'implémentation en ligne de scénarios d'attaques de blocage et la validation hors-ligne de notre module de diagnostic à parti de ces scénarios.

Au terme de nos travaux, plusieurs perspectives de recherche se dégagent en réponse aux limites mises en lumière et discutées dans ce manuscrit.

A court terme, cinq axes de travail peuvent être envisagés :

1. Le fonctionnement du module de diagnostic n'a pas été simulé dans un environnement dynamique au sein duquel des événements de ré-ordonnancement, d'indisponibilité et des attaques ont lieu de manière aléatoire. Une telle simulation serait nécessaire pour mettre en lumière les limites de notre méthode au regard du temps de calcul et à l'explosion des états du diagnostiqueur commun ;
2. Face aux deux limites exposées ci-dessus, l'algorithme de calcul du diagnostiqueur commun pourrait, à court terme, être amélioré selon les deux pistes suivantes :
   - Le calcul des profils d'attaquant pourrait être accéléré grâce au développement de fonctions heuristiques plus informées ;
   - Le diagnostiqueur pourrait être construit sur un espace d'états plus réduit en le restreignant à un ordonnancement partiel et non total ;
3. Le nombre de places de contrôle construites en cas de changement de mode pourrait être réduit en supprimant de manière optimale les places de contrôle redondantes avec celles du $mode_0$ (voir sous-partie 4.1.2) ;
4. L'expérimentation doit être consolidée afin de correspondre plus rigoureusement à nos travaux. D'une part, les temps de trajets devraient être inclus aux temps d'opérations dans le but de respecter le fonctionnement FMS de la plateforme. D'autre part, des scénarios d'attaque plus complexes et de nouvelles recettes devraient être implémentés pour pouvoir expérimenter les attaques de blocage partiel indifférenciables de changements de mode.

A moyen terme, six orientations de recherche peuvent être envisagées :

1. Dans la continuité de la dernière perspective à court terme, une condition nécessaire et suffisante à l'existence des attaques de blocage partiel indifférenciées des changements de mode pourrait être développée. Cette condition s'appuierait sur des propriétés structurelles des modèles S^3PR ;
2. Nous avons exposé dans le chapitre 1 la variété des caractéristiques associées à un FMS (capacité non unitaire des ressources, opérations multi-ressources, ressource requise pour plusieurs opérations successives, etc.). La possibilité d'extension de nos travaux à ces caractéristiques pourrait être étudiée ;

3. Nous avons défini, dans le chapitre 3, que la fonction coût unitaire d'une attaque ne dépendait que des vulnérabilités propres du FMS face aux cyber-attaques. Une méthode rigoureuse d'estimation de ces coûts à partir de nos connaissances techniques sur les vulnérabilités des composants numériques, des ressources et de l'architecture d'un FMS devrait être développée ;
4. L'existence des attaques physiques contres les signaux bas-niveau (sous-partie 2.2.1) pourrait être considérée comme hypohtèse de travail. La robustesse de notre module de diagnostic face à ces attaques devra alors être prouvée ou, en cas de non robustesse, notre module de diagnostic sera amélioré afin de devenir robuste à ces attaques ;
5. La perspective expérimentale à moyen terme que nous envisageons est l'implémentation des scénarios d'attaque directement depuis le réseau industriel de la plateforme et non par simulation dans l'automate de supervision. Par exemple, un module Arduino contenant les différents scénarios pourrait être connecté au switch local du réseau industriel de la plateforme afin de réaliser les attaques d'insertion et de suppression d'événements grâce à un port-mirroring et une attaque "Man-In-The-Middle".
6. Enfin, le fonctionnement en-ligne du module de diagnostic devrait être implémenté sur la plateforme expérimentale. Ainsi, des capteurs indépendants connectés au signaux bas-niveau et capables d'interpréter les variations de ces signaux seraient déployés sur la plateforme et connectés via un réseau indépendant à un automate hébergeant le module de diagnostic. Idéalement, une interface graphique affichant les résultats de diagnostic devrait être développée avec, pour objectif, de notifier les opérateurs lors d'événements suspicieux ou synonymes d'attaques de blocage.

A long terme, quatre perspectives de travail peuvent être envisagées :

1. La notion de profil d'attaquant a été restreinte dans ce manuscrit à la sélection de trois profils pour notre cas d'étude. La généralisation de cette notion à l'ensemble des SEDs pourrait être menée et conduire à une méthode de génération automatisée de profils d'attaquant selon différentes caractéristiques paramétrables de ceux-ci. Ces profils d'attaquant auto-générés pourraient alors être utilisés pour valider des méthodes de cyber-sécurité de la littérature ;
2. Nous avons observé, dans le chapitre 4, que les attaques ciblant un blocage partiel du FMS ne sont pas systématiquement diagnosticables et peuvent être confondues avec des changements de mode non identifiés comme malveillants. Face à cette limite, deux perspectives inspirées des travaux sur la planification des opérations de maintenance préventive [290], [370]-[372] pourraient être envisagées :
    - Premièrement, des règles plus restrictives de planification des opérations de maintenance préventive peuvent interdire les changements de mode non distinguables d'une attaque de blocage partiel. Une méthode robuste de développement automatisé de ces règles à partir des différents modes de fonctionnement et marquages atteignables par le FMS pourrait être développée ;
    - Deuxièmement, une composante stochastique pourrait être associée aux changements de mode afin d'évaluer leur probabilité d'occurrence. Cette composante pourrait permettre au diagnostiqueur d'estimer si un changement de mode a une forte probabilité de correspondre à une opération de maintenance ou non. Les indicateurs

de fiabilité et de maintenabilité des ressources pourraient alors être utilisés pour construire cette composante stochastique. Ainsi, plus la probabilité d'une opération de maintenance sur une ressource du FMS est faible, plus le changement de mode causant l'indisponibilité de cette ressource a de chance d'être malveillant ;

3. Enfin, nos travaux pourraient être adaptés à d'autres domaines d'application au sein desquels le partage de ressources est requis pour le pilotage d'un système physique complexe. Par exemple, au sein d'un navire militaire, les ressources partagées pourraient être définies comme la puissance moteur disponible, les opérateurs humains et les ressources matérielles tandis que les recettes seraient les différentes missions que doit réaliser le navire (direction, propulsion, utilisation de l'armement, télé-communication, refroidissement etc.). Une allocation malveillante non optimale des ressources à ces différentes missions pourrait entrainer un ralentissement voir un blocage des activités du navire. Dans un autre domaine d'application, l'allocation de zones de route aux véhicules autonomes d'une même flotte pourrait être manipulée par un attaquant afin de créer des collisions entre les véhicules ou un blocage de la flotte [356], [374]. Dans ce système, les ressources partagées seraient les zones de route tandis que les recettes correspondraient aux stratégies de conduite et aux destinations des différents véhicules. Toutefois, différemment des FMSs, une flotte de véhicule est pilotée de manière distribuée grâce à la communication des véhicules voisins les uns avec les autres. Par conséquent, la construction de l'attaque devra prendre en compte ce type de pilotage. Ces deux perspectives d'application de nos travaux aux navires militaires et aux flottes de véhicules autonomes nécessiteraient un travail de recherche approfondi pour valider ou non leurs faisabilités.

Chapitre 6

Annexe

6.1 Introduction aux réseaux de Petri

Définition 6.1.1 (Réseau de Petri)**.**
Un réseau de Petri est un 4-uplet $N = (P, T, F, W)$, où $P = \{p_1, p_2, p_3, \ldots, p_n\}$ est un ensemble fini de places avec $n > 0$, où $T = \{t_1, t_2, t_3, \ldots, t_m\}$ est un ensemble fini de transitions avec $m > 0$, $P \cup T \neq \varnothing$ et $P \cap T = \varnothing$. $F \subseteq (P \times T) \cup (T \times P)$ est l'ensemble de tous les arcs orientés , où $P \times T \to N$ est la fonction d'entrée qui définit l'ensemble des arcs de P vers T, et $T \times P \to N$ est la fonction de sortie qui définit l'ensemble des arcs de T vers P, où $N = \{0, 1, 2, ...\}$. $W : F \to \mathbb{N}$ est la fonction de pondération de chaque arc de F.

◇

Définition 6.1.2 (Marquage et marquage initial)**.**
Un marquage M d'un Rdp N est une fonction $M : P \to \mathbb{N}$ associant chaque place $p_i \in P$, $i \in [1, n]$, avec son marquage actuel $M(p_i) \in \mathbb{N}$ où $M(p_i)$ représente le nombre de jetons dans la place p_i. L'ensemble de tous les $M(p_i)$, à savoir le marquage M, symbolise l'état actuel du système modélisé par N.

Si N est marqué, $M_0 : P \to \mathbb{N}$ est le marquage initial de N, associant chaque place $p_i \in P$ avec un marquage initial de jeton $M_0(p_i)$. Dans la suite de ce manuscrit, un RdP sans marquage initial sera noté $N = (P, T, F, W)$. Lorsqu'il est associé à un marquage initial M_0, un RdP N sera noté (N, M_0).

◇

Pour un ensemble de places $Q \subseteq P$, $M(Q) = \sum_{p_i \in Q} M(p_i)$ est le marquage total des places appartenant à Q et $M|_Q$ est le marquage M restreint ou projeté aux places de Q. Dans la suite de ce manuscrit, un marquage M pourra être écrit $M = M(p_1)p_1 + M(p_2)p_2 + ... + M(p_n)p_n$, $n = |P|$. Par conséquent, on obtient $M|_Q = \sum_{p \in Q} M(p)p$.

Définition 6.1.3 (Preset et Postset)**.**
Pour un élément $x \in P \cup T$, le preset de x est noté ${}^\bullet x$, et son postset $x^\bullet$, où ${}^\bullet x = \{y \in P \cup T | (y, x) \in F\}$ et $x^\bullet = \{y \in P \cup T | (x, y) \in F\}$. Pour un ensemble $P_k \subseteq P$, nous définissons ${}^\bullet P_k = \{{}^\bullet p | p \in P_k\}$ et $P_k^\bullet = \{p^\bullet | p \in P_k\}$.

◇

Pour une place p_i, $i \in [1, n]$, l'ensemble des transitions d'entrée est notée ${}^\bullet p_i$, et l'ensemble des transitions de sortie $p_i^\bullet$. De la même manière, pour une transition t_j, $j \in [1, m]$, l'ensemble des places d'entrée et noté ${}^\bullet t_j$, et l'ensemble des places de sortie $t_j^\bullet$.

Définition 6.1.4 (Transition franchissable)**.**
Une transition t_j est dite franchissable ou activable si chaque place d'entrée $p_i \in {}^\bullet t_j$ est marquée par au minimum $W(p_i, t_j)$ jetons, où $W(p_i, t_j)$ est le poids de l'arc orienté de p_i vers t_j.

◇

Le franchissement d'une transition t_j suit deux étapes. D'une part, cela enlève $W(p_i, t_j)$ jetons de chaque place d'entrée $p_i \in {}^\bullet t_j$ et d'autre part, ajoute $W(t_j, p_i)$ jetons à chaque place de sortie $p_i \in t_j^\bullet$, où $W(t_j, p_i)$ est le poids de l'arc de t_j vers p_i. Le procédé de franchissement de la transition t_j conduit le RdP N d'un marquage M à un nouveau marquage M' et est noté $M[t_j > M'$. Si un marquage M'' peut être atteint depuis un marquage M par le franchissement d'une séquence de transitions $\sigma = [t_0, t_1, t_2, ..., tk]$, $k \in [1, m]$, cette succession de franchissements est notée $M[\sigma > M''$. Le marquage M'' est dit accessible depuis M. A partir de la définition du marquage d'une place, si $M[t_j > M'$ avec $t_j \in T$, nous avons alors, pour chaque $p_i \in N$, $M'(p_i) = M(p_i) + W(t_j, p_i) - W(p_i, t_j)$. Cette dernière formule peut être représentée par deux matrice distinctes de dimension $n \times m$, la matrice d'incidence arrière Pre et la matrice d'incidence avant $Post$.

Définition 6.1.5 (Matrice d'incidence)**.**
La matrice d'incidence arrière Pre et la matrice d'incidence avant $Post$ sont définies par :

$$Pre(p_i, t_j) = Pre(i, j) = \begin{cases} W(p_i, t_j), & if\ p_i \in^\bullet t_j \\ 0, & sinon \end{cases}$$

$$Post(p_i, t_j) = Post(i, j) = \begin{cases} W(t_j, p_i), & if\ p_i \in t_j^\bullet \\ 0, & sinon \end{cases}$$

Ainsi, nous avons pour chaque marquage M et M', $M' = M + Post(:, j) - Pre(:, j)$ avec $M[t_j > M'$. Dans un RdP N sans boucle (*une boucle est un couple* (p, t) *tel que* $t \in {}^\bullet p$ *et* $t \in p^\bullet$), la matrice d'incidence est définie par $[N] = Post - Pre$, tel que $M' = M + [N](:, j)$.

◇

Chaque ligne de $[N]$ représente une place et chaque colonne une transition. Une cellule (i, j) de $[N]$, notée $[N](i, j)$, décrit comment le marquage $M(p_i)$ de la place p_i évoluera en $M'(p_i)$ si la transition t_j est franchie.

Définition 6.1.6 (Marquages accessibles)**.**
Soit un RdP N, l'ensemble de tous les marquages accessibles depuis un marquage M est désignée par $\mathcal{R}(N, M) = \{M' \mid \exists\ \sigma = [t_0, t_1, t_2, ..., tk]\ où\ M[\sigma > M'\}$.

L'ensemble des marquages accessibles peut être modélisé à travers un graphe orienté $\mathcal{RG}(N, M) = (V, F)$ avec $V = \mathcal{R}(N, M)$ l'ensemble des nœuds du graphe et $F = \{(M, M'), M, M' \in$

$\mathcal{R}(N,M)|\exists t \in T, M[t > M'\}$ l'ensemble des arcs orientés du graphe. $\mathcal{R}1(N,M)$ est appelé graphe des marquages accessibles de (N,M). ◇

Définition 6.1.7 (Composante fortement connexe)**.**
Une composante fortement connexe (CFC) d'un graphe orienté (V,F) est un ensemble de nœuds $V_1 \subset V$ au sein duquel tous les nœuds sont atteignables depuis n'importe quel nœud $v_1 \in V_1$. ◇

Définition 6.1.8 (Séquence de transitions)**.**
Soit T^* l'ensemble de toutes les séquences de transitions, $T^* = \{\sigma = [t_0, t_1, t_2, ..., tk],\ k \in \mathbb{N}|\forall i \in \{1,k\},\ t_i \in T\}$. L'ensemble de toutes les séquences de transitions franchissables depuis un marquage M est noté $L(N,M) = \{\sigma \in T^*|\exists M' \in \mathcal{R}(N,M),\ M[\sigma > M'\}$. Cet ensemble restreint aux séquences bornées par $|\sigma| \leq n$ est noté $L(N,M)_{\leq n}$.

◇

Définition 6.1.9 (Chemin et circuits d'un RdP)**.**
Un chemin de N est une séquence de places et de transitions $x_1x_2...x_{k+1}$ avec $x_i \in P \cup T$ et $\forall i \in [1,k], (x_i, x_{i+1}) \in F, i = 1,2,...,k$ et k est la longueur du chemin. Un circuit C est un chemin dont le premier et le dernier élément sont identiques, i.e. $x_1 = x_{k+1}$. Un circuit peut contenir plusieurs fois le même élément. Un circuit est dit élémentaire si $\forall i \neq j \in [1, k+1], x_i \neq x_j$.

◇

Les RdPs possèdent des propriétés propres que nous utiliserons dans ce manuscrit.

Définition 6.1.10 (RdP ordinaire ou généralisé)**.**
Un RdP N est un RdP ordinaire si $\forall (x,y) \in (T \times P) \cup (P \times T), W(x,y) = 1$. Sinon, N est appelé un RdP généralisé. ◇

Définition 6.1.11 (RdP borné)**.**
Dans un RdP marqué (N, M_0), une place p est dites k-bornée si pour tout $M \in \mathcal{R}(N, M_0)$, $M(p) \leqslant k$. Un RdP (N, M_0) est dit k-borné si pour toute place $p_i \in P$, p_i est k-bornée. A contrario, un RdP qui n'est pas borné est qualifié de *non-borné* et ne peut pas converger vers un ensemble fini de marquages atteignables. ◇

Définition 6.1.12 (Vivacité)**.**
Une transition t_i est vivante si pour tout marquage $M \in \mathcal{R}(N, M_0)$, il existe une séquence σ de transitions franchissable depuis M qui contient t_i. Un RdP N est considéré vivant si toutes ses transitions sont vivantes. Par extension, on qualifie un circuit de vivant si toutes ses transitions sont vivantes ; ◇

Définition 6.1.13 (Blocage ou marquage bloquant)**.**
Un blocage ou marquage bloquant M est un marquage $M \in \mathcal{R}(N, M_0)$ depuis lequel aucune transition ne peut être franchie. La vivacité d'un RdP garantie l'absence de blocage car depuis n'importe quel marquage $M \in \mathcal{R}(N, M_0)$, il est toujours possible de franchir n'importe quelle transition t_i. Un RdP sans blocage est appelé un RdP libre. ◇

Définition 6.1.14 (RdP réinitialisable)**.**
Un RdP (N, M_0) est dit réinitialisable, si pour chaque marquage $M \in \mathcal{R}(N, M_0)$, il existe une séquence de transitions σ validant $M[\sigma > M_0$. En d'autres termes, le marquage initial de N peut être atteint depuis n'importe quel marquage $M \in \mathcal{R}(N, M_0)$. Par extension, un marquage M depuis lequel M_0 est accessible est qualifié de réinitialisable. Si un RdP possède un marquage

bloquant, il n'est pas réinitialisable. ◇

6.2 Composition de RdPs

Définition 6.2.1. Composition de 2 RdPs
La composition de deux RdPs labellisés $N_1 = (P_1, T_1, F_1, E_1, l_1)$ et $N_2 = (P_2, T_2, F_2, E_2, l_2)$ en un RdP N_3 selon la fusion de leurs transitions $T_1 = T_2$ est notée $N_3 = N_1 \parallel N_2$ et est définie par :

1. $N_3 = (P_3, T_3, F_3, E_3, l_3)$;
2. $P_3 = P_1 \cup P_2$;
3. $T_3 = T_1 = T_2$;
4. $F_3 = \{(X,Y)|X \in P_3 \cap P_1, Y \in T_3 = T_1, (X,Y) \in F_1\} \cup \{(Y,X)|X \in P_3 \cap P_1, Y \in T_3 = T_1, (Y,X) \in F_1\} \cup \{(X,Y)|X \in P_3 \cap P_2, Y \in T_3 = T_2, (X,Y) \in F_2\} \cup \{(Y,X)|X \in P_3 \cap P_2, Y \in T_3 = T_2, (Y,X) \in F_2\}$;
5. $E_3 = E_1 = E_S$;
6. Soit $l_3 : T_3 \to (E_3 \cup \varepsilon)$ où pour une transition $t \in T_3 = T_1 = T_S$, $l_3(t) = l_1(t) = l_2(t) = e \in E_3 = E_1 = E_2$;
7. Un marquage initial M_0^3 est qualifié d'acceptable ssi $M_0^3(p), \forall p \in P_1$ et $M_0^3(p), \forall p \in P_2$ sont acceptables selon la définition d'un modèle N_e.

◇

6.3 Franchissement d'une transition dans un RdP P-temporisé

Définition 6.3.1 (Vecteur des temps de délai restants)**.** Soit une matrice R définie pour représenter l'état temporel des jetons à chaque état du FMS. Si on suppose que les places activités de N sont κ-bornés, $\kappa \in \mathbb{N}$, R est alors une matrice de dimensions $|P| \times \kappa$ où un élément $R(p_i, k)$ de R, $k \in [1, \kappa]$, désigne le temps de délai restant du $k^{ième}$ jeton présent dans la place $p_i \in P$. Si il n'existe pas de $k^{ième}$ jeton dans p_i, $R(p_i, k) = 0$.

◇

Soit un état $M \in \mathcal{R}(N, M_0)$, avec M_0 un marquage initial acceptable de N, R l'état temporel des jetons de N à M et Γ_{M_0} le temps écoulé depuis le début de l'exploration de N à partir de M_0. On désigne par (M, R) l'état global du RdP P-temporisé N. On admet que la transition $t_j \in T$ est franchissable dans le cas non temporisé et souhaite être franchie afin d'atteindre un nouvel état (M', R') et une nouvelle durée écoulée Γ'_{M_0}. Le franchissement de t_j permettant d'obtenir ce nouvel état s'organise selon les étapes ci-après :

1. Calculer l'état M' identiquement au cas non temporisé : $M'(:, j) = M(:, j) + (Post(:, j) - Pre(:, j))$ et $M'(:, j_1) = M(:, j_1)$ avec $j_1 \neq j$;
2. Construire l'ensemble des places précédents t_j dont le temps de délai est non nul : $P_{pre}(t_j) = \{p_i \in^\bullet t_j | D(p_i) > 0\}$. Si $P_{pre}(t_j) = \emptyset$, alors $R' = R$. Sinon :

(a) Calculer le temps de délai restant maximum parmi tous les jetons présents dans les places $p_i \in P_{pre}(t_j)$ et requis pour le franchissement de t_j. On a :

$$\delta = \max_{p_i \in P_{pre}(t_j)} (R(p_i, M(p_i) - W(p_i, t_j) + 1)$$

(b) Calculer R_- en soustrayant δ des éléments non nuls de R. Autrement dit $\forall p_i \in P$ et $\forall k \in [1, \kappa]$, on a

$$R_-(p_i, k) = \begin{cases} R(p_i, k) - \delta, & si\ R(p_i, k) \geq \delta \\ 0, & sinon \end{cases}$$

3. Construire l'ensemble des places succédant t_j dont le temps de délai est non nul : $P_{post}(t_j) = \{p_i \in t_j^\bullet | D(p_i) > 0\}$. Si $P_{post}(t_j) \neq \emptyset$, alors $\forall p_i \in P_{post}(t_j)$, on a $R'(p_i, :) = \{D(p_i)^{W(t_j,p_i)}, R_-(p_i, [1, \kappa - W(t_j, p_i)])\}$ où $D(p_i)^{W(t_j,p_i)}$ est la vecteur ligne de $W(t_j, p_i)$ fois $D(p_i)$.
4. Calculer $\Gamma'_{M_0} = \Gamma_{M_0} + \delta$.

La méthode de franchissement temporisé de t_j dans N présenté ci-dessus sera désignée par la fonction $\Omega_N : T \times \mathcal{R}(N, (M_0, r_0)) \times \mathbb{R}^+ \to \mathcal{R}(N, (M_0, r_0)) \times \mathbb{R}^+$ et notée $\Omega_N(t_j, M, R, \Gamma) = (M', R', \Gamma')$. La fonction Ω_N peut être étendue à $T^\star$ avec pour $\sigma_2 = \sigma_1 t \in T^\star$, $\Omega_N(\sigma_2, M, R, \Gamma) = \Omega_N(t, \Omega_N(\sigma_1, M, R, \Gamma))$.

Dans les étapes de la fonction Ω_N, quelques explications doivent être apportées.

Premièrement, dans l'étape 3., pour les $W(t_j, p_i)$ jetons ajoutés dans p_i après le franchissement de t_j, on insère $W(t_j, p_i)$ fois $D(p_i)$ au début du vecteur $R'(p_i, :)$ puis on le complète par les valeurs de délais de temps des jetons contenus dans p_i avant le franchissement de t_j. Ainsi, dans $R(p_i, :)$, les temps de délai restants sont triés de manière décroissante puisque les nouveaux temps sont ajoutés au début du vecteur. Notons qu'un jeton peut avoir un temps de délai nul si sa valeur dans R est nulle.

Deuxièmement, dans l'étape 2.1, le choix de l'indicateur $M(p_i) - W(p_i, t_j) + 1$ repose sur le fait que R soit trié par ancienneté croissante des jetons. En effet, au sein des $M(p_i)$ jetons présents dans p_i, on sélectionne les $W(p_i, t_j)$ plus anciens et parmi ceux-ci, on sait que celui avec l'indicateur le plus faible dans R est celui avec le temps de délai restant le plus élevé. On obtient ainsi le temps de délai maximum pour les jetons de p_i nécessaires au franchissement de t_j. La calcul de δ repose ensuite sur le calcul du maximum de ces temps entre toutes les places $p_i \in P_{pre}(t_j)$.

Troisièmement, franchir la transition t_j ne signifie pas franchir la première transition franchissable de N. Pour connaitre cette première transition franchissable, il suffit de calculer pour toute transition t_j franchissable dans le cas non temporisé son temps de délai maximum δ_j, puis de choisir la transition t_j ayant le δ_j minimum.

Quatrièmement, cette méthode de franchissement des transitions dans un RdP P-temporisé prend correctement en compte le parallélisme d'exécution des opérations dans le calcul des temps de délai restant R et du temps global d'opération Γ_{M_0}. De fait, supposons que deux opérations O_i et O_j débutent en même temps pour des durées $D(p_i^1) > D(p_j^1)$ et pour une seule unité de ressource. On a donc initialement que $M(p_i^1) = M(p_j^1) = 1$, $R(p_i^1, 1) = D(p_i^1)$

et $R(p_j^1, 1) = D(p_j^1)$. Lorsque O_j se termine, la transition t_j^1 est franchie en consommant le jeton présent dans p_j^1 et la méthode de franchissement met alors à jour les valeurs de R' et Γ'_{M_0} telles que $R'(p_j^1, 1) = 0$, $R'(p_i^1, 1) = D(p_i^2) - D(p_j^2)$ et $\Gamma'_{M_0} = \Gamma_{M_0} + D(p_j^1)$. Ainsi, la durée des exécutions parallèles de O_i et O_j sera $D(p_i^1)$ et non $D(p_i^1) + D(p_j^1)$. De la même manière, cette méthode permet de calculer précisément la durée d'une attaque lorsque les transitions d'attaques $t \in E_a \subset E_e$ sont franchies.

6.4 SC-net

Définition 6.4.1 (Modèle SC-net)**.** Un modèle SC-net noté $N_t = (P_A \cup P_R \cup P_E \cup P_S, T, F_t, D)$ est défini à partir d'un RdP P-temporisé $N = (P_A \cup P_R \cup P^0, T, F, D)$ par :

1. $P_E = \{p_e | p^0 \in P^0\}$ est l'ensemble des places d'entrée du FMS modélisant les produits devant être fabriqués lorsqu'ils arrivent tandis que $P_S = \{p_s | p^0 \in P^0\}$ est l'ensemble des places de sorties du FMS symbolisant les produits correctement fabriqués et évacués hors du FMS ;
2. $T_t = T$;
3. $F_t = \{(X, Y) \in F | X, Y \notin P^0\} \cup F_E \cup F_S$ avec $F_E = \{(p_e, t) | p_e \in P_E, t \in T_t \; et \; (p^0, t) \in F\}$ et $F_S = \{(t, p_s) | p_s \in P_S, t \in T_t \; et \; (t, p^0) \in F\}$ où p_0 est la place dans N associée aux places p_e et p_s ;
4. $D : (P_A \cup P_R) \to \mathbb{R}$ est la fonction temps de délai qui associe à chaque place opération un temps d'opération et à chaque place ressource un temps de libération.
5. Un marquage (M_0^t, R_0^t) est un marquage initial qualifié d'acceptable pour N_t si pour un marquage M_0 acceptable pour N, on a :
   - $\forall p_a \in P_A$, $M_0^t(p_a) = 0$;
   - $\forall p_r \in P_R$, $M_0^t(p_r) = 1$;
   - $\forall p_e \in P_E$, $M_0^t(p_e) = M_0(p^0)$ avec $p^0 \in P^0$ la place d'entrée de N associée à p_e^t ;
   - et $\forall p_s \in P_S$, $M_0^t(p_s) = 0$, aucun produit n'a été réalisé ;
   - $R_0^t = O_{|P_t| \times 1}$, aucune opération n'a été débutée et les places activités ont une capacité de 1.

◇

Définition 6.4.2 (Marquage Objectif)**.** A partir d'un marquage $(M, R) \in \mathcal{R}(N_t, (M_0^t, R_0^t))$, un marquage objectif $(M_{obj}, R_{obj}) \in \mathcal{R}(N_t, (M_0^t, R_0^t))$ est défini par :

- $\forall p_a \in P_A$, $M_{obj}(p_a) = 0$, toutes les opérations ont été terminée ;
- $\forall p_r \in P_R$, $M_{obj}(p_r) = 1$, toutes les ressources ont été libérées ;
- $\forall p_e \in P_E$, $M_0^t(p_e) = 0$, tous les produits en entrée du FMS ont été réalisés ;
- Pour une place $p_s \in P_S$, $M_{obj}(p_s) = M(p_e) + \sum M(p_a)$ avec p_e la place d'entrée associée à la même recette que p_s et p_a une place activité appartenant à un chemin $p_e^t t_1 p_{a_1} .... p_{a_i} t_j p_s^t$ reliant p_e^t et p_s^t. En d'autres termes, pour une recette A, tous les produits en entrée de cette recette devant être fabriqués et ceux en cours de réalisation sont additionnés pour obtenir le nombre de produits finis devant être obtenus à la fin de l'ordonnancement.
- $R_{obj} = O_{|P_t| \times 1}$, toutes les opérations ont été terminées.

◇

6.5 Algorithme de recherche $A^\star BT$

Dans l'algorithme $A^\star BT$ illustré par le pseudo-code 1 ci-contre chaque $OPEN_i$ correspond à une ième recherche $A^\star$. Cette recherche a lieu sur une liste $OPEN_i$ tant que la taille de $OPEN_i$ n'est pas supérieur à $K_{max} \in \mathbb{N}$ (ligne 8 à 11) ou tant que $OPEN_i$ n'est pas vide (ligne 12 à 18). Si la première condition cas est vraie, l'algorithme prend le premier état de $OPEN_i$, l'état le plus prometteur en raison du tri réalisé ligne 41, et créer un nouvel ensemble $OPEN_{i+1}$ à partir de cet état. Une nouvelle recherche $A^\star$ est alors débutée depuis cet état. La variable K_{max} permet donc de limiter la taille de l'espace exploré lors de chaque itération i de la recherche $A^\star$, et par conséquent de moduler le temps de calcul en considérant que plus l'espace exploré est petit, plus l'algorithme convergera rapidement. Notons que si $K_{max} = 0$, l'algorithme est équivalent à un algorithme de recherche en profondeur [365]. Si le second cas est vérifié, le retour sur trace est appliqué et permet de débuter une nouvelle ième recherche sur $OPEN_i$ à partir de l'état le plus prometteur des $OPEN_j, j < i$ n'ayant pas encore été utilisé en entrée d'une recherche $A^\star$.

Entre les lignes 19 à 46, l'exploration de $OPEN_i$ a lieu. Le premier élément de $OPEN_i$, son état le plus prometteur, est assigné à (M, R) (ligne 19), est ôté de $OPEN_i$ (ligne 20) et est ajouté à la liste $CLOSED$ (ligne 21) car considéré comme "déjà visité" par l'algorithme. Si (M, R) est différent de (M_{obj}, R_{obj}) (ligne 22 à 24), toutes les transitions franchissables depuis ce dernier dans N_t sont déterminées (ligne 25), et chacune d'entre elles est franchie (ligne 26 et 27). Le nouvel état atteint après le franchissement temporisé d'une transition est noté (M', R') (ligne 27) et est lié à (M, R) par un arc ajouté à l'ensemble F_{ABT} (ligne 28). Le coût passé g (ligne 29) et l'estimation du coût restant h sont estimés afin de calculé l'estimation du coût total f (ligne 30). A partir de ces coûts, deux actions sur (M', R') et $OPEN_i$ ont lieu dans l'algorithme. Premièrement, selon son coût $g(M', R')$ et l'existence d'un autre état identique dans $OPEN_i$ et $CLOSED$, l'état (M', R') est ajouté soit à $OPEN_i$, soit à $CLOSED$, soit à aucun des deux (ligne 31 à 39). Deuxièmement, une fois (M', R') correctement distribué entre $OPEN_i$ et $CLOSED$, la liste $OPEN_i$ est trié selon les valeurs croissantes de $f(M, R), (M, R) \in OPEN_i$ (ligne 41).

L'algorithme de recherche $A^\star BT$ possède deux conditions d'arrêt : si l'état (M_{obj}, R_{obj}) est atteint (ligne 22) ou si l'algorithme ne trouve aucune solution, i.e. si tous les $OPEN_i$ sont vides (ligne 43).

6.6 Algorithme $A^\star \mathcal{P}$

L'algorithme de recherche $A^\star$ adapté au calcul des profils d'attaquant et noté $A^\star \mathcal{P}$ est présenté à travers le pseudo-code 2 ci-contre. Cet algorithme est adapté au calcul des 3 profils d'attaquant introduits dans la partie 3.1.1. Dans cette annexe, l'algorithme $A^\star \mathcal{P}$ est introduit au regard des ses spécificités par rapport à un algorithme de recherche $A^\star$ traditionnel, à savoir **la paramétrisation de l'algorithme selon le profil** et l'**implémentation de la propriété de sournoiserie d'un profil**. Préalablement, la fonction définie ci-dessous est requise.

Définition 6.6.1 (Fonction Υ d'indice de l'ordonnancement)**.**
Soit $\Upsilon : \mathcal{R}(N_\alpha, (M_{init}, R_{init})) \times \sigma_f \rightarrow \{1, 2, ..., |\sigma_f|\}$ la fonction qui pour un état (M, R) dans N_α accessibles depuis M_{init}, R_{init} et un ordonnancement $\sigma_f \in T^\star$ calculé depuis un état initial

Algorithm 1: Algorithme de recherche $A^{\star}BT$ [365]

Entrées : (M_{init}, R_{init}) l'état initial, (M_{obj}, R_{obj}) l'état objectif, N_t le CS-net, K_{max} la taille maximum de l'espace de solutions à explorer, la matrice WRT ;
Sorties : $\sigma_f = t_1 t_2 ... t_n$ l'ordonnancement final ;
$i = 0$;
$OPEN_i = (M_{init}, R_{init})$;
$CLOSED = ()$; /* $OPEN$ et $CLOSED$ sont deux listes ordonnées */
$F_{ABT} = \{\}$; /* F_{ABT} est un ensemble d'arcs */
while $OPEN_0 \neq \emptyset$ **do**
  if $|OPEN_i| > K_{max}$ **then**
    $i = i + 1$;
    $OPEN_i = OPEN_{i-1}(1)$;
  end if
  while $OPEN_i = \emptyset$ **do**
    if $OPEN_{i-1} \neq \emptyset$ **then**
      $OPEN_i = OPEN_{i-1}(1)$;
    else
      $i = i - 1$;
    end if
  end while
  $(M, R) = OPEN_i(1)$; ; /* On étudie le premier état (M, R) de $OPEN_i$ */
  $OPEN_i = OPEN_i \setminus OPEN_i(1)$; /* On ôte (M, R) de $OPEN_i$ */
  $CLOSED = (CLOSED, (M, R))$; /* On ajoute (M, R) à la fin de $CLOSED$ */
  if $(M, R) = (M_{obj}, R_{obj})$ **then**
    Retourner σ_f tel que $(M_{init}, R_{init})[\sigma_f > (M, R)$; /* La construction de σ_f est réalisée à l'aide des arcs de F_{ABT} */
  end if
  $L(N_t, M)|_1$; /* $L(N_t, M)|_1$ est l'ensemble des transitions franchissables depuis M dans N_t sans considération des délais de temps restants */
  for $t \in L(N_t, M)|_1$ **do**
    $(M', R', \Gamma) = \Omega(t, M, R, 0)$; /* (M', R') est l'état temporisé succédant (M, R) après le franchissement de t */
    $F_{ABT} = \{F_{ABT}, ((M, R), (M', R'))\}$; /* Un arc reliant (M, R) à (M', R') est ajouté à F_{ABT} */
    $g(M', R') = g(M, R) + \Gamma$;
    $f(M', R') = g(M', R') + h(M', R', WRT, N_t)$; /* c.f. 3.2.2 */
    if $\exists (M^o, R^o) \in OPEN_i | M^o = M', R^o = R'$ *et* $g(M', R') < g(M^o, R^o)$ **then**
      $OPEN_i = OPEN_i \setminus (M^o, R^o)$; /* Si (M', R') appartient déjà à $OPEN_i$ et a un coût passé g plus faible */
      $OPENT_i = (OPEN_i, (M', R'))$;
    else if $\exists (M^c, R^c) \in CLOSED | M^c = M', R^c = R'$ *et* $g(M', R') < g(M^c, R^c)$ **then**
      $CLOSED = CLOSED \setminus (M^c, R^c)$; /* Si (M', R') appartient déjà à $CLOSED$ et a un coût passé g plus faible */
      $OPEN_i = (OPEN_i, (M', R'))$;

37: **else if** $(M', R') \notin OPEN_i$ et $(M', R') \notin CLOSED$ **then**
38: $OPEN_i = (OPEN_i, (M', R'))$ /* Si (M', R') n'appartient ni à $OPEN_i$, ni à $CLOSED$ */
39: **end if**
40: **end for**
41: Trier la liste $OPEN_i$ selon les valeurs croissantes de $f(M, R), (M, R) \in OPEN_i$;
42: **end while**
43: Retourner "Erreur" ;

$(M_{init}, R_{init})|_{P_t}$ obtenu à partir du marquage des places $P_{G\alpha} \cup (P_a^0 \cap P_G) \in P_\alpha$, associe un entier naturel indiquant la prochaine décision d'allocation ou transition devant être franchie dans σ_f.

Cette fonction illustre le fait que lorsqu'une attaque σ_a a lieu et possède une durée non nulle, l'ordonnancement prévu est aussi exécuté par le module de supervision relativement à cette durée. Chaque état de N_α traversé par l'attaque correspond donc à une transition de décision d'allocation de σ_f devant être franchie la prochaine. ◇

Au sein de l'algorithme, les 4 paramètres propres à un profil donné sont illustrés en rouge parmi les entrées de l'algorithme et sont appliqués dans celui-ci de la manière suivante :

- Les états ciblés $\mathcal{M}_c \subset \mathcal{R}(N, M_0)$ sont requis ligne 11 afin de déterminer si l'état visiter $(M, R) = OPEN(1)$ appartient à $\mathcal{M}_c$. Notons que M est ici projeté sur $P^{G\alpha} = P_A^{G\alpha} \cup P_R^{G\alpha}$ et $\mathcal{M}_c$ est projeté sur $P_A \cup P_R$ afin d'être comparables entre eux. A cette ligne, si $M|_{P^{G\alpha}} \in \mathcal{M}_c|_{P_A \cup P_R}$, l'algorithme se termine et les séquences d'événements et de transitions de (M_{init}, R_{init}) à (M, R) sont retournées ;
- Les contraintes notées CTR sont implémentées ligne 46 lorsqu'un nouvel état (M', R') est atteint depuis l'état en cours d'exploration (M, R). Si pour cet état (M', R'), une contrainte $Ctr \in CTR$ est vraie, alors l'état (M', R') n'est pas considéré pour l'exploration et n'est ajouté ni à $OPEN$, ni à $CLOSED$. Notons qu'une contrainte peut s'appliquer à l'état (M', R') en lui-même ou à la séquence de transitions menant à (M, R) depuis (M_{init}, R_{init}) ;
- La fonction coût c et l'heuristique h sont appliqués aux lignes 22, 38-39 et 43-44 et permettent le calcul de g et de f. A l'instar de la recherche $A^\star$, le coût passé g est requis pour comparé un état intermédiaire (M'', R'') (lignes 23,26) ou un état atteint (M', R') (lignes 48,51) aux états identiques de $OPEN$ et $CLOSED$ et les remplacer si le nouveau coût passé est inférieur à l'ancien ; en d'autres termes, si le nouvel état est plus optimal que ceux identiques dans $OPEN$ ou $CLOSED$. Notons toutefois que dans le cas où (M'', R'') n'appartient ni à $OPEN$(ligne 23), ni à $CLOSED$(ligne 26), ce dernier n'est pas ajouter à $OPEN$ comme l'est l'état (M', R') puisque l'état que l'on souhaite réellement explorer est (M', R'), soit l'état ciblé par l'attaquant et accessibles depuis (M'', R'') en franchissant t.
- Enfin, les consignes de tri TRI sont implémentées ligne 58 et s'applique à $OPEN$ en amont et en aval du tri selon les valeurs croissantes de f. Dans le cas où une consigne de tri hors première consigne est appliquée, elle ne concerne que le tri des états voisins égaux selon toutes les consignes précédentes. L'ensemble des consignes de tri permettent alors

Algorithm 2: Algorithme de recherche $A^{\star}\mathcal{P}$

Entrées : $(M_{init}, R_{init}) \in \mathcal{R}(N_\alpha, M_0^\alpha)$ l'état initial, $\mathcal{M}_c \in \mathcal{R}(N, M_0)$ l'ensemble des états de G ciblés par l'attaquant, N_α, *Contraintes* les contraintes du profil, σ_f l'ordonnancement à partir de (M_{init}, R_{init}), c et h la fonction coût et l'heuristique du profil, et *Tri* les consignes de tri du profil hors coût ;

Sorties : $\mathcal{P}$ le profil d'attaquant, $\sigma_a = l_\alpha^{-1}(\mathcal{P})$ la séquence de transition correspondante dans $T_\alpha^\star$;

$\Upsilon((M_{init}, R_{init}), \sigma_f) = 1$;

$OPEN = (M_{init}, R_{init})$;

$CLOSED = ()$; /* $OPEN$ et $CLOSED$ sont deux listes ordonnées */

$F_\mathcal{P} = \{\}$; /* $F_\mathcal{P}$ est un ensemble d'arcs */

while $OPEN \neq \emptyset$ **do**

  $(M, R) = OPEN(1)$; /* On étudie le premier état (M, R) de $OPEN$ */

  $OPEN = OPEN \setminus OPEN(1)$; /* On ôte (M, R) de $OPEN$ */

  $CLOSED = (CLOSED, (M, R))$; /* On ajoute (M, R) à $CLOSED$ */

  if $M|_{(P_A^{G_\alpha} \cup P_R^{G_\alpha})} \in \mathcal{M}_c|_{(P_A \cup P_R)}$ **then**

    Retourner σ_a et $\mathcal{P} = l_\alpha(\sigma_a)$ tel que $(M_{init}, R_{init})[\sigma_a > (M, R)$;

  end if

  $T_{fr} = L(N_\alpha, M)|_1 \cap (T_+^\alpha \cup (T \cap \sigma_f(\Upsilon((M, R)), \sigma_f)))$; /* T_{fr} est l'ensemble des transitions franchissables dans N_α à partir de M */

  for $t \in T_{fr}$ **do**

    $(M', R', \Gamma') = \Omega_{N_\alpha}(t, M, R, 0)$; /* Franchissement de t depuis (M, R) dans N_α */

    $T- = t_-^1 t_-^2 ... t_-^n | (t_-^1, t_-^2, ..., t_-^n) \in T_-^\alpha$ *et* $\forall t_- \in T-, \tau_\alpha(t_-) \in \sigma_f(\Upsilon((M, R), \sigma_f) \rightarrow fin), (M'', R'', \Gamma'') = \Omega_{N_\alpha}(t_-^1 t_-^2 ... t_-, M, R, 0)$, *on a* $\Gamma'' \leq \Gamma'$; /* $T-$ est la séquence des transitions de suppression des décisions d'allocation successifs de σ_f depuis (M, R) ayant lieu avant ou simultanément à $t \in T_{fr}$ */

    if $T- \neq \emptyset$ et $t \neq \tau_\alpha(T-)$ **then** /* Si la séquence $T-$ est non vide */

      $(M_1'', R_1'') = (M, R)$; /* un état intermédiaire (M_1'', R_1'') est créé */

      for $t_- \in T-$ **do**

        $(M'', R'', \Gamma'') = \Omega_{N_\alpha}(t_-, M_1'', R_1'', 0)$;/* Franchissement de t_- depuis (M_1'', R_1'') dans N_α */

        $g(M'', R'') = g(M_1'', R_1'') + c((M_1'', R_1''), (M'', R''))$; /* Coût passé de (M'', R'') */

        /* Si l'état (M'', R'') appartient déjà à OPEN ou CLOSED et possède un meilleur coût passé */

        if $\exists (M^o, R^o) \in OPEN | M^o = M'', R^o = R''$ *et* $g(M'', R'') < g(M^o, R^o)$ **then**

          $OPEN = OPEN \setminus (M^o, R^o)$;

          $OPEN = (OPEN, (M'', R''))$;

        else if $\exists (M^c, R^c) \in CLOSED | M^c = M'', R^c = R''$*et* $g(M'', R'') < g(M^c, R^c)$ **then**

          $CLOSED = CLOSED \setminus (M^c, R^c)$;

          $OPEN = (OPEN, (M'', R''))$;

        end if

        $F_\mathcal{P} = \{F_\mathcal{P}, ((M_1'', R_1''), (M'', R''))\}$; /* Un arc de (M_1'', R_1'') à (M'', R'') est créé */

$\Upsilon(M'', R'') = \Upsilon(M''_1, R''_1) + 1$; /* Actualisation de Υ, une nouvelle décision d'allocation a eu lieu et a été supprimée */
$(M''_1, R''_1) = (M'', R'')$; /* L'état intermédiaire devient égal à (M'', R'') pour la prochain transition $t_- \in T-$ */
$g(M''_1, R''_1) = g(M'', R'')$;
end for
$(M', R', \Gamma' - \Gamma'') = \Omega_{N_\alpha}(t, M'', R'', \Gamma'')$; /* Franchissement de t depuis (M'', R'') atteint après l'exécution de $T-$ pour atteindre un nouvel état (M', R') */
$F_{\mathcal{P}} = \{F_{\mathcal{P}}, ((M'', R''), (M', R'))\}$; /* Un arc de (M'', R'') et (M', R') est créé */
$\Upsilon(M', R') = \Upsilon(M'', R'') = \Upsilon(M, R) + |T - |$; /* Actualisation de Υ */
$g(M', R') = g(M, R) + c((M, R), (M'', R'')) + c((M'', R''), (M', R'))$;
$f(M', R') = g(M', R') + h(M', R')$; /* g, h et f sont calculés selon la fonction coût et l'heuristique propres au profil */
else/* Sinon, si t est franchie avant la prochaine transition de l'ordonnancement σ_f */
$F_{\mathcal{P}} = \{F_{\mathcal{P}}, ((M, R), (M', R'))\}$; /* Un arc entre (M, R) et (M', R') est créé */
$\Upsilon(M', R') = \Upsilon(M, R)$; /* Aucune transition de l'ordonnancement n'a été franchie */
$g(M', R') = g(M, R) + c((M, R), (M', R'))$;
$f(M', R') = g(M', R') + h(M', R')$;
end if
if $\exists Ctr \in CTR | Ctr = vraie$ **then**
$(M', R') = \emptyset$; /* Si une contrainte est vérifié en (M', R'), l'état est interdit et n'est pas visité par l'algorithme */
else if $\exists (M^o, R^o) \in OPEN | M^o = M', R^o = R'$ *et* $g(M', R') < g(M^o, R^o)$ **then**
$OPEN = OPEN \setminus (M^o, R^o)$;
$OPEN = (OPEN, (M', R')$;
else if $\exists (M^c, R^c) \in CLOSED | M^c = M', R^c = M' et\ g(M', R') < g(M^c, R^c)$ **then**
$CLOSED = CLOSED \setminus (M^c, R^c)$;
$OPEN = (OPEN, (M', R')$;
else
$OPEN = (OPEN, (M', R'))$;
end if
end for
Trier la liste $OPEN$ selon les valeurs croissantes de $f(M, R), (M, R) \in OPEN$ et selon TRI, les autres consignes de tri du profil ;
end while
Retourner "Erreur' ;

d'orienter l'exploration des états par l'algorithme vers les séquences d'attaques les plus prometteuses pour le profil considéré.

La propriété de sournoiserie d'un profil est pour sa part implémentée entre les lignes 17 et 39 de l'algorithme 2 selon quatre étapes distinctes :

1. Premièrement, ligne 14, l'ensemble T_{fr} des transitions d'attaque franchissables dans N_α depuis l'état en cours d'exploration (M, R) intègre la prochaine décision d'allocation de σ_f, soit le terme $T \cap \sigma_f(\Upsilon((M, R)), \sigma_f))$ de T_{fr}. Si elle est franchissable depuis (M, R), cette transition de l'ordonnancement est associée dans T_{fr} aux transitions T_α^+ des attaques d'insertion de décisions d'allocation. Elles forment l'ensemble des capacités d'un attaquant depuis un état $(M, R) \in \mathcal{R}(N_\alpha, (M_{init}, R_{init})$: laisser l'ordonnancement se dérouler si l'état du FMS le permet ou insérer une décision d'allocation malveillante ;
2. Deuxièmement, pour chaque transition $t \in T_{fr}$ (ligne 15), la séquence de transitions de suppression des décisions d'allocation depuis (M, R) ayant lieu avant ou simultanément à t est identifiée (ligne 17) grâce à la fonction Υ et au calcul de $(M', R', \Gamma') = \Omega_{N_\alpha}(t, M, R, 0)$ représentant le franchissement temporisé de t (ligne 16). Cette séquence, notée $T-$, représente la dernière capacité de l'attaquant depuis (M, R) : la suppression des décisions d'allocation émises par le module de supervision. Cependant, ces décisions ne doivent être supprimées par l'attaquant uniquement si elle ont lieu entre l'état (M, R) et le prochain état (M', R') atteint en franchissant une transition d'attaque de T_{fr} ;
3. Troisièmement, si $T-$ n'est pas vide (ligne 18), toutes les décisions d'allocation qu'elle contient doivent être supprimées en franchissant successivement toutes les transitions $t_- \in T-$ avant de franchir t.
4. Quatrièmement, pour chaque transition $t_- \in T-$, un état intermédiaire (M'', R'') est calculé depuis l'état précédent (M_1'', R_1'') grâce à l'appel à la fonction de franchissement $(M'', R'', \Gamma'') = \Omega_{N_\alpha}(t_-, M_1'', R_1'', 0)$ (ligne 21). Initialement, si $T-$ n'est pas vide, (M_1'', R_1'') est défini égal à l'état en cours d'exploration (M, R) (ligne 19).
5. Cinquièmement, pour chaque transition $t_- \in T-$, l'état intermédiaire atteint (M'', R'') est ajouté lignes 22-29 aux listes $OPEN$ et $CLOSED$ si il leur appartient déjà et présente un coût passé plus faible (c.f. point 3 du paragraphe précédent). Puis, un arc entre (M_1'', R_1'') et (M'', R'') est ajouté à $F_\mathcal{P}$ ligne 30 et la fonction Υ est mise à jour car une nouvelle décision d'allocation a eu lieu et a été supprimée par t_- (ligne 31). Enfin, (M_1'', R_1'') est mis à jour et devient égal à (M'', R'') pour la prochaine transitions de $T-$.
6. Dernièrement, lorsque toutes les transitions de $T-$ on été franchies, la transition t est franchie depuis le dernier état intermédiaire (M'', R'') ligne 35 afin d'obtenir un nouvel état (M', R') intégrant la suppression de toutes les décisions d'allocation ayant lieu entre (M, R) et l'exécution de t. Un arc entre (M'', R'') et (M', R') est ajouté à $F_\mathcal{P}$ ligne 36. Il permettra avec les arcs ajoutés à l'étape 5. d'inclure T_- et les états intermédiaires dans le profil $\mathcal{P}$. Suite au franchissement de t, $\Upsilon((M', R'), \sigma_f)$ est incrémentée de $|T-|$ puisque $|T-|$ décisions d'allocation ont été prises entre (M, R) et (M', R') (ligne 37). Enfin, les fonctions g, f et h sont calculés ligne 38-39 depuis (M', R') afin de l'intégrer ultérieurement à $OPEN$ ou $CLOSED$ (lignes 46-55) et de réaliser le tri de $OPEN$ (ligne 58).

Pour conclure, notons que les spécificités communes aux profils introduites dans la partie 3.1.1 sont respectées dans l'algorithme. En effet, le profil en sortie de l'algorithme est une séquence unitaire (caractéristique 1), l'attaque est minimale (caractéristique 2) car la recherche s'arrête au premier état de $\mathcal{M}_c$ atteint, elle est experte (caractéristique 3) par définition de N_α, et elle est sournoise (caractéristique 4), N_α ne modélisant que les attaques sournoises et la recherche $A^\star\mathcal{P}$ connaissant et trompant l'ordonnancement réel σ_f du FMS (caractéristique 5).

6.7 Algorithme de construction d'un diagnostiqueur

Soit $Diag(N, M_{init}) = (X_d, E_d, f_d, x_0^d)$, le diagnostiqueur construit à partir du RdP N_l et d'un état initial $M_{init} \in \mathcal{R}(N_l, M_0)$. L'algorithme de construction d'un tel diagnostiqueur est détaillé dans le pseudo-code 3 ci-dessous et est présenté dans cet partie.

Dans cet algorithme, les lignes 3 à 13 permettent le calcul de l'état initial x_o^d du diagnostiqueur, puis les lignes 14 à 42 l'exploration de $\mathcal{RG}(N, M_{init})$ pour la construction de $Diag(N, M_{init})$. Au sein de l'exploration, pour chaque état $x_d \in X_d$ déjà exploré et pour chaque événement observable $e_o \in E_o$, l'ensemble $\mathcal{M}_e$ des marquages atteignables depuis les paires $(M_1, \Delta_1) \in x_d$ suite à l'occurrence de e_o est préalablement calculé (ligne 21). Le label Δ_1 est aussi associé aux marquages $M_e \in \mathcal{M}_e$ puisqu'aucun événement inobservable de faute n'appartient à e. Puis, l'ensemble $UR(\mathcal{M}_e)$ des marquages atteignables depuis un état $M_e \in \mathcal{M}_e$ après l'occurrence d'une séquence non observable $\lambda_{io} \in E_{io}^\star$ est construit ligne 22. Notons que les états M de $UR(\mathcal{M}_e)$ sont parcourus ligne 24 selon la séquence d'événements non observables λ_{io} reliant un état M_e à M puisque M peut être atteint selon plusieurs séquences λ_{io} depuis un état de $\mathcal{M}_e$. Chaque marquage de $UR(\mathcal{M}_e)$ est labellisé entre les lignes 23 et 35, avant d'être ajouté joint avec son label à x_d^{new} (lignes 36-38). Singulièrement, l'étape de labellisation est fondée sur le principe de propagation de faute (lignes 28 et 32), à savoir que si l'état $(M_1, \Delta_1) \in x_d$ précédant le marquage $M \in UR(\mathcal{M}_e)$ est labellisé par un ensemble de fautes Δ_1, cet ensemble est conservé au sein de la paire (M, Δ_2) ajoutée à x_d^{new}, i.e. $\Delta_1 \subset \Delta_2$, puisque ces fautes sont toujours contenues dans la séquence d'événements connectant x_0^d à M en passant par x_d. Similairement, le label O est conservé si $\Delta_1 = O$ et si aucune faute n'a lieu dans λ_{io}. Enfin, x_d^{new} est ajouté à X_d^{new} uniquement s'il n'avait pas encore été exploré puis un arc $f_d(x_d, e) = x_d^{new}$ est créé. L'algorithme se termine lorsque plus aucun état x_d^{new} n'est découvert et le diagnostiqueur est alors retourné (ligne 43).

6.8 Algorithme de construction de $Diag_a$

Soit $Diag_a(N_\alpha, M_{init}^\alpha) = (X_d^a, E_d^a, f_d^a, x_0^a)$, le diagnostiqueur des attaques de blocage construit à partir du modèle d'attaque N_α et d'un état initial $M_{init}^\alpha \in \mathcal{R}(N_\alpha, M_0^\alpha)$. L'algorithme de construction d'un tel diagnostiqueur est détaillé dans le pseudo-code 4 ci-dessous et est présenté dans cette partie.

Pour rappel, dans cet algorithme, $UR(\mathcal{M}_e) = \mathcal{M}_e$ puisque les événements non observables par le module de diagnostic ne conduisent pas le FMS vers un état critique de blocage. Ils ne sont par conséquent par considérés par $Diag_a$ et conduisent à omettre $UR(\mathcal{M}_e)$ de l'algorithme.

Algorithm 3: Algorithme de construction d'un diagnostiqueur

Entrées : $(N_l, M_0) = (P, T, F, E, l, M_0)$ avec $E = E_o \cup E_{io}$, $\Delta = \{O\} \cup \Delta_f$, M_{init} ;
Sorties : $Diag(N, M_{init}) = (X_d, E_d, f_d, x_0^d)$;
$UR(M_{init}) = \{M \in \mathcal{R}(N, M_{init}) | \exists \lambda_{io} \in E_{io}^\star, l^{-1}(\lambda_{io}) = \sigma_{io}, M_{init}[\sigma > M\}$; /* Ensemble des états accessibles depuis M_{init} via des événements non observables */
$x_0^d = \{\}$; /* L'état initial x_0^d est défini par un ensemble vide */
for $M \in UR(M_{init})$ **do**
  for $\lambda_{io} \in E_{io}^\star | l^{-1}(\lambda_{io}) = \sigma_{io}, M_{init}[\sigma_{io} > M$ **do**
    if $\nexists \delta \in \Delta_f | e_\delta \in \lambda_{io}$ **then**
      $x_0^d = \{x_0^d, (M, O)\}$; /* (M, O) est ajouté à x_0^d */
    else if $\exists \{e_{\delta_1}, ..., e_{\delta_k}\} \in \lambda_{io}$ avec $\delta_1, ..., \delta_k \in \Delta_f$ **then**
      $x_0^d = \{x_0^d, (M, \{\delta_1, ..., \delta_k\})\}$; /* $(M, \{\delta_1, ..., \delta_k\})$ est ajouté à x_0^d */
    end if
  end for
end for
$X_d = \{x_0^d\}$;
$X_d^{new} = \{\}$;
while $X_d^{new} \neq X_d$ **do**
  $X_d^{new} = X_d$; /* Tant que de nouveaux états sont découverts */
  for $x_d \in X_d$ **do**
    for $e \in E_o$ **do**
      $x_d^{new} = \{\}$;
      $\mathcal{M}_e = \{M_e \in \mathcal{R}(N, M_0) | \exists (M_1, \Delta_1) \in x_d, \Delta_1 \subset \Delta, l^{-1}(e) = t, M_1[t > M_e\}$; /* $\mathcal{M}_e$ est l'ensemble des états atteignables après l'occurence de e depuis un état M tel que $(M, \Delta_1) \in x_d$, et $\Delta_1 \subset \Delta$ est aussi le label associé à M_e */
      $UR(\mathcal{M}_e) = \{M \in \mathcal{R}(N, M_0) | \exists \lambda_{io} \in E_{io}^\star, l^{-1}(\lambda_{io}) = \sigma_{io}, M_e \in \mathcal{M}_e \ t.q. \ M_e[\sigma > M\}$; /* Ici, UR est défini pour tous les états de $\mathcal{M}_e$ */
      for $M \in UR(\mathcal{M}_e)$ **do**
        for $\lambda_{io} \in E_{io}^\star | l^{-1}(\lambda_{io}) = \sigma_{io}, M_e \in \mathcal{M}_e, M_e[\sigma_{io} > M$ **do**
          if $\Delta_1 = O$ et $\nexists \delta \in \Delta_f | e_\delta \in \lambda_{io}$ **then**
            $x_d^{new} = \{x_d^{new}, (M, O)\}$; /* (M, O) est ajouté à x_d^{new} */
          else if $\Delta_1 = \{\delta_1, ..., \delta_k \{\subset \Delta_f$ et $\nexists \delta \in \Delta_f | e_\delta \in \lambda_{io}$ **then**
            $x_d^{new} = \{x_d^{new}, (M, \Delta_1)\}$; /* Le label Δ_1 est propagé de M_e à M */
          else if $\Delta_1 = O$ et $\exists \{e_{\delta_1}, ..., e_{\delta_k}\} \in \lambda_{io}$ avec $\Delta_2 = \{\delta_1, ..., \delta_k\} \in \Delta_f$ **then**
            $x_d^{new} = \{x_d^{new}, (M, \Delta_2\})\}$;
          else if $\Delta_1 \subset \Delta_f$ et $\exists \{e_{\delta_1}, ..., e_{\delta_k}\} \in \lambda_{io}$ avec $\Delta_2 = \{\delta_1, ..., \delta_k\} \in \Delta_f$ **then**
            $x_d^{new} = \{x_d^{new}, (M, \Delta_1 \cup \Delta_2\})\}$;
          end if
        end for
      end for
      if $x_d^{new} \notin X_d^{new}$ **then**
        $X_d^{new} = \{X_d^{new}, x_d^{new}\}$;
      end if
      $f_d(x_d, e) = x_d^{new}$; /* Un arc de x_d à x_d^{new} est créé et labellisé par e */
    end for
  end for
end while
Retourner $Diag(N, M_{init}) = (X_d, E_d, f_d, x_0^d)$;

Algorithm 4: Algorithme de construction d'un diagnostiqueur $Diag_a$

Entrées : $(N_\alpha, M_0^\alpha) = (P_\alpha, T_\alpha, F_\alpha, E_\alpha, l_\alpha, M_0^\alpha)$ avec $E_o = E^\alpha \cup E_+^\alpha \in E_\alpha$, $\Delta_a = \{O, \delta_{A1}, \delta_{A2}\}$, M_{init}^α ;
Sorties : $Diag_a(N_\alpha, M_{init}^\alpha) = (X_d^a, E_d^a, f_d^a, x_0^a)$;
if $M_{init}^\alpha|_{P_A^{G\alpha} \cup P_R^{G\alpha}} \in \mathcal{M}_{DZ}|_{P_A \cup P_R}$ **then**
  $x_0^a = (M_{init}^\alpha|_{P^{G\alpha}}, \delta_{A2})$; /* Si M_{init}^α est un état de la DZ dans G */
else
  $x_0^a = (M_{init}^\alpha|_{P^{G\alpha}}, O)$;
end if
$X_d^a = \{x_0^a\}$;
$X_d^{new} = \{\}$;
while $X_d^{new} \neq X_d^a$ **do**
  $X_d^{new} = X_d^a$; /* Tant que de nouveaux états sont découverts */
  for $x_d \in X_d^a$ **do**
    for $e \in E_o$ **do**
      $x_d^{new} = \{\}$;

$\mathcal{M}_e = \{M \in \mathcal{R}(N_\alpha, M_0^\alpha) | \exists (M_1, \Delta_1) \in x_d, \Delta_1 \subset \Delta_a, t \in (l_\alpha)^{-1}((f_a^1)^{-1}(e)), M_1[t > M|_{P^{G\alpha}}\}$;

      $\mathcal{M}_e = \mathcal{M}_e|_{P^{G\alpha}}$; /* $\mathcal{M}_e$ est projeté sur les places de G dans N_α */
      for $M \in \mathcal{M}_e$ **do**
        /* M est précédé par la paire (M_1, Δ_1) */
        if $\Delta_1 = \{O\}$ et pour λ_o la séquence d'événements observables de $M_{init}^\alpha|_{P^{G\alpha}}$ à M_1, on a $\lambda_o e < \lambda_f$ avec $\lambda_f = l(\sigma_f)$ **then**
          $x_d^{new} = \{x_d^{new}, (M, O)\}$;
        else if $\Delta_1 = \{O\}$ et $\lambda_o e \not< \lambda_f$ **then**
          if $M|_{P_A \cup P_R} \in \mathcal{M}_{DZ}|_{P_A \cup P_R}$ **then**
            $x_d^{new} = \{x_d^{new}, (M, \delta_{A2})\}$; /* (M, δ_{A2}) est ajouté à x_d^{new} */
          else
            $x_d^{new} = \{x_d^{new}, (M, \delta_{A1})\}$;
          end if
        end if
      end for
      if $x_d^{new} \notin X_d^{new}$ **then**
        $X_d^{new} = \{X_d^{new}, x_d^{new}\}$;
      end if
      $f_d(x_d, e) = x_d^{new}$; /* Un arc de x_d à x_d^{new} est créé et labellisé par e */
    end for
  end for
end while
Retourner $Diag_a(N_\alpha, M_{init}^\alpha) = (X_d^a, E_d^a, f_d^a, x_0^a)$;

Par ailleurs, le label δ_{A1} correspond à la détection d'une attaque par non respect de la trajectoire optimale d'ordonnancement tandis que le label δ_{A2} est associé à une attaque conduisant le FMS dans un état interdit de la DZ. La notation $M|_{P_A^{G_\alpha} \cup P_R^{G_\alpha}}$ désigne la projection d'un marquage M sur l'ensemble des places activité et ressources de G dans N_α. Cette projection est nécessaire pour comparer les états de N_α avec les états de blocage de $\mathcal{M}_{DZ}$.

6.9 Algorithme de construction de $Diag_{\mathcal{P}}$

Soit $Diag_{\mathcal{P}}(N_\alpha, M^\alpha_{init}) = (X^{\mathcal{P}}_d, E^{\mathcal{P}}_d, f^{\mathcal{P}}_d, x^{\mathcal{P}}_0)$, le diagnostiqueur des profils d'attaquant construit à partir du modèle d'attaque N_α et d'un état initial $M^\alpha_{init} \in \mathcal{R}(N_\alpha, M^\alpha_0)$. L'algorithme de construction d'un tel diagnostiqueur est détaillé dans le pseudo-code 5 ci-dessous et est présenté dans cette partie.

Dans cet algorithme, le principe de propagation des labels est inversé puisque le label d'un profil cesse d'être propagé lorsque le nouvel état atteint $M \in \mathcal{M}_e$ n'est plus traversé par le profil. Le label $O_{\mathcal{P}}$ désigne l'absence de profils d'attaquant et arrête l'exploration du diagnostiqueur si il est associé à un état M_1 (ligne 19). La boucle FOR débutée à la ligne 22 permet de considérer tous les profils d'attaquant appartenant à $\Delta_{\mathcal{P}}$.

6.10 Composition de diagnostiqueurs

Soit $Diag_c = (X^c_d, E^c_d, f^c_d, x^c_0) = Diag_1 \circ Diag_2$ le diagnostiqueur construit par composition de $Diag_1 = (X^1_d, E^1_d, f^1_d, x^1_0)$ et $Diag_2 = (X^2_d, E^2_d, f^2_d, x^2_0)$, deux diagnostiqueurs débutés à partir d'un même marquage M_{init}. On a $x^c_0 = x^1_0 \otimes x^2_0$ avec $\otimes$ l'opérateur défini par :

Définition 6.10.1 (Fusion d'états de diagnostiqueurs)**.** Pour $x^1_d \in X^1_d, x^2_d \in X^2_d$, on définit la fusion de x^1_d avec x^2_d, notée $x^1_d \otimes x^2_d$, par :

$$x^1_d \otimes x^2_d = \Big(\bigcup_{\substack{(M,\Delta_1)\in x^1_d \\ \nexists (M,\Delta_2)\in x^2_d}} (M,\Delta_1)\Big) \cup \Big(\bigcup_{\substack{(M,\Delta_1)\in x^1_d \\ \nexists (M,\Delta_2)\in x^2_d}} (M,\Delta_2)\Big) \cup \Big(\bigcup_{\substack{(M,\Delta_1)\in x^1_d \\ (M,\Delta_2)\in x^2_d}} (M,\Delta_1 \cup \Delta_2)\Big)$$

$\diamond$

Par conséquent, on obtient $x^c_0 = (M_{init}, \Delta^1_0 \cup \Delta^2_0)$ où $(M_{init}, \Delta^1_0) = x^1_0$ et $(M_{init}, \Delta^2_0) = x^2_0$. Plus globalement, la fonction $\otimes$ permet de fusionner deux états x^1_d et x^2_d entre eux en regroupant au sein d'une même paire $(M, \Delta_1 \cup \Delta_2)$ tous les labels associés à M irrespectivement des états x^1_d et x^2_d.

La construction de $Diag_c$ repose pour sa part sur les langages respectifs des diagnostiqueurs $\mathcal{L}(Diag_1, x^1_0)$, $\mathcal{L}(Diag_2, x^2_0)$ et leur intersection. On définit l'opérateur $\circ$ de composition de deux diagnostiqueurs par :

Définition 6.10.2 (Composition de diagnostiqueurs)**.** Soient $Diag_1 = (X^1_d, E^1_d, f^1_d, x^1_0)$ et $Diag_2 = (X^2_d, E^2_d, f^2_d, x^2_0)$ deux diagnostiqueur. On définit la composition $Diag_c = (X^c_d, E^c_d, f^c_d, x^c_0) = Diag_1 \circ Diag_2$ par :

L'état initial est défini par $x^c_0 = x^1_0 \otimes x^2_0$

Algorithm 5: Algorithme de construction d'un diagnostiqueur $Diag_{\mathcal{P}}$

Entrées : $(N_\alpha, M_0^\alpha) = (P_\alpha, T_\alpha, F_\alpha, E_\alpha, l_\alpha, M_0^\alpha)$ avec $E_o = E^\alpha \cup E_+^\alpha \in E_\alpha$, $\Delta_{\mathcal{P}} = \{O_{\mathcal{P}}, \delta_{\mathcal{P}_1}, \delta_{\mathcal{P}_2}, \delta_{\mathcal{P}_3}\}$, M_{init}^α, tous les profils d'attaquant $\mathcal{P}$;
Sorties : $Diag_{\mathcal{P}}(N_\alpha, M_{init}^\alpha) = (X_d^{\mathcal{P}}, E_d^{\mathcal{P}}, f_d^{\mathcal{P}}, x_0^{\mathcal{P}})$;
if $M_{init}^\alpha|_{P_A^{G_\alpha} \cup P_R^{G_\alpha}} \in \mathcal{M}_{DZ}|_{P_A \cup P_R}$ **then**
  $x_0^{\mathcal{P}} = (M_{init}^\alpha|_{P^{G_\alpha}}, O_{\mathcal{P}})$; /* Si M_{init}^α est un état de la DZ dans G */
else
  $x_0^{\mathcal{P}} = (M_{init}^\alpha|_{P^{G_\alpha}}, \{\delta_{\mathcal{P}_1}, \delta_{\mathcal{P}_2}, \delta_{\mathcal{P}_3}\})$;
end if
$X_d^{\mathcal{P}} = \{x_0^{\mathcal{P}}\}$;
$X_d^{new} = \{\}$;
while $X_d^{new} \neq X_d^{\mathcal{P}}$ **do**
  $X_d^{new} = X_d^{\mathcal{P}}$; /* Tant que de nouveaux états sont découverts */
  for $x_d \in X_d^{\mathcal{P}}$ **do**
    for $e \in E_o$ **do**
      $x_d^{new} = \{\}$;

$\mathcal{M}_e = \{M \in \mathcal{R}(N_\alpha, M_0^\alpha) | \exists (M_1, \Delta_1) \in x_d, \Delta_1 \subset \Delta_a, t \in (l_\alpha)^{-1}((f_a^1)^{-1}(e)), M_1[t > M|_{P^{G_\alpha}}\}$;
      $\mathcal{M}_e = \mathcal{M}_e|_{P^{G_\alpha}}$; /* $\mathcal{M}_e$ est projeté sur les places de G dans N_α */
      for $M \in \mathcal{M}_e$ **do**
        /* M est précédé par la paire (M_1, Δ_1) */
        if $\Delta_1 \neq O_{\mathcal{P}}$ **then**
          /* Si l'état précédent M_1 est labellisé par des profils */
          $\Delta_e = \{\}$;
          for $\delta_{\mathcal{P}} \in \Delta_1$ **do**
            if $\delta_{\mathcal{P}} \in \Delta_1$ et pour λ_o la séquence d'événements observables de $M_{init}^\alpha|_{P^{G_\alpha}}$ à M_1, on a $\lambda_o e < P_o(\mathcal{P})$ **then**
              $\Delta_e = \{\Delta_e, \delta_{\mathcal{P}}\}$; /* Le label $\delta_{\mathcal{P}}$ est ajouté à Δ_e */
            end if
          end for
          if $\Delta_e = \emptyset$ **then**
            $\Delta_e = \{O_{\mathcal{P}}\}$; /* Si aucun profil n'est associé à l'état M */
          end if
          $x_d^{new} = \{x_d^{new}, (M, \Delta_e)\}$;
        end if
      end for
      if $x_d^{new} \notin X_d^{new}$ **then**
        $X_d^{new} = \{X_d^{new}, x_d^{new}\}$;
      end if
      $f_d(x_d, e) = x_d^{new}$; /* Un arc de x_d à x_d^{new} est créé et labellisé par e */
    end for
  end for
end while
Retourner $Diag_{\mathcal{P}}(N_\alpha, M_{init}^\alpha) = (X_d^{\mathcal{P}}, E_d^{\mathcal{P}}, f_d^{\mathcal{P}}, x_0^{\mathcal{P}})$;

L'ensemble des événements observables est défini par $E_d^c = E_d^1 \cup E_d^2$;

La fonction de transition f_d^c est définie par :

- $\forall \lambda \in \mathcal{L}(Diag_1, x_0^1) \setminus \mathcal{L}(Diag_2, x_0^2)$, on $f_d^c(x_0^c, \lambda) = f_d^1(x_0^1, \lambda) = x_d^1 \in X_d^c$;
- $\forall \lambda \in \mathcal{L}(Diag_2, x_0^2) \setminus \mathcal{L}(Diag_1, x_0^1)$, $f_d^c(x_0^c, \lambda) = f_d^2(x_0^2, \lambda) = x_d^2 \in X_d^c$;
- $\forall \lambda \in \mathcal{L}(Diag_1, x_0^1) \cap \mathcal{L}(Diag_2, x_0^2)$, $f_d^c(x_0^c, \lambda) = f_d^1(x_0^1, \lambda) \otimes f_d^2(x_0^2, \lambda) \in X_d^c$

Par conséquent, si $Diag_1$ et $Diag_2$ ont les mêmes langages, seule la dernière règle de composition s'applique ;

L'ensemble X_d^c des états du diagnostiqueur est alors défini à partir de f_d^c par : $X_d^c = \{f_d^1(x_0^1, \lambda) | \lambda \in \mathcal{L}(Diag_1, x_0^1) \setminus \mathcal{L}(Diag_2, x_0^2)\} \cup \{f_d^2(x_0^2, \lambda) | \lambda \in \mathcal{L}(Diag_2, x_0^2) \setminus \mathcal{L}(Diag_1, x_0^1)\} \cup \{f_d^1(x_0^1, \lambda) \otimes f_d^2(x_0^2, \lambda) | \lambda \in \mathcal{L}(Diag_1, x_0^1) \cup \mathcal{L}(Diag_2, x_0^2)\}$.

◇

Cette opération $\circ$ peut être généralisée au cas où les deux diagnostiqueurs ne sont pas calculés à partir d'un même marquage initial mais où $Diag_2$ est débuté à partir d'un marquage $M_2 \in \mathcal{R}(N, M_{init})$ tel que M_2 est atteignable depuis M_{init} via une séquence d'événements $\lambda \in E_d^{1\star}$, i.e. $M_{init}[\sigma > M_2$ et $l(\sigma) = \lambda$. Soit $f_d^1(x_0^1, \lambda) = x_d^1$ l'état atteint par $Diag_1$ après la séquence λ. L'état correspondant à x_d^1 dans $Diag_2$, soit l'état x_d^2, est obtenu par la fonction suivante :

Définition 6.10.3 (Fonction χ de conversion d'états entre deux diagnostiqueurs)**.** Soit $x_d^1 \in X_d^1$ un état du diagnostiqueur $Diag_1$ tel que $f(x_0^1, \lambda) = x_d^1$ avec $\lambda \in E_d^{1\star}$. Soit $Diag_2$, le second diagnostiqueur construit après l'observation de la séquence d'événements λ. Soit $\mathcal{M}_{x_d^1} = \{M | \exists (M, \Delta) \in x_d^1\}$ l'ensemble des marquage contenu dans x_d^1. On définit $UR(\mathcal{M}_{x_d^1}) = \bigcup UR(M) | M \in \mathcal{M}_{x_d^1}$ l'ensemble des marquages atteignables depuis un marquage de x_d^1 après une séquences $\lambda_{io} \in E_{io}^\star$ d'événements non observables. On définit la fonction χ par :

$$\chi_{1,2}(x_d^1) = \{(M, O) | M \in UR(\mathcal{M}_{x_d^1}), \nexists \delta \in \Delta_f \; et \; e_\delta \in \lambda_{io}\}$$
$$\cup \{(M, \Delta_M) | M \in UR(\mathcal{M}_{x_d^1}), \exists \{e_{\delta_1}, ..., e_{\delta_k}\} \in \lambda_{io} \; et \; \Delta_M = \{\delta_1, ..., \delta_k\} \subset \Delta_f\}$$

En d'autres termes, la fonction $\chi_{1,2}(x_d^1)$ est une application de l'étape d'initialisation de x_0^d dans l'algorithme de construction d'un diagnostiqueur(pseudo-code 3 : lignes 4 à 14) à chaque marquage de $UR(\mathcal{M}_{x_d^1})$.

◇

Enfin, à partir de $x_d^2 = \chi_{1,2}(x_d^1)$, le diagnostiqueur $Diag_2(N, x_d^2) = (X_d^2, E_d^2, f_d^2, x_d^2)$ peut être construit. La composition de $Diag_c(N, x_0^c) = Diag_1(N, x_0^1) \circ Diag_2(N, x_d^2)$ est définie par :

- $\forall \lambda_1 \in \mathcal{L}(Diag_1, x_0^1)$, si $\lambda < \lambda_1$, i.e. $\lambda_1 = \lambda\lambda_2, \lambda_2 \in \mathcal{L}(Diag_2, x_d^2)$, alors $f_d^c(x_0^c, \lambda) = f_d^1(x_0^1, \lambda) \otimes f_d^2(x_d^2, \lambda_2) \in X_d^c$,
- Sinon $f_d^c(x_0^c, \lambda) = f_d^1(x_0^1, \lambda) \in X_d^c$;
- De plus, si $\exists \lambda_2 \in \mathcal{L}(Diag_2, x_d^2)$ tel que $\lambda_3 = \lambda\lambda_2 \notin \mathcal{L}(Diag_1, x_0^1)$, alors $f_d^c(x_0^c, \lambda_3) = f_d^2(x_d^2, \lambda_2) \in X_d^c$;
- Notons que si $\lambda = \{\varepsilon\}$, alors $f_d(x, \lambda) = x$.

En prenant $x_d^2 = x_0^2$, on retrouve le résultat précédemment énoncé pour la composition de deux diagnostiqueurs débutés à partir d'un même état M_{init}.

6.11 Diagnostiqueur commun $Diag_c^i$

Soit $Diag_c^i(N, x_0^{c_i}) = (X_d^{c_i}, E_d^{c_i}, f_d^{c_i}, x_0^i)$ la $i^{ème}$ itération du diagnostiqueur commun construit à partir de l'algorithme de $Diag_a$ (annexe 6.8), de celui de $Diag_{\mathcal{P}}^i$ (annexe 6.9) et du diagnostiqueur de l'itération précédente $Diag_c^{i-1}$. L'algorithme de construction du diagnostiqueur $Diag_c^i$ est détaillé dans le pseudo-code 6 ci-dessous. Préliminairement, la restriction du diagnostiqueur $Diag_c^{i-1}$ aux pairs (M, Δ) contenant des labels d'attaques est définie. Cette restriction est implémentée dans l'algorithme 6 puisque seules ces paires sont requises pour le diagnostic dans $Diag_c^i$ de profils d'attaquants hérités de $Diag_c^{i-1}$.

Définition 6.11.1 (Restriction d'un diagnostiqueur à un ensemble de labels Δ_r)**.** La restriction d'un diagnostiqueur $Diag(N, x_0^d) = (X_d, E_d, f_d, x_0^d)$ à un ensemble de label $\Delta_r \subset \Delta$ est notée $Diag(N, x_0^d|_{\Delta_r})|_{\Delta_r} = (X_d^{\Delta_r}, E_d^{\Delta_r}, f_d^{\Delta_r}, x_0^d|_{\Delta_r})$ et est définie par :

$X_d^{\Delta_r} = \{x_{\Delta_r} = \{(M, \Delta_M) \in x | \Delta_M \subset \Delta_r\} | x \in X_d, \exists \lambda \in \mathcal{L}(Diag, x_0^d), f_d(x_0^d, \lambda) = x \text{ et } \exists (M, \Delta_M) \in x, \Delta_M \subset \Delta_r\}$. Les états de X_d sont restreints à leurs pairs dont les labels sont inclus dans Δ_r. Si aucune paire d'un état x_d ne respecte cette contrainte de restriction, x_d n'est pas considéré dans $X_d^{\Delta_r}$;

$E_d^{\Delta_r} = E_d$;

$f_d^{\Delta_r}(x_{\Delta_r}, \lambda) = x'_{\Delta_r}$ si $f_d(x, \lambda) = x'$ existe, $x_{\Delta_r}, x'_{\Delta_r} \in X_d^{\Delta_r}$ et $x, x' \in X_d$;

$x_0^d|_{\Delta_r}$ est défini de la même manière que les états de $X_d^{\Delta_r}$ à partir de x_0^d.

◇

Dans cet algorithme, $Diag_a^i$ est tout d'abord calculé ligne 3 à partir de σ_f^i et du marquage initial (M_{init}, R_{init}) du FMS puis est composé avec le diagnostiqueur précédent $Diag_c^{i-1}$ restreint à ses états atteignables depuis x_c^{i-1} et à l'ensemble de ses labels de profils d'attaquant, soit $Diag_c^{i-1}(N, x_c^{i-1}|_{\Delta_{\mathcal{P}}})|_{\Delta_{\mathcal{P}}}$. Enfin, de manière itérative entre les lignes 6 à 11, un diagnostiqueur $Diag_{\mathcal{P}_k}$ construit à partir des 3 profils $\mathcal{P}_1((M_k, R_k)), \mathcal{P}_2((M_k, R_k)), \mathcal{P}_3((M_k, R_k))$ (ligne 8-9) est intégré à $Diag_c^i$ pour chaque état (M_k, R_k) traversé par l'ordonnancement σ_f^i (ligne 10). La fonction χ est ici requise pour le calcul de $Diag_{\mathcal{P}_k}$ afin d'obtenir son état initial $x_0^{\mathcal{P}_k}$ à partir de l'état x_c^k dans $Diag_c^i$ correspondant à M_k, soit $x_0^{\mathcal{P}_k} = \chi_{c,\mathcal{P}_k}(x_c^k)$ (ligne 7). Pour chaque état (M_k, R_k) traversé par l'ordonnancement σ_f^i, les labels du diagnostiqueur des profils d'attaquant $Diag_{\mathcal{P}_k}$ sont $\Delta_{\mathcal{P}_k} = \{O_{M_k}, \delta_{\mathcal{P}_1}^{M_k}, \delta_{\mathcal{P}_2}^{M_k}, \delta_{\mathcal{P}_3}^{M_k}\}$, et l'ensemble de tous les labels des différents $Diag_{\mathcal{P}_k}, k \in [1, |\sigma_f^i|]$ est noté $\Delta_{\mathcal{P}} = \bigcup \Delta_{\mathcal{P}_k}$.

L'algorithme $Diag_c^i$ est appelé et exécuté à l'initialisation du module de diagnostic et dès que la condition Cn_d est vérifiée lors de l'observation d'une nouvelle décision d'allocation.

6.12 Algorithme de calcul des conséquences de cascade d'indisponibilité

L'algorithme de calcul de toutes les conséquences de cascades d'indisponibilités à partir d'un changement de mode $Mode_{\mathcal{R}_i,\mathcal{R}_j}(M)$ est présenté dans l'algorithme 7 ci-dessous. Préalablement, les deux définitions suivantes sont nécessaires.

Algorithm 6: Algorithme de construction de $Diag_c^i$

Entrées : $(N_\alpha, M_0^\alpha) = (P_\alpha, T_\alpha, F_\alpha, E_\alpha, l_\alpha, M_0^\alpha)$, $M_{init} \in \mathcal{R}(N_\alpha, M_0^\alpha)$, R_{init} la matrice des délais restant, $Diag_c^{i-1} = (X_d^{c_{i-1}}, E_d^{c_{i-1}}, f_d^{c_{i-1}}, x_0^{c_{i-1}})$ le diagnostiqueur de l'itération précédente, x_c^{i-1} l'état actuel de $Diag_c^{i-1}$ et $\sigma_f^i, \lambda_f^i = l(\sigma_f^i)$ l'ordonnancement calculé à partir de 0, les caractéristiques des différents profils;

Sorties : $Diag_c^i(N_\alpha, x_0^{c_i}) = (X_d^{c_i}, E_d^{c_i}, f_d^{c_i}, x_0^{c_i})$;

$Diag_c^i(N_\alpha, x_0^{c_i}) = Diag_a^i(N_\alpha, M_{init}) \circ Diag_c^{i-1}(N_\alpha, x_c^{i-1}|_{\Delta_\mathcal{P}})|_{\Delta_\mathcal{P}}$; /* $Diag_c^{i-1}(N_\alpha, x_c^{i-1}|_{\Delta_\mathcal{P}})|_{\Delta_\mathcal{P}}$ est la restriction à partir de x_c^{i-1} des états du diagnostiqueur précédent $Diag_c^{i-1}$ aux pairs (M, Δ) contenant des labels de profils d'attaques */

$x_0^{a_i} = \{(M_{init}|_{PG_\alpha}, O)\}$;

$x_0^{c_i} = x_0^{a_i} \otimes x_c^{i-1}|_{\Delta_\mathcal{P}}$;

$(M_{init}, R_{init}), (M_2, R_2), ..., (M_{|\sigma_f|}, M_{|\sigma_f^i|})$; /* Séquence d'états traversés par N_α lors de l'exécution de σ_f^i depuis (M_{init}, R_{init}) */

for $M_k \in \{M_{init}, M_2, ..., M_{|\sigma_f|}\} | M_{init}[\sigma > M_k, l(\sigma) = \lambda < \lambda_f^i$ **do**

  $x_c^k \in X_d^{c_i} | f(x_0^{c_i}, \lambda) = x_c^k$; /* x_c^k est l'état correspondant à M_k dans $diag_c^i$, atteint après la séquence λ */

  $\mathcal{P}_1((M_k, R_k)), \mathcal{P}_2((M_k, R_k)), \mathcal{P}_3((M_k, R_k))$; /* Les trois profils d'attaquants étudiés sont calculés depuis (M_k, R_k) */

  $Diag_{\mathcal{P}_k}(N, \chi_{c,\mathcal{P}_k}(x_c^k))$; /* On note $\Delta_{\mathcal{P}_k} = \{O_{M_k}, \delta_{\mathcal{P}_1}^{M_k}, \delta_{\mathcal{P}_2}^{M_k}, \delta_{\mathcal{P}_3}^{M_k}\}$, et $\Delta_\mathcal{P} = \bigcup \Delta_{\mathcal{P}_k}$ */

  $Diag_c^i(N, x_0^{c_i}) = Diag_c^i(N, x_0^{c_i}) \circ Diag_{\mathcal{P}_k}(N, \chi_{c,\mathcal{P}_k}(x_c^k))$;

end for

Retourner $Diag_c^i(N, x_0^{c_i})$;

Définition 6.12.1 (Fonction T_f)**.** Pour un graphe orienté $(V, F) = \mathcal{RG}(N, M)$, soit $T_f : F \to T$ la fonction associant à chaque arc de F la transition de T franchie dans N.

Cette fonction est extensible à une séquence d'arcs de $F^\star$ vers une séquence de transitions de $T^\star$. ◇

Définition 6.12.2 (Projection d'un état M de N à $N_{\mathcal{R}_k}$)**.** Soit $M|_{P_{\mathcal{R}_k}}$ la projection de $M \in \mathcal{R}(N, M_0)$ sur $P_{\mathcal{R}_k}$ définie par :

$\forall r_k \in R_k$, si $\exists p_a \in H_{r_k} | M(p_a) = 1$ alors $M|_{P_{\mathcal{R}_k}}(p_a) = 0$ et $M|_{P_{\mathcal{R}_k}}(p^{ind}_{p_a}) = 1$, sinon $M|_{P_{\mathcal{R}_k}}(p_k) = 0$ et $M|_{P_{\mathcal{R}_k}}(p^{ind}_k)$ avec $p^{ind}_{p_a}$ et p^{ind}_k les places d'indisponibilité associées à la place activité p_a et à la place ressource de r_k.

Ainsi, les places d'indisponibilité des ressources de $\mathcal{R}_k$ sont marquées selon si la ressource est en opération ou hors-opération dans M. ◇

L'algorithme présenté dans cet annexe est inspiré des travaux de Liu [150] et de l'étude des composantes fortement connexes du graphe des marquages accessibles $\mathcal{RG}(N_{\mathcal{R}_k}, M|_{P_{\mathcal{R}_k}})|_{\rho'}$ calculé à partir du RdP incluant les réseaux de reprise modélisant l'indisponibilité des ressources $r_k \in \mathcal{R}_k$ et du marquage M projeté à ce modèle par $M|_{P_{\mathcal{R}_k}}$. La politique de contrôle ρ' est appliquée à l'exploration de $\mathcal{RG}(N_{\mathcal{R}_k}, M|_{P_{\mathcal{R}_k}})|_{\rho'}$ afin d'interdire le franchissement des transitions de reprises $t^{dis}_{p_a} \in (p^{ind}_{p_a})^\bullet | p_a \in H_{\mathcal{R}_k}, M|_{P_{\mathcal{R}_k}}(p^{ind}_{p_a}) = 1$ et $t^{dis}_{p_k} \in (p^{ind}_k) \bullet | r_k \in \mathcal{R}_k, M|_{P_{\mathcal{R}_k}}(p^{ind}_k) = 1$ relatives aux ressource de $\mathcal{R}_k$. Au sein du $mode_{\mathcal{R}_k}$, ces transitions ne sont en effet pas franchissables en raison des indisponibilités des ressources de $\mathcal{R}_k$ mais peuvent le redevenir en cas de changement de mode.

Ainsi, la ligne 6 de l'algorithme introduit l'ensemble des composantes fortement connexes de $(V_r, F_r) = \mathcal{RG}(N_{\mathcal{R}_k}, M|_{P_{\mathcal{R}_k}})|_{\rho'}$ calculés dans ces travaux à l'aide de l'algorithme de Kosaraju [375]. Dans cet algorithme, une composante fortement connexe CFC_i correspond à un ensemble d'états dont l'accessibilité des uns avec les autres est obtenus grâce un sous-RdP vivant de $N_{\mathcal{R}_k}|_{\rho'}$ pour un ensemble $T_f(CFC_i) \subset T$ de transitions vivantes. Dans un S^{3}PR, ce sous-RdP vivant se compose d'un ensemble de circuits non-bloqués et exploitables pour la réalisation tout de même de recettes.

Puis, la boucle *for* de cet algorithme (ligne 7 à 22) cherche alors à identifier pour chaque CFC_i de $\mathcal{RG}(N_{\mathcal{R}_k}, M|_{P_{\mathcal{R}_k}})|_{\rho'}$ les ressources indisponibles de $\mathcal{R}_k$ combinées aux ressources devenues indisponibles par cascades d'indisponibilité en calculant $\mathcal{R}'_k$ (ligne 11 à 18), et les places bloquées hors ressources indisponibles appartenant à P'_{blck} (ligne 20).

Précisément, la boucle *while* entre les lignes 14 et 18 permet de trouver toutes les ressources bloquées $\mathcal{R}'_k$, par indisponibilité ou par cascade, en explorant l'ensemble $P^{csc'}_i$. Cet ensemble regroupe les places activité marquées ne requérant pas de ressources bloquées de $\mathcal{R}'_k$ mais précédant $(P^{pre}_{\mathcal{R}'_k} \setminus H_{\mathcal{R}'_k})$ ou succédant $(P^{post}_{\mathcal{R}'_k})$ des places de $H_{\mathcal{R}'_k}$ requérant ces ressources. Dans la boucle while, $\mathcal{R}'_k$ est mis à jour par l'ajout des ressources requises par les places de $P^{csc'}_i$, puis $P^{csc'}_i$ est recalculée et le boucle se répète jusqu'à ce qu'aucune nouvelle place activité ne puisse être ajoutée à $P^{csc'}_i$. Dans cette boucle, les ressources requises par les places de $P^{csc'}_i$ sont les ressources devenues indisponibles par cascades d'indisponibilités. Dans la définition de $P^{csc'}_i$,

Algorithm 7: Algorithme de calcul des conséquences de cascades d'indisponibilités

Entrées : N, $\mathcal{R}_k$, M ;
Sorties : Csc;
$Csc = \{\}$;
$N_{\mathcal{R}_k} = (P_{\mathcal{R}_k}, T_{\mathcal{R}_k}, F_{\mathcal{R}_k})$; /* Le modèle $N_{\mathcal{R}_k}$ incluant les réseaux de reprise des ressource de $\mathcal{R}_k$ est construit (1.2.2) */
$(V_r, F_r) = \mathcal{RG}(N_{\mathcal{R}_k}, M|_{P_{\mathcal{R}_k}})|_{\rho'}$; /* ρ' est une politique de contrôle empêchant le franchissement des transitions de reprise */
$CFC = \bigcup CFC_i$, $CFC_i = \{v_1, ..., v_i\} \subset V_r$; /* CFC désigne l'ensemble des composantes fortement connexes de (V_r, F_r) */
for $CFC_i \in CFC$ **do**
  $F_i = \{(v_1, v_2) \in F_r | v_1, v_2 \in CFC_i\}$; /* F_i est l'ensemble des arcs reliant des nœuds de CFC_i */
  $T_f(CFC_i) = \{t \in T | \exists (v_1, v_2) \in F_i, T_f((v_1, v_2)) = t\}$; /* Ensemble des transitions franchies dans CFC_i */
  $T_{nf} = T \setminus T_f(CFC_i)$; /* Ensemble des transition non franchies dans CFC_i */
  $P_i^{csc} = \{p \in P_{\mathcal{R}_k}^{pre} \setminus H_{\mathcal{R}_k} | \forall M = v \in CFC_i, M(p) = 1\} \cup \{p \in P_{\mathcal{R}_k}^{post} | \forall M = v \in CFC_i, M(p) = 1 \; et \; \exists M = v \in V_r, M(p) = 0\}$; /* Ensemble des places toujours marquées dans CFC_i de $P_{\mathcal{R}_k}^{pre}$ et $P_{\mathcal{R}_k}^{post}$ hors $H_{\mathcal{R}_k}$ */
  $\mathcal{R}'_k = \mathcal{R}_k \cup \{r | p_r \in {}^{\bullet\bullet}P_i^{csc} \cap P_R\}$; /* Union de $\mathcal{R}_k$ avec les ressources requises par les places de P_i^{csc} */
  $P_i^{csc'} = \{p \in P_{\mathcal{R}'_k}^{pre} \setminus H_{\mathcal{R}'_k} | \forall M = v \in CFC_i, M(p) = 1\} \cup \{p \in P_{\mathcal{R}'_k}^{post} | \forall M = v \in CFC_i, M(p) = 1 \; et \; \exists M = v \in V_r, M(p) = 0\}$;
  while $P_i^{csc} \neq P_i^{csc'}$ **do**
    $P_i^{csc} = P_i^{csc'}$;
    $\mathcal{R}'_k = \mathcal{R}'_k \cup \{r | p_r \in {}^{\bullet\bullet}P_i^{csc} \cap P_R\}$;
    $P_i^{csc'} = P_i^{csc'} \cup \{p \in P_{\mathcal{R}'_k}^{pre} \setminus H_{\mathcal{R}'_k} | \forall M = v \in CFC_i, M(p) = 1\} \cup \{p \in P_{\mathcal{R}'_k}^{post} | \forall M = v \in CFC_i, M(p) = 1 \; et \; \exists M = v \in V_r, M(p) = 0\}$;
  end while
  $P'_{blck} = \{p \in P_A \cap {}^{\bullet}T_{nf} | p \notin P_{\mathcal{R}'_k}^{pre}\}$; /* P'_{blck} est l'ensemble des places activités bloquées de CFC_i n'appartenant pas à $P_{\mathcal{R}'_k}^{pre}$ */
  $M_{rc} = v_1 \in CFC_i, M|_{N_{\mathcal{R}_k}}[e_1^\star > v_1 | \nexists v_2 \in CFC_i, M|_{N_{\mathcal{R}_k}}[e_2^\star > v_2, |v_2| < |v_2|, e_1^\star, e_2^\star \in F^\star$; /* M_{rc} est appelé l'état racine de CFC_i */ ;
  $Csc = \{Csc, (\mathcal{R}'_k, M_{rc}, P'_{blck})\}$; /* La nouvelle conséquence de CFC_i est ajoutée à Csc */
end for

des places marquées de $P^{post}_{\mathcal{R}'_k}$ succédant des places de $H_{\mathcal{R}'_k}$ peuvent créer des effets de cascades d'indisponibilités si elle n'ont pas été vidées préalablement dans le CFC_i alors qu'elles auraient pu l'être (c.f. condition $\exists M = v \in V_r, M(p) = 0$). Cette condition permet d'assurer que ces places marquées ne sont pas bloquées en raison d'un blocage partiel inévitable mais bien en raison de décisions d'allocation bloquantes.

A la ligne 19, P'_{blck} représente les places activités bloquées non en raison d'une ressource indisponible ou devenue indisponible par cascade d'indisponibilité, i.e. $P'_{blck} \cap P^{pre}_{\mathcal{R}'_k} = \emptyset$. Les places de P'_{blck} sont soit des places activités non marquées de $P^{post}_{\mathcal{R}'_k}$ succédant des places de $P^{pre}_{\mathcal{R}'_k}$ dans des circuits bloqués de N, soit correspondent à des places activités bloquées suite à un blocage partiel apparu dans N_{NB} à cause d'une perte de flexibilité et le blocage de circuits parallèles. Par exemple, dans la figure 1.6, si la ressource R_1 devient indisponible et que les places p_7 et p_{15} sont marquées, un état de blocage sera systématiquement atteint lorsque les paires de places (p_7, p_{18}), (p_{12}, p_{17}), (p_{13}, p_{16}) ou (p_{14}, p_{15}) sont marquées. Dans le cas sans indisponibilité, ce blocage aurait été évité grâce au chemin parallèle $p_8p_9p_{10}$.

Enfin, à la ligne 20 de l'algorithme est défini le marquage "racine" M_{rc} de la composante fortement connexe CFC_i, soit le premier marquage appartenant à CFC_i atteint depuis le marquage initial $M|_{P_{\mathcal{R}_k}}$ dans $\mathcal{RG}(N_{\mathcal{R}_k}, M|_{P\mathcal{R}_k})|_{\rho'}$.

Pour chaque CFC_i, le triplet $(\mathcal{R}'_k, M_{rc}, P'_{blck})$ est ajouté à Csc (ligne 21) afin d'être retourné à la fin de l'algorithme, définissant ainsi les caractéristiques des conséquences de cascades d'indisponibilités au sein de CFC_i. Ainsi, pour un changement de mode $(M, Mode_{\mathcal{R}_k})$, **l'algorithme est capable de calculer la totalité des conséquences de cascades d'indisponibilités pouvant avoir lieu et ne bloquant par l'entièreté des recettes**. En effet, toutes les CFCs de $\mathcal{RG}(N_{\mathcal{R}_k}, M|_{P_{\mathcal{R}_k}})|_{\rho'}$ sont étudiées dans cet algorithme et elles représentent tous les comportements vivants que peut avoir un FMS à la suite de ce changement de mode $(M, Mode_{\mathcal{R}_k})$. A l'inverse, si CFC est vide, le FMS est entièrement bloqué suite au changement de mode.

6.13 Algorithme $A^\star\mathcal{P}$ dans un contexte incertain

Dans cette annexe, l'algorithme $A^\star\mathcal{P}$ présenté précédemment (annexe 6.6) est étendu au calcul des profils d'attaquant dans un contexte incertain. Le pseudo-code de cet algorithme est introduit ci-dessous (algorithme 8) et ses nouveautés sont décrites subséquemment.

Dans cette nouvelle version de l'algorithme $A^\star\mathcal{P}$ (algorithme 2, annexe 6.6), l'ensemble T_{fr} des transitions franchissables par l'attaquant est étendu aux transitions des attaque d'insertion des événements d'observation de défaillance(T^{ind}_+) et de reprise(T^{dis}_+) d'une ressource (ligne 16). Puis, cette ensemble T_{fr} est redéfinie à la ligne 20 si il existe des décisions d'allocation instantanées depuis (M, R) devant être supprimées. L'ensemble des transitions de suppression de ces décision instantanées est défini par T_0 (ligne 17). Une de ces décisions instantanées prises par le module de supervision libère et détient deux nouvelles ressources et modifie par conséquent les attaques d'indisponibilités contre ces ressources selon si elles sont en opération ou hors-opération, soient l'ensemble T^{ind}_+. Enfin, sont ôtés de T_{fr} à la ligne 24 les transitions de T^{ind}_+ menant à un mode de fonctionnement bloquant l'entièreté du FMS. Pour cette dernière modification de T_{fr}, la fonction $MODE(M)$ est définie avec l'objectif de déterminer le mode dans lequel est l'état M à partir du marquage de ses places d'indisponibilité.

Algorithm 8: Algorithme de recherche $A^\star\mathcal{P}$ dans un contexte incertain

Entrées : $(M_{init}, R_{init}) \in \mathcal{R}(N^\alpha_{\mathcal{R}_{ind}}, M_0^{\mathcal{R}_{ind}\alpha})$ l'état initial, $\mathcal{M}_c \in \mathcal{R}(N, M_0)$ l'ensemble des états de G ciblés par l'attaquant, $N^\alpha_{\mathcal{R}_{ind}}$, *Contraintes* les contraintes du profil, σ_f l'ordonnancement à partir de (M_{init}, R_{init}), c et h la fonction coût et l'heuristique du profil, Tri les consignes de tri du profil hors coût, K_{max} la profondeur de la recherche $A^\star BT$;

Sorties : $\mathcal{P}$ le profil d'attaquant, $\sigma_a = l_\alpha^{-1}(\mathcal{P})$ la séquence de transition correspondante dans $T^{\alpha\star}_{\mathcal{R}_{ind}}$;

$\Upsilon((M_{init}, R_{init}), \sigma_f) = 1$;

$OPEN = (M_{init}, R_{init})$;

$CLOSED = ()$; /* $OPEN$ et $CLOSED$ sont deux listes ordonnées */

$F_\mathcal{P} = \{\}$; /* $F_\mathcal{P}$ est un ensemble d'arcs */

$\sigma_f^{(M_{init}, R_{init})} = \sigma_f$; /* L'ordonnancement associé à (M_{init}, R_{init}) est égal à σ_f */

while $OPEN \neq \emptyset$ **do**

  $(M, R) = OPEN(1)$; /* On étudie le premier état (M, R) de $OPEN$ */

  $OPEN = OPEN \setminus OPEN(1)$; /* On ôte (M, R) de $OPEN$ */

  $CLOSED = (CLOSED, (M, R))$; /* On ajoute (M, R) à $CLOSED$ */

  if $M|_{(P_A^G \cup P_R^G)} \in \mathcal{M}_c|_{(PA \cup P_R)}$ **then**

    Retourner σ_a et $\mathcal{P} = l_\alpha(\sigma_a)$ tel que $(M_{init}, R_{init})[\sigma_a > (M, R)$;

  end if

  $\sigma_f = \sigma_f^{(M,R)}$; /* L'ordonnancement σ_f considéré est celui associé à (M, R) */

  $T_{fr} = L(N_\alpha, M)|_1 \cap (T_+^\alpha \cup (T_+^{ind} \cup T_+^{dis}) \cup (T \cap \sigma_f(\Upsilon((M, R)), \sigma_f)))$; /* T_{fr} est l'ensemble des transitions franchissables dans N_α à partir de M */

  $T_0 = (t_-^1, t_-^2, ..., t_-^n) | (t_-^1, t_-^2, ..., t_-^n) \in T_-^\alpha$ *et* $\forall t_- \in T-, \tau_\alpha(t_-) \in \sigma_f(\Upsilon((M, R), \sigma_f) \rightarrow fin), (M'', R'', \Gamma'') = \Omega_{N^\alpha_{\mathcal{R}_{ind}}}(t_-^1 t_-^2 ... t_-, M, R, 0)$, *on a* $\Gamma'' = 0$; /* T_0 est l'ensemble des transitions des décision d'allocations instantanées depuis (M, R) */

  if $T_0 \neq \emptyset$ **then**

    $(M'', R'', \Gamma'') = \Omega_{N^\alpha_{\mathcal{R}_{ind}}}(t_-^1 t_-^2 ... t_-^n, M, R, 0)$;

    $T_{fr} = L(N_\alpha, M'')|_1 \cap (T_+^\alpha \cup (T_+^{ind} \cup T_+^{dis}) \cup (T \cap \sigma_f(\Upsilon((M, R), \sigma_f) + |T_0|))$; /* Si plusieurs prochaines décision d'allocation sont instantanées, T_{fr} est recalculé puisque les transitions de $(T_+^{ind} \cup T_+^{dis})$ franchissables changent */

  end if

  for $t \in (T_{fr} \cap T_+^{ind})$ **do**

    if $N_{NB}^C(MODE(M'), M') = \emptyset, M[t > M'$ **then**

      $T_{fr} = T_{fr} \setminus t$; /* Si une attaque d'indisponibilité d'une ressource conduit à bloquer totalement le FMS, elle est ôtée de T_{fr} */

    end if

  end for

  for $t \in T_{fr}$ **do**

    $(M', R', \Gamma') = \Omega_{N^\alpha_{\mathcal{R}_{ind}}}(t, M, R, 0)$; /* Franchissement de t depuis (M, R) */

    $T- = t_-^1 t_-^2 ... t_-^n | (t_-^1, t_-^2, ..., t_-^n) \in T_-^\alpha$ *et* $\forall t_- \in T-, \tau_\alpha(t_-) \in \sigma_f(\Upsilon((M, R), \sigma_f) \rightarrow fin), (M'', R'', \Gamma'') = \Omega_{N^\alpha_{\mathcal{R}_{ind}}}(t_-^1 t_-^2 ... t_-, M, R, 0)$, *on a* $\Gamma'' \leq \Gamma'$; /* $T-$ est la séquence des transitions de suppression des décisions d'allocation successifs de σ_f depuis (M, R) ayant lieu avant ou simultanément à $t \in T_{fr}$ */

30: **if** $T- \neq \emptyset$ et $t \neq \tau_{\alpha}(T-)$ **then** /* Si la séquence $T-$ est non vide */
/* Si la prochaine transition de l'ordonnancement σ_f est franchie avant ou en même temps que t et doit être supprimée via t_- */
31: $(M_1'', R_1'') = (M, R)$; /* Un état intermédiaire (M_1'', R_1'') est créé */
32: **for** $t_- \in T-$ **do**
33: $(M'', R'', \Gamma'') = \Omega_{N^{\alpha}_{\mathcal{R}_{ind}}}(t_-, M_1'', R_1'', 0)$; /* Franchissement de t_- depuis (M_1'', R_1'') */
34: $g(M'', R'') = g(M_1'', R_1'') + c((M_1'', R_1''), (M'', R''))$; /* Coût passé de (M'', R'') */
/* Si l'état (M'', R'') appartient déjà à OPEN ou CLOSED et possède un meilleur coût passé */
35: **if** $\exists (M^o, R^o) \in OPEN | M^o = M'', R^o = R''$ *et* $g(M'', R'') < g(M^o, R^o)$ **then**
36: $OPEN = OPEN \setminus (M^o, R^o)$;
37: $OPEN = (OPEN, (M'', R'')$;
38: **else if** $\exists (M^c, R^c) \in CLOSED | M^c = M'', R^c = R''$*et* $g(M'', R'') < g(M^c, R^c)$ **then**
39: $CLOSED = CLOSED \setminus (M^c, R^c)$;
40: $OPEN = (OPEN, (M'', R'')$;
41: **end if**
42: $F_{\mathcal{P}} = \{F_{\mathcal{P}}, ((M_1'', R_1''), (M'', R''))\}$; /* Un arc de (M_1'', R_1'') à (M'', R'') est créé */
43: $\Upsilon(M'', R'') = \Upsilon(M_1'', R_1'') + 1$; /* Actualisation de Υ, une nouvelle décision d'allocation a eu lieu et a été supprimée */
44: $(M_1'', R_1'') = (M'', R'')$; /* L'état intermédiaire devient égal à (M'', R'') pour la prochaine transition $t_- \in T-$ */
45: $g(M_1'', R_1'') = g(M'', R'')$;
46: **end for**
47: $(M', R', \Gamma' - \Gamma'') = \Omega_{N^{\alpha}_{\mathcal{R}_{ind}}}(t, M'', R'', \Gamma'')$; /* Franchissement de t depuis (M'', R'') pour atteindre un nouveau (M', R') */
48: $F_{\mathcal{P}} = \{F_{\mathcal{P}}, ((M'', R''), (M', R'))\}$; /* Un arc de (M'', R'') et (M', R') est créé */
49: $\Upsilon(M', R') = \Upsilon(M'', R'') = \Upsilon(M, R) + |T - |$; /* Actualisation de Υ */
50: $g(M', R') = g(M, R) + c((M, R), (M'', R'')) + c((M'', R''), (M', R'))$;
51: $f(M', R') = g(M', R') + h(M', R')$; /* g, h et f sont calculés selon la fonction coût et l'heuristique propres au profil */
/* Sinon, si t est franchie avant la prochaine transition de l'ordonnancement σ_f */
52: **else**
53: $F_{\mathcal{P}} = \{F_{\mathcal{P}}, ((M, R), (M', R'))\}$; /* Un arc entre (M, R) et (M', R') est créé */
54: $\Upsilon(M', R') = \Upsilon(M, R)$; /* Aucune transition de l'ordonnancement n'a été franchie */
55: $g(M', R') = g(M, R) + c((M, R), (M', R'))$;
56: $f(M', R') = g(M', R') + h(M', R')$;
57: **end if**

if $\exists Ctr \in CTR | Ctr = vraie$ **then**
  $(M', R') = \emptyset$; /* Si une contrainte est vérifié en (M', R'), l'état est interdit et n'est pas visité par l'algorithme */
else if $\exists (M^o, R^o) \in OPEN | M^o = M', R^o = R'$ *et* $g(M', R') < g(M^o, R^o)$ **then**
  $OPEN = OPEN \setminus (M^o, R^o)$;
  $OPEN = (OPEN, (M', R')$;
else if $\exists (M^c, R^c) \in CLOSED | M^c = M', R^c = M' et\ g(M', R') < g(M^c, R^c)$ **then**
  $CLOSED = CLOSED \setminus (M^c, R^c)$;
  $OPEN = (OPEN, (M', R')$;
else
  $OPEN = (OPEN, (M', R'))$;
end if
if $MODE(M') \neq MODE(M)$ **then**
  $N_{NB}^{C-t}(MODE(M'), M')$; /* Si une attaque de changement de mode a eu lieu, N_{NB}^{C-t} est calculé depuis le nouveau mode (4.1.2) */
  $R' = 0$; /* R' est nul à la suite du changement de mode */
  $\sigma_f^{(M',R')} = A^\star BT(M'|_{P_{NB}^{C-t}}, R'|_{P_{NB}^{C-t}}, N_{NB}^{C-t}, K_{max})$; /* L'ordonnancement associé à (M', R') est calculé depuis le nouveau modèle N_{NB}^{C-t} */
  $\Upsilon(M', R') = 1$;
end if
end for
Trier la liste $OPEN$ selon les valeurs croissantes de $f(M, R), (M, R) \in OPEN$ et selon TRI, les autres consignes de tri du profil ;
end while
Retourner "Erreur' ;

Cette fonction $MODE(M)$ est de nouveau utilisée à la fin de l'algorithme afin de déterminer si un changement de mode a eu lieu entre M et M'. Si un tel changement a eu lieu, le modèle N_{NB}^{C-t} est construit à partir de la paire $(MODE(M'), M')$ (ligne 70) dans le but de calculer un nouvel ordonnancement depuis (M', R') avec $R' = 0$ (lignes 71-72). La fonction $\Upsilon(M', R')$ est alors réinitialisée à 1 (ligne 73). Ce nouvel ordonnancement sera celui pris en compte ultérieurement (ligne 15) pour la suppression de décision d'allocation ayant lieu après (M', R').

6.14 Algorithme de construction du diagnostiqueur des changements de mode

Dans cette annexe, l'algorithme de construction du diagnostiqueur des changements de mode $Diag_m$ est présenté. Le pseudo-code de cet algorithme est introduit ci-dessous (algorithme 9) et ses nouveautés sont décrites subséquemment.

Cette algorithme se fonde sur l'algorithme de construction d'un diagnostiqueur présenté précédemment (annexe 6.7).

Dans un premier temps, l'état initial de $Diag_m$, soit x_0^m, est défini parallèlement à un second état $x_m^{\in} X_m$. Ce second état représente les changements de mode conduisant à un blocage complet du FMS. Ainsi, cet état est composé uniquement du marquage M_{init} et est connecté à x_m^0 par un arc labellisé ε (ligne 6). Puis, pour chaque mode $Mode_{\mathcal{R}_j}$ atteignable depuis le mode initial $Mode_{\mathcal{R}_i}$ (ligne 4), le label $\delta_{Mode_{\mathcal{R}_j}}^{M_{init}}$ est associé à M_{init} dans x_0^m (ligne 9). Ce label est aussi associé à M_{init} dans x_m^{ε} (ligne 11) si l'ordonnancement $\sigma(M_{init}, Mode_{\mathcal{R}_j})$ (ligne 8) calculé depuis la paire $(M_{init}, Mode_{\mathcal{R}_j})$ est vide (ligne 10), soit si un état de blocage ou de pré-blocage est atteint lors du changement de mode vers $Mode_{\mathcal{R}_j}$.

Dans un second temps, le diagnostiqueur est construit par l'exploration des marquages de N_t. Dans cet algorithme, les marquages de N_t sont retenus et ajoutés aux états du diagnostiqueur au détriment des marquages du S^{3}PR N afin de correctement distinguer les marquages successifs au sein des ordonnancements $\sigma(M_{init}, Mode_{\mathcal{R}_j})$. En effet, sans distinction du marquage des places d'entrée et de sortie de N_t, un marquage de N peut être traversé plusieurs fois au sein d'un même ordonnancement. Ainsi, $\mathcal{M}_e$ (annexe 6.7) est calculé ligne 21 à partir de marquages non temporisés de N_t et non de N.

Pour chaque état M appartenant à $\mathcal{M}_e$, leur labellisation repose sur l'ordonnancement propre à chaque mode $Mode_{\mathcal{R}_j} \in \mathcal{RM}(Mode_{\mathcal{R}_i})$ (c.f. méthode de prévention et d'ordonnancement 4.1.2). Ainsi, pour chaque $Mode_{\mathcal{R}_j}$ (ligne 24), si M est atteint depuis M_{init} en suivant l'ordonnancement $\sigma(M_{init}, Mode_{\mathcal{R}_j})$ (ligne 25), le label $\delta_{\mathcal{R}_j}^{M_{init}}$ est ajouté à Δ_e (ligne 27) afin de labelliser M (ligne 33). A l'inverse, si aucun ordonnancement $\sigma(M_{init}, Mode_{\mathcal{R}_j})$ ne relie M_{init} à M (aucun changement de mode ne correspond à M), le label $O_{\mathcal{R}_i}^{M_{init}}$ est associé à M (ligne 31).

La construction de $Diag_m$ est partielle puisqu'elle s'arrête dès que le label $O_{\mathcal{R}_i}^{M_{init}}$ est assigné à M. En effet, lors de la construction de $\mathcal{M}_e$ (ligne 21), un état M y est ajouté si l'état le précédent n'est pas labellisé par $O_{\mathcal{R}_i}^{M_{init}}$. Ainsi, seuls les états reflétant un changement de mode depuis $(M_{init}, \mathcal{R}_i)$ sont étudiés dans $Diag_m$.

Algorithm 9: Algorithme de construction de $Diag_m$

Entrées : $(N_t, M_0^t) = (P_t, T, F, E, l, D_t, M_0^t)$ un modèle SC-net, $E_m = E_o$, $Mode_{\mathcal{R}_i}$ le mode étudié, (M_{init}, R_{init}) l'état étudié du SC-net ;
Sortie : $Diag_m(N, x_0^m) = (X_m, E_m, f_m, x_0^m)$;
$x_0^m = \{\}$;
$\mathcal{RM}(Mode_{\mathcal{R}_i}) = \{Mode_{\mathcal{R}_j} | ((\mathcal{R}_j \setminus \mathcal{R}_i = r_j) \vee (\mathcal{R}_i \setminus \mathcal{R}_j = r_i)) \wedge (\exists C \in \mathcal{C}, C \cap H_{\mathcal{R}_j} = emptyset)\}$; /* Construction de l'ensemble des modes atteignables depuis $Mode_{\mathcal{R}_i}$ selon les hypothèses 1 et 2 (4.3.1) */
$x_m^\varepsilon = \{\}$;/* État si un changement de mode depuis M_{init} bloque tout le FMS */
$f_m(x_m^0, \varepsilon) = x_m^\varepsilon$; /* Arc labellisé par ε entre x_m^0 et x_m^ϵ */
for $Mode_{\mathcal{R}_j} \in \mathcal{RM}(Mode_{\mathcal{R}_i})$ **do**
  $\sigma(M_{init}, Mode_{\mathcal{R}_j})$; /* L'ordonnancement depuis $(M_{init}, Mode_{\mathcal{R}_j})$ est calculé à partir des étapes de la partie 4.1.2 */
  $x_0^m = x_0^m \otimes \{(M_{init}, \delta_{\mathcal{R}_j}^{M_{init}})\}$; /* On ajoute $(M_{init}, \delta_{\mathcal{R}_j}^{M_{init}})$ à x_0^m */
  if $\sigma(M_{init}, Mode_{\mathcal{R}_j}) = \emptyset$ **then**
    $x_m^\varepsilon = x_m^\varepsilon \otimes \{(M_{init}, \delta_{\mathcal{R}_j}^{M_{init}})\}$; /* On compose $(M_{init}, \delta_{\mathcal{R}_j}^{M_{init}})$ avec x_0^ε */
  end if
end for
$X_m = \{x_0^m, x_m^\}$;
$X_m^{new} = \{\}$;
while $X_m \neq X_m^{new}$ **do**
  $X_m^{new} = X_m$;
  for $x_m \in X_m$ **do**
    for $e \in E_o$ **do**
      $x_m^{new} = \{\}$;
      $\mathcal{M}_e = \{M \in \mathcal{R}(N_t, M_{init}) | \exists (M_1, \Delta_1) \in x_d, \Delta_1 \neq O_{\mathcal{R}_j}^{M_{init}}, t = (l)^{-1}(e), M_1[t > M\}$;
      for $M \in \mathcal{M}_e$ **do**
        $\Delta_e = \{\}$;/* Ensemble des labels associés à M */
        for $Mode_{\mathcal{R}_j} \in \mathcal{RM}(Mode_{\mathcal{R}_i})$ **do**
          if $(\delta_{\mathcal{R}_j}^{M_{init}} \in \Delta_1) \wedge (\exists \lambda_o \in E_o^\star, l(\lambda_o) = \sigma_o, M_{init}[\sigma_o > M_1, \lambda_o e < l(\sigma(M_{init}, Mode_{\mathcal{R}_j}))$ **then**
            $\Delta_e = \{\Delta_e, \delta_{\mathcal{R}_j}^{M_{init}}\}$;
          end if
        end for
        if $\Delta_e = \emptyset$ **then**
          $\Delta_e = \{O_{\mathcal{R}_i}^{M_{init}}\}$;
        end if
        $x_m^{new} = \{x_m^{new}, (M, \Delta_e)\}$;
      end for
      if $x_m^{new} \notin X_m^{new}$ **then**
        $X_m^{new} = \{X_m^{new}, x_m^{new}\}$;
      end if
      $f_m(x_m, e) = x_m^{new}$; /* Un arc de x_m à x_m^{new} est créé */
    end for
  end for
end while
Retourner $Diag_m(N, x_0^m) = (X_m, E_m, f_m, x_0^m)$;

6.15 Algorithme de construction du diagnostiqueur commun dans un contexte incertain

Dans cette annexe, l'algorithme de construction de la ième itération du diagnostiqueur commun dans un contexte incertain $Diag^i_{cu}$ est présenté. Cet algorithme est fondé sur celui de $Diag^i_c$ (voir annexe 6.11) introduit dans le chapitre 3 (3.3.2). Son objectif est le diagnostic conjoint des changements de mode, des attaques de blocage et des profils d'attaquant dans un contexte incertain. Le pseudo-code de ce nouvel algorithme est introduit ci-dessous (algorithme 10) et ses nouveautés sont décrites subséquemment.

Dans cette algorithme, la phase d'initialisation (lignes 3 à 11) consiste à conserver du diagnostiqueur précédent $Diag^{i-1}_{cu}$ les états pertinents pour le diagnostic au sein de la nouvelle itération $Diag^i_{cu}$ du diagnostiqueur commun. Par conséquent, la phase d'initialisation cherche en premier lieu à identifier l'origine du déclenchement de la nouvelle itération, entre ré-ordonnancement (lignes 3 à 6) et changement de mode (lignes 7 à 11). Dans le premier cas, il existe un label d'état optimal $O^{\mathcal{R}_i}_a/O^{M_k,\mathcal{R}_i}_a$ au sein de l'état actuel x^{i-1}_{cu} du diagnostiqueur précédent. Dans le second cas, il n'existe pas de tels labels au sein de x^{i-1}_{cu} puisque M_{init} est un état ne respectant pas l'ordonnancement précédemment calculé.

Une fois l'origine du déclenchement identifiée, la phase d'initialisation défini le nouveau diagnostiqueur à partir d'une réduction du précédent. Dans le premier cas (lignes 4-5), sont conservés de $Diag^{i-1}_{cu}$ les états atteignables depuis x^{i-1}_{cu} et labellisés par des profils d'attaquant($\delta^{M_k,\mathcal{R}_i}_{\mathcal{P}}$) ou des changements de mode ($\delta^{M_k}_{\mathcal{R}_i}$). Les labels des attaques de blocage ne se propagent pas entre les deux diagnostiqueurs puisque le FMS se trouve dans un état optimal lors du ré-ordonnancement. Puis, tous les modes dans lesquelles peut se trouver le FMS sont identifiés à l'aide des labels de trajectoire optimale ($O^{\mathcal{R}_i}_a/O^{M_k,\mathcal{R}_i}_a$) de x^{i-1}_{cu} et sont rassemblés dans $LABEL(x^{cu_i}_0)$ (ligne 6). Dans le second cas (ligne 8-9), sont conservés de $Diag^{i-1}_{cu}$ les états labellisés par les attaques de blocage ($\delta^{\mathcal{R}_i}_{A1}/\delta^{M_k,\mathcal{R}_i}_{A2}$) et les profils d'attaquant ($\delta^{M_k,\mathcal{R}_i}_{\mathcal{P}}$). Dans $Diag^{i-1}_{cu}$, x^{i-1}_{cu} est labellisé par un label d'attaque de blocage puisqu'il correspond à un état ne respectant pas l'ordonnancement attendu. Les labels de changement de mode et les états labellisés uniquement par ceux-ci ne sont pas conservés dans $Diag^i_{cu}$ car un deuxième changement de mode ne peut avoir lieu immédiatement (hypothèse 4). Puis, tous les modes dans lesquelles peut se trouver le FMS sont identifiés à l'aide des labels de changement de mode de x^{i-1}_{cu} et sont rassemblés dans $LABEL(x^{cu_i}_0)$ (ligne 10).

La construction complète de $Diag^i_{cu}$ est réalisé par les lignes 12 à 30 de l'algorithme. Dans $Diag^i_{cu}$, chaque ordonnancement éligible depuis M_{init} représente une trajectoire que peut suivre le FMS. Ces ordonnancements correspondent aux labels identifiés précédemment dans $LABEL(x^{cu_i}_0)$, soit un ordonnancement par mode actuel dans x^{i-1}_{cu} dans le cas d'un ré-ordonnancement et un ordonnancement par changement de mode possible dans x^{i-1}_{cu} dans le cas d'un changement de mode. Dans le second cas, il est à noté qu'un même nouveau mode $Mode_{\mathcal{R}_j}$ peut générer plusieurs ordonnancements si son changement de mode peut avoir lieu depuis différents états M_k précédents M_{init}. Ainsi pour chaque label de $LABEL(x^{cu_i}_0)$, l'ordonnancement correspondant σ_{R_i} depuis M_{ini} est déterminé (lignes 14-15 pour le premier cas, lignes 17-18 pour le second). A partir de cet ordonnancement, le diagnostiqueur des attaques de blocage $Diag^{\sigma_{\mathcal{R}_i}}_a$ est composé à $Diag^i_{cu}$ (ligne 20) et représente la trajectoire optimal et les attaques de blocage contre un FMS suivant cet ordonnancement. Les labels associés à ce diagnostiqueur sont notés $\Delta^{\mathcal{R}_i}_a = \{O^{\mathcal{R}_i}_a, \delta^{\mathcal{R}_i}_{A1}, \delta^{\mathcal{R}_i}_{A2}\}$ ou $\Delta^{\mathcal{R}_i}_a = \{O^{M_k,\mathcal{R}_i}_a, \delta^{M_k,,\mathcal{R}_i}_{A1}, \delta^{M_k,\mathcal{R}_i}_{A2}\}$ dans le cas d'un label de changement

Algorithm 10: Algorithme de construction de $Diag^i_{cu}$

Entrées : $(N_t) = (P_t, T, F, E, l, D_t)$ le modèle SC-net du FMS, $N^{\alpha}_{\mathcal{R}_{ind}} = (P^{\alpha}_{\mathcal{R}_{ind}}, T^{\alpha}_{\mathcal{R}_{ind}}, F^{\alpha}_{\mathcal{R}_{ind}}, E^{\alpha}_{\mathcal{R}_{ind}}, l^{\alpha}_{\mathcal{R}_{ind}}, Ct^{\alpha}_{\mathcal{R}_{ind}}, D^{\alpha}_{\mathcal{R}_{ind}})$ le modèle d'attaque, $Diag^{i-1}_{cu}(N_t, x_0^{cu_{i-1}})$ le précédent diagnostiqueur, x^{i-1}_{cu} l'état actuel de $Diag^{i-1}_{cu}$, (M_{init}, R_{init}) l'état actuel du FMS, K_{max} ;

Sorties : $Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i}) = (X^i_{cu}, E^i_{cu}, f^i_{cu}, x_0^{cu_i})$;

if $\exists (M, \Delta) \in x^{i-1}_{cu}|_{\Delta_a}$ *tel que* $O^{\mathcal{R}_i}_a / O^{M_k, \mathcal{R}_i}_a \in \Delta$ **then** /* Si un ré-ordonnancement a lieu */

  $\Delta_{cu} = \Delta_m \cup \Delta_{\mathcal{P}}$;

  $Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i}) = Diag^{i-1}_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x^{i-1}_{cu}|_{\Delta_{cu}})|_{\Delta_{cu}})$; /* On conserve de $Diag^{i-1}_{cu}$ les états atteignables depuis M_{init} et labellisés par des profils d'attaquant ou des changements de mode (c.f. 6.11). */

  $LABEL(x_0^{cu_i}) = \{O^{\mathcal{R}_i}_a / O^{M_k, \mathcal{R}_i}_a | \exists (M, \Delta) \in x^{i-1}_{cu}, O^{\mathcal{R}_i}_a / O^{M_k, \mathcal{R}_i}_a \in \Delta\}$; /* Ensemble des labels de modes actuels si un ré-ordonnancement est à l'origine de $Diag^i_{cu}$ */

else/* Si un changement de mode a lieu */

  $\Delta_{cu} = \Delta_{\mathcal{P}} \cup \Delta_a$;

  $Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i}) = Diag^{i-1}_{cu}(N^{\mathcal{R}_{ind}, x^{i-1}_{cu}|_{\Delta_{cu}})|_{\Delta_{cu}}}_{\alpha}$; /* On conserve de $Diag^{i-1}_{cu}$ les états atteignables depuis M_{init} et labellisés par des profils d'attaquant (c.f. 6.11) car un changement de mode vient d'avoir lieu. Le label d'attaque de Δ_a associé à M_{init} dans x^{i-1}_{cu} est aussi conservé */

  $LABEL(x_0^{cu_i}) = \{\delta^{M_k}_{\mathcal{R}_i} | \exists (M, \Delta) \in x_0^{cu_i}, \delta^{M_k}_{\mathcal{R}_i} \in \Delta\}$: /* Ensemble des labels de modes actuels dans $x_0^{cu_i}$ si un changement de mode est à l'origine de $Diag^i_{cu}$ */

end if

for $\delta \in LABEL(x_0^{cu_i})$ **do** /* Pour chaque label de mode dans $x_0^{cu_i}$ */

  if $\delta = O^{\mathcal{R}_i}_a$ ou $\delta = O^{M_k, \mathcal{R}_i}_a$ **then**

    $N^{C-t}_{NB}(Mode_{\mathcal{R}_i}, M_{init})$; /* Construction de N^{C-t}_{NB} depuis $(Mode_{\mathcal{R}_i}, M_{init})$ */

    $\sigma_{\mathcal{R}_i} = A^{\star}BT(M_{init}|_{P^{C-t}_{NB}}, R_{init}|_{P^{C-t}_{NB}}, N^{C-t}_{NB}, K_{max})$; /* nouvel ordonnancement calculé depuis (M_{init}, R_{init}) en cas de ré-ordonnancement */

  else if $\delta = \delta^{M_k}_{\mathcal{R}_i}$ **then**

    $\sigma_{\mathcal{R}_i} | \sigma(M_k, \mathcal{R}_i) = \sigma_{M_k} \sigma_{\mathcal{R}_i}, M_k[\sigma_{M_k} > M_{init}$; /* En cas de changement de mode labellisé par $\delta^{M_k}_{\mathcal{R}_i}$, l'ordonnancement est extrait à partir de celui précédemment calculé depuis la paire $(M_k, \mathcal{R}_i)$ */

    $R_{init} = 0_{1 \times |P_t|}$;

  end if

  $Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i}) = Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i}) \circ Diag^{\sigma_{\mathcal{R}_i}}_a(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i})$; /* Le diagnostiqueur des attaques de blocage calculé depuis M_{init} et $\sigma_{\mathcal{R}_i}$ est composé à $Diag^i_{cu}$. Ses labels sont $\Delta^{\mathcal{R}_i}_a = \{O^{\mathcal{R}_i}_a, \delta^{\mathcal{R}_i}_{A1}, \delta^{\mathcal{R}_i}_{A2}\}$ ou $\Delta^{\mathcal{R}_i}_a = \{O^{M_k, \mathcal{R}_i}_a, \delta^{M_k, , \mathcal{R}_i}_{A1}, \delta^{M_k, \mathcal{R}_i}_{A2}\}$ dans le cas d'un label $\delta^{M_k}_{\mathcal{R}_i}$ */

  $\mathcal{RM}(Mode_{\mathcal{R}_i})$; /* Ensemble des modes atteignables depuis $Mode_{\mathcal{R}_i}$ */

  for $(M_k, R_k) \in \mathcal{R}(N_t, (M_{init}, R_{init}) | (M_{init}, R_{init})[\sigma_k > (M_k, R_k), l(\sigma_k) < l(\sigma_{\mathcal{R}_i})$ **do** /* Pour chaque état du FMS traversé par l'ordonnancement $\sigma_{\mathcal{R}_i}$ */

    $x^k_{cu} \in X^i_{cu} | f^i_{cu}(x_0^{cu_i}, l(\sigma_k)) = x^k_{cu}$; /* x^k_{cu} est l'état correspondant à M_k dans $Diag^i_{cu}$ */

    $\mathcal{P}_1((M_k, R_k)), \mathcal{P}_2((M_k, R_k)), \mathcal{P}_3((M_k, R_k))$; /* Calcul des trois profils d'attaquant dans un contexte incertain (4.2.3 et 6.13) */

25: $Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i}) = Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i}) \circ Diag^{\mathcal{R}_i}_{\mathcal{P}_k}(N^{\mathcal{R}_{ind}}_{\alpha}, \chi_{cu,\mathcal{P}_k}(x^k_{cu}))$;
/* Composition de $Diag^i_{cu}$ avec le diagnostiqueur des profils d'attaquant calculés depuis la paire $((M_k, R_k), Mode_{\mathcal{R}_i})$. On note $\Delta^{\mathcal{R}_i,k}_{\mathcal{P}} = \{O^{\mathcal{R}_i,k}_{\mathcal{P}}, \delta^{M_k,\mathcal{R}_i}_{\mathcal{P}_1}, \delta^{M_k,\mathcal{R}_i}_{\mathcal{P}_2}, \delta^{M_k,\mathcal{R}_i}_{\mathcal{P}_3}\}$ */
26: **if** $\nexists \sigma_{\mathcal{R}_j} \neq \sigma_{\mathcal{R}_i} | l(\sigma_k) < l(\sigma_{\mathcal{R}_j}), Mode_{\mathcal{R}_j} \in \mathcal{RM}(Mode_{\mathcal{R}_i}), Mode_{\mathcal{R}_j} \neq Mode_{\mathcal{R}_i}$ **then**
/* Si il existe un ordonnancement unique depuis (M_k, R_k) pour tous les modes de $\mathcal{RM}(Mode_{\mathcal{R}_i})$ */
27: $x^k_{cu} \in X^i_{cu} | f^i_{cu}(x_0^{cu_i}, l(\sigma_k)) = x^k_{cu}$;
28: $Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cui}) = Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cui}) \circ Diag^{M_k,\mathcal{R}_i}_{m}(N_t, \chi_{cu,m}(x^k_{cu}))$; /* Le diagnostiqueur des changements de mode construits depuis $(M_k, Mode_{\mathcal{R}_i})$ est composé à $Diag^i_{cu}$. Fonction χ, c.f. 6.11 */
29: **end if**
30: **end for**
31: Retourner $Diag^i_{cu}(N^{\mathcal{R}_{ind}}_{\alpha}, x_0^{cu_i}) = (X^i_{cu}, E^i_{cu}, f^i_{cu}, x_0^{cu_i})$;

de mode $\delta^{M_k}_{\mathcal{R}_i}$.

Puis, pour chaque état (M_k, R_k) traversé par l'ordonnancement $\sigma_{\mathcal{R}_i}$ depuis $(M_{init}, R_{init}$ (ligne 22), les diagnostiqueurs de profils d'attaquant (lignes 23-25) et de changements de mode (lignes 26-28) sont construits et composés à $Diag^i_{cu}$. Les labels du diagnostiqueur des profils d'attaquant sont notés $\Delta^{\mathcal{R}_i,k}_{\mathcal{P}} = \{O^{\mathcal{R}_i,k}_{\mathcal{P}}, \delta^{M_k,\mathcal{R}_i}_{\mathcal{P}_1}, \delta^{M_k,\mathcal{R}_i}_{\mathcal{P}_2}, \delta^{M_k,\mathcal{R}_i}_{\mathcal{P}_3}\}$ tandis que ceux du diagnostiqueur des changement de mode sont notés $\delta^{M_k}_{\mathcal{R}_i}$. Dans le cas des diagnostiqueurs de changement de mode, ils sont construits et composés à $Diag^i_{cu}$ uniquement si l'hypothèse 4 est respecté au sein de l'état (M_k, R_k), en d'autres termes si il existe un ordonnancement unique depuis (M_k, R_k) pour tous les modes atteignables de $\mathcal{RM}(Mode_{\mathcal{R}_i})$ (ligne 26).

Le diagnostiqueur $Diag^i_{cu}$ ainsi construit est partiel puisque les diagnostiqueurs qui le composent le sont et car celui-ci ne considère que les ordonnancements éligibles pouvant avoir lieu depuis (M_{init}, R_{init}) selon x^{i-1}_{cu} et les modes qui le caractérisent.

6.16 Expérimentation : logiciel et programmes

Dans cette dernière partie de l'annexe, les différents programmes SFC développés pour l'expérimentation des scénarios attaques sont présentés dans les figures suivantes. Par soucis de concision, seuls les programmes des attaques sont illustrés. Les programmes de commande de la plateforme "transfert-libre" sont disponible sur demande.

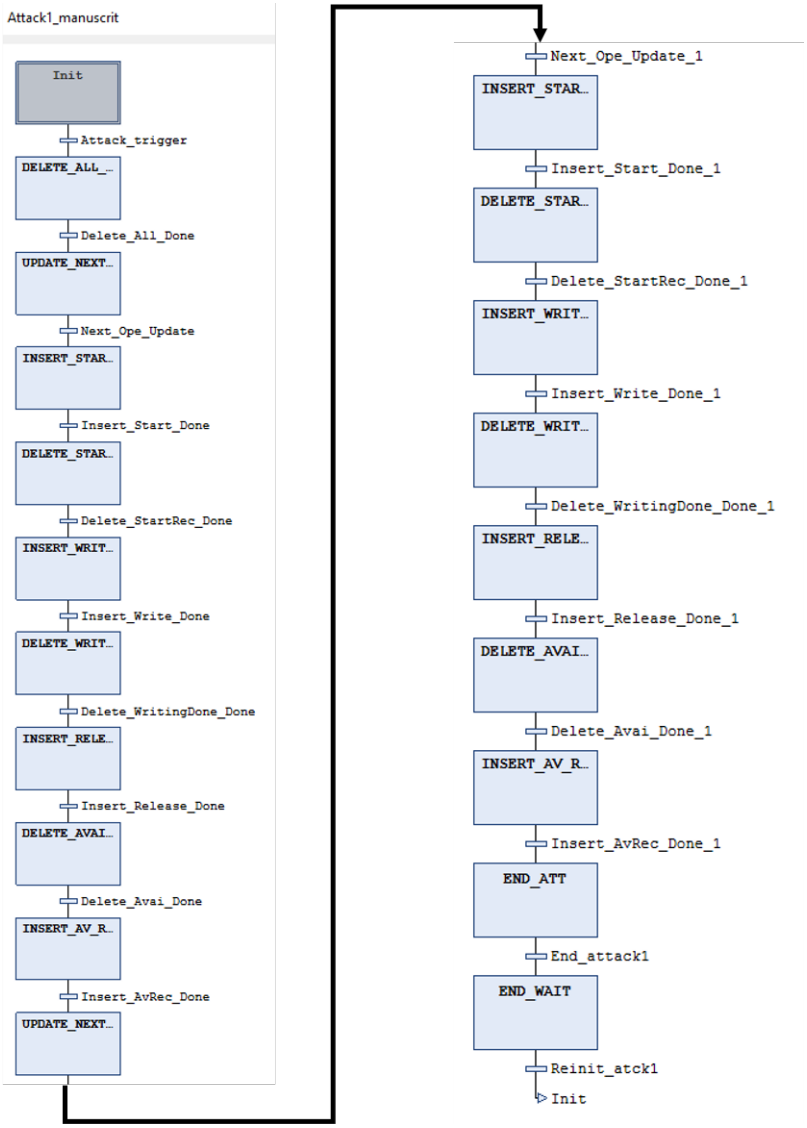


FIGURE 6.1 – Programme Grafcet sous SoMachine du scénario d'attaque 1

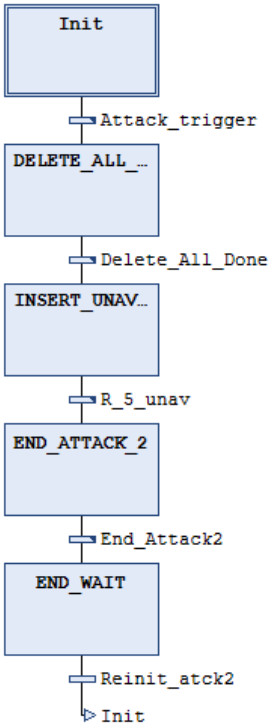


FIGURE 6.2 – Programme Grafcet sous SoMachine du scénario d'attaque 2

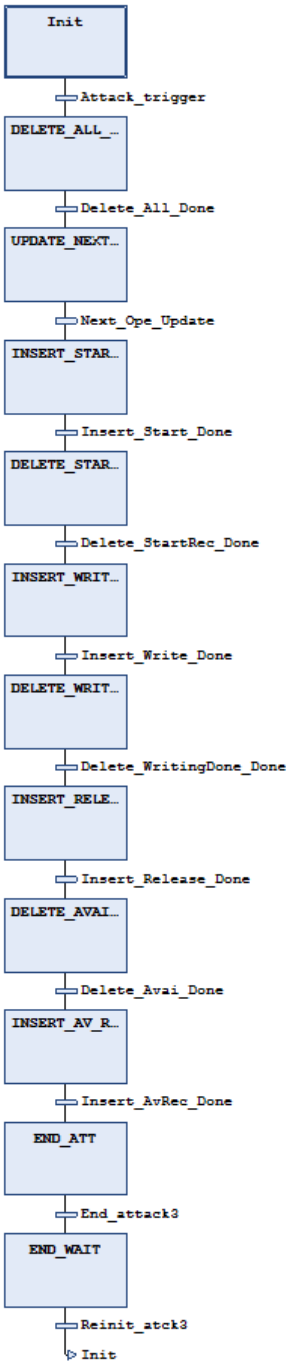


FIGURE 6.3 – Programme Grafcet sous SoMachine du scénario d'attaque 3

Bibliographie

[1] Z. Li, N. Wu et M. Zhou, "Deadlock Control of Automated Manufacturing Systems Based on Petri Nets—A Literature Review," p. 437-462, juill. 2012.

[2] Y. Koren, "General RMS Characteristics. Comparison with Dedicated and Flexible Systems," in *Reconfigurable Manufacturing Systems and Transformable Factories*, A. I. Dashchenko, éd., Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 27-45.

[3] Y. Koren, X. Gu et W. Guo, "Reconfigurable manufacturing systems : Principles, design, and future trends," *Frontiers of Mechanical Engineering*, p. 121-136, 1er juin 2018.

[4] T. J. Williams, "The Purdue enterprise reference architecture," en, *Computers in Industry*, p. 141-158, sept. 1994.

[5] D. Trentesaux, "Distributed control of production systems," en, *Engineering Applications of Artificial Intelligence*, Distributed Control of Production Systems, p. 971-978, oct. 2009.

[6] Y. Yang, H. Hu et Y. Liu, "A Petri net-based distributed control of automated manufacturing systems with assembly operations," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, ISSN : 2161-8089, août 2015, p. 1090-1097.

[7] H. Hu, C. Chen, R. Su, Y. Liu et M. Zhou, "Distributed supervisor synthesis for automated manufacturing systems using Petri nets," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, ISSN : 1050-4729, mai 2014, p. 4423-4429.

[8] N. Du, H. Hu et M. Zhou, "Robust Deadlock Avoidance and Control of Automated Manufacturing Systems With Assembly Operations Using Petri Nets," *IEEE Transactions on Automation Science and Engineering*, p. 1961-1975, oct. 2020, Conference Name : IEEE Transactions on Automation Science and Engineering.

[9] H. Gimbert, C. Mascle, A. Muscholl et I. Walukiewicz, *Distributed controller synthesis for deadlock avoidance*, avr. 2022.

[10] C. Cassandras et S. Lafortune, "Introduction to Discrete Event Systems," in *Introduction to Discrete Event Systems*, Journal Abbreviation : Introduction to Discrete Event Systems, jan. 2010, p. 800.

[11] A. Yalcin et T. Boucher, "Deadlock avoidance in flexible manufacturing systems using finite automata," *IEEE Transactions on Robotics and Automation*, p. 424-429, août 2000, Conference Name : IEEE Transactions on Robotics and Automation.

[12] A. Yalcin, "Supervisory control of automated manufacturing cells with resource failures," 2004.

[13] H. Golmakani, J. Mills et B. Benhabib, "Deadlock-free scheduling and control of flexible manufacturing cells using automata theory," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 327-337, mars 2006, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[14] H. Cho, T. Kumaran et R. Wysk, "Graph-theoretic deadlock detection and resolution for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, p. 413-421, juin 1995, Conference Name : IEEE Transactions on Robotics and Automation.

[15] M. Fanti, B. Maione, S. Mascolo et A. Turchiano, "Event-based feedback control for deadlock avoidance in flexible production systems," *IEEE Transactions on Robotics and Automation*, p. 347-363, juin 1997, Conference Name : IEEE Transactions on Robotics and Automation.

[16] M. Fanti et M. Zhou, "Deadlock control methods in automated manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 5-22, jan. 2004, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[17] G. Liu et K. Barkaoui, "A survey of siphons in Petri nets," en, *Information Sciences*, p. 198-220, oct. 2016.

[18] B. Huang, M. Zhou, X. S. Lu et A. Abusorrah, "Scheduling of Resource Allocation Systems with Timed Petri Nets : A Survey," *ACM Computing Surveys*, nov. 2022.

[19] M. Colledani, T. Tolio, A. Fischer, B. Iung, G. Lanza, R. Schmitt et J. Váncza, "Design and management of manufacturing systems for production quality," en, *CIRP Annals*, p. 773-796, jan. 2014.

[20] Y. K. Son et C. S. Park, "Economic measure of productivity, quality and flexibility in advanced manufacturing systems," en, *Journal of Manufacturing Systems*, p. 193-207, jan. 1987.

[21] R. Cordero, "Changing human resources to make flexible manufacturing systems (FMS) successful," en, *The Journal of High Technology Management Research*, p. 263-275, sept. 1997.

[22] V. Jain et T. Raj, "Modelling and analysis of FMS productivity variables by ISM, SEM and GTMA approach," en, *Frontiers of Mechanical Engineering*, p. 218-232, sept. 2014.

[23] A. Villemeur, *Sûreté de fonctionnement des systèmes industriels : fiabilité, facteurs humains, informatisation*, fr. Eyrolles, 1988.

[24] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou et Y. Halgand, "A survey of approaches combining safety and security for industrial control systems," *Reliability Engineering et System Safety*, p. 156-178, juill. 2015.

[25] L. J. Wells, J. A. Camelio, C. B. Williams et J. White, "Cyber-physical security challenges in manufacturing systems," en, *Manufacturing Letters*, p. 74-77, avr. 2014.

[26] N. Tuptuk et S. Hailes, "Security of smart manufacturing systems," en, *Journal of Manufacturing Systems*, p. 93-106, avr. 2018.

[27] A. E. Elhabashy, L. J. Wells, J. A. Camelio et W. H. Woodall, "A cyber-physical attack taxonomy for production systems : a quality control perspective," *Journal of Intelligent Manufacturing*, p. 2489-2504, 1er août 2019.

[28] M. A. Youssef et B. Al-Ahmady, "Quality management practices in a Flexible Manufacturing Systems (FMS) environment," *Total Quality Management*, p. 877-890, sept. 2002.

[29] F. Chen et E. Adam, "The impact of flexible manufacturing systems on productivity and quality," *IEEE Transactions on Engineering Management*, p. 33-45, fév. 1991, Conference Name : IEEE Transactions on Engineering Management.

[30] D. Stockton et N. Bateman, "Measuring the production range flexibility of a FMS," *Integrated Manufacturing Systems*, jan. 1995, Publisher : MCB UP Ltd.

[31] T. Raj, R. Attri et V. Jain, "Modelling the factors affecting flexibility in FMS," *International Journal of Industrial and Systems Engineering*, p. 350-374, jan. 2012.

[32] K. Mahmood, T. Karaulova, T. Otto et E. Shevtshenko, "Performance Analysis of a Flexible Manufacturing System (FMS)," en, *Procedia CIRP*, Manufacturing Systems 4.0 – Proceedings of the 50th CIRP Conference on Manufacturing Systems, p. 424-429, jan. 2017.

[33] H. O. Fried, S. S. Schmidt et C. A. K. Lovell, *The Measurement of Productive Efficiency : Techniques and Applications*, en. Oxford University Press, avr. 1993.

[34] J. LIU et B. L. MacCARTHY, "The classification of FMS scheduling problems," *International Journal of Production Research*, p. 647-656, mars 1996.

[35] M. Godinho Filho, C. F. Barco et R. F. Tavares Neto, "Using Genetic Algorithms to solve scheduling problems on flexible manufacturing systems (FMS) : a literature survey, classification and analysis," en, *Flexible Services and Manufacturing Journal*, p. 408-431, sept. 2014.

[36] F. T. S. Chan et H. K. Chan, "A comprehensive survey and future trend of simulation study on FMS scheduling," en, *Journal of Intelligent Manufacturing*, p. 87-102, fév. 2004.

[37] K. Miriyala et N. Viswanadham, "Reliability analysis of flexible manufacturing systems," en, *International Journal of Flexible Manufacturing Systems*, p. 145-162, déc. 1989.

[38] P. Berruet, A. Toguyeni et E. Craye, "Structural and functional approach for dependability in FMS," ISSN : 1062-922X, oct. 1999, 486-491 vol.4.

[39] M. G. Mehrabi, A. G. Ulsoy, Y. Koren et P. Heytler, "Trends and perspectives in flexible and reconfigurable manufacturing systems," en, *Journal of Intelligent Manufacturing*, p. 135-146, avr. 2002.

[40] A. Zakarian et A. Kusiak, "Modeling manufacturing dependability," *IEEE Transactions on Robotics and Automation*, p. 161-168, avr. 1997, Conference Name : IEEE Transactions on Robotics and Automation.

[41] G. Cherif, E. Leclercq et D. Lefebvre, "Scheduling of a class of partial routing FMS in uncertain environments with beam search," en, *Journal of Intelligent Manufacturing*, p. 493-514, fév. 2023.

[42] M. Mattila, M. Perala et V. Vannas, "Flexible Manufacturing Systems' compliance to the safety standards," en, *The International Journal of Advanced Manufacturing Technology*, p. 60-65, jan. 1996.

[43] V. Vannas et M. Mattila, "Safety aspects in FMS implementations," en, *International Journal of Human Factors in Manufacturing*, p. 163-179, 1996.

[44] Y. Zhang et K. Rasmussen, "Detection of Electromagnetic Interference Attacks on Sensor Systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, p. 203-216.

[45] G. Y. Liu, Z. W. Li, K. Barkaoui et A. M. Al-Ahmari, "Robustness of deadlock control for a class of Petri nets with unreliable resources," en, *Information Sciences*, Data-based Control, Decision, Scheduling and Fault Diagnostics, juin 2013.

[46] H. Kaid, A. Al-Ahmari, Z. Li et R. Davidrajuh, "Automatic Supervisory Controller for Deadlock Control in Reconfigurable Manufacturing Systems with Dynamic Changes," en, *Applied Sciences*, p. 5270, jan. 2020.

[47] B. Huang, M. Zhou, C. Wang, A. Abusorrah et Y. Al-Turki, "Deadlock-free Supervisor Design for Robotic Manufacturing Cells With Uncontrollable and Unobservable Events," *IEEE/CAA Journal of Automatica Sinica*, p. 597-605, mars 2021, Conference Name : IEEE/CAA Journal of Automatica Sinica.

[48] G. Cloutier et J.-J. Paques, "GEMMA, the complementary tool of the GRAFCET," in *Fourth Annual Canadian Conference Proceedings., Programmable Control and Automation Technology Conference and Exhibition*, 1988, 12A1-5/1.

[49] M. Uzam, "An Optimal Deadlock Prevention Policy for Flexible Manufacturing Systems Using Petri Net Models with Resources and the Theory of Regions," en, *The International Journal of Advanced Manufacturing Technology*, p. 192-208, fév. 2002.

[50] M. Uzam et M. Zhou, "An Iterative Synthesis Approach to Petri Net-Based Deadlock Prevention Policy for Flexible Manufacturing Systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 362-371, mai 2007, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[51] J. Ezpeleta, J. Colom et J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, p. 173-184, avr. 1995, Conference Name : IEEE Transactions on Robotics and Automation.

[52] Y. Hou et K. Barkaoui, "Deadlock analysis and control based on Petri nets : A siphon approach review," *Advances in Mechanical Engineering*, 2017, Publisher : Sage Journals.

[53] G. Tuncel et G. M. Bayhan, "Applications of Petri nets in production scheduling : a review," en, *The International Journal of Advanced Manufacturing Technology*, p. 762-773, oct. 2007.

[54] D. Lei, "Multi-objective production scheduling : a survey," en, *The International Journal of Advanced Manufacturing Technology*, p. 926, oct. 2008.

[55] J. Branke, S. Nguyen, C. W. Pickardt et M. Zhang, "Automated Design of Production Scheduling Heuristics : A Review," *IEEE Transactions on Evolutionary Computation*, p. 110-124, fév. 2016, Conference Name : IEEE Transactions on Evolutionary Computation.

[56] I. Hatono, K. Yamagata et H. Tamura, "Modeling and online scheduling of flexible manufacturing systems using stochastic Petri nets," *IEEE Transactions on Software Engineering*, p. 126-132, fév. 1991, Conference Name : IEEE Transactions on Software Engineering.

[57] G. H. Hu, Y. S. Wong et H. T. Loh, "An FMS scheduling and control decision support system based on generalised stochastic Petri nets," en, *The International Journal of Advanced Manufacturing Technology*, p. 52-58, jan. 1995.

[58] A. K. Chincholkar et O. V. Krishnaiah Chetty, "Stochastic coloured Petri nets for modelling and evaluation, and heuristic rule base for scheduling of FMS," en, *The International Journal of Advanced Manufacturing Technology*, p. 339-348, sept. 1996.

[59] A. Camurri, P. Franchi, F. Gandolfo et R. Zaccaria, "Petri net based process scheduling : A model of the control system of flexible manufacturing systems," en, *Journal of Intelligent and Robotic Systems*, p. 99-123, août 1993.

[60] B. Huang, R. Jiang et G. Zhang, "Search strategy for scheduling flexible manufacturing systems simultaneously using admissible heuristic functions and nonadmissible heuristic functions," en, *Computers et Industrial Engineering*, p. 21-26, mai 2014.

[61] J. Zhou, J. Luo, D. Lefebvre et Z. Li, "Modeling and Scheduling Methods for Batch Production Systems Based on Petri Nets and Heuristic Search," *IEEE Access*, p. 163 458-163 471, 2020, Conference Name : IEEE Access.

[62] G. Cherif, E. Leclercq et D. Lefebvre, "Scheduling Problems for a Class of Hybrid FMS Using T-TPN and Beam Search," en, *Journal of Control, Automation and Electrical Systems*, p. 591-604, juin 2021.

[63] D. Y. Lee et F. DiCesare, "Scheduling flexible manufacturing systems using Petri nets and heuristic search," *IEEE Transactions on Robotics and Automation*, p. 123-132, avr. 1994, Conference Name : IEEE Transactions on Robotics and Automation.

[64] B. Huang et M. Zhou, *Supervisory Control and Scheduling of Resource Allocation Systems : Reachability Graph Perspective*. John Wiley et Sons, 28 juill. 2020, 288 p.

[65] Q. Chen, J. Y. S. Luh et L. Shen, "Complexity Reduction for Optimization of Deterministic Timed Petri-Net Scheduling by Truncation," *Cybernetics and Systems*, p. 643-695, sept. 1994, Publisher : Taylor et Francis _eprint : https ://doi.org/10.1080/01969729408902348.

[66] M. Zhou, H.-S. Chiu et H. Xiong, "Petri net scheduling of FMS using branch and bound method," in *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, nov. 1995, 211-216 vol.1.

[67] R. Fritz, N. Krebs et P. Zhang, "A Monte-Carlo Tree Search based Tracking Control Approach for Timed Petri Nets," en, *IFAC-PapersOnLine*, 21st IFAC World Congress, p. 2095-2100, jan. 2020.

[68] A. Reyes, H. Yu, G. Kelleher et S. Lloyd, "Integrating Petri Nets and hybrid heuristic search for the scheduling of FMS," en, *Computers in Industry*, p. 123-138, jan. 2002.

[69] K. Z. Gao, Z. M. He, Y. Huang, P. Y. Duan et P. N. Suganthan, "A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing," en, *Swarm and Evolutionary Computation*, p. 100 719, sept. 2020.

[70] V. Kesavan, R. Kamalakannan, R. Sudhakarapandian et P. Sivakumar, "Heuristic and meta-heuristic algorithms for solving medium and large scale sized cellular manufacturing system NP-hard problems : A comprehensive review," en, *Materials Today : Proceedings*, International Conference on Recent Trends in Nanomaterials for Energy, Environmental and Engineering Applications, p. 66-72, jan. 2020.

[71] T. R. Chinnusamy, P. Kammar, F. Praveen, T. Karthikeyan, M. Krishnan, N. Varshitha et A. A. Shetty, "Scheduling of Flexible Manufacturing System by Hybridizing Petri Net with Improved Scatter Search Algorithm," en, in *Recent Trends in Mechanical Engineering*, G. S. V. L. Narasimham, A. V. Babu, S. S. Reddy et R. Dhanasekaran, éd., sér. Lecture Notes in Mechanical Engineering, Singapore : Springer, 2020, p. 305-332.

[72] X. Guo, S. Liu, M. Zhou et G. Tian, "Disassembly Sequence Optimization for Large-Scale Products With Multiresource Constraints Using Scatter Search and Petri Nets," *IEEE Transactions on Cybernetics*, p. 2435-2446, nov. 2016, Conference Name : IEEE Transactions on Cybernetics.

[73] Y. Mati, N. Rezg et X. Xie, "Geometric approach and taboo search for scheduling flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, p. 805-818, déc. 2001, Conference Name : IEEE Transactions on Robotics and Automation.

[74] V Jain, R Swarnkar et M. K Tiwari, "Modelling and analysis of wafer fabrication scheduling via generalized stochastic Petri net and simulated annealing," *International Journal of Production Research*, p. 3501-3527, jan. 2003, Publisher : Taylor et Francis _eprint : https ://doi.org/10.1080/0020754031000118152.

[75] Z. Huang et Z. Wu, "Deadlock-free scheduling method for automated manufacturing systems using genetic algorithm and Petri nets," in *IEEE International Conference on*

Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, ISSN : 1050-4729, avr. 2004, 566-571 Vol.1.

[76] A. Li et N. Xie, "A Robust Scheduling for Reconfigurable Manufacturing System Using Petri Nets and Genetic Algorithm," in *2006 6th World Congress on Intelligent Control and Automation*, juin 2006, p. 7302-7306.

[77] G. Mejia, C. Montoya, J. Cardona et A. L. Castro, "Petri nets and genetic algorithms for complex manufacturing systems scheduling," *International Journal of Production Research*, p. 791-803, fév. 2012, Publisher : Taylor et Francis _eprint : https ://doi.org/10.1080/00207543.

[78] F. Qiao, Y.-m. Ma, L. Li et H.-x. Yu, "A Petri Net and Extended Genetic Algorithm Combined Scheduling Method for Wafer Fabrication," *IEEE Transactions on Automation Science and Engineering*, p. 197-204, jan. 2013, Conference Name : IEEE Transactions on Automation Science and Engineering.

[79] H. Liu, W. Wu, X. Han, H. Yang, W. Ma et C. Wang, "Deadlock-free genetic scheduling algorithm for flexible manufacturing systems with key resources," in *2019 Chinese Control Conference (CCC)*, ISSN : 1934-1768, juill. 2019, p. 1772-1776.

[80] X. M. Liu, L. Pan et H. Zheng, "Schedule optimization of time petri nets based on ant colony systems," *Applied Mechanics and Materials*, 2013, Conference Name : Information Technology Applications in Industry ISBN : 9783037855744 Publisher : Trans Tech Publications Ltd.

[81] S. Tian, T. Wang, L. Zhang et X. Wu, "Real-time shop floor scheduling method based on virtual queue adaptive control : Algorithm and experimental results," en, *Measurement*, p. 106 689, déc. 2019.

[82] L. Han, K. Xing, X. Chen et F. Xiong, "A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems," en, *Journal of Intelligent Manufacturing*, p. 1083-1096, juin 2018.

[83] X. Li, K. Xing et Q. Lu, "Hybrid particle swarm optimization algorithm for scheduling flexible assembly systems with blocking and deadlock constraints," en, *Engineering Applications of Artificial Intelligence*, p. 104 411, oct. 2021.

[84] M. Drakaki et P. Tzionas, "Manufacturing Scheduling Using Colored Petri Nets and Reinforcement Learning," en, *Applied Sciences*, p. 136, 2017, Number : 2 Publisher : Multidisciplinary Digital Publishing Institute.

[85] S. Baer, J. Bakakeu, R. Meyes et T. Meisen, "Multi-Agent Reinforcement Learning for Job Shop Scheduling in Flexible Manufacturing Systems," in *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*, sept. 2019, p. 22-25.

[86] L. Hu, Z. Liu, W. Hu, Y. Wang, J. Tan et F. Wu, "Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network," en, *Journal of Manufacturing Systems*, p. 1-14, avr. 2020.

[87] J.-H. Lee et H.-J. Kim, "Reinforcement learning for robotic flow shop scheduling with processing time variations," *International Journal of Production Research*, p. 2346-2368, avr. 2022, Publisher : Taylor et Francis _eprint : https ://doi.org/10.1080/00207543.2021.1887533.

[88] J. Li, J. Luo, X. Li, S. Yi et C. Pan, "Heuristic Scheduling Method for FMSs Based on P-timed Petri Nets and Deep Learning," déc. 2022, p. 1-6.

[89] Y. Chen, Z. Li, M. Khalgui et O. Mosbahi, "Design of a Maximally Permissive Liveness-Enforcing Petri Net Supervisor for Flexible Manufacturing Systems," *IEEE Transactions on Automation Science and Engineering*, p. 374-393, avr. 2011, Conference Name : IEEE Transactions on Automation Science and Engineering.

[90] Z. Li et M. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 38-51, jan. 2004, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[91] H. Liu, K. Xing, W. Wu, M. Zhou et H. Zou, "Deadlock Prevention for Flexible Manufacturing Systems via Controllable Siphon Basis of Petri Nets," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, p. 519-529, mars 2015, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics : Systems.

[92] F. Chu et X.-L. Xie, "Deadlock analysis of Petri nets using siphons and mathematical programming," *IEEE Transactions on Robotics and Automation*, p. 793-804, déc. 1997, Conference Name : IEEE Transactions on Robotics and Automation.

[93] K. Xing, M. Zhou, F. Wang, H. Liu et F. Tian, "Resource-Transition Circuits and Siphons for Deadlock Control of Automated Manufacturing Systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 74-84, jan. 2011, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[94] S. Wang, C. Wang, M. Zhou et Z. Li, "A Method to Compute Strict Minimal Siphons in a Class of Petri Nets Based on Loop Resource Subsets," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 226-237, jan. 2012, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[95] R. Cordone, L. Ferrarini et L. Piroddi, "Enumeration algorithms for minimal siphons in Petri nets based on place constraints," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 844-854, nov. 2005.

[96] D. You, O. Karoui et S. Wang, "Computation of Minimal Siphons in Petri Nets Using Problem Partitioning Approaches," *IEEE/CAA Journal of Automatica Sinica*, p. 329-338, fév. 2022, Conference Name : IEEE/CAA Journal of Automatica Sinica.

[97] D. You, S. Wang et M. Zhou, "Synthesis of Monitor-Based Liveness-Enforcing Supervisors for \rmS^3 PR With \boldsymbol\xi -Resources," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, p. 967-975, juin 2015, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics : Systems.

[98] Y. Huang, M. Jeng, X. Xie et S. Chung, "Deadlock prevention policy based on Petri nets and siphons," *International Journal of Production Research*, p. 283-305, 1er jan. 2001, Publisher : Taylor et Francis _eprint : https ://doi.org/10.1080/00207540010002405.

[99] X. Guo, S. Wang, D. You, Z. Li et X. Jiang, "A Siphon-Based Deadlock Prevention Strategy for S3PR," *IEEE Access*, p. 86 863-86 873, 2019, Conference Name : IEEE Access.

[100] K. Xing, F. Tian, H. Xu et B. Hu, "Optimal Polynomial Complexity Deadlock Avoidance Policies for Manufacturing Systems with Flexible Routings," in *2006 IEEE International Conference on Automation Science and Engineering*, ISSN : 2161-8089, oct. 2006, p. 448-453.

[101] K. Xing, M. Zhou, H. Liu et F. Tian, "Optimal Petri-Net-Based Polynomial-Complexity Deadlock-Avoidance Policies for Automated Manufacturing Systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 188-199, jan. 2009, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[102] N. Wu, M. Zhou et G. Hu, "One-Step Look-Ahead Maximally Permissive Deadlock Control of AMS by Using Petri Nets," *ACM Transactions on Embedded Computing Systems*, 10 :1-10 :23, 1er jan. 2013.

[103] C. Gu, Z. Li et A. Al-Ahmari, "A multistep look-ahead deadlock avoidance policy for automated manufacturing systems," *Discrete Dynamics in Nature and Society*, e8687035, 21 août 2017, Publisher : Hindawi.

[104] R. Lin, Z. Yu, X. Shi, L. Dong et E. A. Nasr, "On Multi-Step Look-Ahead Deadlock Prediction for Automated Manufacturing Systems Based on Petri Nets," *IEEE Access*, p. 170 421-170 432, 2020, Conference Name : IEEE Access.

[105] B. Yang et H. Hu, "Maximally Permissive Deadlock and Livelock Avoidance for Automated Manufacturing Systems via Critical Distance," *IEEE Transactions on Automation Science and Engineering*, p. 1-15, 2022, Conference Name : IEEE Transactions on Automation Science and Engineering.

[106] J. Ezpeleta, F. Tricas, F. Garcia-Valles et J. Colom, "A Banker's solution for deadlock avoidance in FMS with flexible routing and multiresource states," *IEEE Transactions on Robotics and Automation*, p. 621-625, août 2002, Conference Name : IEEE Transactions on Robotics and Automation.

[107] J. Luo, Z. Liu, S. Wang et K. Xing, "Robust deadlock avoidance policy for automated manufacturing system with multiple unreliable resources," *IEEE/CAA Journal of Automatica Sinica*, p. 812-821, mai 2020, Conference Name : IEEE/CAA Journal of Automatica Sinica.

[108] F.-S. Hsieh et S.-C. Chang, "Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, p. 196-209, avr. 1994, Conference Name : IEEE Transactions on Robotics and Automation.

[109] X. Fan, H. Hu, B. Yang, Y. Liu et G. He, "Event Circuit Structures for Deadlock Avoidance in Flexible Manufacturing Systems," *IEEE Transactions on Automation Science and Engineering*, p. 1-14, 2022, Conference Name : IEEE Transactions on Automation Science and Engineering.

[110] R. A. Wysk, N.-S. Yang et S. Joshi, "Resolution of deadlocks in flexible manufacturing systems : Avoidance and recovery approaches," *Journal of Manufacturing Systems*, p. 128-138, 1994.

[111] M. Fanti, G. Maione et B. Turchiano, "Deadlock detection and recovery in flexible production systems with multiple capacity resources," in *Proceedings of 8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications (MELECON 96)*, 1996, 237-241 vol.1.

[112] A. Giua, C. Seatzu et F. Basile, "Observer-based state-feedback control of timed Petri nets with deadlock recovery," *IEEE Transactions on Automatic Control*, p. 17-29, jan. 2004, Conference Name : IEEE Transactions on Automatic Control.

[113] Y. Lu, Y. Chen, Z. Li et N. Wu, "An Efficient Method of Deadlock Detection and Recovery for Flexible Manufacturing Systems by Resource Flow Graphs," *IEEE Transactions on Automation Science and Engineering*, p. 1-12, 2021, Conference Name : IEEE Transactions on Automation Science and Engineering.

[114] I. Grobelna et A. Karatkevich, "A Deadlock Recovery Policy for Flexible Manufacturing Systems with Minimized Traversing within Reachability Graph," in *2022 21st International Symposium INFOTEH-JAHORINA (INFOTEH)*, ISSN : 2767-9470, mars 2022, p. 1-6.

[115] Y. Chen, Z. Li, A. Al-Ahmari, N. Wu et T. Qu, "Deadlock recovery for flexible manufacturing systems modeled with Petri nets," *Information Sciences*, p. 290-303, 2017.

[116] H. Aytug, M. A. Lawley, K. McKay, S. Mohan et R. Uzsoy, "Executing production schedules in the face of uncertainties : a review and some future directions," *European Journal of Operational Research*, IEPM : Focus on Scheduling, p. 86-110, 16 fév. 2005.

[117] Z. Li et M. Ierapetritou, "Process scheduling under uncertainty : review and challenges," *Computers et Chemical Engineering*, Festschrift devoted to Rex Reklaitis on his 65th Birthday, p. 715-727, 5 avr. 2008.

[118] D. Ouelhadj et S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, p. 417-431, 1er août 2009.

[119] T. Chaari, S. Chaabane, N. Aissani et D. Trentesaux, "Scheduling under uncertainty : Survey and research directions," in *2014 International Conference on Advanced Logistics and Transport (ICALT)*, mai 2014, p. 229-234.

[120] D. Lefebvre, "Evaluating the robustness of scheduling in uncertain environment with Petri nets," in *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, sér. VALUETOOLS 2017, New York, NY, USA : Association for Computing Machinery, 5 déc. 2017, p. 170-177.

[121] D. Lefebvre et G. Mejia, "Robust scheduling in uncertain environment with petri nets and beam search," *IFAC-PapersOnLine*, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018, p. 1077-1082, 1er jan. 2018.

[122] G. Mejia et D. Lefebvre, "Robust scheduling of flexible manufacturing systems with unreliable operations and resources," *International Journal of Production Research*, p. 6474-6492, 1er nov. 2020, Publisher : Taylor Francis _eprint : https ://doi.org/10.1080/00207543.2019.16827

[123] G. Liu, D. Lefebvre et Z. Li, "Robust Deadlock-free Scheduling for FMS with Liveness-enforcing Supervisor Combined with Beam Search Controller," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, ISSN : 2577-1655, oct. 2019, p. 1825-1830.

[124] S. Himmiche, P. Marangé, A. Aubry et J.-F. Pétin, "Robust production scheduling under machine failures - a DES based evaluation approach," *IFAC-PapersOnLine*, 14th IFAC Workshop on Discrete Event Systems WODES 2018, p. 271-276, 1er jan. 2018.

[125] S. Himmiche, P. Marangé, A. Aubry et J.-F. Pétin, "Robustness evaluation process for scheduling under uncertainties," *Processes*, p. 371, fév. 2023, Number : 2 Publisher : Multidisciplinary Digital Publishing Institute.

[126] D. R. Denzler, W. J. Boe et E. Duplaga, "An experimental investigation of FMS scheduling rules under uncertainty," *Journal of Operations Management*, p. 139-151, 1987, _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1016/0272-6963%2887%2990013-1.

[127] I. Sabuncuoglu et S. Karabuk, "Rescheduling frequency in an FMS with uncertain processing times and unreliable machines," *Journal of Manufacturing Systems*, Special issue on scheduling : From Research Into Practice, p. 268-283, 1er jan. 1999.

[128] D. Lefebvre, "Dynamical scheduling and robust control in uncertain environments with petri nets for DESs," *Processes*, p. 54, déc. 2017, Number : 4 Publisher : Multidisciplinary Digital Publishing Institute.

[129] J. C. Luo, K. Y. Xing, M. C. Zhou, X. L. Li et X. N. Wang, "Scheduling of deadlock and failure-prone automated manufacturing systems via hybrid heuristic search," *International Journal of Production Research*, p. 3283-3293, 3 juin 2017, Publisher : Taylor et Francis _eprint : https ://doi.org/10.1080/00207543.2017.1306132.

[130] O. Hayane et D. Lefebvre, "Reconfigurable Timed Extended Reachability Graphs for scheduling problems in uncertain environments," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, ISSN : 2161-8089, août 2021, p. 310-315.

[131] F. Wang, K.-Y. Xing, M.-C. Zhou, X.-P. Xu et L.-B. Han, "A robust deadlock prevention control for automated manufacturing systems with unreliable resources," *Information Sciences*, p. 243-256, 1er juin 2016.

[132] M. Qin, Z. Li, M. Zhou, M. Khalgui et O. Mosbahi, "Deadlock Prevention for a Class of Petri Nets With Uncontrollable and Unobservable Transitions," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 727-738, mai 2012, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[133] Y. Wu, K. Xing, J. Luo et Y. Feng, "Robust deadlock control for automated manufacturing systems with an unreliable resource," *Information Sciences*, p. 17-28, 10 juin 2016.

[134] N. Du et H. Hu, "A Robust Prevention Method for Automated Manufacturing Systems With Unreliable Resources Using Petri Nets," *IEEE Access*, p. 78 598-78 608, 2018, Conference Name : IEEE Access.

[135] X. Li, G. Liu, M. Qin et Z. Li, "Robust liveness enforcement of petri nets with uncontrollable and unobservable transitions based on structural analysis," *IET Control Theory et Applications*, n/a, _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1049/cth2.12384.

[136] Y. Feng, K. Xing, H. Liu et Y. Wu, "Two-stage design method of robust deadlock control for automated manufacturing systems with a type of unreliable resources," *Information Sciences*, p. 286-301, 1er mai 2019.

[137] Y. Feng, K. Xing, M. Zhou, X. Wang et H. Liu, "Robust Deadlock Prevention for Automated Manufacturing Systems With Unreliable Resources by Using General Petri Nets," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, p. 3515-3527, oct. 2020, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics : Systems.

[138] G. Liu, L. Zhang, L. Chang, A. Al-Ahmari et N. Wu, "Robust deadlock control for automated manufacturing systems based on elementary siphon theory," *Information Sciences*, p. 165-182, 1er fév. 2020.

[139] G. Liu, Y. Liu et Z. Li, "Robust liveness-enforcing supervisor for petri nets with unreliable resources based on mixed integer programming," *Soft Computing*, 19 nov. 2021.

[140] N. Du, Y. Yang et H. Hu, "Robust deadlock control in automated manufacturing systems with unreliable resources based on an algebraic way," *International Journal of Production Research*, p. 1-15, 3 oct. 2022, Publisher : Taylor et Francis _eprint : https ://doi.org/10.1080/00207543.2022.2127965.

[141] H. Liu, Y. Feng, J. Li et J. Luo, "Robust Petri Net Controllers for Flexible Manufacturing Systems With Multitype and Multiunit Unreliable Resources," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, p. 1-14, 2022, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics : Systems.

[142] J. Luo, Z. Liu, M. Zhou, K. Xing, X. Wang, X. Li et H. Liu, "Robust deadlock control of automated manufacturing systems with multiple unreliable resources," *Information Sciences*, p. 401-415, 1er avr. 2019.

[143] Z. Zhang, G. Liu, K. Barkaoui et Z. Li, "Adaptive Deadlock Control for a Class of Petri Nets With Unreliable Resources," *IEEE Transactions on Systems, Man, and Cybernetics :*

Systems, p. 3113-3125, mai 2022, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics : Systems.

[144] M. Lawley et W. Sulistyono, "Robust supervisory control policies for manufacturing systems with unreliable resources," *IEEE Transactions on Robotics and Automation*, p. 346-359, juin 2002, Conference Name : IEEE Transactions on Robotics and Automation.

[145] F.-S. Hsieh, "Fault-tolerant deadlock avoidance algorithm for assembly processes," *IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans*, p. 65-79, jan. 2004, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics - Part A : Systems and Humans.

[146] Y. Cheng, H. Hu et Y. Liu, "Robust supervisor synthesis for automated manufacturing systems using Petri nets," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, ISSN : 2161-8089, août 2015, p. 1029-1035.

[147] H. Yue, K. Xing, H. Hu, W. Wu et H. Su, "Petri-net-based robust supervisory control of automated manufacturing systems," *Control Engineering Practice*, p. 176-189, 1er sept. 2016.

[148] X. Wang et H. Hu, "A robust control approach to AMSs allowing multiple types of resources via Petri Nets," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, déc. 2017, p. 2354-2359.

[149] J. Luo, K. Xing et Y. Wu, "Robust supervisory control policy for automated manufacturing systems with a single unreliable resource," *Transactions of the Institute of Measurement and Control*, p. 793-806, 1er juin 2017, Publisher : SAGE Publications Ltd STM.

[150] G. Liu, P. Li, N. Wu et L. Yin, "Two-step approach to robust deadlock control in automated manufacturing systems with multiple resource failures," *Journal of the Chinese Institute of Engineers*, p. 452-462, 18 août 2018, Publisher : Taylor et Francis _eprint : https ://doi.org/10.1080/02533839.2018.1498023.

[151] J. Luo, K. Xing et M. Zhou, "Deadlock and blockage control of automated manufacturing systems with an unreliable resource," *Asian Journal of Control*, p. 334-345, 2020, _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/asjc.1856.

[152] H. Yue, K. Xing, H. Hu, W. Wu et H. Su, "Supervisory Control of Deadlock-Prone Production Systems With Routing Flexibility and Unreliable Resources," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, p. 3528-3540, oct. 2020, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics : Systems.

[153] X. Wang et H. Hu, "A Robust Control Approach to Automated Manufacturing Systems Allowing Multitype and Multiquantity of Resources With Petri Nets," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, p. 3499-3514, oct. 2020, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics : Systems.

[154] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams et A. Hahn, "Guide to industrial control systems (ICS) security," National Institute of Standards et Technology, juin 2015, NIST SP 800-82r2.

[155] R. Langner, "Stuxnet : Dissecting a Cyberwarfare Weapon," *IEEE Security Privacy*, p. 49-51, mai 2011.

[156] C. Escudero, F. Sicard et E. Zamai, "Process-Aware Model based IDSs for Industrial Control Systems Cybersecurity : Approaches, Limits and Further Research," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2018, p. 605-612.

[157] T. Alladi, V. Chamola et S. Zeadally, "Industrial control systems : cyberattack trends and countermeasures," *Computer Communications*, p. 1-8, 1er avr. 2020.

[158] M. D. Firoozjaei, N. Mahmoudyar, Y. Baseri et A. A. Ghorbani, "An evaluation framework for industrial control system cyber incidents," *International Journal of Critical Infrastructure Protection*, p. 100 487, 4 oct. 2021.

[159] T. Miller, A. Staves, S. Maesschalck, M. Sturdee et B. Green, "Looking back to look forward : lessons learnt from cyber-attacks on industrial control systems," *International Journal of Critical Infrastructure Protection*, p. 100 464, 15 juill. 2021.

[160] A. Hassanzadeh, A. Rasekh, S. Galelli, M. Aghashahi, R. Taormina, A. Ostfeld et M. K. Banks, "A Review of Cybersecurity Incidents in the Water Sector," *Journal of Environmental Engineering*, p. 03 120 003, 1er mai 2020, Publisher : American Society of Civil Engineers.

[161] A. Ribeiro. "Hackers attempt to poison water treatment plant in florida," Industrial Cyber. (10 fév. 2021), adresse : https://staging.industrialcyber.co/news/hackers-attempt-to-poison-water-treatment-plant-in-florida/ (visité le 29/03/2023).

[162] P. l. r. d. Z. à 12 :24. "NotPetya a coûté cher à Saint-Gobain," ZDNet France. (), adresse : https://www.zdnet.fr/actualites/notpetya-a-coute-cher-a-saint-gobain-39855594.html (visité le 02/04/2019).

[163] billbriggs. "Hackers hit norsk hydro with ransomware. the company responded with transparency," Transform. Section : Feature. (16 déc. 2019), adresse : https://news.microsoft.com/transform/hackers-hit-norsk-hydro-ransomware-company-responded-transparency/ (visité le 23/09/2020).

[164] "Japanese Carmaker Honda Hit by Cyberattack | SecurityWeek.Com." (), adresse : https://www.securityweek.com/japanese-carmaker-honda-hit-cyberattack (visité le 12/03/2021).

[165] A. Ribeiro. "WestRock faces ransomware incident on its IT-OT systems," Industrial Cyber. (26 jan. 2021), adresse : https://staging.industrialcyber.co/news/westrock-faces-ransomware-incident-on-its-it-ot-systems/ (visité le 30/03/2023).

[166] I. Arghire. "Boat building giant beneteau says cyberattack disrupted production," SecurityWeek. (1er mars 2021), adresse : https://www.securityweek.com/boat-building-giant-beneteau-says-cyberattack-disrupted-production/ (visité le 30/03/2023).

[167] "Bombardier suffers cybersecurity attack in isolated IT network," Industrial Cyber. (25 fév. 2021), adresse : https://industrialcyber.co/threats-attacks/industrial-cyber-attacks/bombardier-suffers-cybersecurity-attack-away-from-its-main-it-network/ (visité le 03/03/2021).

[168] "Cyberattack Forces Brewery Shutdown at Molson Coors | SecurityWeek.Com." (), adresse : https://www.securityweek.com/cyberattack-forces-brewery-shutdown-molson-coors (visité le 12/03/2021).

[169] "H1 2022 – a brief overview of the main incidents in industrial cybersecurity | kaspersky ICS CERT." (8 sept. 2022), adresse : https://ics-cert.kaspersky.com/publications/reports/2022/09/08/h1-2022-a-brief-overview-of-the-main-incidents-in-industrial-cybersecurity/ (visité le 30/03/2023).

[170] "H2 2022 – brief overview of main incidents in industrial cybersecurity | kaspersky ICS CERT." (15 mars 2023), adresse : https://ics-cert.kaspersky.com/publications/reports/2023/03/15/h2-2022-brief-overview-of-main-incidents-in-industrial-cybersecurity/ (visité le 30/03/2023).

[171] I. Arghire. "Ransomware attack disrupts manufacturing at KP snacks," SecurityWeek. (3 fév. 2022), adresse : https://www.securityweek.com/ransomware-attack-disrupts-manufacturing-kp-snacks/ (visité le 30/03/2023).

[172] I. Arghire. "Wind turbine giant nordex shuts down IT systems in response to cyberattack," SecurityWeek. (5 avr. 2022), adresse : https://www.securityweek.com/wind-turbine-giant-nordex-shuts-down-it-systems-response-cyberattack/ (visité le 30/03/2023).

[173] A. Press. "Toyota's japan production halted over suspected cyberattack," SecurityWeek. (28 fév. 2022), adresse : https://www.securityweek.com/toyotas-japan-production-halted-over-suspected-cyberattack/ (visité le 30/03/2023).

[174] I. Arghire. "Ransomware attack hits production facilities of agricultural equipment giant AGCO," SecurityWeek. (9 mai 2022), adresse : https://www.securityweek.com/ransomware-attack-hits-production-facilities-agricultural-equipment-giant-agco/ (visité le 30/03/2023).

[175] I. Arghire. "Foxconn confirms ransomware hit factory in mexico," SecurityWeek. (3 juin 2022), adresse : https://www.securityweek.com/foxconn-confirms-ransomware-hit-factory-mexico/ (visité le 30/03/2023).

[176] A. Press. "Meat producer ransomware attack disrupts global production," SecurityWeek. (1er juin 2021), adresse : https://www.securityweek.com/meat-producer-ransomware-attack-disrupts-global-production/ (visité le 30/03/2023).

[177] A. Ribeiro. "Ransomware attack temporarily shuts down dole production, disrupts food supplies," Industrial Cyber. (23 fév. 2023), adresse : https://industrialcyber.co/news/ransomware-attack-temporarily-shuts-down-dole-production-disrupts-food-supplies/ (visité le 30/03/2023).

[178] R. S. Manager Account. "The SolarWinds cyberattack and its effect on manufacturing," Global Electronic Services. (30 déc. 2020), adresse : https://gesrepair.com/the-solarwinds-cyberattack-and-its-effect-on-manufacturing/ (visité le 30/03/2023).

[179] E. Kovacs. "Hundreds of industrial organizations received sunburst malware in SolarWinds attack," SecurityWeek. (27 jan. 2021), adresse : https://www.securityweek.com/hundreds-industrial-organizations-received-sunburst-malware-solarwinds-attack/ (visité le 30/03/2023).

[180] E. Kovacs. "Industrial organizations targeted in log4shell attacks," SecurityWeek. (14 déc. 2021), adresse : https://www.securityweek.com/industrial-organizations-targeted-log4shell-attacks/ (visité le 30/03/2023).

[181] A. Ribeiro. "Log4shell vulnerability may have affected close to 10 percent of ICS systems globally," Industrial Cyber. (3 jan. 2022), adresse : https://staging.industrialcyber.co/news/log4shell-vulnerability-may-have-affected-close-to-10-percent-of-ics-systems-globally/ (visité le 30/03/2023).

[182] "Claroty biannual ICS risk et vulnerability report : 2h 2020," Claroty. (), adresse : https://claroty.com/resources/reports/claroty-biannual-ics-risk-vulnerability-report-2h-2020 (visité le 30/03/2023).

[183] "State of XIoT security : 2h 2022," Claroty. (), adresse : https://claroty.com/resources/reports/state-of-xiot-security-2h-2022 (visité le 30/03/2023).

[184] "Cybersecurity research report january 2023," Nozomi Networks. (), adresse : https://www.nozominetworks.com/iot-ot-cybersecurity-research-report-january-2023/ (visité le 30/03/2023).

[185] "ICS vulnerabilities and CVEs second half of 2022," SynSaber | Industrial Cybersecurity. (), adresse : https://synsaber.com/resources/datasheets/ics-cve-reports/ics-vulnerabilities-and-cves-second-half-2022/ (visité le 30/03/2023).

[186] "The State of OT/ICS Cybersecurity in 2022 and Beyond." (), adresse : https://www.sans.org/white-papers/state-ics-ot-cybersecurity-2022-beyond/ (visité le 30/03/2023).

[187] "IBM security x-force threat intelligence index 2023." (20 mars 2023), adresse : https://www.ibm.com/reports/threat-intelligence (visité le 30/03/2023).

[188] "The 2022 state of operational technology," SCADAfence. (), adresse : https://www.scadafence.com/scadafence-ot-industrial-report-2022/ (visité le 28/09/2023).

[189] "Threat landscape for industrial automation systems. statistics for h2 2022 | kaspersky ICS CERT." (6 mars 2023), adresse : https://ics-cert.kaspersky.com/publications/reports/2023/03/06/threat-landscape-for-industrial-automation-systems-statistics-for-h2-2022/ (visité le 30/03/2023).

[190] "APT attacks on industrial organizations in h2 2022 | kaspersky ICS CERT." (24 mars 2023), adresse : https://ics-cert.kaspersky.com/publications/reports/2023/03/24/apt-attacks-on-industrial-organizations-in-h2-2022/ (visité le 30/03/2023).

[191] "2022 ICS/OT cybersecurity year in review report | dragos." (), adresse : https://hub.dragos.com/ics-cybersecurity-year-in-review-2022 (visité le 30/03/2023).

[192] "The state of industrial cybersecurity survey report." (), adresse : https://resources.trendmicro.com/IoT-survey-report.html (visité le 30/03/2023).

[193] R. Lee et M. Assante, "The industrial control system cyber kill chain," *SANS Survey*, p. 23, 2015.

[194] "Matrix | MITRE ATTetCK®." (), adresse : https://attack.mitre.org/matrices/ics/ (visité le 31/03/2023).

[195] I. Nai Fovino, M. Masera et A. De Cian, "Integrating cyber attacks within fault trees," *Reliability Engineering et System Safety*, ESREL 2007, the 18th European Safety and Reliability Conference, p. 1394-1402, 1er sept. 2009.

[196] E. J. Byres, M. Franz et D. Miller, "The use of attack trees in assessing vulnerabilities in SCADA systems," p. 10,

[197] M. H. Rahman, T. Wuest et M. Shafae, "Manufacturing cybersecurity threat attributes and countermeasures : review, meta-taxonomy, and use cases of cyberattack taxonomies," *Journal of Manufacturing Systems*, p. 196-208, 1er juin 2023.

[198] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang et S. Sastry, "Attacks Against Process Control Systems : Risk Assessment, Detection, and Response," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, sér. ASIACCS '11, event-place : Hong Kong, China, New York, NY, USA : ACM, 2011, p. 355-366.

[199] L. K. Carvalho, Y.-C. Wu, R. Kwong et S. Lafortune, "Detection and mitigation of classes of attacks in supervisory control systems," *Automatica*, p. 121-133, 1er nov. 2018.

[200] Z. Basnight, J. Butts, J. Lopez et T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, p. 76-84, 1er juin 2013.

[201] C. Schuett, J. Butts et S. Dunlap, "An evaluation of modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, p. 61-68, 1er mars 2014.

[202] N. Govil, A. Agrawal et N. O. Tippenhauer, "On ladder logic bombs in industrial control systems," in *Computer Security*, S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, C. Kalloniatis, J. Mylopoulos, A. Antón et S. Gritzalis, éd., sér. Lecture Notes in Computer Science, Cham : Springer International Publishing, 2018, p. 110-126.

[203] L. Cheng, K. Tian, D. Yao, L. Sha et R. A. Beyah, "Checking is Believing : Event-Aware Program Anomaly Detection in Cyber-Physical Systems," *IEEE Transactions on Dependable and Secure Computing*, p. 1-1, 2019, Conference Name : IEEE Transactions on Dependable and Secure Computing.

[204] H. Yoo, S. Kalle, J. Smith et I. Ahmed, *Overshadow PLC to Detect Remote Control-Logic Injection Attacks*. 23 avr. 2019.

[205] B. Zhu, A. Joseph et S. Sastry, "A taxonomy of cyber attacks on SCADA systems," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, Dalian, China : IEEE, oct. 2011, p. 380-388.

[206] B. Green, M. Krotofil et A. Abbasi, "On the Significance of Process Comprehension for Conducting Targeted ICS Attacks," in *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, sér. CPS '17, New York, NY, USA : Association for Computing Machinery, 3 nov. 2017, p. 57-67.

[207] Y. Ashibani et Q. H. Mahmoud, "Cyber physical systems security : Analysis, challenges and solutions," *Computers et Security*, p. 81-97, 1er juill. 2017.

[208] R. Fritz et P. Zhang, "Modeling and detection of cyber attacks on discrete event systems," *IFAC-PapersOnLine*, p. 285-290, 1er jan. 2018.

[209] M. Wu et Y. B. Moon, "Taxonomy for secure CyberManufacturing systems," **presentedat**ASME 2018 International Mechanical Engineering Congress and Exposition, American Society of Mechanical Engineers Digital Collection, 15 jan. 2019.

[210] M. Yampolskiy, W. E. King, J. Gatlin, S. Belikovetsky, A. Brown, A. Skjellum et Y. Elovici, "Security of additive manufacturing : attack taxonomy and survey," *Additive Manufacturing*, p. 431-457, 1er mai 2018.

[211] P. M. Lima, L. K. Carvalho et M. V. Moreira, "CONFIDENTIALITY OF CYBER-PHYSICAL SYSTEMS USING EVENT-BASED CRYPTOGRAPHY," *IFAC-PapersOnLine*, 21th IFAC World Congress, p. 1735-1740, 1er jan. 2020.

[212] L. Lin et R. Su, "Synthesis of covert actuator and sensor attackers," *Automatica*, p. 109 714, 1er août 2021.

[213] P. M. Lima, L. K. Carvalho et M. V. Moreira, "Ensuring confidentiality of cyber-physical systems using event-based cryptography," *Information Sciences*, p. 119-135, 1er avr. 2023.

[214] A. Abbasi et M. Hashemi, "Ghost in the PLC designing an undetectable programmable logic controller rootkit via pin control attack," p. 35,

[215] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg et R. Candell, "A Survey of Physics-Based Attack Detection in Cyber-Physical Systems," *ACM Comput. Surv.*, 76 :1-76 :36, juill. 2018.

[216] U. P. D. Ani, H. M. He et A. Tiwari, "Review of cybersecurity issues in industrial critical infrastructure : manufacturing in perspective," *Journal of Cyber Security Technology*, p. 32-74, 2 jan. 2017.

[217] C. Escudero, P. Massioni, E. Zamai et B. Raison, "Analysis, prevention, and feasibility assessment of stealthy ageing attacks on dynamical systems," *IET Control Theory et Applications*, p. 381-397, 2022. eprint : https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/cth2.12178.

[218] E. M. Merouane, C. Escudero, F. Sicard et E. Zamai, "Aging Attacks against Electro-Mechanical Actuators from Control Signal Manipulation," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, ISSN : 2643-2978, fév. 2020, p. 133-138.

[219] A. Beaudet, Zamaï, C. Escudero et E. Dumitrescu, "Malicious Origin of Deadlocks in Flexible Manufacturing Systems," *IFAC-PapersOnLine*, 16th IFAC Workshop on Discrete Event Systems WODES 2022, p. 100-107, 1er jan. 2022.

[220] Q. Zhang, Z. Li, C. Seatzu et A. Giua, "Stealthy Attacks for Partially-Observed Discrete Event Systems," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, ISSN : 1946-0759, sept. 2018, p. 1161-1164.

[221] A. Khoumsi, "Sensor and Actuator Attacks of Cyber-Physical Systems : A Study Based on Supervisory Control of Discrete Event Systems," in *2019 8th International Conference on Systems and Control (ICSC)*, ISSN : 2379-0067, oct. 2019, p. 176-182.

[222] Q. Zhang, C. Seatzu, Z. Li et A. Giua, "Stealthy sensor attacks for plants modeled by labeled petri nets," *IFAC-PapersOnLine*, 15th IFAC Workshop on Discrete Event Systems WODES 2020 - Rio de Janeiro, Brazil, 11-13 November 2020, p. 14-20, 1er jan. 2020.

[223] R. Ammour, L. Brenner, I. Demongodin, S. Amari et D. Lefebvre, "Costs analysis of stealthy attacks with bounded output synchronized Petri nets," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, ISSN : 2161-8089, août 2021, p. 799-804.

[224] Z. Jakovljevic, V. Lesi et M. Pajic, "Attacks on Distributed Sequential Control in Manufacturing Automation," *IEEE Transactions on Industrial Informatics*, p. 775-786, fév. 2021, Conference Name : IEEE Transactions on Industrial Informatics.

[225] *HyPLC : Hybrid Programmable Logic Controller Program Translation for Verification*. 4 avr. 2019.

[226] Z. Drias, A. Serhrouchni et O. Vogel, "Taxonomy of attacks on industrial control protocols," in *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, ISSN : 2162-190X, juill. 2015, p. 1-6.

[227] P. A. S. Ralston, J. H. Graham et J. L. Hieb, "Cyber security risk assessment for SCADA and DCS networks," *ISA Transactions*, p. 583-594, 1er oct. 2007.

[228] Y. Cherdantseva, P. Burnap, A. Blyth, P. Eden, K. Jones, H. Soulsby et K. Stoddart, "A review of cyber security risk assessment methods for SCADA systems," *Computers et Security*, p. 1-27, 1er fév. 2016.

[229] C. Escudero et E. Zamaï, "Prevention of Aging Attacks : Malicious Nature of the Control Signal," in *2019 International Automatic Control Conference (CACS)*, nov. 2019, p. 1-6.

[230] C. Escudero, P. Massioni, G. Scorletti et E. Zamaï, "Security of Control Systems : Prevention of Aging Attacks by means of Convex Robust Simulation Forecasts," *IFAC-PapersOnLine*, 21st IFAC World Congress, p. 4452-4459, 1er jan. 2020.

[231] C. Escudero, C. Murguia, P. Massioni et E. Zamaï, "Enforcing Safety under Actuator Injection Attacks through Input Filtering," in *2022 European Control Conference (ECC)*, juill. 2022, p. 1521-1528.

[232] A. Saboori et C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *2007 46th IEEE Conference on Decision and Control*, ISSN : 0191-2216, déc. 2007, p. 5056-5061.

[233] R. Jacob, J.-J. Lesage et J.-M. Faure, "Overview of discrete event systems opacity : models, validation, and quantification," *Annual Reviews in Control*, p. 135-146, 1er jan. 2016.

[234] Y. Tong, Z. Li, C. Seatzu et A. Giua, "Verification of State-Based Opacity Using Petri Nets," *IEEE Transactions on Automatic Control*, p. 2823-2837, juin 2017, Conference Name : IEEE Transactions on Automatic Control.

[235] S. Lafortune, F. Lin et C. N. Hadjicostis, "On the history of diagnosability and opacity in discrete event systems," *Annual Reviews in Control*, p. 257-266, 1er jan. 2018.

[236] C. Gao, C. Seatzu, Z. Li et A. Giua, "Multiple Attacks Detection on Discrete Event Systems," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, ISSN : 2577-1655, oct. 2019, p. 2352-2357.

[237] Q. Zhang, C. Seatzu, Z. Li et A. Giua, "A framework for the analysis of supervised discrete event systems under attack," in *15th European Workshop on Advanced Control and Diagnosis (ACD 2019)*, E. Zattoni, S. Simani et G. Conte, éd., sér. Lecture Notes in Control and Information Sciences - Proceedings, Cham : Springer International Publishing, 2022, p. 529-546.

[238] R. Ammour, S. Amari, L. Brenner, I. Demongodin et D. Lefebvre, "Observer design for labeled finite automata with inputs under stealthy actuators attacks," *IFAC-PapersOnLine*, 16th IFAC Workshop on Discrete Event Systems WODES 2022, p. 46-51, 1er jan. 2022.

[239] Y. Wang, Y. Li, Z. Yu, N. Wu et Z. Li, "Supervisory control of discrete-event systems under external attacks," *Information Sciences*, p. 398-413, 1er juill. 2021.

[240] D. You, S. Wang, M. Zhou et C. Seatzu, "Supervisory Control of Petri Nets in the Presence of Replacement Attacks," *IEEE Transactions on Automatic Control*, p. 1-1, 2021, Conference Name : IEEE Transactions on Automatic Control.

[241] D. You, S. Wang et C. Seatzu, "A Liveness-Enforcing Supervisor Tolerant to Sensor-Reading Modification Attacks," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, p. 2398-2411, avr. 2022, Conference Name : IEEE Transactions on Systems, Man, and Cybernetics : Systems.

[242] R. Meira-Goes, S. Lafortune et H. Marchand, "Synthesis of Supervisors Robust Against Sensor Deception Attacks," *IEEE Transactions on Automatic Control*, p. 1-1, 2021, Conference Name : IEEE Transactions on Automatic Control.

[243] R. Su, "About Existence of Resilient Supervisors against Smart Sensor Attacks," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, ISSN : 2576-2370, déc. 2022, p. 4263-4269.

[244] J. Zaddach, L. Bruno, A. Francillon et D. Balzarotti, "Avatar : a framework to support dynamic security analysis of embedded systems' firmwares," in *Proceedings 2014 Network and Distributed System Security Symposium*, San Diego, CA : Internet Society, 2014.

[245] H. C. Barroso et J. A. Gaspar, "Validation of Discrete Event Processes implemented on PLCs based on Petri Nets," in *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, avr. 2021, p. 143-148.

[246] S. McLaughlin, S. Zonouz, D. Pohly et P. McDaniel, "A trusted safety verifier for process controller code," in *Proceedings 2014 Network and Distributed System Security Symposium*, San Diego, CA : Internet Society, 2014.

[247] L. Garcia, S. Zonouz, D. Wei et L. P. de Aguiar, "Detecting PLC control corruption via on-device runtime verification," in *2016 Resilience Week (RWS)*, août 2016, p. 67-72.

[248] M. Zhang, C.-Y. Chen, B.-C. Kao, Y. Qamsane, Y. Shao, Y. Lin, E. Shi, S. Mohan, K. Barton, J. Moyne et Z. M. Mao, "Towards Automated Safety Vetting of PLC Code in Real-World Plants," in *2019 IEEE Symposium on Security and Privacy (SP)*, ISSN : 2375-1207, mai 2019, p. 522-538.

[249] R. Sun, A. Mera, L. Lu et D. Choffnes, *SoK : Attacks on Industrial Control Logic and Formal Verification-Based Defenses*. 9 juin 2020.

[250] E. D. Knapp et J. Langill, *Industrial Network Security : Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 9 déc. 2014, 460 p., Google-Books-ID : V2RzAwAAQBAJ.

[251] T. Bartman et K. Carson, "Securing communications for SCADA and critical industrial systems," in *2016 69th Annual Conference for Protective Relay Engineers (CPRE)*, avr. 2016, p. 1-10.

[252] Z. Yang, S. Adepu et J. Zhou, "Opportunities and Challenges in Securing Critical Infrastructures Through Cryptography," *IEEE Security Privacy*, p. 2-11, 2021, Conference Name : IEEE Security Privacy.

[253] A. Ayub, H. Yoo et I. Ahmed, *Empirical Study of PLC Authentication Protocols in Industrial Control Systems*. 24 avr. 2021.

[254] A. A. Mercado-Velazquez, P. J. Escamilla-Ambrosio et F. Ortiz-Rodríguez, "A Moving Target Defense Strategy for Internet of Things Cybersecurity," *IEEE Access*, p. 118 406-118 418, 2021, Conference Name : IEEE Access.

[255] J. S. Li, C. G. Liu, C. J. Wu, C. C. Wu, C. W. Huang, C. F. Li et I.-H. Liu, "Design of industrial control system secure communication using moving target defense with legacy infrastructure," *Sensors and Materials*, p. 3415-3424, 2021.

[256] M. Cheminod, L. Durante, L. Seno, F. Valenza, A. Valenzano et C. Zunino, "Leveraging SDN to improve security in industrial networks," in *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, mai 2017, p. 1-7.

[257] D. Henneke, L. Wisniewski et J. Jasperneite, "Analysis of realizing a future industrial network by means of Software-Defined Networking (SDN)," in *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, mai 2016, p. 1-4.

[258] A. Tsuchiya, F. Fraile, I. Koshijima, A. Ortiz et R. Poler, "Software defined networking firewall for industry 4.0 manufacturing systems," *Journal of Industrial Engineering and Management (JIEM)*, p. 318-333, 2018, Publisher : Barcelona : OmniaScience.

[259] A. Melis, D. Berardi, C. Contoli, F. Callegati, F. Esposito et M. Prandini, "A Policy Checker Approach for Secure Industrial SDN," in *2018 2nd Cyber Security in Networking Conference (CSNet)*, oct. 2018, p. 1-7.

[260] R. Mitchell et I.-R. Chen, "A Survey of Intrusion Detection Techniques for Cyber-physical Systems," *ACM Comput. Surv.*, 55 :1-55 :29, mars 2014.

[261] S. V. B. Rakas, M. D. Stojanovic et J. D. Markovic-Petrovic, "A Review of Research Work on Network-Based SCADA Intrusion Detection Systems," *IEEE Access*, p. 93 083-93 108, 2020, Conference Name : IEEE Access.

[262] E. Vasilomanolakis, S. Srinivasa, C. G. Cordero et M. Muhlhauser, "Multi-stage attack detection and signature generation with ICS honeypots," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, ISSN : 2374-9709, avr. 2016, p. 1227-1232.

[263] *Intrusion Detection of the ICS Protocol EtherCAT*. 27 mars 2017.

[264] R. Colelli, S. Panzieri et F. Pascucci, "Securing connection between IT and OT : the Fog Intrusion Detection System prospective," in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)*, juin 2019, p. 444-448.

[265] L. O. Nweke, "A Survey of Specification-based Intrusion Detection Techniques for Cyber-Physical Systems," *12*, 2021.

[266] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. N. Fovino et A. Trombetta, "A Multidimensional Critical State Analysis for Detecting Intrusions in SCADA Systems," *IEEE Transactions on Industrial Informatics*, p. 179-186, mai 2011.

[267] A. Khalili, A. Sami, A. Khozaei et S. Pouresmaeeli, "SIDS : state-based intrusion detection for stage-based cyber physical systems," *International Journal of Critical Infrastructure Protection*, p. 113-124, 1er sept. 2018.

[268] Y. Wang, Z. Xu, J. Zhang, L. Xu, H. Wang et G. Gu, "SRID : state relation based intrusion detection for false data injection attacks in SCADA," in *Computer Security - ESORICS 2014*, M. Kutylowski et J. Vaidya, éd., sér. Lecture Notes in Computer Science, Cham : Springer International Publishing, 2014, p. 401-418.

[269] Q. Lin, S. Adepu, S. Verwer et A. Mathur, "TABOR : A Graphical Model-based Approach for Anomaly Detection in Industrial Control Systems," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, sér. ASIACCS '18, New York, NY, USA : Association for Computing Machinery, 29 mai 2018, p. 525-536.

[270] D. Hadziosmanovic, R. Sommer, E. Zambon et P. H. Hartel, "Through the eye of the PLC : semantic security monitoring for industrial processes," **presentedat**Proceedings of the 30th Annual Computer Security Applications Conference, ACM, 12 août 2014, p. 126-135.

[271] O. Koucham, S. Mocanu, G. Hiet, J.-M. Thiriet et F. Majorczyk, "Detecting process-aware attacks in sequential control systems," in *Secure IT Systems*, B. B. Brumley et J. Röning, éd., sér. Lecture Notes in Computer Science, Cham : Springer International Publishing, 2016, p. 20-36.

[272] G. Delaval, A. Hore, S. Mocanu, L. Muller et E. Rutten, "Discrete Control of Response for Cybersecurity in Industrial Control," in *IFAC 2020 - IFAC World Congress 2020*, sér. Proc. of the 21st IFAC World Congress, Berlin, Germany, juill. 2020, p. 1-8.

[273] M. Hussain, E. Foo et S. Suriadi, "An Improved Industrial Control System Device Logs Processing Method for Process-Based Anomaly Detection," in *2019 International Conference on Frontiers of Information Technology (FIT)*, ISSN : 2334-3141, déc. 2019, p. 150-1505.

[274] B. Ferling, J. Chromik, M. Caselli et A. Remke, "Intrusion detection for sequence-based attacks with reduced traffic models," in *Measurement, Modelling and Evaluation of Computing Systems*, R. German, K.-S. Hielscher et U. R. Krieger, éd., sér. Lecture Notes in Computer Science, Cham : Springer International Publishing, 2018, p. 53-67.

[275] M. Caselli, E. Zambon, J. Amann, R. Sommer et F. Kargl, "Specification mining for intrusion detection in networked control systems," **presentedat**25th Security Symposium Security 16, 2016, p. 791-806.

[276] D. Myers, K. Radke, S. Suriadi et E. Foo, "Process discovery for industrial control system cyber attack detection," in *ICT Systems Security and Privacy Protection*, S. De Capitani di Vimercati et F. Martinelli, éd., sér. IFIP Advances in Information and Communication Technology, Cham : Springer International Publishing, 2017, p. 61-75.

[277] C. Ahmed, M. Ochoa, J. Zhou et A. Mathur, *Scanning the Cycle : Timing-based Authentication on PLCs*. 17 fév. 2021.

[278] M. Faisal, A. A. Cardenas et A. Wool, "Modeling Modbus TCP for intrusion detection," in *2016 IEEE Conference on Communications and Network Security (CNS)*, oct. 2016, p. 386-390.

[279] A. Kleinmann et A. Wool, "Accurate Modeling of the Siemens S7 SCADA Protocol for Intrusion Detection and Digital Forensics," en, *Journal of Digital Forensics, Security and Law*, 2014.

[280] R. Berthier et W. H. Sanders, "Specification-Based Intrusion Detection for Advanced Metering Infrastructures," in *2011 IEEE 17th Pacific Rim International Symposium on Dependable Computing*, déc. 2011, p. 184-193.

[281] N. Goldenberg et A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *International Journal of Critical Infrastructure Protection*, p. 63-75, juin 2013.

[282] M. Caselli, E. Zambon, J. Petit et F. Kargl, "Modeling message sequences for intrusion detection in industrial control systems," in *Critical Infrastructure Protection IX*, M. Rice et S. Shenoi, éd., sér. IFIP Advances in Information and Communication Technology, Cham : Springer International Publishing, 2015, p. 49-71.

[283] B. Genge, D. A. Rusu et P. Haller, "A connection pattern-based approach to detect network traffic anomalies in critical infrastructures," in *Proceedings of the Seventh European Workshop on System Security*, sér. EuroSec '14, New York, NY, USA : Association for Computing Machinery, 13 avr. 2014, p. 1-6.

[284] Q. S. Qassim, A. R. Ahmad, R. Ismail, A. A. Bakar, F. A. Rahim, M. Z. Mokhtar, R. Ramli, B. M. Yusof et M. N. Mahdi, "An Anomaly Detection Technique for Deception Attacks in Industrial Control Systems," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, mai 2019, p. 267-272.

[285] W. Yusheng, F. Kefeng, L. Yingxu, L. Zenghui, Z. Ruikang, Y. Xiangzhen et L. Lin, "Intrusion Detection of Industrial Control System Based on Modbus TCP Protocol," in *2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS)*, mars 2017, p. 156-162.

[286] D. I. Urbina, J. Giraldo, A. A. Cardenas, J. Valente, M. Faisal, N. O. Tippenhauer, J. Ruths, R. Candell et H. Sandberg, "Survey and new directions for physics based attack detection in control systems," National Institute of Standards et Technology, Gaithersburg, MD, nov. 2016.

[287] Y. Mo, R. Chabukswar et B. Sinopoli, "Detecting Integrity Attacks on SCADA Systems," *IEEE Transactions on Control Systems Technology*, p. 1396-1407, juill. 2014.

[288] C. Escudero, M. S. Chong, P. Massioni et E. Zamaï, "Safety monitoring under stealthy sensor injection attacks using reachable sets," *IFAC-PapersOnLine*, 22nd IFAC World Congress, p. 1833-1840, 1er jan. 2023.

[289] S. Abdellaoui, D. D. K. Nguyen, E. Dumitrescu, C. Escudero et E. Zamai, "Cybersécurité des Systèmes de Surveillance des Appareils de Voie Ferroviaires," in *14ème colloque sur la Modélisation des Systèmes Réactifs*, Toulouse, France, nov. 2023.

[290] S. Abdellaoui, E. Dumitrescu, C. Escudero et E. Zamaï, "Cyber Threat Assessment in Monitoring Turnout Railway Systems," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, ISSN : 1946-0759, sept. 2023, p. 1-8.

[291] F. Sicard, E. Zamaï et J.-M. Flaus, "Distance Concept Based Filter Approach for Detection of Cyberattacks on Industrial Control Systems," in *20th World Congress of the International Federation of Automatic Control (IFAC 2017)*, sér. Preprints IFAC WC 2017 Toulouse. Toulouse, France : IFAC, juill. 2017.

[292] F. Sicard, C. Escudero, Zamaï et J.-M. Flaus, "From ICS Attacks' Analysis to the S.A.F.E. Approach : Implementation of Filters based on Behavioral Models and Critical State Distance for ICS Cybersecurity," in *2nd Cyber Security In Networking Conference*, sér. 2nd Cyber Security In Networking Conference (CSNet'18) Proceedings, Paris, France : IEEE, oct. 2018, p. 8.

[293] F. Sicard, Zamai et J.-M. Flaus, "Critical states distance filter based approach for detection and blockage of cyberattacks in industrial control systems," in *Diagnosability, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems*, M. Sayed-Mouchaweh, éd., Cham : Springer International Publishing, 2018, p. 117-145.

[294] F. Sicard, E. Zamai et J.-M. Flaus, "Filters based Approach with Temporal and Combinational Constraints for Cybersecurity of Industrial Control Systems," *IFAC-PapersOnLine*, 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018, p. 96-103, 1er jan. 2018.

[295] F. Sicard, E. Zamai et J.-M. Flaus, "An approach based on behavioral models and critical states distance notion for improving cybersecurity of industrial control systems," *Reliability Engineering System Safety*, p. 584-603, 1er août 2019.

[296] F. Sicard, E. Hotellier, J. S. Pérez-Olivares et E. Zamaï, "Optimization of Process-Aware Attack Detection for Industrial Control Systems Security," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, ISSN : 1946-0759, sept. 2020, p. 889-896.

[297] A. Beaudet, F. Sicard, C. Escudero et E. Zamaï, "Process-Aware Model-based Intrusion Detection System on Filtering Approach : Further Investigations," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, ISSN : 2643-2978, fév. 2020, p. 310-315.

[298] R. Ammour, S. Amari, L. Brenner, I. Demongodin et D. Lefebvre, "Observer Design for Bounded Output Synchronized Petri Nets," in *2021 European Control Conference (ECC)*, juin 2021, p. 746-751.

[299] Q. Zhang, C. Seatzu, Z. Li et A. Giua, *State estimation under attack in partially observed discrete event systems*. 14 jan. 2021.

[300] R. Fritz et P. Zhang, "Detection and Localization of Stealthy Cyber Attacks in Cyber Physical Discrete Event Systems," *IEEE Transactions on Automatic Control*, p. 1-8, 2023, Conference Name : IEEE Transactions on Automatic Control.

[301] A. E. Elhabashy, L. J. Wells et J. A. Camelio, "Cyber Physical Security Research Efforts in Manufacturing : A Literature Review," en, *Procedia Manufacturing*, 47th SME

North American Manufacturing Research Conference, NAMRC 47, Pennsylvania, USA. P. 921-931, jan. 2019.

[302] M. Wu et Y. B. Moon, "Intrusion detection system for cyber manufacturing system," *Journal of Manufacturing Science and Engineering*, 1er mars 2019, Publisher : American Society of Mechanical Engineers Digital Collection.

[303] M. H. Rahman et M. Shafae, "Physics based detection of cyber attacks in manufacturing systems : a machining case study," *Journal of Manufacturing Systems*, 28 avr. 2022.

[304] A. Robles Durazno, N. Moradpoor, J. McWhinnie, G. Russell et I. Maneru Marin, "PLC memory attack detection and response in a clean water supply system," *International Journal of Critical Infrastructure Protection*, p. 100 300, 1er sept. 2019.

[305] A. Beaudet, C. Escudero et E. Zamai, "Malicious anomaly detection approaches robustness in manufacturing ICSs," *IFAC-PapersOnLine*, 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021, p. 146-151, 1er jan. 2021.

[306] E. Kovacs. "Serious Vulnerabilities Found in Schneider Electric Power Meters | SecurityWeek.Com." (), adresse : https://www.securityweek.com/serious-vulnerabilities-found-schneider-electric-power-meters (visité le 12/03/2021).

[307] E. Kovacs. "Unprotected Private Key Allows Remote Hacking of Rockwell Controllers | SecurityWeek.Com." (), adresse : https://www.securityweek.com/unprotected-private-key-allows-remote-hacking-rockwell-controllers (visité le 01/04/2021).

[308] A. Ribeiro. "Several DoS vulnerabilities detected in mitsubishi GOT, tension controller," Industrial Cyber. (7 sept. 2021), adresse : https://industrialcyber.co/article/several-dos-vulnerabilities-detected-in-mitsubishi-got-tension-controller/ (visité le 13/09/2021).

[309] E. Kovacs. "Over 100 Building Controllers in Russia Vulnerable to Remote Hacker Attacks | SecurityWeek.Com." (), adresse : https://www.securityweek.com/over-100-building-controllers-russia-vulnerable-remote-hacker-attacks (visité le 25/03/2022).

[310] E. Kovacs. "Tractor-Trailer Brake Controllers Vulnerable to Remote Hacker Attacks | SecurityWeek.Com." (), adresse : https://www.securityweek.com/tractor-trailer-brake-controllers-vulnerable-remote-hacker-attacks (visité le 02/05/2022).

[311] E. Kovacs. "Hackers Can Make Siemens Building Automation Controllers 'Unavailable for Days' | SecurityWeek.Com." (), adresse : https://www.securityweek.com/hackers-can-make-siemens-building-automation-controllers-unavailable-days (visité le 16/05/2022).

[312] E. Kovacs. "Vulnerabilities in HID Mercury Access Controllers Allow Hackers to Unlock Doors | SecurityWeek.Com." (), adresse : https://www.securityweek.com/vulnerabilities-hid-mercury-access-controllers-allow-hackers-unlock-doors (visité le 13/06/2022).

[313] A. Ribeiro. "Vedere labs updates OT :ICEFALL findings, adds three more vulnerabilities in festo automation controllers, CODESYS equipment," Industrial Cyber. (30 nov. 2022), adresse : https://industrialcyber.co/vulnerabilities/vedere-labs-updates-oticefall-findings-adds-three-more-vulnerabilities-in-festo-automation-controllers-codesys-equipment/ (visité le 05/12/2022).

[314] E. Kovacs. "Several DoS, Code Execution Vulnerabilities Found in Rockwell Automation Controllers | SecurityWeek.Com." (), adresse : https://www.securityweek.com/

several-dos-code-execution-vulnerabilities-found-rockwell-automation-controllers (visité le 08/01/2023).

[315] E. Kovacs. "Encryption Vulnerabilities Allow Hackers to Take Control of Schneider Electric PLCs | SecurityWeek.Com." (), adresse : https://www.securityweek.com/encryption-vulnerabilities-allow-hackers-take-control-schneider-electric-plcs (visité le 19/01/2021).

[316] E. Kovacs. "Siemens Releases Patches to Prevent Remote Takeover of SIMATIC HMI Panels | SecurityWeek.Com." (), adresse : https://www.securityweek.com/siemens-releases-patches-prevent-remote-takeover-simatic-hmi-panels (visité le 16/02/2021).

[317] E. Kovacs. "Newly Disclosed Vulnerability Allows Remote Hacking of Siemens PLCs | SecurityWeek.Com." (), adresse : https://www.securityweek.com/newly-disclosed-vulnerability-allows-remote-hacking-siemens-plcs (visité le 08/06/2021).

[318] E. Kovacs. "Critical Vulnerability Can Be Exploited to Hack Schneider Electric's Modicon PLCs | SecurityWeek.Com." (), adresse : https://www.securityweek.com/critical-vulnerability-can-be-exploited-hack-schneider-electrics-modicon-plcs (visité le 14/07/2021).

[319] A. Ribeiro. "Critical vulnerability in schneider electric modicon PLCs can bypass authentication mechanisms," Industrial Cyber. (14 juill. 2021), adresse : https://industrialcyber.co/article/critical-vulnerability-in-schneider-electric-modicon-plcs-can-bypass-authentication-mechanisms/ (visité le 26/07/2021).

[320] E. Kovacs. "Vulnerability Exposes MicroLogix PLCs to Remote DoS Attacks | SecurityWeek.Com." (), adresse : https://www.securityweek.com/vulnerability-exposes-micrologix-plcs-remote-dos-attacks (visité le 26/07/2021).

[321] A. Ribeiro. "Critical vulnerability found in ThroughTek kalay p2p SDK ; can remotely exploit millions of IoT devices," Industrial Cyber. (19 août 2021), adresse : https://industrialcyber.co/article/critical-vulnerability-found-in-throughtek-kalay-p2p-sdk-can-remotely-exploit-millions-of-iot-devices/ (visité le 24/08/2021).

[322] E. Kovacs. "New Vulnerabilities Can Allow Hackers to Remotely Crash Siemens PLCs | SecurityWeek.Com." (), adresse : https://www.securityweek.com/new-vulnerabilities-can-allow-hackers-remotely-crash-siemens-plcs (visité le 21/02/2022).

[323] E. Kovacs. "High-Severity Vulnerabilities Patched in Omron PLC Programming Software | SecurityWeek.Com." (), adresse : https://www.securityweek.com/high-severity-vulnerabilities-patched-omron-plc-programming-software (visité le 14/03/2022).

[324] E. Kovacs. "New Vulnerabilities Allow Stuxnet-Style Attacks Against Rockwell PLCs | SecurityWeek.Com." (), adresse : https://www.securityweek.com/new-vulnerabilities-allow-stuxnet-style-attacks-against-rockwell-plcs (visité le 04/04/2022).

[325] E. Kovacs. "Details Disclosed After Schneider Electric Patches Critical Flaw Allowing PLC Hacking | SecurityWeek.Com." (), adresse : https://www.securityweek.com/details-disclosed-after-schneider-electric-patches-critical-flaw-allowing-plc-hacking (visité le 30/09/2022).

[326] A. Ribeiro. "Claroty's team82 can extract cryptographic keys embedded within siemens SIMATIC PLC, TIA portal," Industrial Cyber. (13 oct. 2022), adresse : https://

industrialcyber.co/vendor/clarotys-team82-can-extract-cryptographic-keys-embedded-within-siemens-simatic-plc-tia-portal/ (visité le 17/10/2022).

[327] E. Kovacs. "Unpatchable hardware vulnerability allows hacking of siemens PLCs," SecurityWeek. (11 jan. 2023), adresse : https://www.securityweek.com/unpatchable-hardware-vulnerability-allows-hacking-siemens-plcs/ (visité le 24/01/2023).

[328] E. Kovacs. "Critical vulnerabilities allow hackers to take full control of wago PLCs," SecurityWeek. (6 mars 2023), adresse : https://www.securityweek.com/critical-vulnerabilities-allow-hackers-to-take-full-control-of-wago-plcs/ (visité le 12/03/2023).

[329] A. Ribeiro. "BadAlloc vulnerabilities wreck havoc in IoT, OT devices in industrial, medical, enterprise networks," Industrial Cyber. (7 mai 2021), adresse : https://industrialcyber.co/article/badalloc-vulnerabilities-wreck-havoc-in-iot-ot-devices-in-industrial-medical-enterprise-networks/ (visité le 10/05/2021).

[330] E. Kovacs. "Moxa MXview Vulnerabilities Expose Industrial Networks to Attacks | SecurityWeek.Com." (), adresse : https://www.securityweek.com/moxa-mxview-vulnerabilities-expose-industrial-networks-attacks (visité le 21/02/2022).

[331] E. Kovacs. "Schneider Relay Flaws Can Allow Hackers to Disable Electrical Network Protections | SecurityWeek.Com." (), adresse : https://www.securityweek.com/schneider-relay-flaws-can-allow-hackers-disable-electrical-network-protections (visité le 14/03/2022).

[332] E. Kovacs. "Flaws in ABB Network Interface Modules Expose Industrial Systems to DoS Attacks | SecurityWeek.Com." (), adresse : https://www.securityweek.com/flaws-abb-network-interface-modules-expose-industrial-systems-dos-attacks (visité le 14/04/2022).

[333] E. Kovacs. "Vulnerabilities in Aruba and Avaya Switches Expose Enterprise Networks to Attacks | SecurityWeek.Com." (), adresse : https://www.securityweek.com/vulnerabilities-aruba-and-avaya-switches-expose-enterprise-networks-attacks (visité le 04/05/2022).

[334] A. Ribeiro. "Nozomi detects 13 BMC firmware vulnerabilities on lanner hardware exposing OT, IoT devices to RCE attacks," Industrial Cyber. (23 nov. 2022), adresse : https://industrialcyber.co/news/nozomi-detects-13-bmc-firmware-vulnerabilities-on-lanner-hardware-exposing-ot-iot-devices-to-rce-attacks/ (visité le 28/11/2022).

[335] E. Kovacs. "InHand Industrial Router Vulnerabilities Expose Internal OT Networks to Attacks |." (), adresse : https://www.securityweek.com/inhand-industrial-router-vulnerabilities-expose-internal-ot-networks-attacks (visité le 24/01/2023).

[336] A. Ribeiro. "ICS vulnerabilities found in siemens, sewio, InHand networks, SAUTER controls, hitachi energy, johnson controls hardware," Industrial Cyber. (16 jan. 2023), adresse : https://industrialcyber.co/cisa/ics-vulnerabilities-found-in-siemens-sewio-inhand-networks-sauter-controls-hitachi-energy-johnson-controls-hardware/ (visité le 24/01/2023).

[337] E. Kovacs. "Flaws in PcVue SCADA Product Can Facilitate Attacks on Industrial Organizations | SecurityWeek.Com." (), adresse : https://www.securityweek.com/flaws-pcvue-scada-product-can-facilitate-attacks-industrial-organizations (visité le 19/01/2021).

[338] "[SCADA] Multiples vulnérabilités dans les produits Schneider Electric – CERT-FR." (), adresse : https://www.cert.ssi.gouv.fr/avis/CERTFR-2020-AVI-209/ (visité le 03/02/2021).

[339] A. Ribeiro. "Applied risk analyses vulnerabilities found in GE iFIX HMI/SCADA equipment," Industrial Cyber. (11 fév. 2021), adresse : https://industrialcyber.co/threats-attacks/vulnerabilities/applied-risk-analyses-vulnerabilities-found-in-ge-ifix-hmi-scada-equipment/ (visité le 16/02/2021).

[340] A. Ribeiro. "Security defects found in ICS equipment from aveva, xArrow, siemens," Industrial Cyber. (20 août 2021), adresse : https://industrialcyber.co/article/security-defects-found-in-ics-equipment-from-aveva-xarrow-siemens/ (visité le 24/08/2021).

[341] E. Kovacs. "Several Critical Vulnerabilities Found in myPRO HMI/SCADA Product | SecurityWeek.Com." (), adresse : https://www.securityweek.com/several-critical-vulnerabilities-found-mypro-hmiscada-product (visité le 03/01/2022).

[342] E. Kovacs. "GE SCADA Product Vulnerabilities Show Importance of Secure Configurations | SecurityWeek.Com." (), adresse : https://www.securityweek.com/ge-scada-product-vulnerabilities-show-importance-secure-configurations (visité le 28/02/2022).

[343] E. Kovacs. "Organizations notified of remotely exploitable vulnerabilities in aveva HMI, SCADA products," SecurityWeek. (21 mars 2023), adresse : https://www.securityweek.com/organizations-notified-of-remotely-exploitable-vulnerabilities-in-aveva-hmi-scada-products/ (visité le 26/03/2023).

[344] M. H. Rahman, T. Wuest et M. Shafae, *Review, Meta-Taxonomy, and Use Cases of Cyberattack Taxonomies of Manufacturing Cybersecurity Threat Attributes and Countermeasures*, 17 jan. 2023. arXiv : 2301.07303[cs,eess].

[345] R. Tai, L. Lin et R. Su, "Synthesis of optimal covert sensor–actuator attackers for discrete-event systems," *Automatica*, p. 110 910, 1er mai 2023.

[346] C. N. Hadjicostis, S. Lafortune, F. Lin et R. Su, "Cybersecurity and Supervisory Control : A Tutorial on Robust State Estimation, Attack Synthesis, and Resilient Control," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, ISSN : 2576-2370, déc. 2022, p. 3020-3040.

[347] R. Ammour, S. Amari, L. Brenner, I. Demongodin et D. Lefebvre, "Robust stealthy attacks based on uncertain costs and labeled finite automata with inputs," *IEEE Robotics and Automation Letters*, p. 1-8, 2023, Conference Name : IEEE Robotics and Automation Letters.

[348] Y. Li, C. N. Hadjicostis et N. Wu, "Tamper-tolerant diagnosability under bounded or unbounded attacks," *IFAC-PapersOnLine*, 16th IFAC Workshop on Discrete Event Systems WODES 2022, p. 52-57, 1er jan. 2022.

[349] D. Thorsley et D. Teneketzis, "Intrusion Detection in Controlled Discrete Event Systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, ISSN : 0191-2216, déc. 2006, p. 6047-6054.

[350] P. M. Lima, L. K. Carvalho et M. V. Moreira, "Detectable and undetectable network attack security of cyber-physical systems," *IFAC-PapersOnLine*, 14th IFAC Workshop on Discrete Event Systems WODES 2018, p. 179-185, 1er jan. 2018.

[351] A. Rashidinejad, B. Wetzels, M. Reniers, L. Lin, Y. Zhu et R. Su, "Supervisory Control of Discrete-Event Systems under Attacks : An Overview and Outlook," in *2019 18th European Control Conference (ECC)*, p. 1732-1739.

[352] R. Su, *On Decidability of Existence of Nonblocking Supervisors Resilient to Smart Sensor Attacks*, 1er avr. 2022. arXiv : 2009.02626[cs].

[353] R. Meira-Goes et S. Lafortune, "Moving target defense based on switched supervisory control : a new technique for mitigating sensor deception attacks11the work of r.m.g. and s.l. was supported in part by US NSF grants CNS-1738103 and CNS-1801342.," *IFAC-PapersOnLine*, 15th IFAC Workshop on Discrete Event Systems WODES 2020 — Rio de Janeiro, Brazil, 11-13 November 2020, p. 317-323, 1er jan. 2020.

[354] Z. Yu, X. Duan, X. Cong, X. Li et L. Zheng, "Detection of actuator enablement attacks by petri nets in supervisory control systems," *Mathematics*, p. 943, jan. 2023, Number : 4 Publisher : Multidisciplinary Digital Publishing Institute.

[355] R. Fritz, P. Schwarz et P. Zhang, "Modeling of Cyber Attacks and a Time Guard Detection for ICS based on Discrete Event Systems," in *2019 18th European Control Conference (ECC)*, juin 2019, p. 4368-4373.

[356] R. Meira-Goes, C. Keroglou et S. Lafortune, "Towards probabilistic intrusion detection in supervisory control of discrete event systems," *IFAC-PapersOnLine*, 21th IFAC World Congress, p. 1776-1782, 1er jan. 2020.

[357] J. Zaytoon et S. Lafortune, "Overview of fault diagnosis methods for discrete event systems," *Annual Reviews in Control*, p. 308-320, 1er déc. 2013.

[358] M. P. Cabasino, A. Giua, M. Pocci et C. Seatzu, "Discrete event diagnosis using labeled petri nets. an application to manufacturing systems," *Control Engineering Practice*, Special Section : DCDS 09 – The 2nd IFAC Workshop on Dependable Control of Discrete Systems, p. 989-1001, 1er sept. 2011.

[359] N. F. Shamloo, E. De Santis et M. D. Di Benedetto, "Critical Observability of Finite State Machines Under Attacks," in *2021 29th Mediterranean Conference on Control and Automation (MED)*, ISSN : 2473-3504, juin 2021, p. 162-166.

[360] D. Lefebvre, Z. Li et Y. Liang, "Verifiers for the detection of timed patterns in discrete event systems," *IFAC-PapersOnLine*, 16th IFAC Workshop on Discrete Event Systems WODES 2022, p. 264-269, 1er jan. 2022.

[361] J. Yao, X. Yin et S. Li, "On Attack Mitigation in Supervisory Control Systems : A Tolerant Control Approach," in *2020 59th IEEE Conference on Decision and Control (CDC)*, ISSN : 2576-2370, déc. 2020, p. 4504-4510.

[362] K. Yamalidou, J. Moody, M. Lemmon et P. Antsaklis, "Feedback control of petri nets based on place invariants," *Automatica*, p. 15-28, 1er jan. 1996.

[363] T. Y. Elmekkawy et H. A. Elmaraghy, "Efficient search of Petri Nets for deadlock-free scheduling in FMSs using heuristic functions," *International Journal of Computer Integrated Manufacturing*, p. 14-24, 1er jan. 2003.

[364] T. Y. ElMekkawy, I. B. Abdallah et H. A. ElMaraghy, "A heuristic search approach for deadlock-free scheduling in FMSS using petri nets and AI techniques," **presentedat**ASME 1998 Design Engineering Technical Conferences, American Society of Mechanical Engineers Digital Collection, 12 fév. 2021.

[365] B. Huang, Y. Sun, Y.-M. Sun et C.-X. Zhao, "A hybrid heuristic search algorithm for scheduling FMS based on petri net model," *The International Journal of Advanced Manufacturing Technology*, p. 925-933, 1er juin 2010.

[366] H. Lei, K. Xing, L. Han et Z. Gao, "Hybrid heuristic search approach for deadlock-free scheduling of flexible manufacturing systems using petri nets," *Applied Soft Computing*, p. 413-423, 1er juin 2017.

[367] H. H. Xiong et M. Zhou, "Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search," *IEEE Transactions on Semiconductor Manufacturing*, p. 384-393, août 1998, Conference Name : IEEE Transactions on Semiconductor Manufacturing.

[368] S. Li, M. Uzam, L. Yin, Z. Zhong, L. Zheng et N. Wu, "Robust Diagnosability Analysis of Discrete Event Systems Using Labeled Petri Nets," *IEEE Access*, p. 163 504-163 515, 2021, Conference Name : IEEE Access.

[369] D. Lefebvre et C. Delherm, "Diagnosis of DES With Petri Net Models," *IEEE Transactions on Automation Science and Engineering*, p. 114-118, jan. 2007, Conference Name : IEEE Transactions on Automation Science and Engineering.

[370] B. Ostadi, "An optimal preventive maintenance model to enhance availability and reliability of flexible manufacturing systems," *Journal of Industrial and Systems Engineering*, p. 47-61, 23 août 2018, Publisher : Iranian Institute of Industrial Engineering.

[371] A. Salonen et M. Gopalakrishnan, "Practices of preventive maintenance planning in discrete manufacturing industry," *Journal of Quality in Maintenance Engineering*, p. 331-350, 1er jan. 2020, Publisher : Emerald Publishing Limited.

[372] E. I. Basri, I. H. Abdul Razak, H. Ab-Samat et S. Kamaruddin, "Preventive maintenance (PM) planning : a review," *Journal of Quality in Maintenance Engineering*, p. 114-143, 1er jan. 2017, Publisher : Emerald Publishing Limited.

[373] M. Uzam, A. Jones et N. Ajlouni, "Conversion of Petri net controllers for manufacturing systems into ladder logic diagrams," in *Proceedings 1996 IEEE Conference on Emerging Technologies and Factory Automation. ETFA '96*, nov. 1996, 649-655 vol.2.

[374] T. Yang, C. Murguia et C. Lv, "Risk Assessment for Connected Vehicles Under Stealthy Attacks on Vehicle-to-Vehicle Networks," *IEEE Transactions on Intelligent Transportation Systems*, p. 13 627-13 638, 2023.

[375] M. Sharir, "A strong-connectivity algorithm and its applications in data flow analysis," *Computers et Mathematics with Applications*, p. 67-72, 1er jan. 1981.

FOLIO ADMINISTRATIF

THESE DE L'INSA LYON, MEMBRE DE L'UNIVERSITE DE LYON

NOM : BEAUDET
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 07/06/2024

Prénoms : Amaury

TITRE : Méthode de détection et de diagnostic des attaques de blocage dans les systèmes manufacturiers flexibles incertains

NATURE : Doctorat

Numéro d'ordre : 2024ISAL0048

Ecole doctorale : ED 160 : Electronique, Electrotechnique, Automatique

Spécialité : Automatique

RESUME : Les systèmes manufacturiers flexibles (FMSs en anglais) sont conçus avec l'objectif de pouvoir réaliser différentes recettes en parallèle en utilisant conjointement des ressources flexibles et un superviseur allouant ces ressources aux différentes recettes en cours. Par conception, un FMS évolue dans un environnement critique, en présence d'états d'allocation des ressources bloquants pour la réalisation des recettes, et incertain, en raison d'évènements imprévus conduisant à l'indisponibilité temporaire des ressources. Au sein des FMSs modernes, l'utilisation pour l'allocation des ressources de composants de contrôle hautement interconnectés entre eux et avec des réseaux internet ouverts a rendu ces systèmes vulnérables aux cyber-attaques. Aux origines de ces cyber-attaques, différents profils d'attaquant peuvent être distingués selon leurs objectifs, leurs origines, leurs compétences et leurs moyens.
Ainsi, bien qu'un FMS soit initialement construit pour faire face aux états de blocage et aux indisponibilités de ressources, un profil d'attaquant expert pourrait être capable de manipuler les décisions d'allocation et les disponibilités des ressources pour conduire ce FMS dans un état de blocage ciblé. A partir de cette observation, la problématique de recherche suivante émerge : au sein du contexte incertain des FMSs, comment diagnostiquer correctement l'origine, naturelle ou malveillante, d'un état de blocage détecté et identifier le profil d'attaquant à l'origine de l'attaque ?
En réponse à cette problématique, nous proposons trois contributions principales. Premièrement, les attaques de blocage et les profils d'attaquant sont définis et modélisés dans un contexte certain afin de structurer le développement d'un module de diagnostic de ces derniers. Puis, ce module est étendu au contexte incertain des FMS au sein duquel l'indisponibilité des ressources est prise en compte et peut être manipulée par un attaquant. Enfin, ce module de diagnostic est implémenté sur une plateforme manufacturière expérimentale afin d'être évalué.

MOTS-CLÉS :
Cybersécurité, Systèmes manufacturiers flexibles, Attaques de blocage, Diagnostic d'attaques, Systèmes à évènements discrets

Laboratoire (s) de recherche : Laboratoire Ampère

Directeur de thèse: Pr. Éric ZAMAÏ

Président de jury :

Composition du jury :

LEFEBVRE, Dimitri, Professeur des universités, Université de Normandie	Rapporteur.e
ESPES, David, Professeur des universités, Université de Bretagne Occidentale	Rapporteur.e
BERRUET, Pascal, Prof. des universités, Université de Bretagne Sud	Examinateur.rice
MARANGÉ, Pascale, Maître de conférences, Université de Lorraine	Examinateur.rice
HENRY, Sébastien, Maître de conférences, IUT Lyon 1	Examinateur.rice