Synthèse de contrôleurs pour des classes de réseau de Petri à contrôlabilité et observabilité partielles : application au contrôle automatisé des trains

Paul Cazenave

Centrale Lille

Thèse

présentée en vue d'obtenir le grade de

Docteur

En

Spécialité : Automatique, Génie informatique, traitement du signal et des images

Par

Paul CAZENAVE

Doctorat délivré par Centrale Lille

Titre de la thèse :

Synthèse de contrôleurs pour des classes de Réseau de Petri à contrôlabilité et observabilité partielles : Application au contrôle automatisé des trains

Soutenance le 26 novembre 2020 :

Rapporteur	Kamel BARKAOUI, Professeur des universités	Conservatoire national des arts et métiers
Rapporteur	Sébastien LAHAYE, Professeur des universités	Polytech Angers
Examinateur	Nidhal REZG, Professeur des universités	Université de Lorraine
Examinateur	Laurent PIÉTRAC, Professeur des universités	SIGMA Clermont
Examinateur	Belkacem OULD BOUAMAMA, Professeur des universités	Polytech Lille
Examinatrice	Lilian KAWAKAMI CARVALHO, Maître de Conférences	Escola Politécnica da UFRJ
Directeur de Thèse	Armand TOGUYÉNI, Professeur des universités	Centrale Lille
Co-encadrante	Manel KHLIF-BOUASSIDA, Maître de Conférences	Centrale Lille

Thèse préparée dans le Centre de Recherche en Informatique, Signal et Automatique de Lille, CRIStAL, CNRS UMR 9189
École Doctorale Sciences pour l'Ingénieur (SPI) 072

Table des matières

Table des figures

Liste des tableaux

Nomenclature

Abréviations et Acronymes

COP Contrôlabilité et Observabilité Partielle, page 107

GMA Graphe des marquages accessibles, page 18

L-S3PR *Linear Simple Sequential Process with Resources*, page 69

Max-cs Siphon Max-contrôlé *Max-controled siphon*, page 49

Min-cs Siphon Min-contrôlé *Min-controled siphon*, page 50

MIP Problème d'optimisation en nombres mixtes (*Mixed Integer Programing*), page 56

RAS Système d'allocations de ressources (RAS : *Resource Allocation System*), page 61

RdP Réseaux de Petri, page 15

S3PMR *Simple Sequential Process with Multiple Resources*, page 69

S3PR *Simple Sequential Process with Resources*, page 69

SCT Théorie du Contrôle Supervisé (*Supervisory Control Theory*), page 28

SED Système à Événements Discrets, page 7

SFRM Système Flexible de Production Manufacturière, page 51

SMS Siphon Strict Minimal (*strict minimal siphon*), page 23

SMS *Flexible Manufacturing Systems*, page 51

Symboles et notations

(w, b) Notation standard pour une contrainte avec w le vecteur de pondération et b la borne, page 38

δ Notation standard pour la fonction de transition d'un automate, page 10

η_S T-vecteur caractéristique d'un siphon S, page 51

$\Gamma(x)$ Fonction d'événements actifs à un état x, page 10

$\mathcal{A}$ Notation standard d'un automate, page 10

$\mathcal{A}_1 \parallel \mathcal{A}_2$ Composition parallèle de deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$, page 14

$\mathcal{A}_1 \times \mathcal{A}_2$ Produit totalement synchrone entre deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$, page 14

$\mathcal{C}^C$ Ensemble des vecteurs admissibles vis-à-vis de la contrôlabilité, cas général, page 42, cas des RAS-PN, page 112

$\mathcal{C}^O$ Ensemble des vecteurs admissibles vis-à-vis de l'observabilité, cas général, page 42, cas des RAS-PN, page 112

$\mathcal{C}_{in}(K)$ La classe des sous-langages de K qui respectent la relation de contrôlabilité, page 30

$\mathcal{I}$ Ensemble des indices des processus, page 63

$\mathcal{L}(\mathcal{A})$ Langage d'un automate, page 10

$\mathcal{L}(\mathcal{A})$ Langage marqué d'un automate, page 10

$\mathcal{M}(w,b)$ Marquages autorisés par la contrainte (w,b), page 38

$\mathcal{N}$ Notation standard pour un Réseau de Petri, page 16

$\mathcal{N}_i^P$ Un Réseau de processus d'indice i, page 70

$\mathcal{N}_i^R$ Sous-Réseau de la ressource i, page 67

$\mathcal{P}^E_{\downarrow\Sigma}(L)$ Projection d'un langage L défini sur l'alphabet E vers l'alphabet Σ, page 9

$\mathcal{P}^E_{\uparrow\Sigma}(L)$ Projection inverse d'un langage L défini sur l'alphabet Σ vers l'alphabet E, page 9

$\mathcal{P}_{\circlearrowleft}(\mathcal{N})$ Ensemble des vecteurs P-flots d'un Réseau de Petri $\mathcal{N}$, page 21

$\mathcal{P}^+_{\circlearrowleft}(\mathcal{N})$ Ensemble des vecteurs P-semiflots d'un Réseau de Petri $\mathcal{N}$, page 21

$\mathcal{R}(\mathcal{N},M)$ Ensemble des marquages accessibles d'un Réseau de Petri $\mathcal{N}$ à partir de l'état M, page 18

$\mathcal{T}_{\circlearrowleft}(\mathcal{N})$ Ensemble des vecteurs T-flots d'un Réseau de Petri $\mathcal{N}$, page 21

$\mathcal{T}^+_{\circlearrowleft}(\mathcal{N})$ Ensemble des vecteurs T-semiflots d'un Réseau de Petri $\mathcal{N}$, page 21

$\mathfrak{C}$ Ensemble des composants d'un noeud ferrovaire, page 136

$\mathfrak{J}$ Ensemble des connexions d'un noeud ferrovaire, page 136

$\overline{L}$ Ensemble des préfixes du langage L, page 8

$\overrightarrow{\mathfrak{C}}$ ($\overleftarrow{\mathfrak{C}}$) Ensemble des composants d'un nœud ferroviaire se traversant uniquement de la gauche vers la droite (droite vers la gauche), page 136

$\overrightarrow{\mathfrak{Ins}}(\mathfrak{j})$ ($\overleftarrow{\mathfrak{Ins}}(\mathfrak{j})$) Instrumentation de la connexion $\mathfrak{j}$ liée au sens de passage de la gauche vers la droite (droite vers la gauche), page 136

$\overrightarrow{\mathfrak{J}}$ ($\overleftarrow{\mathfrak{J}}$) Ensemble des connexions d'un nœud ferroviaire se traversant uniquement de la gauche vers la droite (droite vers la gauche), page 136

$\Pi(\mathcal{N})$ Ensemble des siphons d'un Réseau de Petri, page 23

$\Pi_D(\mathcal{N})$ Ensemble des siphons dépendants d'un Réseau de Petri $\mathcal{N}$, page 51

$\Pi_E(\mathcal{N})$ Ensemble des siphons élémentaires d'un Réseau de Petri $\mathcal{N}$, page 51

$\Pi_{\check{m}}(\mathcal{N})$ Ensemble des siphons minimaux d'un Réseau de Petri, page 23

$\Pi_{\hat{M}}(\mathcal{N})$ Ensemble des siphons maximaux d'un Réseau de Petri, page 23

$\succ$ Relation d'ordre partielle entre des places de processus liées par un chemin de transitions incontrôlables, page 109, 110

$\Theta(\mathcal{N})$ Ensemble de tous les ensembles connexes inobservables maximaux d'un Réseau de Petri $\mathcal{N}$, page 110

$\vec{\mathcal{R}}(\mathcal{N}, M)$ Ensemble des marquages d'un Réseau de Petri $\mathcal{N}$ qui vérifient l'équation d'état à partir de l'état M, page 19

$\vec{p}^R_{i\circlearrowleft}$ P-semiflot généré par une place de ressource p^R_i, page 67

$Ac(\mathcal{A})$ Opération d'accessibilité appliquée à un automate $\mathcal{A}$, page 11

C Notation standard pour la matrice d'incidence d'un Réseau de Petri, page 16

$CoAc(\mathcal{A})$ Opération de co-accessibilité appliquée à un automate $\mathcal{A}$, page 11

E Notation standard d'un alphabet, page 8

E^* Opérateur étoile de Kleen appliqué à un alphabet E, page 8

F Notation standard pour l'ensemble des arcs d'un Réseau de Petri, page 16

$H(p^R)$ Ensemble détenteur d'une place de ressources p^R, page 73

$K^{\uparrow C}$ Langage suprême-contrôlable de K, page 30

L Notation standard d'un langage, page 8

$Obs(\mathcal{A})$ Observation d'un automate $\mathcal{A}$, page 12

P Notation standard pour l'ensemble des places d'un Réseau de Petri, page 16

P^0 Ensemble des places de repos, page 72

P^P Ensemble des places de processus, page 72

P^R Ensemble des places de ressources, page 64

$Post$ Notation standard pour la matrice d'incidence arrière d'un Réseau de Petri, page 16

Pre Notation standard pour la matrice d'incidence avant d'un Réseau de Petri, page 16

S Notation standard pour un siphon, page 22

S/G Système G supervisé par S, page 29

T Notation standard pour l'ensemble des transitions d'un Réseau de Petri, page 16

$Trim(\mathcal{A})$ Opération $Trim$ appliqué à un automate $\mathcal{A}$, page 12

V Notation standard pour une place monitrice, page 39

W Notation standard pour l'application de pondération des arcs d'un Réseau de Petri, page 16

X Notation standard pour les états d'un automate, page 10

X_0 Notation standard pour les états initiaux d'un automate, page 10

X_m Notation standard pour les états marqués d'un automate, page 10

$x^\bullet$ Ensemble des noeuds successeurs d'un noeud x, page 16

${}^\bullet x$ Ensemble des noeuds prédécesseurs d'un noeud x, page 16

Chapitre 1

Introduction

Cette thèse traite de méthodes de supervision des Réseaux de Petri en vue d'une application aux systèmes ferroviaires. Ce travail a été effectué dans l'équipe de recherche MOSES (Modèles et Outils formels pour des Systèmes à Événements discrets Sûrs) du laboratoire de recherche CRIStAL (Centre de Recherche en Informatique, Signal et Automatique de Lille, UMR 9189) dirigée par Armand TOGUYÉNI et encadrée par Manel KHLIF-BOUASSIDA.

1.1 Préambule et motivations

La théorie du contrôle supervisé a été introduite par Ramadge et Wonham à la fin des années 90 [RW87]. Cette théorie a continué d'être développée jusqu'à nos jours pour engendrer une base solide et diversifiée d'apports scientifiques. Aujourd'hui ces approches se trouvent être regroupées sous la dénomination de « méthodes de synthèse de contrôleurs ». Ces approches permettent, à partir de la connaissance d'un système libre (ou système en boucle ouverte) et de spécifications sur ce système, de définir un contrôleur ou un superviseur qui va venir « contrôler » ce système, pour que celui-ci respecte les spécifications. Ce sont en premier lieu les systèmes manufacturiers qui ont bénéficié de ces méthodologies, ce qui a contribué au développement des systèmes flexibles de production manufacturière.

Cette thèse a été motivée par l'interrogation suivante : « Comment appliquer la théorie du contrôle supervisé aux systèmes ferroviaires pour un contrôle automatisé garantissant une sécurité maximale ? ». Cette interrogation peut mener vers un nombre important de résultats. Mais ces travaux de thèse ont finalement abouti à une méthode de routage dynamique des trains dans un nœud ferroviaire. Il a été observé que les modèles obtenus pouvaient être rapportés à des systèmes d'allocations de ressources, bien connus dans le domaine manufacturier, qui bénéficient d'un développement scientifique avancé. Pour pouvoir appliquer ces méthodes aux systèmes ferroviaires, les méthodes de la littérature ont dû être étendues à des systèmes plus complexes, notamment vis-à-vis de la prise en compte des événements incontrôlables et inobservables.

Cette thèse présente des théories sur le contrôle de classes de Réseaux de Petri pour la modélisation de systèmes d'allocations de ressources. Elle vise à étendre une partie de ces méthodes de contrôle à des Réseaux de Petri comportant des transitions inobservables et incontrôlables. Les systèmes ferroviaires ont influé l'orientation technique et les apports scientifiques de cette thèse, même s'ils sont uniquement présentés comme application.

1.2 Contexte

Dans cette thèse nous nous intéresserons uniquement à une classe particulière des systèmes dynamiques : les systèmes à événements discrets. Ces systèmes correspondent à un certain nombre de systèmes : systèmes manufacturiers, réseaux de communication, systèmes informatiques, etc. Beaucoup de systèmes comportant une dynamique continue peuvent être ramenés à des systèmes à événements discrets par abstraction de certains comportements. Ainsi les systèmes à événements discrets se révèlent d'une grande utilité pour modéliser les systèmes de grande tailles, avec un haut niveau d'abstraction.

En 1989, Ramadge et Wonham développent la théorie du contrôle supervisé (*Supervisory Control Theory*), une méthode qui permet de générer un superviseur à partir de la connaissance d'un système libre et d'une spécification exprimée sous la forme d'un des formalismes de modélisation des systèmes à événements discrets. Deux types d'événements sont considérés : les événements contrôlables dont le superviseur peut empêcher l'apparition et ceux incontrôlables que le superviseur ne peut inhiber. Grâce à cette méthode, un superviseur est généré automatiquement.

La méthode est développée à la base sous forme de langages mais par la suite surtout appliquée sous le formalisme des automates. Cette méthode atteint rapidement ses limites pour des problèmes complexes et de grandes tailles. En effet, cette approche cherche à explorer les états d'un système de manière à identifier les états qui violent les spécifications. Les systèmes complexes peuvent comporter un nombre important de sous-systèmes qui évoluent en parallèle, ce qui fait augmenter le nombre d'états existant exponentiellement en fonction du nombre de ces sous-systèmes. D'une manière générale les méthodes de contrôle basées sur le formalisme des automates souffrent de l'explosion combinatoire.

Par la suite, ont été développées, des méthodes considérant des événements inobservables (événements dont le superviseur ne peut détecter l'occurrence). Ces méthodes se trouvent être encore plus complexes algorithmiquement que celles ou les événements inobservables ne sont pas considérés.

Les Réseaux de Petri ont été introduits par Carl Adam Petri dans sa thèse de Doctorat [Pet62]. Les Réseaux de Petri sont des outils graphiques et mathématiques permettant de modéliser et de vérifier le comportement dynamique des systèmes à événements discrets. Des méthodes de synthèse de contrôleurs ont été par la suite développées pour les Réseaux de Petri, avec des approches autres que l'exploration des états, ce qui permet ainsi d'éviter l'explosion combinatoire.

Même si les méthodes de synthèse de contrôleurs des Réseaux de Petri bénéficient d'une plus faible complexité algorithmique que celles des automates, cela est aussi souvent au prix d'une plus faible permissivité. Le formalisme des Réseaux de Petri permet, entre autres, d'exprimer deux grandes classes de spécifications : l'évitement d'états interdits et l'évitement de blocages. L'évitement d'états interdits peut être ramené à un problème structurel, ce qui permet de se passer de l'exploration des états. La présence de transitions incontrôlables (et inobservables) vient complexifier considérablement la synthèse d'un contrôleur. L'évitement de blocage est typiquement un problème comportemental et il est compliqué de le ramener à un problème structurel dans le cadre général.

Il a été remarqué que certains types de systèmes étaient plus enclins à entrainer des blocages, comme les systèmes d'allocations de ressources. Les systèmes d'allocations de ressources sont composés de différents processus et d'un ensemble de ressources. Les processus se partagent les ressources, qu'ils sont amenés à utiliser dans certains états de leur évolution. Ces systèmes peuvent modéliser plusieurs types de systèmes notamment les systèmes manufacturiers ou les systèmes ferroviaires. On trouve dans la littérature un nombre important de classes de Réseaux de Petri pour la modélisation des systèmes d'allocations de ressources. Ces classes ont l'avantage de disposer de propriétés intéressantes pour générer des lois de contrôles non-bloquants. D'une manière générale, ces méthodes ne considèrent pas les évènements incontrôlables et inobservables, car elles ont été développées pour des systèmes manufacturiers avec un très haut niveau d'abstraction.

1.3 Contributions

Dans le cadre de cette thèse, nous nous intéresserons particulièrement aux méthodes basées sur des systèmes d'allocations de ressources pour l'évitement des blocages. Ces méthodes ne prennent généralement pas en compte le contexte complet de la supervision. En effet, le niveau d'abstraction étant relativement élevé, ces modèles ne prennent pas en compte les événements incontrôlables et inobservables. La prise en compte de ces événements étant cruciale pour modéliser les systèmes ferroviaires, nous avons étendu des méthodes de contrôles pour des systèmes d'allocations de ressources au cas de la contrôlabilité et observabilité partielles. Cette étude a mené à l'élaboration de deux algorithmes de synthèse de contrôleurs pour des Réseaux de Petri sous observabilité et contrôlabilité partielles modélisant des systèmes d'allocation de ressources.

Une méthodologie de modélisation a été développée au cours de cette thèse. Cette modélisation introduit un nouveau concept : celui de traversée, n'imposant pas de route préétablie à un train, mais laissant la possibilité à ce dernier de définir son itinéraire, au fur-et-à-mesure, en fonction de la saturation du réseau. À partir de la topologie d'un nœud ferroviaire quelconque, cette méthode générique permet d'obtenir un Réseau de Petri modélisant un système d'allocations de ressources. Par la suite ce système peut être contrôlé, de manière à ce qu'il ne puisse pas y avoir au sein du nœud ferroviaire, ni collisions entre les trains, ni blocages. Le modèle ainsi obtenu permet d'établir une base où la sécurité est garantie, pour par la suite appliquer des méthodes de contrôle de plus haut niveau : optimisation, régulation de la vitesse ...

1.4 Structure du manuscrit

La figure 1.1 illustre les dépendances entre les sections et les sous-sections de chaque chapitre. Ce manuscrit est structuré comme suit :

— Le chapitre 2 présente les systèmes à événements discrets. Trois formalismes y sont présentés : les langages, les automates et les Réseaux de Petri. Des opérations de base pour les automates y sont présentées ainsi que des méthodes d'analyses pour les Réseaux de Petri.

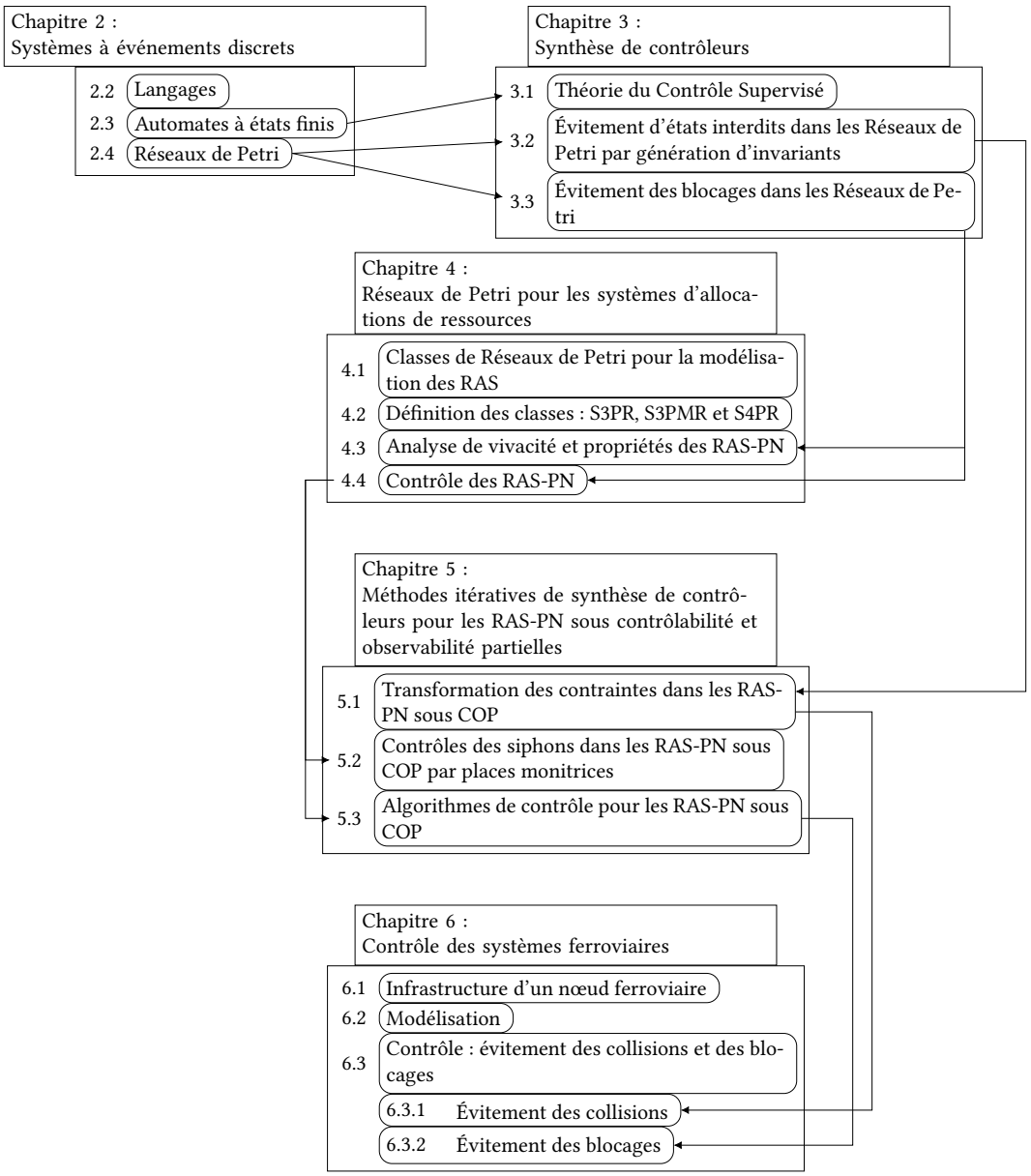


FIGURE 1.1 – Structure du manuscrit

— Le chapitre 3 fait l'état de l'art de la théorie du contrôle supervisé. Dans un premier temps, la méthode historique de Ramadge et Wonham, appliquée aux automates, y est introduite. Ensuite des méthodes appliquées aux Réseaux de Petri sont présentées, divisées en deux parties ; une concernant l'évitement de marquages interdits et la seconde concernant l'évitement des blocages.

— Le chapitre 4 introduit les classes de Réseaux de Petri pour la modélisation des systèmes d'allocations de ressources. Dans un premier temps, plusieurs classes sont présentées de manière à donner au lecteur une vue d'ensemble du domaine. Ensuite trois classes sont détaillées et leurs propriétés vis-à-vis de leurs relations aux blocages sont énumérées.

— Le chapitre 5 vise à étendre les méthodes de synthèse de contrôle spécifiques aux Réseaux de Petri modélisant les systèmes d'allocations de ressources au

cas de la contrôlabilité et de l'observabilité partielles. D'une manière générale, ce chapitre donne une méthodologie pour étendre les méthodes dites "itératives". Deux méthodes de la littérature sont étendues aux systèmes sous observabilité et contrôlabilité partielles.

— Le chapitre 6 présente comment synthétiser une loi de contrôle pour le routage dynamique des trains dans un nœud ferroviaire. À partir de la topologie d'un nœud ferroviaire et des points d'entrée et de sortie des trains dans le nœud, un modèle Réseau de Petri est construit en suivant une méthode générique. Le réseau ainsi obtenu comporte des transitions inobservables et incontrôlables, directement influencées par la configuration de l'instrumentation dans le nœud. Une première étape de synthèse permet d'obtenir un Réseau de Petri correspondant à une classe de systèmes d'allocations de ressources. Une deuxième phase de synthèse applique une méthode d'évitement de blocages qui permet d'obtenir un Réseau de Petri vivant, assurant qu'aucun blocage ne peut arriver.

— Le chapitre 7 conclut ce travail de recherche et présente des perspectives.

Chapitre 2

Systèmes à événements discrets

2.1 Introduction

Un système peut être décrit comme un ensemble de composants, interagissant entre eux, formant une entité complexe. Le terme système est fréquemment utilisé en ingénierie pour définir la conception d'un élément composé de sous-ensembles. L'automatique est la discipline qui traite de la modélisation, de l'analyse, de l'identification des systèmes dynamiques. Elle propose de considérer un système connu, que l'on est capable de synthétiser sous forme d'un modèle mathématique, et de lui appliquer une configuration d'entrées de façon à ce qu'il exécute une fonction donnée. L'automatique concerne un grand nombre d'applications (robotique, biologie, aérospatiale, ferroviaire ...). En fonction du type de modèles mathématiques obtenu lors de la modélisation, les systèmes peuvent être classés en plusieurs catégories. Une classification des différents types de systèmes est proposée dans [CL09] et est représentée en figure 2.1.

Les Systèmes à Événements Discrets (SED) sont des systèmes dont l'espace d'états est discret et dont l'évolution est déclenchée par l'occurrence d'événements. Les SED sont donc des systèmes dont l'état n'évolue pas de façon continue en fonction du temps. Le changement d'état se fait instantanément quand l'événement survient. Un événement correspond à une action instantanée, qui peut être interne ou externe au système, contrôlée ou subie. Selon l'état actuel du système, l'apparition d'un événement peut déclencher ou non son évolution.

2.2 Langages

Les langages représentent la manière la plus élémentaire de décrire un système à événements discrets. Ils consistent à énumérer l'ensemble des mots qu'un système peut générer compte-tenu d'un alphabet constitué par ses événements. La théorie des automates tient une place importante dans les disciplines des mathématiques discrètes et de l'informatique. Un automate à états finis est en effet un générateur de langage.

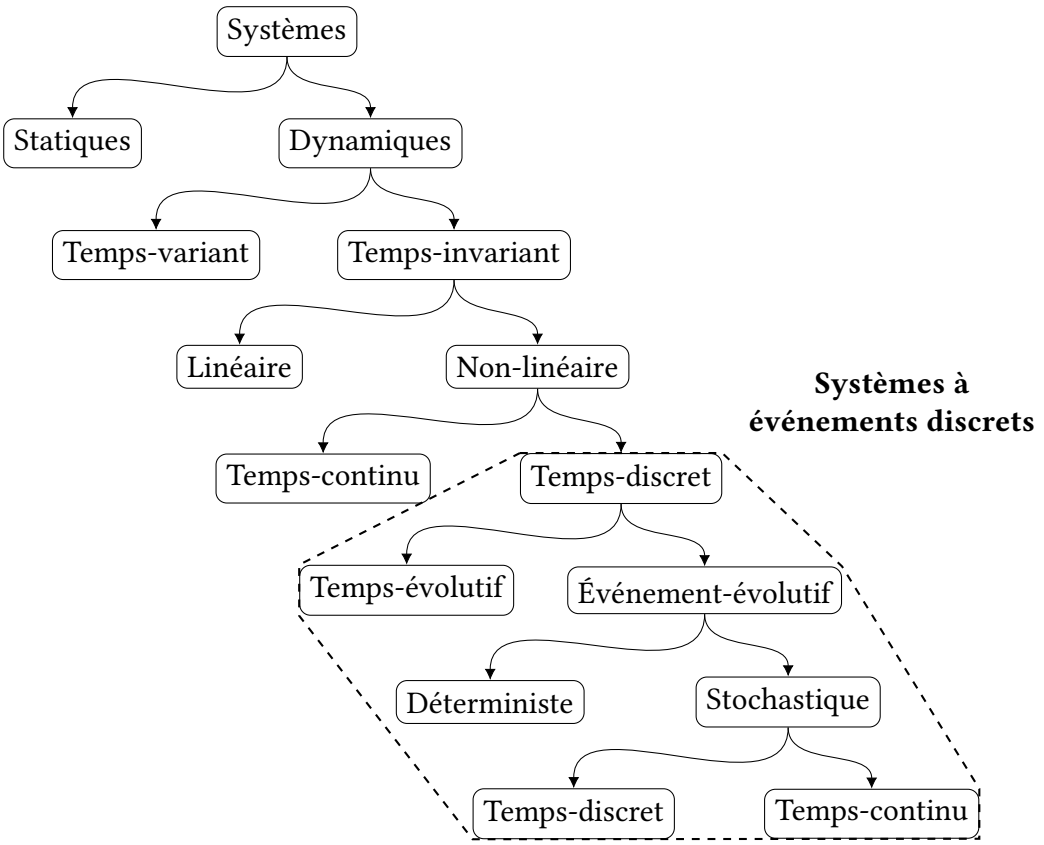


FIGURE 2.1 – Classification des systèmes

2.2.1 Notations et définitions

Un alphabet E est un ensemble fini d'événements. Un mot est une concaténation d'événements, la concaténation de deux événements ou de deux mots est simplement notée par leur juxtaposition. On peut donc former un mot s par la concaténation de deux événements e_1 et e_2 qui sera notée de la manière suivante : $s = e_1e_2$. Un mot peut être constitué d'aucun événement, dans ce cas là il est noté ϵ. On note $|s|$ la longueur du mot s. La longueur d'ϵ est considérée comme nulle.

Définition 1. *Un langage L est défini à partir d'un alphabet E comme étant un ensemble de mots de tailles finies formés par des événements de E.*

On note par E^* l'ensemble de tous les mots finis construits avec des événements de E. Cette opération est appelée étoile de Kleen.

$$E^* = \{e_1e_2...e_n \,|\, n \geq 0, e_i \in E \; \forall i \in [1..n]\}$$

Définition 2. *On note par $\overline{L}$ l'ensemble des préfixes du langage L tel que :*

$$\overline{L} = \{s \,|\, \sigma \in E^* \text{ avec } s\sigma \in L\}$$

Propriété 1. *Par définition on a toujours :*

$$\overline{L} \supseteq L$$

On dit qu'un langage est préfixe-clos si $L = \overline{L}$

2.2.2 Opérations sur les langages

Définition 3. *La projection $\mathcal{P}^{E}_{\downarrow_{\Sigma}} : E^* \to \Sigma^*$ permet de faire passer un langage défini dans un alphabet E à un alphabet réduit $\Sigma \subseteq E$. Il s'agit d'un masque qui « filtre » les événements qui n'appartiennent pas à Σ pour tous les mots d'un langage L. Pour un événement $e \in E$ la projection dans Σ est définie comme :*

$$\mathcal{P}^{E}_{\downarrow_{\Sigma}}(e) = \begin{cases} e & si\ e \in \Sigma \\ \epsilon & si\ e \in E \backslash \Sigma \end{cases}$$

Dans le cas d'un mot projeté de E dans Σ la projection est définie récursivement :

$$\mathcal{P}^{E}_{\downarrow_{\Sigma}}(se) = \mathcal{P}^{E}_{\downarrow_{\Sigma}}(s)\mathcal{P}^{E}_{\downarrow_{\Sigma}}(e)\ pour\ s \in E^*\ et\ e \in E$$

Enfin la projection d'un langage $L \in E^*$ est défini comme suit :

$$\mathcal{P}^{E}_{\downarrow_{\Sigma}}(L) = \{s' \in \Sigma^* | \exists s \in L \wedge \mathcal{P}^{E}_{\downarrow_{\Sigma}}(s) = s'\}$$

Définition 4. *De manière opposée la projection inverse d'un langage $L \in \Sigma^*$ vers $E^* \supseteq \Sigma^*$ est définie comme*

$$\mathcal{P}^{E}_{\uparrow_{\Sigma}}(L) = \{s \in E^* | \mathcal{P}^{E}_{\downarrow_{\Sigma}}(s) \in L\}$$

2.3 Automates à états finis

2.3.1 Notations et définitions

Un automate à états finis permet de représenter de façon graphique un langage. On dit qu'un automate est un générateur de langage, mais un langage donné peut être défini par différents automates.

Automate déterministe

Définition 5. *Un automate déterministe à états finis $\mathcal{A}$ est défini par un sextuplé $\mathcal{A} = (X, E, \delta, x_0, X_m)$,*

X *est l'ensemble des états.*

E *est l'ensemble des événements associés à $\mathcal{A}$*

$\delta : X \times E \to X$ *est la fonction de transition ; elle définit un arc orienté reliant deux états consécutifs. L'arc est étiqueté par un événement de E,*

$x_0 \in X$ *est l'état initial.*

$X_m \subseteq X$ *est l'ensemble des états marqués.*

Automate non-déterministe

Définition 6. *Un automate non-déterministe se distingue d'un automate déterministe par la possibilité, à partir d'un état donné, d'atteindre des états différents par le même événement. Il est aussi possible pour un automate non-déterministe d'avoir plusieurs états initiaux. Il est défini comme un quintuplé* $\mathcal{A}_{nd} = (X, E \cup \epsilon, \delta_{nd}, X_0, X_m)$

X *est l'ensemble des états.*

$E \cup \epsilon$ *est l'ensemble des événements associés à* $\mathcal{A}$.

$\delta_{nd} : X \times (E \cup \epsilon) \to 2^X$ *est la fonction de transition, représentée par des arcs amenant d'un à un ou plusieurs états. Le mot vide* ϵ *peut faire évoluer d'un état vers un autre état : on parle alors de* ϵ*-transition.*

$X_0 \subseteq X$ *l'ensemble des états initiaux.*

$X_m \subseteq X$ *est l'ensemble des états marqués.*

Remarque 1. *On associe généralement la fonction d'événements actifs* $\Gamma : X \to 2^E$ *à la définition d'un automate* $\mathcal{A}$. *Cette fonction associe à un état un ensemble d'événements actifs. Elle est formellement décrite de la manière suivante* $\Gamma(x) = \{e \in E, \ |\exists \delta(x, e) = x'\}$

Un automate transite d'un état x à un autre état en cas d'occurrence d'un événement actif $e \in \Gamma(x)$. Ce nouvel état est déterminé par la fonction de transition $\delta(x, e) = x'$. Dans le cas d'un automate déterministe un événement ne peut amener qu'à un seul état. De manière récursive, la fonction de transition peut être étendue des événements $\delta : X \times E \to X$ aux mots $\delta : X \times E^* \to X$. Si on considère un mot $s = e_0 e_1 ... e_n$ et qu'il existe pour chaque valeur de $i \in [1..n]$ un état $\delta(x_i, e_i) = x_{i+1}$ alors il est permis de noter $\delta(x_0, s) = x_n$.

Définition 7. *Le langage d'un automate* $\mathcal{A}$ *est l'ensemble des mots qu'un automate peut reconnaître.*

$$\mathcal{L}(\mathcal{A}) = \{s \in E^* | \exists x_n \in X \text{ avec } \delta(x_0, s) = x_n\}$$

Définition 8. *Le langage marqué d'un automate* $\mathcal{A}$ *est l'ensemble des mots qui permettent d'atteindre un état marqué.*

$$\mathcal{L}_m(\mathcal{A}) = \{s \in E^* | \exists x_n \in X_m \text{ avec } \delta(x_0, s) = x_n\}$$

Remarque 2. *Le langage marqué généré par un automate est préfixe-clos.*

Exemple

Les figures 2.2 et 2.3 représentent respectivement un automate déterministe et un automate non déterministe. Ces deux automates sont chacun composés de trois états x_0, x_1 et x_2, représentés par des cercles. Ils ont tous deux un alphabet constitué de l'ensemble d'événements $\{a, b\}$. Ces événements sont notés au-dessus des arcs représentant la fonction de transition δ. Les automates ont pour état initial x_0, noté avec une flèche et le label « start » et x_2 comme état marqué noté avec un double cercle. L'automate de la figure 2.3 est un automate non-déterministe. En effet depuis

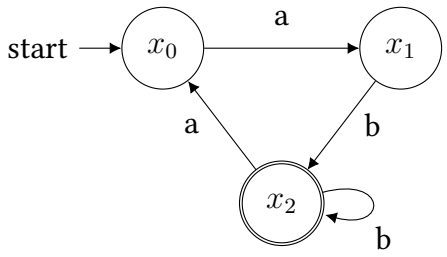


FIGURE 2.2 – Exemple d'un automate déterministe

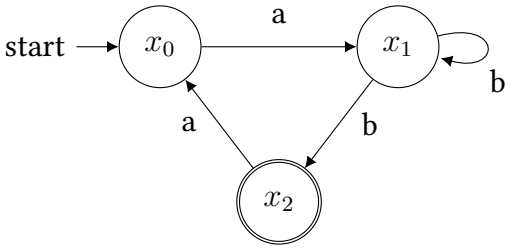


FIGURE 2.3 – Exemple d'un automate non-déterministe

l'état x_2 la fonction de transition sur b emmène vers les états x_2 et x_3 ($\delta(x_2, b) = \{x_2, x_3\}$). L'automate de la figure 2.2, dans chacun de ses états ne peut amener qu'à un seul état pour un événement donné, il est donc déterministe. Les deux automates reconnaissent le même langage $L = \overline{(ab^*ba)^*}$ et le même langage marqué $L_m = ab^*b$;

Remarque 3. *Pour tout automate non-déterministe, il existe un automate déterministe qui génère le même langage et le même langage marqué.*

2.3.2 Opérations unitaires

Accessibilité

Il se peut qu'un certain nombre d'états n'ait aucune influence sur le langage généré par l'automate. Les états qui ne sont pas atteignables depuis l'état initial, n'interviennent pas dans la génération des mots du langage de l'automate. L'opération $Ac(\mathcal{A})$ supprime ces états. Elle est décrite de la manière suivante :

$$
\begin{aligned}
Ac(\mathcal{A}) &= (X_{Ac}, E, \delta_{Ac}, x_0, X_{Ac,m}) \text{ avec} \\
X_{Ac} &= \{x \in X \mid \exists s \in E^* \wedge \delta(x_0, s) = x\} \\
X_{Ac,m} &= X_m \cap X_{Ac} \\
\delta_{Ac} &= \delta \text{ réduit au domaine et co-domaine : } X_{Ac} \times E \to X_{Ac}
\end{aligned}
$$

L'opération d'accessibilité n'a aucune influence sur les langages : $\mathcal{L}(\mathcal{A}) = \mathcal{L}(Ac(\mathcal{A}))$ et $\mathcal{L}_m(\mathcal{A}) = \mathcal{L}_m(Ac(\mathcal{A}))$.

Co-Accessibilité

L'opération de co-accessibillité vise à supprimer les états depuis lesquels on ne peut pas atteindre un état marqué. Cette opération appliquée à un automate $\mathcal{A}$ et notée $CoAc(\mathcal{A})$ et génère un automate défini par :

$$
\begin{aligned}
Ac(\mathcal{A}) &= (X_{CoAc}, E, \delta_{CoAc}, x_{0,CoAc}, X_m) \text{ avec} \\
X_{CoAc} &= \{x \in X \mid \exists s \in E^* \wedge f(x, s) \in X_m\} \\
X_{Ac,m} &= \begin{cases} x_o & \text{si } x_o \in X_{CoAc} \\ \text{non-defini} & \text{sinon} \end{cases} \\
\delta_{CoAc} &= \delta \text{ réduit au domaine et co-domaine : } X_{CoAc} \times E \to X_{CoAc}
\end{aligned}
$$

L'opération de co-accessibilité n'a pas d'effet sur le langage marqué $\mathcal{L}_m(CoAc(\mathcal{A})) = \mathcal{L}_m(\mathcal{A})$. Par contre le langage est réduit $\mathcal{L}(CoAc(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$. L'automate $CoAc(\mathcal{A})$ a la propriété suivante : $\mathcal{L}(CoAc(\mathcal{A}) = \overline{\mathcal{L}_m(CoAc(\mathcal{A}))}$; on dit que l'automate est non bloquant.

Trim (parfait)

L'opération $Trim$ permet d'obtenir un automate à la fois accessible et co-accessible à partir d'un automate $\mathcal{A}$

$$Trim(\mathcal{A}) = CoAc(Ac(\mathcal{A})) = Ac(CoAc(\mathcal{A}))$$

Observateur

Les automates non-déterministes se distinguent des automates déterministes par la possibilité d'atteindre plusieurs états avec une même transition d'états. D'autre part, ils autorisent les ϵ-transitions voire des transitions basées sur des mots.

D'un point de vue informel, on peut dire qu'un observateur permet de « suivre » l'évolution d'un automate non-déterministe avec pour unique information les événements générés par l'automate. L'automate non-déterministe ne permet pas dans le cas général de suivre l'état courant de l'automate en observant les événements émis. L'observateur nous renseigne uniquement sur les états possibles de l'automate non-déterministe après avoir observé un mot. Ainsi chaque état de l'observateur peut être vu comme un macro-état constitué par un sous-ensemble d'états de l'automate non-déterministe initial.

Partant d'un automate non-déterministe $\mathcal{A}_{nd} = (X, E \cup \epsilon, \delta_{nd}, X_0, X_m)$ il est possible de trouver un automate déterministe $Obs(\mathcal{A}_{nd})$ tel que $\mathcal{L}(\mathcal{A}_{nd}) = \mathcal{L}(Obs(\mathcal{A}_{nd})$ et $\mathcal{L}_m(\mathcal{A}_{nd}) = \mathcal{L}_m(Obs(\mathcal{A}_{nd}))$. On appelle l'opération Obs, l'observation et l'automate $Obs(\mathcal{A}_{nd})$ l'observateur. On notera l'automate déterministe généré par l'observation $\mathcal{A}_{Obs} = Obs(\mathcal{A}_{nd})$ avec $\mathcal{A}_{Obs} = (X_{Obs}, E, \delta_{Obs}, x_{0,Obs}, X_{m,Obs})$. Les états de $\mathcal{A}_{Obs}$ correspondent aux états possibles dans lequels peut être l'automate $\mathcal{A}_{nd}$ pour chaque mot de $\mathcal{L}(\mathcal{A}_{nd})$. Chaque état $x_{Obs} \in X_{Obs}$ est donc un ensemble d'états de A_{nd}, $x_{Obs} \in 2^X$ et donc $X_{Obs} \subseteq 2^X$.

Le processus de génération d'un observateur est décrit par l'algorithme 1. La définition 9 est nécessaire à la compréhension de l'algorithme.

Définition 9. *Pour un automate non-déterministe $\mathcal{A}_{nd} = (X, E \cup \epsilon, \delta_{nd}, X_0, X_m)$, on définit la fonction d'epsilon atteignabilité $\mathcal{R}^\epsilon$. La fonction est définie par $\mathcal{R}^\epsilon : X \to 2^X$, qui à partir d'un état $x \in X$, retourne tous les états accessibles par des arcs labellisés par ϵ. Elle est définie de manière récursive de la manière suivante :*

1. $x \in \mathcal{R}^\epsilon(x)$
2. $\forall x' \in \mathcal{R}^\epsilon(x), \delta_{nd}(x', \epsilon) \subset \mathcal{R}^\epsilon(x)$

Le domaine de $\mathcal{R}^\epsilon$ peut être étendu par $\mathcal{R}^\epsilon : 2^X \to 2^X$. Dans ce cas là, pour $Y \subseteq X$, $\mathcal{R}^\epsilon(Y) = \bigcup_{x \in Y} \mathcal{R}^\epsilon(x)$.

Algorithme 1 Construction d'un Observateur

Entrée : $\mathcal{A}_{nd} = (X, E \cup \epsilon, \delta_{nd}, x_0, X_m)$ ▷ Un automate non-déterministe
Sortie : $\mathcal{A}_{Obs} = (X_{Obs}, E, \delta_{Obs}, x_{0,Obs}, X_{m,Obs})$ ▷ Un automate déterministe

$x_{0,Obs} \leftarrow \mathcal{R}^{\epsilon}(x_0)$
$X_{Obs} \leftarrow \{x_{0,Obs}\}$
$X_{m,Obs} \leftarrow \varnothing$
$B \leftarrow \varnothing$ ▷ États de l'observateur déjà explorés
pour $x_{Obs} \in X_{Obs} \backslash B$ **faire**
  $B \leftarrow B \cup x_{Obs}$
  pour $e \in E$ **faire**
    si $\exists x_{nd} \in x_{Obs} \,|\, \delta_{nd}(x_{nd}, e) \neq \varnothing$ **alors**
      $x'_{Obs} \leftarrow \bigcup_{x_{nd} \in x_{Obs}} \mathcal{R}^{\epsilon}(\delta_{nd}(x_{nd}, e))$
      $X_{Obs} \leftarrow X_{Obs} \cup x'_{Obs}$
      définir $\delta_{Obs}(x_{Obs}, e) = x'_{Obs}$
    fin si
  fin pour
fin pour
pour $x_{Obs} \in X_{Obs}$ **faire**
  si $x_{Obs} \cap X_m \neq \varnothing$ **alors**
    $X_{m,Obs} \leftarrow X_{m,Obs} \cup x_{Obs}$
  fin si
fin pour

Exemple

Dans l'exemple de la figure 2.4 un automate non-déterministe $\mathcal{A}$ et son observateur $\mathcal{A}_{Obs} = Obs(\mathcal{A})$ sont représentés. L'automate est non-déterministe dans l'état x_0 ; l'événement a peut faire évoluer $\mathcal{A}$ vers deux états différents, x_1 et x_2. D'autre part, à partir de l'état x_2, il comprend un arc labellisé par ϵ. Sur occurrence de a à partir de x_0, l'automate peut évoluer dans les états x_1, x_2 ou x_3 ($\delta_{nd}(x_0, a) = \{x_1, x_2\}$ et $\mathcal{R}^{\epsilon}(x_2) = \{x_2, x_3\}$). L'état $\{x_1, x_2, x_3\} \in X_{Obs}$ ne peut être quitté sur occurrence de b ($\Gamma_{nd}(x_1) = \Gamma_{nd}(x_3) = b$ et $\Gamma_{nd}(x_2) = \epsilon$). Comme $\delta_{nd}(x_1, b) = x_3$ et $\delta_{nd}(x_3, b) = x_0$, on a $\delta_{Obs}(\{x_1, x_2, x_3\}, b) = \{x_0, x_3\}$. Depuis l'état $\{x_0, x_3\}$, de la même manière l'état initial $\Gamma_{nd}(x_0) = \{a\}$ et $\mathcal{R}_{\epsilon}(\delta_{nd}(x_0, a)) = \{x_1, x_2, x_3\}$ on définit donc l'arc $\delta_{Obs}(\{x_0, x_3\}, a) = \{x_1, x_2, x_3\}$. Depuis l'état $\{x_0, x_3\}$, $\Gamma_{nd}(x_3) = \{b\}$ et $\mathcal{R}_{\epsilon}(\delta_{nd}(x_3, b)) = \{x_0\}$. On définit donc l'arc $\delta_{Obs}(\{x_0, x_3\}, b) = \{x_0\}$.

2.3.3 Compositions entre automates

Dans cette partie sont présentées des compositions entre automates. Nous allons les formaliser avec les deux automates suivants : $\mathcal{A}_1 = (X_1, E_1, \delta_1, x_{0,1}, X_{m,1})$ et $\mathcal{A}_2 = (X_2, E_2, \delta_2, x_{0,2}, X_{m,2})$. Deux exemples d'automates sont représentés en figure 2.5 qui serviront d'illustration aux différentes compositions.

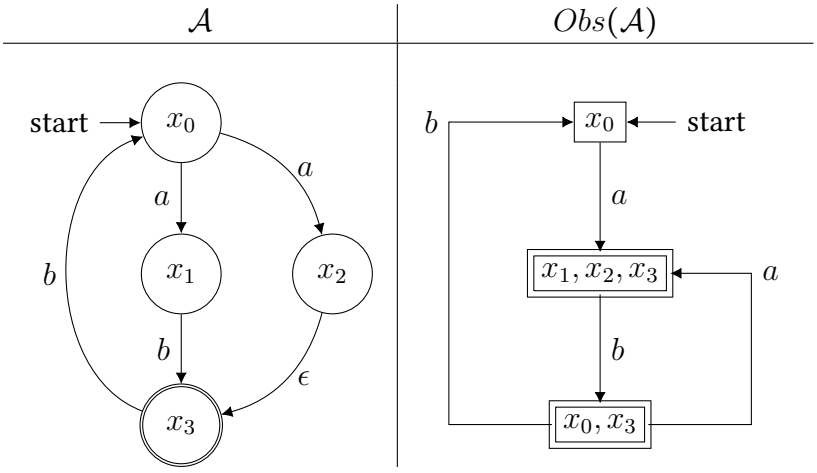


FIGURE 2.4 – Automate non-déterministe et son observateur

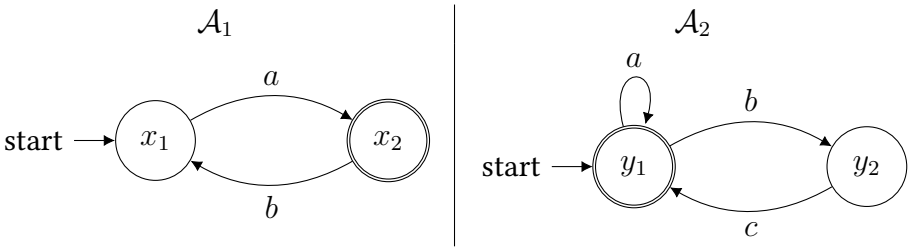


FIGURE 2.5 – Deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$

Produit totalement synchrone

Le produit totalement synchrone d'automates force les automates à synchroniser leurs événements. L'occurrence d'un événement produit une transition dans l'automate résultat seulement si cette transition est autorisée à partir de l'état courant de chacun des deux automates initiaux. Le produit d'automate génère un automate qui a pour alphabet uniquement l'ensemble des événements communs. Le produit de deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$ est noté $\mathcal{A}_1 \times \mathcal{A}_2$

$$\mathcal{A}_1 \times \mathcal{A}_2 = Ac(X_1 \times X_2, E_1 \cap E_2, \delta, (x_{0,1}, x_{0,2}), X_{m,1} \times X_{m_2})$$

avec

$$\delta((x_1, x_2), e) = \begin{cases} (\delta_1(x_1, e), \delta_2(x_2, e)) & \text{si } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ \text{non-defini} & \text{sinon} \end{cases}$$

Propriété 2. *Le langage de l'automate généré a les propriétés suivantes :*

$$\mathcal{L}(\mathcal{A}_1 \times \mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$$

$$\mathcal{L}_m(\mathcal{A}_1 \times \mathcal{A}_2) = \mathcal{L}_m(\mathcal{A}_1) \cap \mathcal{L}_m(\mathcal{A}_2)$$

Composition parallèle d'automates

La composition parallèle d'automates ou produit synchrone force les automates à synchroniser les événements faisant partie de leur l'alphabet commun. Pour les

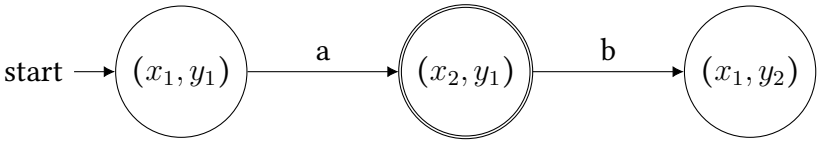


FIGURE 2.6 – Automate du produit $\mathcal{A}_1 \times \mathcal{A}_2$

événements étant hors de cet alphabet commun, la composition parallèle n'impose aucune restriction. Le résultat de la composition parallèle est un automate défini par

$$\mathcal{A}_1 \parallel \mathcal{A}_2 = Ac(X_1 \times X_2, E_1 \cup E_2, \delta, (x_{0,1}, x_{0,2}), X_{m,1} \times X_{m_2})$$

avec

$$\delta((x_1, x_2), e) = \begin{cases} (\delta_1(x_1, e), \delta_2(x_2, e)) & \text{si } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (\delta_1(x_1, e), x2) & \text{si } e \in \Gamma_1(x_1) \backslash E_2 \\ (x_1, \delta_2(x_2, e)) & \text{si } e \in \Gamma_2(x_2) \backslash E_1 \\ \text{non-definie} & \text{sinon} \end{cases}$$

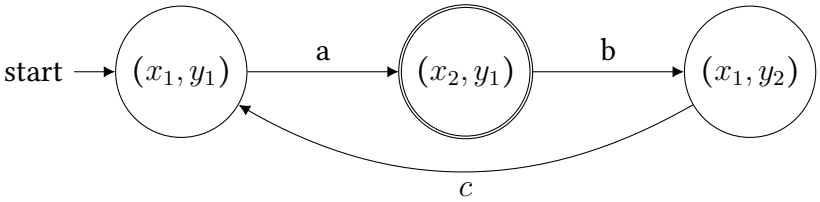


FIGURE 2.7 – Automate de la composition parallèle $\mathcal{A}_1 \| \mathcal{A}_2$

Propriété 3. *Le langage de l'automate généré a les propriétés suivantes :*

$$\mathcal{L}(\mathcal{A}_1 \parallel \mathcal{A}_2) = \mathcal{P}^{E_\cup}_{\uparrow E_1}(\mathcal{L}(\mathcal{A}_1) \cap \mathcal{P}^{E_\cup}_{\uparrow E_2}(\mathcal{L}(\mathcal{A}_2))$$

$$\mathcal{L}_m(\mathcal{A}_1 \parallel \mathcal{A}_2) = \mathcal{P}^{E_\cup}_{\uparrow E_1}(\mathcal{L}_m(\mathcal{A}_1) \cap \mathcal{P}^{E_\cup}_{\uparrow E_2}(\mathcal{L}_m(\mathcal{A}_2))$$

avec $E_\cup = E_1 \cup E_2$

2.4 Réseaux de Petri

Les Réseaux de Petri (RdP dans la suite du document) sont un outil de modélisation des SED. Ils ont été développés dans les années 60 par Carl Adam Petri [Pet62]. Les SED modélisés par des RdP ont l'avantage d'avoir une représentation graphique relativement compacte. Il est possible de modéliser les Réseaux de Petri sous une forme matricielle, ce qui permet de développer de nombreuses techniques d'analyse.

2.4.1 Notations et définitions

Définition 10. *Un Réseau de Petri est un graphe biparti composé de places et de transitions. Il est défini par* $\mathcal{N} = (P, T, F, W)$*. Avec :*

P *l'ensemble des places,*

T *l'ensemble des transitions,*

F *l'ensemble des arcs avec* $F \in (P \times T) \cup (T \times P)$*,*

W *la fonction de pondération qui associe un entier à un arc* $W : F \to \mathbb{N}^*$*.*

Il est impossible qu'un nœud soit à la fois place et transition $P \cap T = \varnothing$*.*

Un Réseau de Petri peut aussi être décrit de manière matricielle. Dans ce cas le Réseau de Petri aura la forme d'un quadruplé $\mathcal{N} = (P, T, Pre, Post)$*.* Pre *est la matrice d'incidence avant qui représente les arcs prédécesseurs reliant les places aux transitions. Elle est définie formellement par l'application* $Pre : P \times T \to \mathbb{N} | \forall f = (p,t) \in F, Pre(p,t) = W(f)$*. De façon analogue on définit* $Post$ *la matrice d'incidence arrière qui représente les arcs postérieurs reliant les transitions à leurs places de sorties. Elle est définie formellement par l'application* $Post = P \times T \to \mathbb{N} | \forall f = (t,p) \in F, Post(p,t) = W(f)$*. On appelle* C *la matrice d'incidence avec* $C = Post - Pre$*. La notation* $C(p,)$ *est utilisée pour faire référence à la ligne concernant la place* p *de la matrice* C*. De manière analogue pour faire référence à la colonne concernant la transition* t*, la notation* $C(,t)$ *sera utilisée.*

Remarque 4. *Dans le cas où il n'existe pas une paire d'arc « aller-retour » sur une transition,* $\nexists \delta_1, \delta_2 \in F | \delta_1 = (p,t)$ *et* $\delta_2 = (t,p)$*, le Réseau de Petri est dit pur. Dans ce cas là le triplé* $\mathcal{N} = (P, T, C)$ *est suffisant pour décrire le modèle.*

Remarque 5. *Un Réseau de Petri qui ne comporte que des arcs avec un poids égal à 1 est dit ordinaire. Dans le cas contraire, on parle de Réseau de Petri généralisé.*

Définition 11. *Dans le cas où un vecteur* $X \in \mathbb{N}^{|P|}$ *fait référence à l'ensemble des places, il est possible de calculer son support* $||X||$*, c'est-à-dire les places dont la valeur de* X *est non nulle*

$$||X|| = \{p | X(p) > 0\}$$

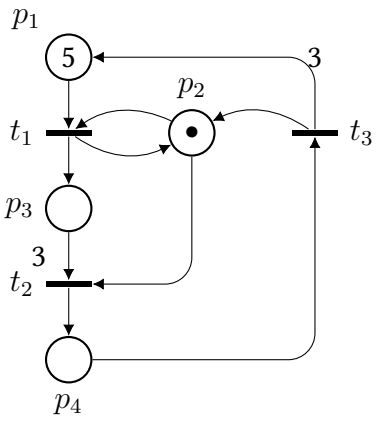
Des jetons peuvent être associés aux places du modèle. Une configuration de jetons dans les places est nommée marquage. Le marquage d'un Réseau de Petri représente l'état courant du système modélisé. Un marquage est donc une application $M : P \to \mathbb{N}^*$.

La notation ${}^\bullet x$ est adoptée pour faire référence à l'ensemble des nœuds prédécesseurs $x \in P \cup T$, ${}^\bullet x = \{y \in P \cup T | \exists (y,x) \in F\}$. De manière analogue on note $x^\bullet$ l'ensemble des nœuds successeurs de x, $x^\bullet = \{y \in P \cup T | \exists (x,y) \in F\}$. Ces notations peuvent être étendues à un ensemble de places : $X \subseteq P$ avec $X^\bullet = \bigcup_{x \in X} x^\bullet$ et ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$. De même pour un ensemble de transitions $Y \subseteq T$ on définit les places successeurs et prédécesseurs : $Y^\bullet = \bigcup_{y \in Y} y^\bullet$ et ${}^\bullet Y = \bigcup_{y \in Y} {}^\bullet y$.

Pour un ensemble de places X, son ensemble de transitions successeures strictes est défini par $X^\circ = X^\bullet \backslash {}^\bullet X$ et son ensemble de transitions prédécesseures strictes ${}^\circ X = {}^\bullet X \backslash X^\bullet$.

Exemple

Un exemple de RdP est présenté en figure 2.8. Pour cet exemple, l'ensemble des places est $P = \{p_1, p_2, p_3, p_4\}$ représenté par des cercles et l'ensemble des transitions représenté par des barres noires est $T = \{t_1, t_2, t_3\}$. Les arcs allant de (p_3, t_2) et de (t_3, p_1) ont une pondération de 3. Les autres arcs, par convention, ont une pondération de 1. Le marquage du Réseau de Petri est $M(p_1) = 5$, $M(p_2) = 1$ et $M(p_3) = M(p_4) = 0$.



$$Pre = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Post = \begin{bmatrix} 0 & 0 & 3 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} -1 & 0 & 3 \\ 0 & -1 & 1 \\ 1 & -3 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

$$M_0 = [5, 1, 0, 0]$$

FIGURE 2.8 – Exemple de Réseau de Petri

2.4.2 Dynamique des Réseaux de Petri

L'évolution d'un Réseau de Petri se fait par le tir (franchissement) des transitions. Une transition est dite sensibilisée si toutes ses places amonts ont un marquage supérieur ou égal au poids de l'arc prédécesseur les reliant.

Définition 12. *L'ensemble des transitions sensibilisées pour un marquage M est noté $\Gamma^{\mathcal{N}}(M) : \mathbb{N}^{|P|} \to 2^T$ et est défini de la manière suivante :*

$$\Gamma^{\mathcal{N}}(M) = \{t \in T \mid \forall p \in {}^\bullet t,\, W(p,t) \leq M(p)\}$$

La notation $M[t\rangle$ sera adoptée dans le cas où $t \in \Gamma^{\mathcal{N}}(M)$.

Si une transition est sensibilisée alors elle peut être *tirée* (franchie) entraînant l'évolution du marquage.

Définition 13. *La fonction de tir $f^{\mathcal{N}} : \mathbb{N}^{|P|} \times T \to \mathbb{N}^{|P|}$ correspondant au tir d'une transition est définie de la manière suivante :*

$$f^{\mathcal{N}}(M, t) = M + C(, t)$$

Cette fonction n'est définie que dans l'ensemble de définition (M, t) qui satisfait $t \in \Gamma^{\mathcal{N}}(M)$.

On notera $M[t\rangle M'$ le tir d'une transition t amenant le marquage de M à M'. De la même façon on considère une séquence de tir $\sigma \in T^*$ telle que $\sigma = t_1, t_2 ... t_n$ avec $t_1, t_2 ... t_n \in T$. On note $M_0[\sigma\rangle M_n$ s'il existe une succession d'états telle que : $\forall i = [1..n],\, M_i[t_i\rangle M_{i+1}$.

Remarque 6. *On peut associer un vecteur de tir $\vec{\sigma} \in \mathbb{N}^{|T|}$ à une séquence de tir $\sigma \in T^*$.*

$$\vec{\sigma}(t) = |\mathcal{P}^{T}_{\downarrow_{\{t\}}}(\sigma)|$$

Où $\mathcal{P}^{T}_{\downarrow_{\{t\}}}$ est l'opérateur de projection de la définition 3, qui projette un mot d'un alphabet défini sur T à $\{t\}$.

2.4.3 Graphe des marquages accessibles

Définition 14. *L'ensemble des marquages accessibles d'un Réseau de Petri $\mathcal{N}$ depuis un marquage M_0 est noté $\mathcal{R}(\mathcal{N}, M_0)$.*

$$\mathcal{R}(\mathcal{N}, M_0) = \{M \in \mathbb{N}^{|P|} | \exists \sigma \in T^*, \text{ avec } M_0[\sigma\rangle M\}$$

Définition 15. *Le graphe des marquages accessibles (GMA) est un automate noté $\mathcal{G}^A(\mathcal{N}, M_0) = (\mathcal{R}(\mathcal{N}, M_0), T, f, M_0)$. Avec*

- *Les états du GMA correspondent aux marquages accessibles $\mathcal{R}(\mathcal{N}, M_0)$ du Réseau de Petri $\mathcal{N}$,*
- *L'ensemble des événements correspond au tir des transitions de T,*
- *La fonction de transition f de l'automate correspond à la fonction de tir $f^{\mathcal{N}}$ définie uniquement depuis les marquages de $\mathcal{R}(\mathcal{N}, M_0)$,*
- *L'état initial correspond au marquage initial M_0.*

Algorithme 2 Calcul du graphe d'accessibilité

Entrée : Un Réseau de Petri marqué ($\mathcal{N}, M_0$)
Sortie : Un automate $\mathcal{G}^A(\mathcal{N}, M_0)$

1: $\mathcal{R}(\mathcal{N}, M_0) \leftarrow \varnothing$ ▷ Marquages explorés
2: $f : \mathbb{N}^{|P|} \times T \to \mathbb{N}^{|P|}$ avec un ensemble de définition vide ▷ Initialisation de la fonction de transition
3: $\mathcal{B} \leftarrow \{M_0\}$ ▷ Marquages inexplorés
4: **tant que** $\mathcal{B} \neq \varnothing$ **faire**
5: Sélectionner $M \in \mathcal{B}$
6: $\mathcal{B} \leftarrow \mathcal{B} \backslash M$
7: $\mathcal{R}(\mathcal{N}, M_0) \leftarrow \mathcal{R}(\mathcal{N}, M_0) \cup M$
8: **pour** $t \in \Gamma^{\mathcal{N}}(M)$ **faire** ▷ Itération sur les transitions sensibilisées par le marquage
9: Définir f sur (M, t) tel que $f(M, t) = f^{\mathcal{N}}(M, t)$
10: **si** $f^{\mathcal{N}}(M, t) \notin \mathcal{R}(\mathcal{N}, M_0)$ **alors**
11: $\mathcal{B} \leftarrow \mathcal{B} \cup \{f^{\mathcal{N}}(M, t)\}$ ▷ Ajout d'un marquage à explorer défini par la fonction de tir
12: **fin si**
13: **fin pour**
14: **fin tant que**

Remarque 7. *Le graphe des marquages accessibles permet de vérifier un nombre important de propriétés d'un Réseau de Petri. Il est cependant potentiellement coûteux de le calculer voir impossible dans le cas d'un Réseau de Petri non-borné.*

La figure 2.9 correspond au graphe des marquages accessibles du Réseau de Petri de la figure 2.8.

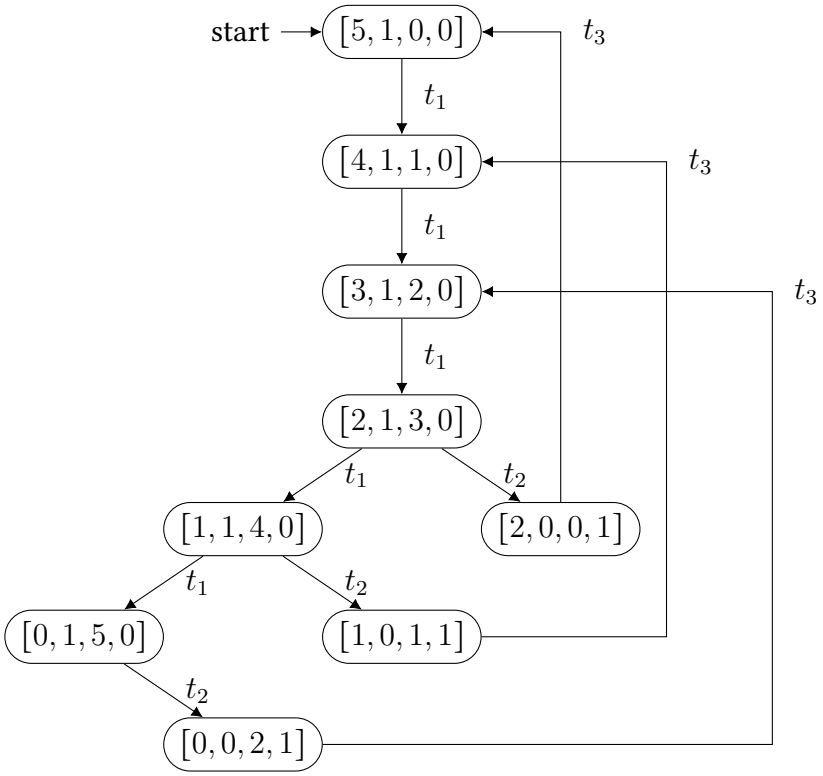


FIGURE 2.9 – Graphe des marquages accessibles

Définition 16. *L'équation d'état est une condition nécessaire à l'accessibilité d'un état. Pour un Réseau de Petri marqué $(\mathcal{N}, M_0)$ avec une matrice d'incidence C, elle est définie par :*

$$M = M_0 + C.u, \quad u \in \mathbb{N}^{|T|},\ M \geq 0$$

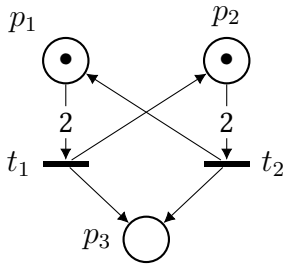
Ici u est un vecteur de tir.

Définition 17. *On définit $\vec{\mathcal{R}}(\mathcal{N}, M_0)$ l'ensemble des marquages qui vérifient l'équation d'états.*

$$\vec{\mathcal{R}}(\mathcal{N}, M_0) = \{M \in \mathbb{N}^{|P|} \mid \exists u \in \mathbb{N}^{|T|} \text{ avec } M = M_0 + C.u\}$$

L'équation d'états étant une condition nécessaire à l'existence d'un état accessible, cela implique la relation suivante $\mathcal{R}(\mathcal{N}, M_0) \subseteq \vec{\mathcal{R}}(\mathcal{N}, M_0)$. Cependant l'équation d'états n'est pas une condition suffisante pour l'existence d'un état.

La figure 2.10 montre un Réseau de Petri marqué $(\mathcal{N}, M_0)$. Dans son état M_0 il est impossible de tirer les transitions t_1 ou t_2 donc $\mathcal{R}(\mathcal{N}, M_0) = \{M_0\}$. Cependant le vecteur de tir $[1, 1]^\intercal$ vérifie l'équation d'état et permet d'atteindre un état $M = [0, 0, 2]^\intercal$. $M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)$ mais $M \notin \mathcal{R}(\mathcal{N}, M_0)$ ce qui implique qu'il n'y a pas une séquence de transition $\sigma \in T^*$ telle que $M_0[\sigma\rangle M$



$$C = \begin{bmatrix} -2 & 1 \\ 1 & -2 \\ 1 & 1 \end{bmatrix} \quad M_0 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

FIGURE 2.10 – Réseau de Petri avec M_0 bloquant

2.4.4 Propriétés usuelles des Réseaux de Petri

Certaines propriétés sont importantes à vérifier dans les Réseaux de Petri. Un certain nombre d'entre elles qui seront utiles par la suite, est présenté ici.

Propriété des états

Des qualificatifs peuvent être appliqués aux états du système. Ces qualificatifs seront utiles par la suite pour décrire des propriétés usuelles des Réseaux de Petri :

- M est un état *accessible* si M est un état du graphe d'accessibilité

$$M \in \mathcal{R}(\mathcal{N}, M_0)$$

- M est un état *d'accueil* si depuis n'importe quel marquage accessible il est possible de trouver une séquence de franchissements permettant d'atteindre M.

$$\forall M' \in \mathcal{R}(\mathcal{N}, M_0), M \in \mathcal{R}(\mathcal{N}, M'))$$

Bornitude ou finitude

Un Réseau de Petri est n-borné si le marquage de ces places est borné par un entier n, $\forall M \in \mathcal{R}(\mathcal{N}, M_0), \forall p \in P, \exists n \in \mathbb{N} \,|\, M(p) \leq n$. Dans le cas où toutes les places d'un Réseau de Petri sont bornées à un jeton (1-borné) nous dirons que le RdP est sauf. Les RdP non bornés comportent une infinité d'états. Le graphe des marquages accessibles est alors impossible à représenter.

Vivacité

Le concept de vivacité d'un Réseau de Petri $\mathcal{N}$ est lié à la possibilité de franchir une transition t dans une des évolutions possibles du système depuis un état M donné. Plusieurs classes de vivacités sont définies. Pour un RdP dans un état M, il est dit que :

- $(\mathcal{N}, M)$ est *bloquant* par rapport à t, s'il est impossible depuis M de trouver une séquence de transitions amenant au tir de t, autrement dit :

$$\nexists \sigma \in T^* | M[\sigma t\rangle$$

- $(\mathcal{N}, M)$ est *potentiellement vivant* par rapport à t si depuis M, il existe une séquence de transitions qui amène au tir de t, autrement dit :

$$\exists \sigma \in T^* | M[\sigma t\rangle$$

- $(\mathcal{N}, M)$ est *pseudo-vivant* s'il est toujours possible de franchir une transition depuis n'importe quel état accessible, autrement dit :

$$\forall m \in \mathcal{R}(\mathcal{N}, M), \exists t \in T | m[t\rangle$$

- $(\mathcal{N}, M)$ est *vivant* par rapport à t, si quel que soit l'état atteint depuis M, il est possible de trouver une séquence de transitions qui amène à franchir t :

$$\forall \sigma \in T^* | M[\sigma\rangle, \exists \sigma' \in T^* \wedge M[\sigma\sigma' t\rangle$$

Réversibilité

Un Réseau de Petri marqué $(\mathcal{N}, M_0)$ est dit réversible s'il est possible depuis n'importe quel état de revenir à M_0.

$$\forall M \in \mathcal{R}(\mathcal{N}, M_0), \quad M_0 \in \mathcal{R}(\mathcal{N}, M)$$

Cette propriété implique que tous les états sont des états d'accueil.

2.4.5 Analyse des Réseaux de Petri par Invariants linéaires

Définition 18. *On considère différentes catégories d'annulateurs de la matrice d'incidence*

- *La classe des T-flots* $\mathcal{T}_{\circlearrowleft}(\mathcal{N}) = \{Y \in \mathbb{Z}^{|T|} | C.Y = 0 \wedge Y \neq 0\}$
- *La classe des T-semiflots* $\mathcal{T}^+_{\circlearrowleft}(\mathcal{N}) = \{Y \in \mathbb{N}^{|T|} | C.Y = 0 \wedge Y \neq 0\}$
- *La classe des P-flots* $\mathcal{P}_{\circlearrowleft}(\mathcal{N}) = \{X \in \mathbb{Z}^{|P|} | X^\top.C = 0 \wedge X \neq 0\}$
- *La classe des P-semiflots* $\mathcal{P}^+_{\circlearrowleft}(\mathcal{N}) = \{X \in \mathbb{N}^{|P|} | X^\top.C = 0 \wedge X \neq 0\}$

Remarque 8. *La définition 18 implique que pour un Réseau de Petri* $\mathcal{N}$ *les P-semiflots sont inclus dans les P-flots,* $\mathcal{P}^+_{\circlearrowleft}(\mathcal{N}) \subseteq \mathcal{P}_{\circlearrowleft}(\mathcal{N})$*. De même que pour les T-semiflots et T-flots on a* $\mathcal{T}^+_{\circlearrowleft}(\mathcal{N}) \subseteq \mathcal{T}_{\circlearrowleft}(\mathcal{N})$

Propriété 4. *Soit un T-flot* $Y \in \mathcal{T}^+_{\circlearrowleft}(\mathcal{N})$ *d'un Réseau de Petri* $\mathcal{N}$ *et une séquence de franchissements* σ *telle que* $\vec{\sigma} = Y$*, alors*

$$M[\sigma\rangle M$$

Dans ce cas, σ *est une séquence répétitive stationnaire.*

Propriété 5. *Considérant un P-flot* $X \in \mathcal{P}_{\circlearrowleft}(\mathcal{N})$ *d'un Réseau de Petri* $\mathcal{N}$ *et une séquence de franchissements* $\sigma \in T^*$ *vérifiant* $M[\sigma\rangle M'$*, alors :*

$$X^\top.M = X^\top.M'$$

Un P-flot $X \in \mathcal{P}_{\circlearrowleft}(\mathcal{N})$ est dit minimal s'il n'existe pas d'autres P-flot $X' \in \mathcal{P}_{\circlearrowleft}(\mathcal{N})$ tels que $X' \leq X$.

Si on considère un Réseau de Petri marqué $(\mathcal{N}, M_0)$, et un P-flot $X \in \mathcal{P}_{\circlearrowleft}(\mathcal{N})$ on parle d'invariant de marquages si la suivante relation sur le marquage est vérifiée :

$$\forall M \in \mathcal{R}(\mathcal{N}, M_0), \quad X^\top.M = X^\top.M_0$$

Dans le cas où X est un P-semiflot on parle alors d'invariant positif. Les places du support de $\|X\|$ sont alors bornées par : $M(p) \leq (X^\top.M_0)/X(p)$

Exemple

Dans la figure 2.11, le Réseau de Petri comporte un P-semiflot positif de marquage $X = [1, 4, 2]^\top$. Ce RdP a pour marquage initial $M_0 = [3, 0, 1]^\top$. Dans la figure 2.11, les marquages accessibles sont représentés par des points avec le vecteur marquage associé. On remarque que tous les points se retrouvent dans le plan gris qui correspond à l'invariant positif formé par X : $M(p_1) + 4M(p_2) + 2M(p_3) = M_0(p_1) + 4M_0(p_2) + 2M_0(p_3) = 5$. Le Réseau de Petri comporte aussi un T-semiflot $Y \in \mathcal{T}^+_{\circlearrowleft}(\mathcal{N})$ tel que $Y = [1, 1, 1]$. On voit que les séquences ayant pour vecteur de tir Y ramènent bien à l'état de départ. Trois séquences de $(\mathcal{N}, M_0)$ admettent Y comme vecteur de tir : $\sigma_1 = t_1t_2t_3$, $\sigma_2 = t_2t_3t_1$ et $\sigma_3 = t_3t_1t_2$. Ces séquences vérifient bien la propriété 4 car $M_0[\sigma_1\rangle M_0$, $M_1[\sigma_2\rangle M_1$, et $M_2[\sigma_2\rangle M_2$.

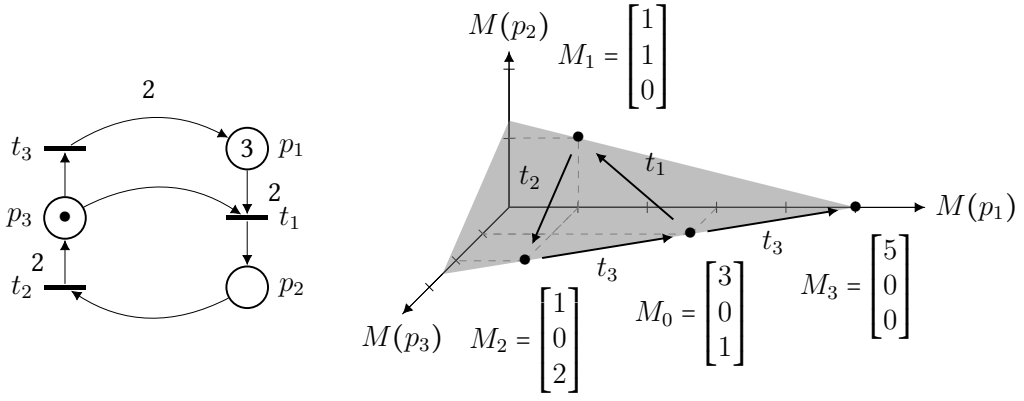


FIGURE 2.11 – Un Réseau de Petri et son invariant de marquages

2.4.6 Siphons et Pièges

Définition 19. *Un siphon est un ensemble de places S qui est tel que l'ensemble des transitions d'entrées ${}^\bullet S$ est inclus dans l'ensemble des transitions de sorties $S^\bullet$ (${}^\bullet S \subseteq S^\bullet$). Par conséquent, l'ensemble des transitions prédécesseures strictes est vide :*

$${}^\circ S = \varnothing$$

Propriété 6. *Une fois qu'un siphon S d'un Réseau de Petri $\mathcal{N}$ est vidé de ses jetons, le siphon ne pourra plus jamais se remplir :*

$$\forall M \in \mathbb{N}^{|P|} \mid \sum_{p \in S} M(p) = 0 \implies \nexists M' \in \mathcal{R}(\mathcal{N}, M) \mid \sum_{p \in S} M'(p) > 0$$

Cela implique qu'une fois le siphon vidé, les transitions successeurs ne seront plus jamais sensibilisées et sont donc mortes :

$$\forall M \in \mathbb{N}^{|P|} \mid \sum_{p \in S} M(p) = 0 \implies \forall t \in S^\bullet, \nexists \sigma \in T^* | M[\sigma t\rangle$$

Si un Réseau de Petri marqué $(\mathcal{N}, M_0)$ comporte un siphon S et qu'il existe un marquage $M \in \mathcal{R}((\mathcal{N}, M_0))$ tel que $\sum_{p \in S} M(p) = 0$, alors toutes les transitions $t \in S^\bullet$ ne sont pas vivantes.

Remarque 9. *Considérant un Réseau de Petri $\mathcal{N}$ et un sous-ensemble de places $X \subseteq P$, on se permet de noter le nombre de jetons dans X pour un marquage M par $M(X) = \sum_{p \in X} M(p)$. Cette notation est particulièrement utile dans le cadre des siphons où le marquage du siphon est une donnée critique.*

L'ensemble des siphons d'un Réseau de Petri $\mathcal{N}$ est noté $\Pi(\mathcal{N})$. La notion de siphon est uniquement structurelle et ne dépend pas du marquage. Un siphon minimal $S_{\check{m}} \in \Pi_{\check{m}}(\mathcal{N})$ est un siphon qui n'est contenu dans aucun autre siphon et $\Pi_{\check{m}}(\mathcal{N})$ est l'ensemble des siphons minimaux de $\mathcal{N}$. $\Pi_{\check{m}}(\mathcal{N})$ est défini comme suit : $\Pi_{\check{m}}(\mathcal{N}) = \{S_{\check{m}} \in \Pi(\mathcal{N}) | \nexists S \in \Pi(\mathcal{N}) \wedge S \subset S_{\check{m}}\}$. De la même manière on définit l'ensemble des siphons maximaux $\Pi_{\hat{M}}(\mathcal{N})$ qui est défini comme $\Pi_{\hat{M}}(\mathcal{N}) = \{S_{\hat{M}} \in \Pi(\mathcal{N}) | \nexists S \in \Pi(\mathcal{N}) \wedge S_{\hat{M}} \subset S\}$. Un siphon est dit strict s'il possède au moins une transition de sortie stricte. L'ensemble des siphons stricts est noté $\Pi_s(\mathcal{N}) = \{S \in \Pi(\mathcal{N}) \, | \, |S^\circ| > 0\}$. Lorsqu'un siphon est strict et minimal, on dira que c'est un siphon strict minimal (*strict minimal siphon* ou SMS). L'ensemble des siphons stricts minimaux est noté $\Pi_{SMS}(\mathcal{N}) = \Pi_{\check{m}}(\mathcal{N}) \cap \Pi_s(\mathcal{N})$.

Définition 20. *Un piège (ou trappe) est un ensemble de places Q tel que son ensemble de transitions de sorties $Q^\bullet$ est inclus dans son ensemble de transition d'entrées $S^\bullet$ ($Q^\bullet \subseteq {}^\bullet Q$). Par conséquent, l'ensemble des transitions successeures strictes de Q est vide :*

$$Q^\circ = \varnothing$$

Propriété 7. *Dans un Réseau de Petri $\mathcal{N}$, un piège Q qui est non vide ne se videra jamais :*

$$\forall M \in \mathbb{N}^{|P|} \mid \sum_{p \in Q} M(p) > 0 \implies \nexists M' \in \mathcal{R}(\mathcal{N}, M) \mid \sum_{p \in Q} M'(p) = 0$$

L'ensemble des pièges d'un Réseau de Petri $\mathcal{N}$ est noté $\Omega(\mathcal{N})$.

Exemple

Dans l'exemple de la figure 2.12, un RdP marqué $(\mathcal{N}, M_0)$ et son graphe d'accessibilité sont présentés. Dans le graphe d'accessibilité, une représentation sous

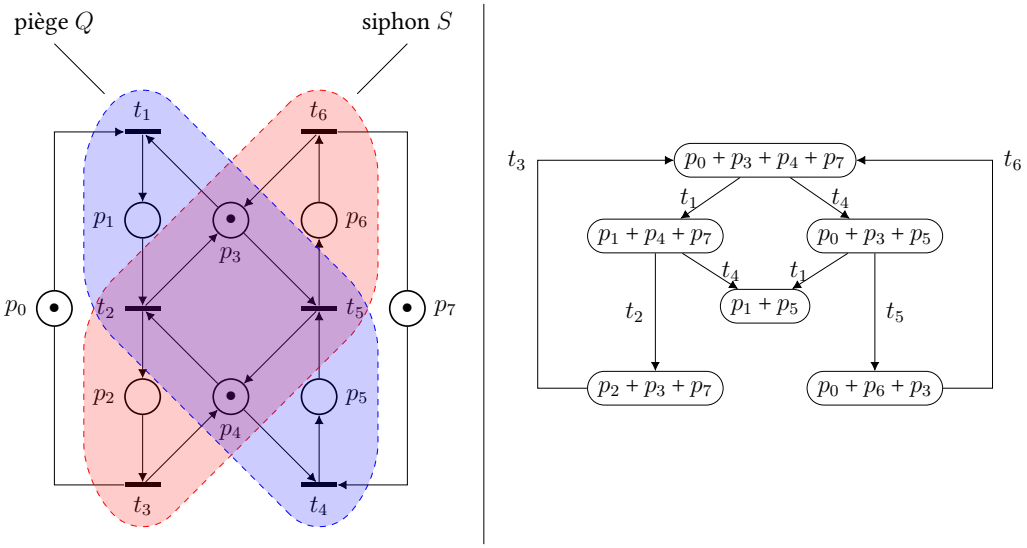


FIGURE 2.12 – Siphon et piège d'un Réseau de Petri et son graphe d'accessibilité

forme de multi-set est adoptée pour des raisons de clarté. Le RdP comporte un siphon strict minimal (SMS), $S \in \Pi_{SMS}(\mathcal{N})$ avec $S = \{p_2, p_3, p_4, p_6\}$. Le siphon S et ses transitions prédécesseurs sont entourés en rouge. S comporte deux transitions successeures strictes, $S^\circ = \{t_1, t_4\}$, mais aucune transition successeure stricte ($^\circ S = \varnothing$ ce qui correspond à la définition du siphon strict).

Le RdP $\mathcal{N}$ comporte lui aussi un piège $Q \in \Omega(\mathcal{N})$ avec $Q = \{p_1, p_3, p_4, p_5\}$. Sur la figure 2.12, Q est représenté en bleu avec ses transitions successeurs. Q comporte deux transitions prédécesseures strictes $^\circ Q = \{t_3, t_6\}$ et aucune transition successeure stricte ($Q^\circ = \varnothing$).

Si on se réfère au graphe d'accessibilité, $(\mathcal{N}, M_0)$ est dans un état bloquant en $p_1 + p_5$. Dans cet état, S est vide, ce qui empêche le tir des transitions de $S^\bullet = T$. Dans cet état, toutes les transitions sont bloquées et donc le RdP n'est pas vivant. Le piège Q quant à lui reste marqué tout au long de l'évolution du RdP et peut capturer les jetons dans l'état bloquant $p_1 + p_5$.

2.5 Conclusion

Ce chapitre introduit les SED. Trois formalismes de modélisation des SED y sont présentés : les langages, les automates et les Réseaux de Petri. Les langages sont relativement compliqués à utiliser en pratique : ils ne bénéficient pas d'une représentation graphique. Cependant les langages se trouvent être d'excellents outils de preuves pour les Réseaux de Petri et les automates. Les automates sont d'ailleurs considérés comme des générateurs de langages (une extension des Réseaux de Petri : les Réseaux de Petri étiquetés sont aussi des générateurs de langages). Les automates ont historiquement été développés avant les Réseaux de Petri et ils bénéficient d'un développement très abouti. Ce chapitre n'introduit que les notions élémentaires des automates, nécessaires au chapitre 3. Les Réseaux de Petri, grâce à la notion de jeton et au parallélisme, permettent d'avoir une représentation plus compacte que celle

des automates dans le cas général. Leur expression sous forme matricielle permet d'utiliser des méthodes issues de l'algèbre linéaire ce qui fait des Réseaux de Petri un outil puissant. Dans ce chapitre, seules les notions et méthodes d'analyses élémentaires ont été présentées.

Chapitre 3

Synthèse de contrôleurs

La synthèse de contrôleurs considère un système connu dont on est capable de modéliser le fonctionnement grâce à un formalisme SED quelconque. Ce modèle correspond au comportement non contraint du système. Contrôler le système, consiste à contraindre le comportement libre du système par le biais de spécifications. Les spécifications doivent également pouvoir être modélisées de manière formelle. Elles peuvent être diverses : des états à atteindre ou à éviter, l'évitement des blocages, des évolutions interdites ... À partir du modèle et de ses spécifications, un contrôleur est synthétisé. Ce contrôleur définit pour le système un cadre de fonctionnement sûr dans lequel les spécifications ne peuvent être violées. Pour avoir de l'influence sur le système, le contrôleur doit avoir des possibilités d'action sur le système. Dans [RW87], Ramadge et Wonham proposent de distinguer les événements en deux sous-ensembles : ceux sur lequel le contrôleur a de l'influence, appelés événements contrôlables, et ceux sur lesquels il n'en a pas, qualifiés d'événements incontrôlables. Ce chapitre présente la Théorie du Contrôle Supervisé (*Supervisory Control Theory* (SCT)), qui dans un premier temps a été présentée sous le formalisme des automates, et qui par la suite a été étendue aux Réseaux de Petri.

Un contrôleur, pour un système à événements discrets, peut toujours être représenté par le schéma fonctionnel de la figure 3.1. Le contrôleur, à partir d'une observation partielle ou totale des événements générés par le système, déduit l'état courant du système. À partir de cette déduction, le contrôleur autorise ou interdit l'apparition d'événements dans le système. Dans ce chapitre, nous présentons une

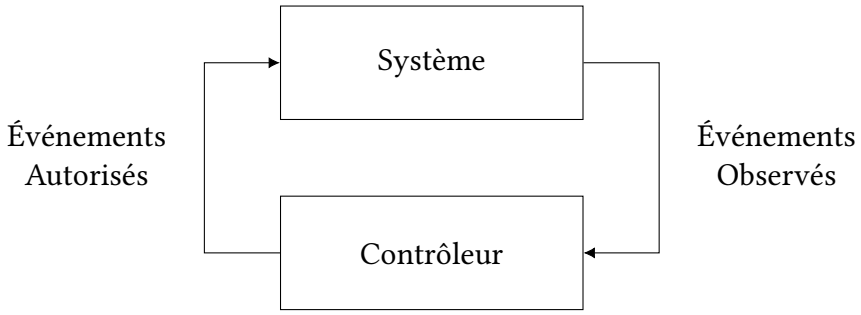


Figure 3.1 – Schéma fonctionnel d'un SED et de son contrôleur

synthèse des principaux résultats de la synthèse de contrôleurs que nous exploiterons dans le cadre de cette thèse. Les méthodes de synthèse de contrôleurs peuvent être classifiées selon différents critères parmi lesquels le formalisme utilisé ou le type de spécification. Dans ce chapitre, nous commencerons par rappeler la Théorie du Contrôle Supervisé. C'est la première méthode de synthèse de contrôleurs

qui a été proposée par Ramadge et Wonham en 1987. L'intérêt de la présentation de cette méthode est de nous permettre d'introduire les concepts de base de la synthèse de contrôleurs. D'autre part, cette méthode basée sur le formalisme des automates à états finis est une méthode optimale du point de vue de la permissivité, l'un des principaux critères de comparaison des méthodes. La SCT sera utilisée pour la comparer aux méthodes que nous proposons d'un point de vue permissivité. Au niveau formalisme, par la suite, nous ferons le choix des Réseaux de Petri. Par rapport à ce formalisme, il existe différents types de spécifications basés soit sur les états du système, soit sur les transitions, soit sur les séquences de transitions. Dans notre travail, nous allons nous intéresser en particulier aux spécifications d'états interdits. Une section de ce chapitre sera donc consacrée à rappeler les principales méthodes structurelles pour la synthèse de contrôleurs en se basant sur ce type de spécification. Mais les Réseaux de Petri possèdent également des propriétés intrinsèques (finitude, vivacité) qu'il est nécessaire de garantir afin d'avoir des contrôleurs performants. La dernière section de ce chapitre s'intéressera donc aux méthodes permettant d'éviter les blocages d'un Réseau de Petri afin d'en garantir la vivacité.

3.1 Théorie du Contrôle Supervisé

La Théorie du Contrôle Supervisé s'appuie sur la théorie des langages, et permet d'établir un cadre général pour la synthèse de contrôleurs des SED. Une implémentation directe de cette théorie peut être mise en place au moyen d'automates à états finis. Le modèle du système et les spécifications sont alors modélisés au moyen d'un automate à états finis. Le contrôleur synthétisé est appelé superviseur.

Dans sa formulation originale, le travail de Ramadge et Wonham s'appliquait aux langages et considérait des langages non-préfixes-clos. Dans le contexte de ce travail nous ne considèrerons que des langages qui sont obtenus à partir d'automates, par nature préfixes-clos.

3.1.1 Supervision sous contrôlabilité partielle

Contrôlabilité

La fonction du superviseur pourrait être vue comme une fonction S qui absorbe une séquence d'événements générée par le système G avec $\mathcal{L}(G) \subseteq E^*$ et qui en fonction de cette séquence choisit les événements à autoriser.

$$S : \mathcal{L}(G) \rightarrow 2^E$$

Définition 21. *Le superviseur ne peut empêcher l'apparition que d'un certain nombre d'événements appelés événements contrôlables. L'ensemble de ces événements est appelé E_c. A l'inverse les événements incontrôlables sont toujours autorisés par le contrôleur et sont représentés par l'ensemble E_{uc}.*

$$E_c \cap E_{uc} = \varnothing \quad \textit{et} \quad E = E_c \uplus E_{uc}$$

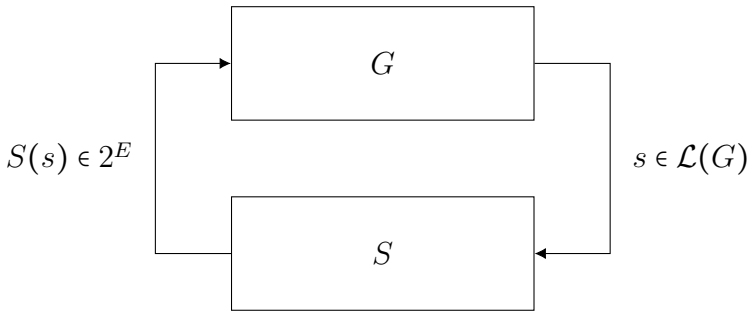


FIGURE 3.2 – Schéma d'un superviseur

Pour chaque séquence $s \in \mathcal{L}(G)$ la relation suivante doit être vérifiée :

$$E_{uc} \cap \Gamma(\delta(x_0, s)) \subseteq S(s)$$

Définition 22. *On note S/G le système G supervisé par S. Le langage généré par S/G est défini récursivement :*

1. $\epsilon \in \mathcal{L}(S/G)$
2. $\mathcal{L}(S/G) = \{s\sigma \mid s \in \mathcal{L}(S/G) \wedge \sigma \in S(s)\}$

Avec le langage marqué ne dépendant uniquement que de G :

$$\mathcal{L}_m(S/G) = \mathcal{L}(S/G) \cap \mathcal{L}_m(G)$$

Théorème de contrôlabilité

Dans le cas de la contrôlabilité partielle, on considère un langage $\bar{K} = \mathcal{L}(S/G)$ d'un système G soumis à l'action d'un superviseur S. Connaissant le langage généré par G, si K est défini sans précautions, il est possible que le superviseur S ne respecte pas la définition 21 et tente d'interdire un événement incontrôlable. Le langage d'un système supervisé respecte la condition de contrôlabilité si une séquence d'événements incontrôlables apparaissant après n'importe quel mot de $\bar{K}$ amène à quitter $\bar{K}$. Le théorème de contrôlabilité définit si un superviseur respecte la condition de contrôlabilité.

Théorème 1. ***Théorème de contrôlabilité*** *: Un langage $K \neq \varnothing$ et $K \subseteq E^*$ est contrôlable par rapport à $\mathcal{L}(G)$ et E_{uc} si :*

$$\overline{K}.E_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$$

Définition 23. *Soit un langage $K \neq \varnothing$ et $K \subseteq \mathcal{L}(G)$. Si K est contrôlable par rapport à $L(G)$ et E_{uc}, il existe un superviseur S tel que $\mathcal{L}(S/G) = \overline{K}$.*

Contrôle non-bloquant

Pour que le superviseur impose un contrôle non-bloquant, le langage généré par le système supervisé doit respecter :

— $K = \mathcal{L}_m(S/G)$ et $\mathcal{L}(S/G) = \overline{K}$

— $K = \overline{K} \cap \mathcal{L}_m(G)$

Cette relation s'appelle la $\mathcal{L}_m(G)$-clôture.

Langage contrôlable suprême

Définition 24. *On définit $\mathcal{C}_{in}(K)$ la classe des sous-langages de K qui respectent la relation de contrôlabilité*

$$\mathcal{C}_{in}(K) = \{L \subseteq K \mid \overline{L}E_{uc} \cap \mathcal{L}(G) \subseteq \overline{L}\}$$

Définition 25. *Le suprême-contrôlable est le langage généré par le contrôleur le plus permissif, incluant toutes les séquences des langages de la classe $\mathcal{C}_{in}(K)$.*

$$K^{\uparrow C} = \bigcup_{L \in \mathcal{C}_{in}(K)} L$$

3.1.2 Réalisation du superviseur

D'un point de vue pratique la définition d'un superviseur comme étant une fonction $S : \mathcal{L}(G) \to 2^E$ est difficile à mettre en œuvre. On appelle R_S la réalisation d'un superviseur S, un automate qui va représenter l'effet du superviseur de façon à ce que $\mathcal{L}(R_S \parallel G) = \mathcal{L}(S/G)$.

Dans le cas d'un système totalement contrôlable (qui ne comprend pas d'événement incontrôlable i.e $E_{uc} = \varnothing$), une réalisation d'un contrôleur maximalement permissif H_0 est obtenue par le produit synchrone de H, l'automate des spécifications et G, l'automate modélisant le comportement du système : $H_0 = H \times G$. H_0 génère le langage admissible $L_a = \mathcal{L}(H_0)$, qui respecte les spécifications ($L_a \subseteq \mathcal{L}(H)$) mais ne respecte ni la condition de contrôlabilité, ni la condition de contrôle non-bloquant.

L'algorithme 3 présente comment modifier de manière itérative l'automate H_0 de manière à ce qu'il respecte les conditions de contrôlabilité et de non-blocage.

On considère que le système est modélisé par un automate $G = (X, E, \delta, \Gamma, x_0, X_m)$ et, les spécifications par un automate $H = (Y, E, \theta, \Gamma_H, y_0, Y_m)$ avec $Y_m = Y$ et $\mathcal{L}_m(H) = K$.

L'automate H_i obtenu après application de l'algorithme 3 a les propriétés suivantes :

$$\mathcal{L}_m(H_i) = K^{\uparrow C} \quad \text{et} \quad \mathcal{L}(H_i) = \overline{K^{\uparrow C}}$$

3.1.3 Exemple

On considère le système présenté en figure 3.3. Deux robots a et b sont chargés de concevoir des pièces différentes : la pièce a_4 dans le cas du robot a et la pièce b_3 dans le cas du robot b. Les processus de création des pièces nécessitent de construire des pièces intermédiaires a_1, a_2 et a_3 dans le cas du robot a, b_1 et b_2 dans le cas du robot b. La pièce a_4 est obtenue par assemblage des pièces a_1, a_2 et a_3 (respectivement b_3 par assemblage de b_1 et b_2 pour le robot b). Les robots construisent les pièces dans l'ordre suivant $a_1 \to a_2 \to a_3 \to a_4$ pour le robot a et $b_1 \to b_2 \to b_3$ pour le robot b. Une fois les pièces assemblées les robots recommencent leur processus. Lorsque les pièces a_1, a_2, a_3, b_1 et b_2 sont construites, elles sont placées dans une zone de

Algorithme 3 Réalisation d'un superviseur

Entrée : G : L'automate modélisant le système et H : l'automate modélisant les spécifications **Sortie :** H_i : la réalisation du superviseur

1: $H_0 = (Y_0, E, \theta_0, (y_0, x_0), Y_{0,m}) = H \times G$ avec Γ_{H_0} la fonction d'événements actifs associée à H_0
2: $i = 0$
3: $Y_i' = \{(y,x) \in Y_i \,|\, \Gamma(x) \cap E_{uc} \subseteq \Gamma_{H_i}((y,x))\}$
4: $\theta_i' = \theta_i$ réduit au domaine de départ Y_i'
5: $Y_{i,m}' = Y_{i,m} \cap Y_i'$
6: $H_{i+1} = Trim(Y_i', E, \theta_i', (y_0, x_0), Y_{i,m}')$
7: **si** H_{i+1} est un automate vide **alors**
8: STOP, il n'existe pas de superviseur qui satisfait les spécifications
9: **sinon**
10: $(Y_{i+1}, E, \theta_{i+1}, (y_0, x_0), Y_{i+1,m}) = H_{i+1}$
11: **fin si**
12: **si** $H_{i+1} = H_i$ **alors**
13: STOP, H_i est la réalisation du superviseur
14: **sinon**
15: $i \leftarrow i + 1$
16: retour à la ligne 3
17: **fin si**

stockage où elles attendent d'être assemblées. Cette zone de stockage a une capacité de trois pièces qui ne peut pas être dépassée.

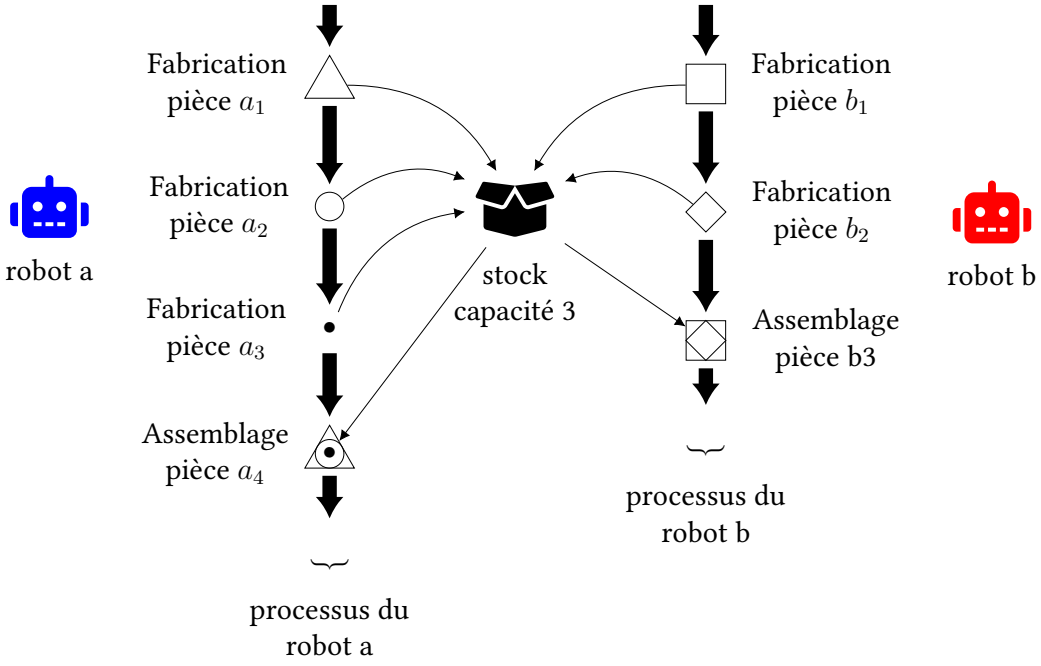


FIGURE 3.3 – Processus de fabrication de pièces par des robots

Les automates des figures 3.4 modélisent le fonctionnement du robot a. Les événements a_1, a_2, a_3 et a_4 représentent des événements correspondant à la fin de la

construction des pièces a_1, a_2,a_3 et a_4. De même, les états x_1^a, x_2^a, x_3^a et x_4^a correspondent à la construction des pièces a_1, a_2, a_3 et a_4. L'état x_1^a est marqué, car on souhaite qu'après l'assemblage de a_4, le robot a puisse toujours revenir au début de son cycle de production. La description est équivalente pour le robot b et l'automate de la figure 3.5. On suppose que la dépose de la pièce a_2 ne peut pas être empêchée par la partie opérative. Dans le cas des autres pièces, les robots peuvent attendre un ordre de dépôt dans la zone de stockage.

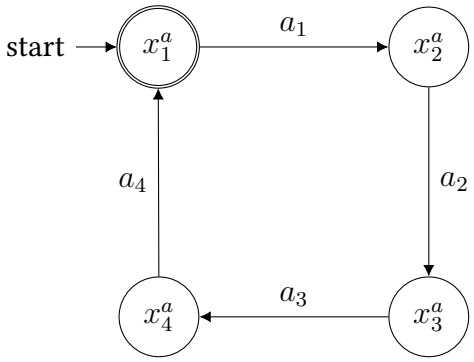


FIGURE 3.4 – Modèle de comportement du robot a

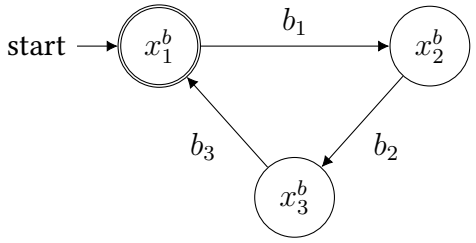


FIGURE 3.5 – Modèle de comportement du robot b

L'automate G qui modélise le fonctionnement des deux robots est obtenu par le produit synchrone des automates des figures 3.4 et 3.5. Cet automate est nommé G, son alphabet associé $E = \{a_1, a_2, a_3, a_4, b_1, b_2, b_3\}$ est séparé en deux sous-ensembles, les événements contrôlables $E_c = \{a_1, a_3, a_4, b_1, b_2, b_3\}$ et les événements incontrôlables $E_{uc} = \{a_2\}$.

L'automate de spécification de la figure 3.7 modélise la limite à trois pièces dans l'espace de stockage. Les états 0, 1, 2 et 3 correspondent au nombre de pièces dans la zone de stockage. Ainsi les événements a_1, a_2, a_3, b_1 et b_2 font passer de l'état 0 à l'état 1, de l'état 1 à l'état 2 et de l'état 2 à l'état 3, et correspondent à l'ajout d'une pièce dans la zone de stockage. Par contre l'ajout d'une pièce est interdit lorsqu'il y a trois pièces en stock, les événements a_1, a_2,a_3,b_1 et b_2 ne peuvent donc pas apparaître dans l'état 3. L'événement b_3 qui correspond à la fin de l'assemblage des pièces b_1 et b_2 consomme deux pièces du stock faisant passer de l'état 2 à l'état 0 et, de l'état 3 à l'état 1. De la même façon, l'assemblage de a_4, consomme 3 pièces, et fait donc passer de l'état 3 à l'état 0.

L'automate H_0 générant le langage admissible est représenté en figure 3.8. Il est obtenu par le produit totalement synchrone $H_0 = G \times H$. Les arcs associés aux événements autorisés par le système G mais interdits par le superviseur sont marqués en pointillé. Cet automate n'est pas une réalisation de superviseur. Il ne satisfait pas les conditions de contrôlabilité et de contrôle non-bloquant. L'état $x_2^a, x_3^b, 3$ viole la relation de contrôlabilité, car l'événement incontrôlable a_2 est permis depuis d'état x_3^a, x_2^b de G mais est interdit depuis l'état 3 de H. L'algorithme 3 à l'étape 3 ne garde pas $x_2^a, x_3^b, 3$ dans Y_1. L'état $x_3^a, x_2^b, 3$ est bloquant ($\Gamma_{H_0}(x_3^a, x_2^b, 3) = \varnothing$ et $x_3^a, x_2^b, 3$ n'est pas marqué). Il est supprimé par l'étape 6 qui effectue l'opération $Trim$. Dans l'itération suivante, l'état $x_2^a, x_2^b, 2$ est supprimé pour non-respect de la relation de contrôlabilité.

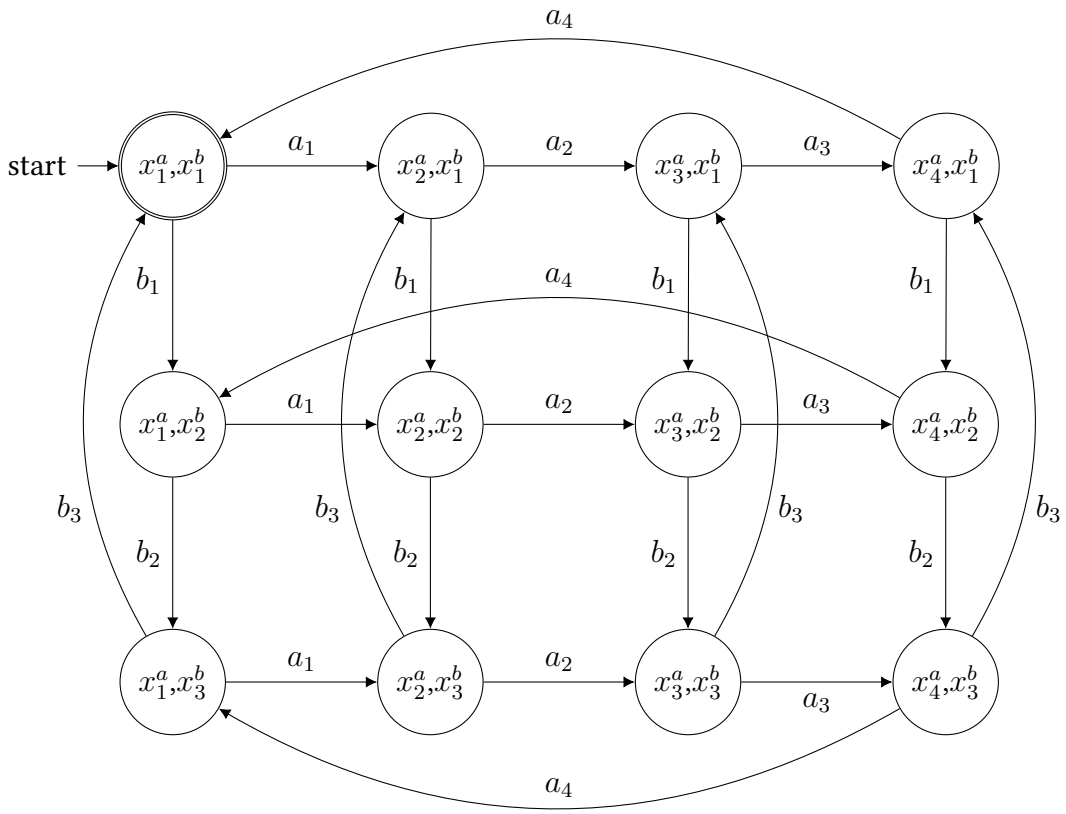


FIGURE 3.6 – Automate G modélisant le fonctionnement des robots a et b

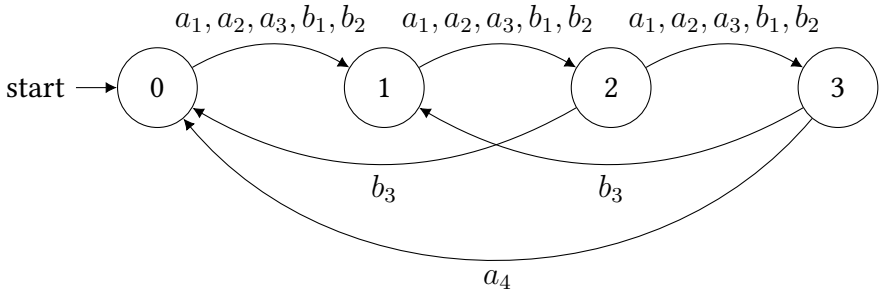


FIGURE 3.7 – Automate H de spécification de la capacité de stockage

L'algorithme 3, s'arrête au bout de deux itérations. L'automate H_2 illustré en figure 3.9 est la réalisation du superviseur qui est maximalement permissif pour G par rapport à H. La relation suivante $\mathcal{L}(H_2 \parallel G) \subseteq \mathcal{L}(H)$ est donc respectée ainsi que la relation de contrôlabilité et de contrôle non-bloquant.

3.1.4 Contrôle sous observabilité et contrôlabilité partielles

Dans le contexte de la contrôlabilité partielle, les événements sont divisés en événements contrôlables et événements incontrôlables. Dans le cas de l'observation partielle les événements sont en plus séparés en événements observables et

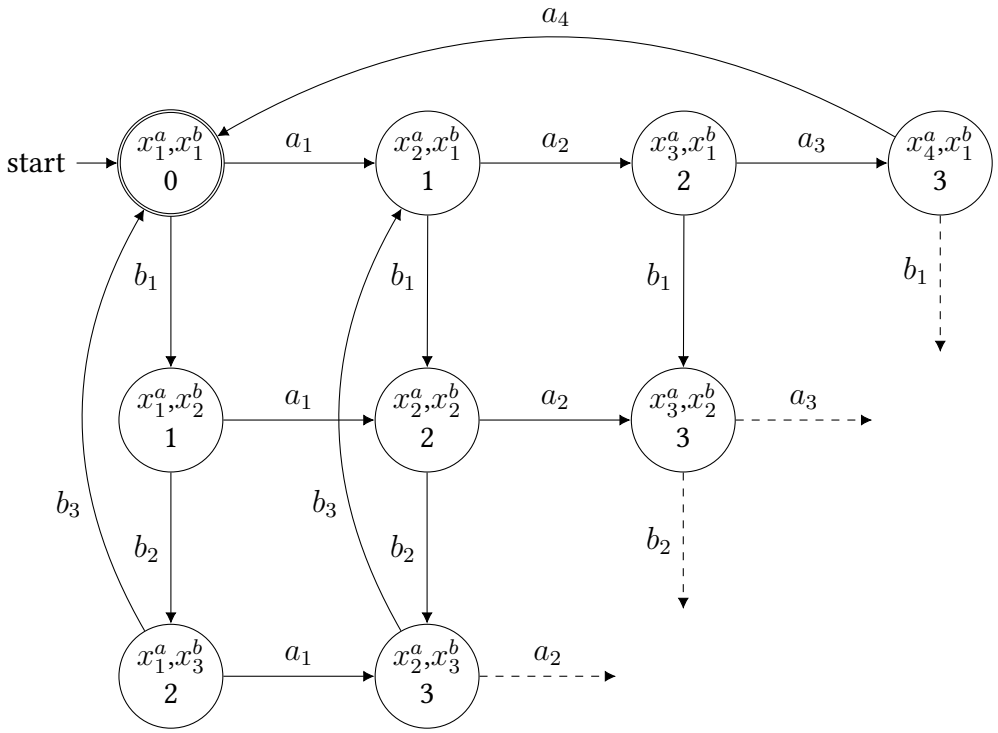


FIGURE 3.8 – Automate H_0 générant le langage admissible

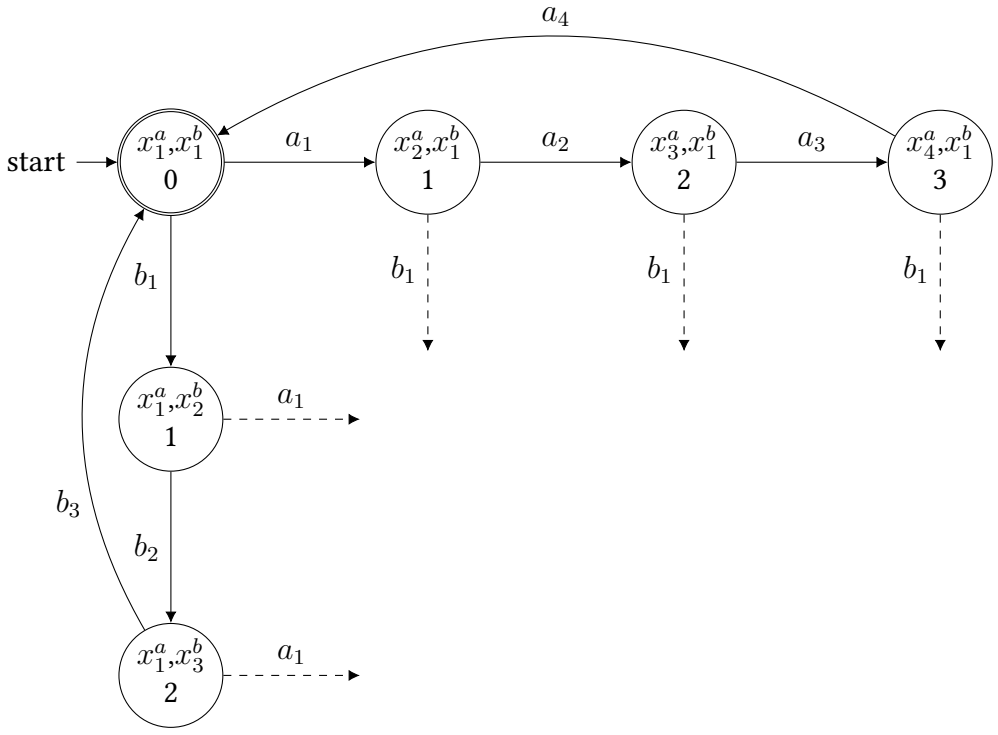


FIGURE 3.9 – Automate H_2 générant le langage maximalement permissif

inobservables. On nomme l'ensemble des événements observables E_o et l'ensemble des événements inobservables E_{uo} avec $E = E_o \cup E_{uo}$. Le superviseur ne peut pas baser son contrôle sur l'occurrence d'événements inobservables. En d'autres termes

le superviseur ne voit pas l'occurrence d'événements inobservables. La boucle de rétroaction présentée au début du chapitre 3 doit être modifiée. Un "filtre" est ajouté qui empêche le superviseur de voir les événements inobservables.

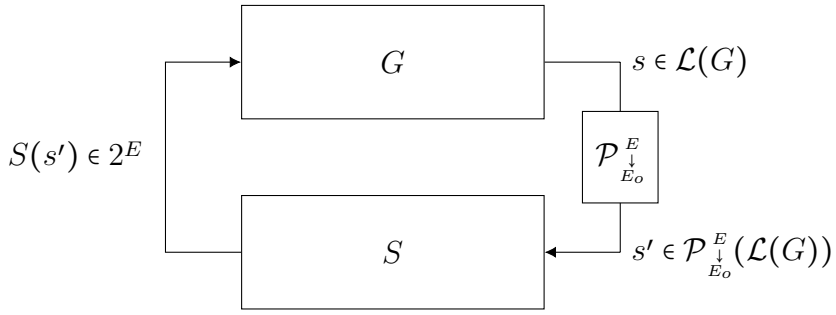


FIGURE 3.10 – Boucle de rétroaction du superviseur en observation partielle

Dans le cas de l'observabilité partielle la définition du superviseur de la sous-section 3.1.1 doit être revue pour prendre en compte les événements inobservables. Dans le cas de l'observabilité partielle, le contrôle est exercé par un P-superviseur S_P et est défini comme suit :

$$S_P : \mathcal{P}^E_{\downarrow E_o}(\mathcal{L}(G)) \to 2^E$$

Le langage du système contrôlé par un P-superviseur est défini récursivement par :

1. $\epsilon \in \mathcal{L}(S_P/G)$
2. $\mathcal{L}(S_P/G) = \{s\sigma \,|\, s \in \mathcal{L}(S_P/G) \wedge \sigma \in S_P(\mathcal{P}^E_{\downarrow E_o}(s))\}$

Théorème de contrôlabilité et d'observabilité

Les notions de contrôlabilité et de contrôle non-bloquant doivent être étendues au cas des systèmes partiellement observables. L'idée est que le contrôle associé à deux séquences d'événements qui ne peuvent être différenciées doit être identique.

Définition 26. *Soit $\mathcal{L}(G)$ le langage du système et $K = \mathcal{L}_m(S_P/G)$ le langage marqué du système contrôlé tous deux définis sur l'alphabet E. E_o est le sous-ensemble des événements observables de E et E_c est le sous-ensemble des événements contrôlables de E. K est dit observable par rapport à G, E_o et E_c si :*

$$\forall s \in \overline{K}, \forall \sigma \in E_c \,|\, (s\sigma \notin \overline{K}) \text{ et } (s\sigma \in \mathcal{L}(G)) \implies \mathcal{P}^E_{\uparrow E_o}(\mathcal{P}^E_{\downarrow E_o}(s))\sigma \cap \overline{K} = \varnothing$$

Remarque 10. *L'observation n'est pas close par union, c'est-à-dire que si deux langages K_1 et K_2 respectent la condition d'observabilité (26), l'union des deux langages $K_3 = K_1 \cup K_2$ ne le respecte pas forcément. Cette propriété empêche d'avoir un langage observable suprême défini de la même manière que la définition 25 pour le langage suprême contrôlable. Il existe potentiellement plusieurs langages maximaux observables.*

3.1.5 Normalité et Observabilité

Le fait que l'observabilité ne soit pas close sous l'union empêche la définition d'un langage suprême observable. Cependant il existe une propriété plus forte, qui implique l'observabilité du langage et qui est close sous l'union. C'est la notion de normalité qui est définie ci-après.

Définition 27. *$K \subseteq E^*$ est normal par rapport à $\mathcal{L}(G) = \overline{\mathcal{L}(G)}$ et E_o si :*

$$\overline{K} = \mathcal{P}^{E}_{\uparrow E_o}(\mathcal{P}^{E}_{\downarrow E_o}(K)) \cap \mathcal{L}(G)$$

Propriété 8. *La normalité possède les propriétés suivantes :*

- *La normalité d'un langage implique l'observabilité de ce langage,*
- *La normalité est close sous l'union,*
- *Lorsque $E_c \subseteq E_o$, si K est contrôlable par rapport à E_c et $\mathcal{L}(G)$ et observable par rapport à E_o, E_c et $\mathcal{L}(G)$ alors K est normal.*

On notera le langage suprême normal $K^{\uparrow N}$. Il peut être exprimé sous forme d'unions comme pour le langage contrôlable suprême $K^{\uparrow C}$(Définition 25).

3.1.6 Réalisation d'un P-superviseur

On se place dans le contexte ou l'on souhaite générer un superviseur non-bloquant et contrôlable d'un système G comportant des événements non-observables E_{uo} et non-contrôlables E_{uc}. Si on considère un langage admissible L_a préfixe-clos ($L_a = \overline{L_a}$) , on souhaite que le superviseur restreigne le comportement de G de la manière suivante, en respectant les points suivants :

1. $\mathcal{L}_m(S_P/G) \subseteq L_{a,m}$
2. $\mathcal{L}_m(S_P/G)$ est le plus englobant possible ; si on considère n'importe quel P-superviseur non-bloquant S'_P tel que $\mathcal{L}_m(S'_P/G) \subseteq L_{a,m}$, il respecte :

$$\mathcal{L}_m(S'_P/G) \subseteq \mathcal{L}_m(S_P/G)$$

La réalisation d'un superviseur n'est pas aussi directe dans le cas d'un système partiellement observable qu'en observation totale. On peut identifier au moins trois approches qui permettent de proposer un contrôleur :

a. Le langage suprême observable n'existe pas du fait de la remarque 10. Il n'est pas possible de trouver un contrôleur qui respecte le point 2. Cependant il est possible de trouver un langage maximal (on pourrait parler de maximal local), le point 2 est donc modifié en :

   2' $\mathcal{L}_m(S_P/G)$ est maximal, il n'existe pas de P-superviseur non-bloquant S'_P tel que $\mathcal{L}_m(S'_P/G) \subseteq L_{a,m}$, qui respecte :

$$\mathcal{L}_m(S'_P/G) \subseteq \mathcal{L}_m(S'_P/G)$$

   Il existe donc plusieurs contrôleurs maximaux possibles qui peuvent être sélectionnés. Un algorithme de génération de contrôleur est présenté en [TU03].

b. La construction du superviseur suprême normal et contrôlable garantit que le superviseur respecte la condition d'observabilité. Le contrôleur est unique, cependant la notion de normalité étant plus forte que celle de l'observabilité, il est possible qu'il existe un contrôleur maximal (au sens de a.) plus permissif. La méthode pour générer la réalisation du superviseur normal suprême non-bloquant est décrite en [CM89].

c. Dans le cas ou $E_c \subseteq E_o$ il existe un langage suprême observable. Dans ce cas, le calcul est immédiat, ne requérant que le calcul de $Obs(G \parallel H)$ et l'ajout de self-loop.

Remarque 11. *Toutes les méthodes requièrent le calcul d'un observateur. L'opération Obs étant de complexité exponentielle, la supervision des systèmes partiellement observables devient vite impossible lorsque le système est de grande taille et/ou complexe.*

3.2 Évitement d'états interdits dans les Réseaux de Petri par génération d'invariants

Comme pour les automates, il est possible de synthétiser un superviseur par identification des états interdits. La théorie des régions est une approche directe qui consiste à calculer le graphe des marquages accessibles et à appliquer une méthode de type Ramadge et Wonham sur le graphe des marquages accessibles. À partir de l'automate superviseur, un Réseau de Petri qui aura un contrôle équivalent, est généré. Cette méthode est d'abord présentée par [GRX03] pour les Réseaux de Petri partiellement contrôlables puis elle a été étendue par [ARX04] aux Réseaux de Petri partiellement contrôlables et observables. Cette méthode permet d'avoir un contrôle maximalement permissif et peut être non-bloquant si souhaité. L'inconvénient majeur de ces méthodes est qu'elles se basent sur le calcul du graphe des marquages accessibles. D'autre part, la méthode de [ARX04] nécessite de calculer l'observateur du graphe des marquages accessibles. Le calcul du graphe des marquages accessibles et le calcul de son observateur sont des opérations de grande complexité, incompatibles avec des systèmes de grandes tailles. On notera tout de même que la méthode présentée en [RAR17] permet de réduire grandement la complexité du calcul, en évitant la construction du graphe des marquages accessibles grâce à l'exploitation de certaines propriétés des Réseaux de Petri. Malheureusement cette méthode ne traite pas, à notre connaissance, les systèmes sous observation partielle et n'est pas en mesure de donner un contrôle non-bloquant.

La supervision des RdP par génération d'invariants est une appellation donnée à un ensemble de méthodes. Elles ont pour point commun d'être structurelles, ce qui les rend relativement efficaces pour la synthèse de contrôleurs dans des systèmes de grande taille et/ou complexes. D'une manière générale, ces méthodes se basent sur des spécifications de type "états interdits" (voir la sous-section 3.2.1) et génèrent un contrôle sous forme de places monitrices. L'ajout de ces places monitrices génère un invariant supplémentaire dans le Réseau de Petri. Le lecteur pourra se référer à l'article de Iordache et Antsaklis [IA06] sur la supervision à base d'invariants pour des compléments d'information.

3.2.1 Contraintes généralisées d'exclusion mutuelle

Les *Contraintes généralisées d'exclusion mutuelle* ou GMECs (Generalised Mutual Exclusion Constraints) sont des contraintes d'évitement d'états interdits présentées dans [GDS92]. Elles consistent à définir un ensemble d'états autorisés respectant une contrainte sur le nombre de jetons dans des places d'un Réseau de Petri. Une pondération est associée à chaque place, la somme pondérée des jetons dans les places devant être inférieure ou égale à une limite. Ce type de contraintes peut donc être exprimée sous forme vectorielle comme une inégalité.

Définition 28. *Une Contrainte généralisée d'exclusion mutuelle est définie par un doublet* (w, b) *avec un vecteur de pondération de place* $w : P \to \mathbb{Z}$ *associé à une limite* $b \in \mathbb{Z}$*. L'ensemble des marquages autorisés par la contrainte est noté* $\mathcal{M}(w, b)$*.*

$$\mathcal{M}(w, b) = \{M \in N^{|P|} \,|\, w^\intercal.M \leq b\} \tag{3.1}$$

Une *Contrainte généralisée d'exclusion mutuelle* contraint le marquage d'un Réseau de Petri à évoluer dans un sous-espace de $\mathbb{N}^{|P|}$. L'espace des marquages $\mathbb{N}^{|P|}$ étant divisé en deux sous-espaces, les états autorisés et ceux interdits. La limite entre les deux sous-espaces étant le plan $w^\intercal.M = b$.

Un ensemble de n contraintes peut aussi être défini grâce à une matrice de contraintes $W \in \mathbb{Z}^{|P|\times n}$. Dans ce cas-là, le marquage autorisé est contraint de la manière suivante :

$$\mathcal{M}(W, B) = \{M \in N^{|P|} \,|\, W^\intercal.M \leq B\} \tag{3.2}$$

Ici chacune des colonnes de W correspond à un vecteur de pondération et $B \in \mathbb{Z}^n$ est un vecteur dont chaque composante est liée à une des n contraintes. Une définition équivalente à l'équation 3.2 peut être exprimée grâce à une intersection sur les marquages $\mathcal{M}(W, B) = \bigcap_{i\in\mathcal{I}} \mathcal{M}(W(, i), B(i))$, dans ce cas les contraintes sont dites **conjonctives**.

Remarque 12. *Dans [IA02], l'auteur propose d'étendre les spécifications en ajoutant des conditions sur le tir des transitions. Les contraintes sont alors nommées Contraintes linéaires généralisées (Generalized Linear Constraints) et sont exprimées de la manière suivante :*

$$\mathcal{M}(W, H, S, B) = \{(M, q, \vec{\sigma}) | W^\intercal.M + H.q + S.\vec{\sigma} \leq B\}$$
$$\text{avec} \quad M_0[\sigma\rangle M$$
$$q(t) = \begin{cases} 1 & \text{si } M[t\rangle \\ 0 & \text{sinon} \end{cases} \quad \forall t \in T$$

L'expression des spécifications est étendue par la prise en compte de deux dimensions supplémentaires. Le vecteur $q \in \mathbb{N}^{|T|}$ *associé à la matrice* $H \in \mathbb{N}^{n\times|T|}$ *autorise le tir d'une transition* $M[t_i\rangle M'$ *uniquement si* $W^\intercal.M + H.q \leq B$ *et* $W^\intercal.M' \leq B$*.* q *est donc le vecteur de tir d'une transition. Le cas où* $|q| > 1$ *correspond au tir de plusieurs transitions simultanées. Le vecteur* $\vec{\sigma} \in \mathbb{N}^{|T|}$ *est le porteur de la séquence de transitions* $\sigma \in T^*$*. La matrice* S *permet donc d'ajouter un poids au tir de chaque transition.*

3.2.2 Génération de places monitrices

A partir d'une contrainte GMEC, il est possible de générer une place qui contraint le marquage du Réseau de Petri à rester dans l'ensemble des marquages autorisés. Le contrôle prend donc la forme d'une place que l'on nomme place monitrice.

Définition 29. *On considère un Réseau de Petri* ***pur*** *marqué* $(\mathcal{N}, M_0)$ *avec* $\mathcal{N} = (P, T, Pre, Post)$ *et un GMEC (w,b).* v *est la place monitrice qui fait respecter le GMEC à* $\mathcal{N}$. $(\mathcal{N}^C, M_0^C)$ *est le Réseau de Petri marqué obtenu après l'ajout de la place monitrice avec* $\mathcal{N}^C = (P \cup \{V\}, T, Pre^C, Post^C)$. *La matrice d'incidence de* $\mathcal{N}^C$ *est obtenue de la manière suivante :*

$$C^C = \begin{bmatrix} C \\ -w^\intercal.C \end{bmatrix}$$

Le marquage de la place monitrice dépend du marquage initial du Réseau de Petri.

$$M_0^C = \begin{bmatrix} M_0 \\ b - w^\intercal.M_0 \end{bmatrix}$$

Dans le cas d'un ensemble de n *contraintes conjonctive, le calcul est identique :*

$$C^C = \begin{bmatrix} C \\ -W^\intercal.C \end{bmatrix} \quad M_0^C = \begin{bmatrix} M_0 \\ B - W^\intercal.M_0 \end{bmatrix}$$

n *places monitrices sont générées formant l'ensemble des places de contrôle* V. *Pour une place de contrôle* $i \in [1..n]$, *le vecteur d'incidence de* i *est* $[-W^\intercal.C](i,)$ *et son marquage initial* $[B - W^\intercal.M_0](i)$.

Remarque 13. *Dans le cas où une contrainte* (w, b) *est violée dès l'état initial, il est alors impossible pour le contrôleur de faire respecter la contrainte* (w, b). *La condition suivante est donc nécessaire pour la synthèse du contrôleur :*

$$w^\intercal.M_0 \leq b \tag{3.3}$$

Remarque 14. *On parle de contrôle par génération d'invariant lors de la génération d'une place monitrice. L'ajout d'une place monitrice génère un invariant dans le Réseau de Petri contrôlé* $(\mathcal{N}^C, M_0^C)$. $X \in \mathcal{P}_{\circlearrowleft}(\mathcal{N}^C)$ *est le P-flot généré après l'ajout de* V *avec* $X(P) = w$ *et* $X(V) = 1$. *Si on se réfère à la propriété 5 et à la définition 29, on obtient la relation suivante :*

$$\forall M \in \mathcal{R}(\mathcal{N}^C, M_0^C), \quad X^\intercal.M = X^\intercal M_0 = b$$

Remarque 15. *Nous utiliserons le terme de* ***place de contrôle*** *pour faire référence à une place ayant pour but d'exercer une fonction de contrôle quelconque, tandis que le terme* ***place monitrice*** *fait référence à une place de contrôle obtenue à partir d'une contrainte GMEC et générant un invariant (présenté dans ce chapitre).*

On utilise la notation $(\mathcal{N}^C, M_0) = (\mathcal{N}, M_0) \oplus (V, M_0(V))$ (ou simplement $\mathcal{N}^C = \mathcal{N} \oplus V$) lorsque une place de contrôle V marquée initialement par $M_0(V)$ est ajoutée à un Réseau de Petri marqué $\mathcal{N} = (P, T, F, W)$ pour obtenir le Réseau contrôlé $(\mathcal{N}^C, M_0^C)$. Dans le cas où $(V, M_0(V))$ est décrite via sa matrice d'incidence comme

pour la définition 29, l'obtention de $(\mathcal{N}^C, M_0)$ est directe. Cependant V peut aussi être décrite d'un point de vue structurel, par des arcs $\{f|\forall t \in V^\bullet, f = (V,t)\} \cup \{f|\forall t \in {}^\bullet V, f = (t,V)\} = F^V$ et l'application de pondération W^V sur F^V. On a alors $\mathcal{N}^C = (P \cup \{V\}, T, F \cup F^V, W^C)$ avec $W^C : F^V \to \mathbb{N}^+$ et $W^C(f) = W(f)$ si $f \in F$ et $W^C(f) = W^V(f)$ si $f \in F^V$.

Le contrôle par places monitrices avec contrôle et observation partiels offre une solution de contrôle suprêmement permissive au sens de Ramadge et Wonam [RW87]. De plus le contrôle est obtenu par une simple multiplication matricielle, ce qui est d'une faible complexité.

3.2.3 Transitions incontrôlables et inobservables

De la même manière que dans la théorie du contrôle supervisé où les systèmes peuvent comprendre des événements incontrôlables et inobservables, les Réseaux de Petri peuvent aussi être partiellement observables ou contrôlables. Dans le cas des RdP, ce sont les transitions qui sont séparées en sous-ensembles. L'ensemble des transitions contrôlables est noté T_c et à l'inverse, les transitions incontrôlables sont notées T_{uc}. Une transition ne peut être à la fois incontrôlable et contrôlable, et une transition doit forcément appartenir à un de ces sous-ensembles. On obtient donc l'union jointe suivante : $T = T_c \cup T_{uc}$. De la même manière les transitions peuvent être séparées en deux sous-ensembles T_o pour les transitions observables et T_{uo} pour les transitions inobservables avec $T = T_o \cup T_{uo}$

Transitions incontrôlables

Une transition incontrôlable est une transition dont la loi de contrôle ne peut empêcher le tir. Donc le tir de la transition ne peut être influencé que par la dynamique du système. Les transitions incontrôlables doivent elles aussi être sensibilisées pour pouvoir être tirées. Dans le contexte du contrôle par places monitrices, il ne peut pas y avoir d'arc liant une place de contrôle à une transition incontrôlable.

En effet, avoir un arc entre une transition et une place monitrice permet à la place monitrice d'influer sur la sensibilisation de la transition et ainsi d'empêcher son tir. Pour un Réseau de Petri contrôlé $\mathcal{N}^\mathcal{C} = (P \cup V, T, F, W)$ où P est l'ensemble des places du système libre et V les places de contrôle.

$$\forall t \in T_{uc} \text{ et } \forall v \in V, \quad \nexists f \in F \,|\, f = (v,t)$$

De manière équivalente, cette relation peut être traduite de manière matricielle pour un Réseau de Petri contrôlé $\mathcal{N}^C = (P \cup V, T, C^C)$, avec $C^C = \left[C^\intercal \, C^{V\intercal}\right]^\intercal$ C étant la matrice d'incidence correspondant au RdP libre et C^V la matrice d'incidence des places de contrôles V. Il en découle les notations Pre^V et $Post^V$ pour les matrices d'incidence Pre-tir et Post-tir relatives aux places de contrôles.

$$Pre^V(, T_{uc}) = 0 \tag{3.4}$$

Remarque 16. *Dans le cas de la génération de places monitrices, ces places ne génèrent pas d'arcs aller-retours. Le sous-Réseau de Petri formé par les places monitrices est donc*

pur. L'équation 3.4 peut donc être modifiée en :

$$C^V(,T_{uc}) \geq 0 \tag{3.5}$$

Transitions inobservables

Il est possible que le tir de certaines transitions ne soit pas visible du point de vue du contrôle. Le contrôleur ne peut donc pas faire évoluer son état courant sur le tir d'une transition inobservable. Pour un Réseau de Petri contrôlé $\mathcal{N}^{\mathcal{C}} = (P \cup V, T, F, W)$, c'est équivalent à dire que le tir d'une transition incontrôlable n'entraîne aucune modification du marquage. L'incidence des places de contrôle vis-à-vis des transitions inobservables doit être nulle, ce qui se traduit par :

$$C^V(,T_{uo}) = 0 \tag{3.6}$$

D'un point de vue structurel cela peut se traduire par deux types de relations entre les transitions :

1. Il n'y a pas d'arcs entre les places de contrôle et une transition inobservable ou
2. Une paire d'arcs "aller-retour" peut exister entre une place de contrôle et une transition si les arcs aller et retour ont la même pondération.

Remarque 17. *Dans le cadre de la génération de places monitrices présentée en 3.2.2, les places générées n'induisent pas de paires d'arcs "aller-retour". Dans ce contexte on pourra se restreindre au cas 1, donc éviter tout arc entre une place monitrice et une transition inobservable :*

$$\forall v \in V,\ \forall t \in T_{uo}, \quad \nexists f \in F \mid f = (v,t) \vee f = (t,v) \tag{3.7}$$

ou de manière équivalente

$$Pre^V(,T_{uo}) = 0 \quad \textit{et} \quad Post^V(,T_{uo}) = 0 \tag{3.8}$$

3.2.4 Contraintes admissibles et places monitrices

Pour une contrainte donnée, la génération directe d'une place monitrice peut-être rendue impossible par la présence de transitions incontrôlables et/ou inobservables.

Définition 30. *La condition d'admissibilité est obtenue grâce aux équations 3.5 et 3.6 et à la définition d'une place de contrôle pour une contrainte (w,b). Les deux conditions respectivement liées aux transitions incontrôlables et inobservables doivent être respectées :*

$$w^\top.C(,T_{uc}) \leq 0 \tag{3.9}$$

et

$$w^\top.C(,T_{uo}) = 0 \tag{3.10}$$

Définition 31. *On définit la classe des contraintes $\mathcal{C}^C(\mathcal{N})$ qui respectent la condition de contrôlabilité 3.9*

$$\mathcal{C}^C(\mathcal{N}) = \{(w, b) \in \mathbb{Z}^{|P|} \times \mathbb{Z} \,|\, w^\top.C(, T_{uc}) \leq 0\}$$

De la même manière on définit la classe des contraintes $\mathcal{C}^O(\mathcal{N})$, qui respectent la condition d'observabilité. 3.10

$$\mathcal{C}^O(\mathcal{N}) = \{(w, b) \in \mathbb{Z}^{|P|} \times \mathbb{Z} \,|\, w^\top.C(, T_{uo}) = 0\}$$

Les contraintes qui respectent les deux conditions appartiennent à la classe $\mathcal{C}^{CO} = \mathcal{C}^C \cap \mathcal{C}^O$.

3.2.5 Méthodes de supervision par transformation des contraintes

Lorsqu'une contrainte n'est pas admissible, il est possible de donner une solution de contrôle qui empêchera le système d'atteindre les états interdits. Plusieurs auteurs ont proposé des solutions de contrôle pour des contraintes non-admissibles.

Observation sur la transformation des contraintes

Dans l'article introduisant le contrôle par place monitrice [GDS92], une importante réflexion est faite. L'ajout d'une place monitrice liée à une contrainte (w, b) limite les marquages $M \in \mathcal{N}^{|P|}$ en les restreignant dans un sous-espace dont la limite est définie par un hyperplan $\{M \in N^{|P|} \,|\, w^\top.M \leq b\}$. Lors de l'ajout de plusieurs places monitrices, l'espace des états est donc réduit de manière convexe. Cependant, à partir d'une spécification de type GMEC dans un Réseau de Petri avec des transitions incontrôlables, l'ensemble des marquages admissibles n'est pas nécessairement convexe.

Un exemple permet de justifier ce résultat. Le RdP de la figure 3.11 comporte une transition incontrôlable : t_3. On souhaite restreindre le marquage dans p_3 de façon à ce que cette place ne soit jamais marquée, ce qui correspond au GMEC suivant $M(p_3) \leq 0$. Dès lors qu'il y a plus d'un jeton dans p_1 et dans p_2 la transition t_3 peut être tirée, violant le GMEC $M(p_3) \leq 0$. L'ensemble optimal des états admissibles peut donc être exprimé grâce à une conjonction $\mathcal{M}_a = \{M \in \mathbb{N}^{|P|} | (M(p_1) \leq 1) \wedge (M(p_2) \leq 1))\}$ (représenté en rouge sur le graphique de la figure 3.11). Cet ensemble étant non-convexe, il est impossible de réduire l'accessibilité du graphe de manière optimale grâce à des places monitrices. Cependant il est possible de trouver une place monitrice v qui évite que la contrainte ne soit violée, mais le contrôle est alors plus restrictif que nécessaire. Dans le cas de l'exemple de la figure 3.11, une place monitrice v avec $M_0(v) = 0$, $v^\bullet = t_1$ et ${}^\bullet v = t_3$, limite à zéro le nombre de jetons dans p_1 et limite les marquages accessibles à une partie des marquages admissibles. Les marquages admissibles avec $M(p_1) > 1)$ sont donc inaccessibles dans le système contrôlé, le contrôle n'est donc pas optimal. À partir de cette observation, différents auteurs ont cherché à transformer les contraintes GMECs inadmissibles, en contraintes admissibles. Le passage de contraintes au calcul du contrôle (génération de place monitrice dans le cas des GMEC) étant généralement direct, la transformation des contraintes est devenue un objet important de recherche.

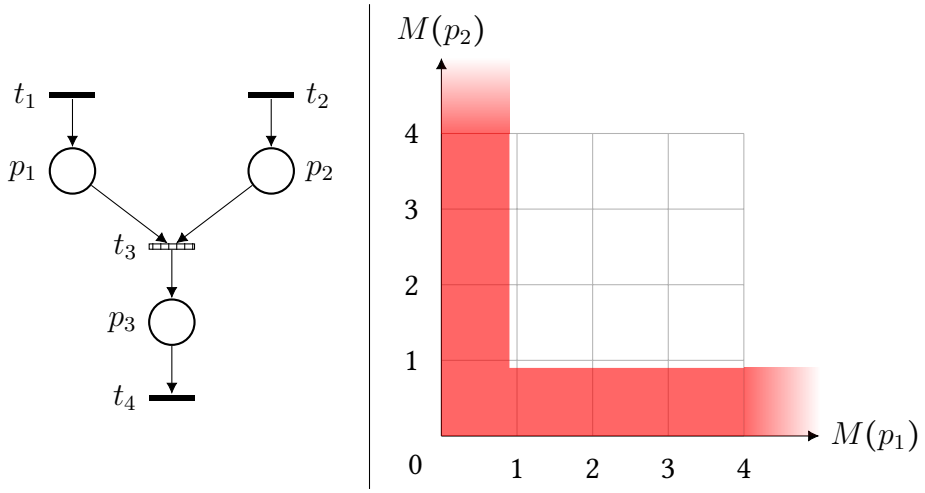


FIGURE 3.11 – Réseau de Petri $\mathcal{N}$, et espace admissible associé à un GMEC

Dans [BCG06], les auteurs s'intéressent à la transformation de contraintes GMECs non-admissibles en contraintes admissibles. Ils ont démontré qu'il n'existe pas qu'un seul contrôle maximal possible. Etant donné un RdP marqué $(\mathcal{N}, M_0)$ et un ensemble de n contraintes GMEC $(W, B) \notin \mathcal{C}^C$, il est alors possible de trouver un ensemble de contraintes respectant la condition de contrôlabilité $(W', B') \in \mathcal{C}^C$, garantissant de rester dans les marquages autorisés $\mathcal{M}(W', B') \subseteq \mathcal{M}(W, B)$. Les auteurs définissent la classe $\mathcal{C}^C_{(W,B)}(\mathcal{N})$ des GMEC qui respectent ces conditions de la manière suivante :

$$\mathcal{C}^C_{(W,B)}(\mathcal{N}) = \{(W', B') \in \mathbb{Z}^{n\times|P|} \times \mathbb{Z}^n \mid \mathcal{M}(W', B') \subseteq \mathcal{M}(W, B) \wedge (W', B') \in \mathcal{C}^C\}$$

Un GMEC ayant un contrôle suprême est plus permissif que tout autre GMEC dans $\mathcal{C}^C_{(W,B)}(\mathcal{N})$. On dit qu'un GMEC (W', B') effectue un contrôle suprême si :

$$\forall (W'', B'') \in \mathcal{C}^C_{(W,B)}(\mathcal{N}),\ \mathcal{M}(W', B') \supseteq \mathcal{M}(W'', B'')$$

Les auteurs démontrent que dans le cas général, un tel GMEC n'existe pas. Un exemple permet de justifier ce résultat si on considère le RdP de la figure 3.11 et le GMEC non-admissible $M(p_3) \leq 0$. Deux GMEC admissibles maximaux émergent $(w_1, b_1) : M(p_1) + M(p_3) \leq 0$ et $(w_2, b_2) : M(p_2) + M(p_3) \leq 0$. Ces deux GMEC sont maximaux et cependant ne sont pas égaux.

Moody and Antsaklis [MA00]

Dans [MA00] les auteurs proposent de transformer une contrainte inadmissible (w, b) en une contrainte admissible (w', b') qui empêche le système d'atteindre les états interdits. Dans le cadre de cette méthode, il est supposé que toutes les transitions inobservables sont aussi incontrôlables ($T_{uo} \subseteq T_{uc}$). Dans [MA00], la méthode est présentée pour traiter directement un ensemble de contraintes. Par souci de simplicité, nous présenterons la méthode pour une seule contrainte. Cette méthode

propose de transformer les contraintes grâce à deux matrices :

$$R_1 \in \mathbb{Z}^{|P|} \quad \text{satisfaisant} \quad R_1^{\intercal}.M \geq 0,\ \forall M \in \mathcal{R}(\mathcal{N}, M_0) \tag{3.11}$$

$$R_2 \in \mathbb{N} \tag{3.12}$$

La nouvelle contrainte (w', b') est calculée de la manière suivante :

$$w' = R_1 + R_2.w \tag{3.13}$$

$$b' = R_2.(b+1) - 1 \tag{3.14}$$

La construction de (w', b), et les propriétés de R_1 et R_2 , garantissent l'implication suivante : $\forall M \in \mathcal{R}(\mathcal{N}, M_0)$, $w'.M \leq b' \implies w.M \leq b$. En effet par injection des équations 3.13 et 3.14 dans $w'.M \leq b'$, on trouve l'égalité stricte suivante (tous les éléments étant des entiers) :

$$(R_1 + R_2.w)M < R_2(b+1)$$

R_2 étant un scalaire :

$$\frac{1}{R_2}R_1.M + w.M < b + 1$$

$\frac{1}{R_2}R_1.M$ étant forcément positive de par la nature de R_1 et R_2, la relation $w.M \leq b$ en est induite.

Il faut s'assurer que la nouvelle contrainte (w', b') ne soit pas violée à l'état initial. Il faut donc s'assurer que $w'^{\intercal}.M_0 \leq b'$. Grâce aux équations 3.13 et 3.14 on obtient :

$$0 \leq R_1.M_0 \leq R_2.(b + 1 - w.M_0) - 1 \tag{3.15}$$

Les valeurs de R_1 et R_2 sont calculées grâce à un **problème d'optimisation en nombre entier**. On définit R :

$$R = \begin{bmatrix} R_1 \\ R_2' \\ R_3 \end{bmatrix}$$

avec $R_2' = R_2 - 1$ et $R_3 \in \mathbb{Z}^{|T_{uo}|}$ est utilisée comme variable d'écart : R_3 étant défini positif sa présence permet de transformer une égalité en une inégalité. Pour que la nouvelle contrainte admissible respecte les conditions des contraintes admissibles sur les transitions incontrôlables (3.9) et inobservables (3.10) un ensemble de contraintes sur R_1 et R_2 est défini.

$z(R)$ est la fonction à minimiser et est définie comme suit :

$$\min_R \left(z(R) = R.\begin{bmatrix} M_0 \\ w^T.M_0 - b - 1 \\ 0 \end{bmatrix} \right)$$

restreint par les contraintes :

$$\begin{cases} R.\begin{bmatrix} C(,T_{uc}) & C(,T_{uo}) \\ w^{\top}.C(,T_{uc}) & w^{\top}.C(,T_{uo}) \\ I & 0 \end{bmatrix} = w^{\top}.\begin{bmatrix} C(,T_{uc}) & C(,T_{uo}) \end{bmatrix} \\ R \leq 0 \quad \text{entier} \end{cases}$$

La fonction à minimiser $z(R)$ reprend l'équation 3.15. Dans le cas où la valeur de $R_1.M_0 + R'_2(w.M_0 - b - 1) \leq w.M_0 - b$, il n'existe pas de contrainte (w', b') qui puisse satisfaire les conditions d'admissibilité. Pour forcer la solution à respecter cette contrainte, la fonction aurait pu être mise en contrainte simple plutôt qu'en fonction à minimiser. [MA00] ne justifie pas clairement ce choix mais laisse à penser que la fonction à minimiser permet de réduit l'écart entre (w', b') et (w, b).

Les contraintes d'égalité permettent de faire respecter les contraintes d'admissibilité vis-à-vis des transitions incontrôlables 3.9 et inobservables 3.10. La première série d'équations, relative aux transitions incontrôlables :

$$R_1.C(,T_{uc}) + R'_2.(w^{\top}.C(,T_{uc})) + R_3 = w^{\top}.C(,T_{uc})$$

Peut être factorisée grâce à l'équation 3.13 de w' et ramenée à la contrainte d'admissibilité relative aux transitions incontrôlables 3.9 :

$$w'^{\top}.C(,T_{uc}) \leq 0$$

De même pour les transitions inobservables :

$$R_1.C(,T_{uo}) + R'_2.(w^{\top}.C(,T_{uo})) = w^{\top}.C(,T_{uo})$$

est équivalente à la relation d'admissibilité 3.10

$$w'^{\top}.C(,T_{uo}) = 0$$

Remarque 18. *Les auteurs font remarquer qu'il peut y avoir quelques difficultés à utiliser cette méthode. Notamment la solution de problème d'optimisation peut avoir une solution non finie. Dans ce cas-là, c'est à l'utilisateur de fixer une nouvelle contrainte pour obtenir une solution finie. De même le problème peut converger vers* $R_1 + R_2.w = 0$ *lorsque il n'existe pas d'autres solutions possibles.*

Les auteurs proposent une solution alternative basée sur une manipulation matricielle de façon à annuler les colonnes relatives aux transitions incontrôlables et inobservables.

3.3 Évitement des blocages dans les Réseaux de Petri

La vivacité d'un Réseau de Petri est une propriété qui peut être souhaitable. L'absence de blocages dans un RdP est une condition nécessaire pour en garantir la vivacité. La présence d'un blocage dans des applications informatiques, ferroviaires,

manufacturières ... peut avoir des répercussions désastreuses sur l'ensemble de l'application. La mise en place de méthodes d'évitement des blocages se fait en général via deux techniques majeures : par analyse du graphe d'accessibilité ou par analyse structurelle. Trois critères importants sont alors à prendre en compte lors de la mise en place d'une méthode d'évitement des blocages dans un RdP :

La Permissivité : Elle permet de juger la restrictivité de la méthode de contrôle. On préférera avoir une méthode la plus permissive possible qui laisse le plus de liberté d'actions au RdP tout en évitant les situations de blocages. La permissivité est généralement mesurée grâce au nombre d'états du graphe d'accessibilité du RdP contrôlé. Plus ce nombre d'états est proche de celui du graphe d'accessibilité du RdP non contrôlé, plus la méthode est permissive.

La complexité structurelle : Elle mesure à quel point le contrôle obtenu peut être difficile à comprendre pour un être humain. Ce critère est intéressant pour les systèmes critiques dont le fonctionnement du contrôle doit être compris via une phase de rétro-ingénierie.

Complexité computationnelle : C'est une notion classique d'algorithmique qui mesure l'efficacité temporelle et/ou spatiale d'un algorithme vis-à-vis du nombre d'entrées.

Les méthodes basées sur l'analyse du graphe d'accessibilité permettent d'obtenir de très bons résultats en termes de permissivité mais souffrent par contre d'une très mauvaise complexité. Cela les rend complètement inutilisables sur des systèmes de grandes tailles. Pour cette raison, cette sous-section sera centrée sur les méthodes structurelles.

3.3.1 Conditions de contrôle d'un siphon

La notion de blocage est fortement liée à la présence de siphon(s) (également appelé verrou(s)). Dans le cas des RdPs ordinaires, un siphon est dit contrôlé si on l'empêche de se vider. Dans le cas des RdPs généralisés, en raison de la pondération des arcs, s'assurer qu'un siphon ne se vide pas n'est pas suffisant pour certifier que ce siphon ne produira pas de blocage.

Propriété 9. *Soient S_1 et S_2 deux siphons d'un Réseau de Petri. Alors, $S_1 \cup S_2$ est un siphon.*

Propriété 10. *Étant donné $M \in \mathcal{R}(\mathcal{N}, M_0)$ un marquage d'un Réseau de Petri $(\mathcal{N}, M_0)$ et S un siphon. Si $M(S) = 0$, alors $\forall M' \in \mathcal{R}(\mathcal{N}, M)$, $M'(S) = 0$.*

Réseaux de Petri ordinaires

Théorème 2. *Soit un Réseau de Petri Ordinaire marqué $(\mathcal{N}, M_0)$ avec son ensemble de siphons associés $\Pi(\mathcal{N})$. Alors le RdP est sans blocage si $\forall S \in \Pi(\mathcal{N}), \forall M \in \mathcal{R}(\mathcal{N}, M_0)$, $M(S) > 0$.*

Théorème 3. *Soit un Réseau de Petri Ordinaire $(\mathcal{N}, M_0)$ dans un état de blocage complet ($\nexists t \in T | M[t\rangle$), alors $\{p \in P | M(p) = 0\}$ est un siphon.*

Corollaire 1. *Un Réseau de Petri bloquant contient au moins un siphon.*

Définition 32. *Un siphon S est dit contrôlé pour un Réseau de Petri ordinaire marqué $(\mathcal{N}, M_0)$ si $\forall M \in \mathcal{R}(\mathcal{N}, M_0)$, $M(S) > 0$*

Dans [LZ09] et [BPP96], sont présentées les conditions nécessaires pour qu'un invariant empêche un siphon de se vider. Si un tel invariant existe, on dit alors que le siphon est contrôlé par invariant.

Propriété 11. *Considérant $X \in \mathcal{P}_{\circlearrowleft}(\mathcal{N})$ et $S \in \Pi(\mathcal{N})$, pour que S soit contrôlé par X, alors il est nécessaire que les relations suivantes soient vérifiées : ,*

$$||X||^+ \cap S \neq \varnothing$$
$$||X||^- \cap S = \varnothing$$

Avec $||X||^+ = \{x \in X \,|X(x) > 0\}$ et $||X||^- = \{x \in X \,|X(x) < 0\}$

Les notations suivantes sont adoptées pour définir des ensembles de places :

$$\begin{aligned} ||X||^+ \cap S &= \Xi \\ ||X||^+ / S &= \Upsilon \\ ||X||^- &= \Phi \end{aligned}$$

Propriété 12. *S est contrôlé par l'invariant X si :*

$$X^\top.M_0 - \max_{M \in \mathcal{R}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Upsilon} X(p)M(p) \right\} + \min_{M \in \mathcal{R}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Phi} -X(p).M_(p) \right\} > 0$$

Corollaire 2. *Si $\Upsilon = \varnothing$, alors S est contrôlé si $X^\top.M_0 > 0$*

Remarque 19. *La solution de $\max_{M \in \mathcal{R}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Upsilon} X(p)M(p) \right\}$ et $\min_{M \in \mathcal{R}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Phi} -X(p).M_(p) \right\}$ peut être complexe à trouver car elle nécessite l'exploration de $\mathcal{R}(\mathcal{N}, M_0)$. Cependant il est possible d'utiliser l'approximation $\max_{M \in \vec{\mathcal{R}}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Upsilon} X(p)M(p) \right\}$ et $\min_{M \in \vec{\mathcal{R}}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Phi} -X(p).M_(p) \right\}$. En effet comme $\mathcal{R}(\mathcal{N}, M_0) \subseteq \vec{\mathcal{R}}(\mathcal{N}, M_0)$ on a :*

$$\max_{M \in \mathcal{R}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Upsilon} X(p)M(p) \right\} \leq \max_{M \in \vec{\mathcal{R}}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Upsilon} X(p)M(p) \right\}$$
$$\min_{M \in \mathcal{R}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Phi} -X(p).M_(p) \right\} \geq \min_{M \in \vec{\mathcal{R}}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Phi} -X(p).M_(p) \right\}$$

Ainsi si la relation suivante est vraie, alors elle implique que la propriété 12 est vraie :

$$X^\top.M_0 - \max_{M \in \vec{\mathcal{R}}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Upsilon} X(p)M(p) \right\} + \min_{M \in \vec{\mathcal{R}}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Phi} -X(p).M(p) \right\} > 0$$

La valeur de $\max_{M \in \vec{\mathcal{R}}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Upsilon} X(p)M(p) \right\}$ et $\min_{M \in \vec{\mathcal{R}}(\mathcal{N},M_0)} \left\{ \sum_{p \in \Phi} -X(p).M_(p) \right\}$ peuvent être trouvées en résolvant les problèmes d'optimisation linéaires suivants :

$$\max\left\{\sum_{p\in\Upsilon} X(p)M(p)\right\} \qquad \min\left\{\sum_{p\in\Phi} -X(p).M_(p)\right\}$$
$$M = M_0 + C.Y \qquad M = M_0 + C.Y$$
$$M \geq 0 \qquad M \geq 0$$
$$Y \geq 0 \qquad Y \geq 0$$

La génération d'un invariant peut être faite grâce à la génération de places monitrices (définition 29). Dans le cas d'un RdP ordinaire il est possible d'éviter qu'un siphon se vide par l'ajout d'une place monitrice résultant d'une GMEC correctement choisie.

Définition 33. *Dans un Réseau de Petri ordinaire $\mathcal{N}$ il est possible de générer une place de contrôle pour éviter qu'un siphon S se vide. Pour cela une contrainte (w, b) est définie telle que $\forall p \in S,\ w(p) = -1$ sinon $w(p) = 0$ et $b = 1$. Une place de contrôle $V(S)$ qui respecte la définition 29, permet ainsi d'éviter que le siphon ne se vide.*

Propriété 13. *Dans un Réseau de Petri contrôlé $\mathcal{N}^C$, une place monitrice $V(S)$ qui empêche qu'un siphon S ne se vide, génère un invariant avec un P-flot avec $X \in \mathcal{P}_{\circlearrowleft}(\mathcal{N}^C)$ tel que :*

$$\forall p \in S,\ X(p) = 1 \quad X(V(S)) = -1 \text{ et } \forall p \notin S \cup \{V(S)\},\ X(p) = 0$$

Remarque 20. *La génération d'une place de contrôle dans un RdP ordinaire peut engendrer une place de contrôle qui comporte des arcs pondérés. Le RdP contrôlé n'est ainsi plus ordinaire. Cette remarque entraîne des complications et empêche d'utiliser des algorithmes se basant sur des méthodes d'analyse des RdPs ordinaires qui généreraient des places de contrôles itérativement.*

La figure 3.12 montre un RdP $\mathcal{N}^C$ contrôlé composé de places $P = \bigcup_{i=1}^{6}\{p_i\}$ et d'une place de contrôle $V(S)$, avec $\mathcal{N}^C = (P \cup \{V(S)\}, T, F^C, W^C)$. Le RdP avant contrôle $\mathcal{N} = (P, T, F, W)$ avec $\mathcal{N} \subseteq \mathcal{N}^C$ est un RdP bloquant. En effet, il existe $\sigma = t_1t_4$ avec $M_0[\sigma\rangle M$ tel que pour M, toutes les transitions sont mortes. Le théorème 3 implique que l'ensemble de places $S = \{p_2, p_4, p_5, p_6\}$ est un siphon avec $M(S) = 0$. Pour éviter que ce siphon ne se vide, une place de contrôle $V(S)$ est générée suivant la définition 33.

La figure 3.13 montre un RdP $\mathcal{N} = (P, T, F, W)$ dont l'ensemble de places $S = \{p_2, p_4, p_5, p_6\}$ forme un siphon strict car ${}^\bullet S \subseteq S^\bullet$. Cependant il n'existe pas de marquage $M \in \mathcal{R}(\mathcal{N}, M_0)$ tel que $M(S) = 0$. En effet il existe un P-semiflot, $X \in \mathcal{P}(\mathcal{N})^+_{\circlearrowleft}$ tel que $\forall p \in \{p_1, p_2, p_3, p_4\},\ X(p) = 2$, $\forall p \in \{p_5, p_6, p_7\},\ X(p) = 1$. Si on se réfère à la propriété 12, dans le cas de notre exemple $\Xi = S$, $\Upsilon = \{p_2, p_4, p_7\}$ et $\Phi = \varnothing$. Nous avons donc $X^\intercal.M_0 = 3$ et $\max_{M\in\mathcal{R}(\mathcal{N},M_0)}\left\{\sum_{p\in\Upsilon} X(p)M(p)\right\} = 2$, la propriété 12 est donc vérifiée, et S est contrôlé. Dans le cas ou $M_0(p_7) = 2$, on remarque que ce marquage ne suffit plus à contrôler S car $X^\intercal.M_0 = 4$ et $\max_{M\in\mathcal{R}(\mathcal{N},M_0)}\left\{\sum_{p\in\Upsilon} X(p)M(p)\right\} = 4$.

Réseaux de Petri Généralisés

Dans le cas des RdPs généralisés une transition peut être désactivée même si un siphon n'est pas complètement vide. Dans [BPP96, BCK05] est introduit la notion de max-marquage et min-marquage d'un siphon.

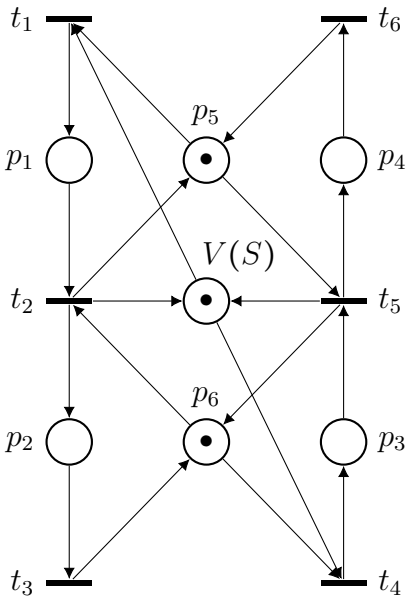


FIGURE 3.12 – Contrôle d'un siphon S par génération d'une place de contrôle

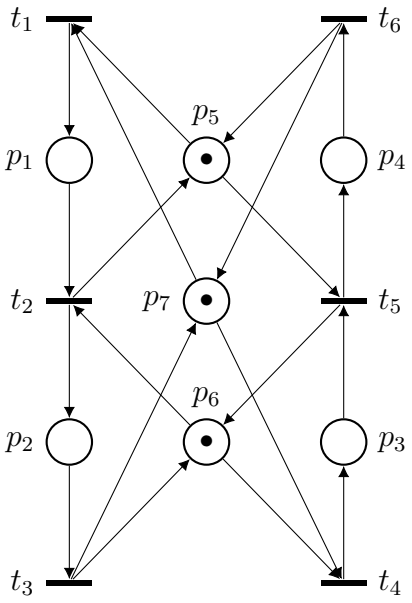


FIGURE 3.13 – Siphon contrôlé par invariant

Définition 34. *Soit un sous-Réseau $\mathcal{N}' = (P', T', F', W')$ d'un RdP $\mathcal{N} = (P, T, F, W)$ avec $\mathcal{N}' \subseteq \mathcal{N}$. En fonction de la pondération des arcs, nous dirons que $\mathcal{N}'$ est :*

- *homogène : si et seulement si $\forall p \in P', \forall t_1, t_2 \in p^\bullet, W'(p, t_1) = W'(p, t_2)$*
- *non-bloquant : si et seulement si : $\forall p \in P', p^\bullet \neq \varnothing \implies min_{\bullet_p} \geq min_{p^\bullet}$*
- *fortement non-bloquant : si et seulement si : $\forall p \in P', p^\bullet \neq \varnothing \Rightarrow min_{\bullet_p} \geq max_{p^\bullet}$*

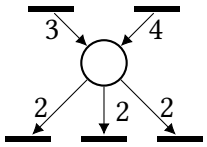


FIGURE 3.14 – Pondération homogène

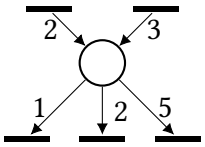


FIGURE 3.15 – Pondération non-bloquante

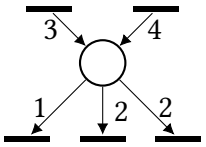


FIGURE 3.16 – Pondération fortement non-bloquante

Définition 35. *Considérons un Réseau de Petri marqué $(\mathcal{N}, M_0)$, avec S un siphon de $\mathcal{N}$. S est dit max-marqué à un marquage $M \in \mathcal{R}(\mathcal{N}, M_0)$, si $\exists p \in S$ tel que $M(p) \geq max_{p^\bullet}$ avec $max_{p^\bullet} = \max\{W(p,t) | \forall t \in p^\bullet\}$.*

Définition 36. *Un siphon est dit max-contrôlé s'il est max-marqué pour chacun de ses marquages accessibles. Lorsque tous les siphons d'un Réseau de Petri $(\mathcal{N}, M)$ sont max-marqués ont dit alors que le RdP satisfait la propriété de Max-cs (max controled siphon)*

Propriété 14. *Si un Réseau de Petri $(\mathcal{N}, M_0)$ satisfait la propriété Max-cs il est alors quasi-vivant.*

Définition 37. *Considérons un Réseau de Petri marqué $(\mathcal{N}, M_0)$, avec S un siphon de $\mathcal{N}$. S est dit min-marqué à un marquage $M \in \mathcal{R}(\mathcal{N}, M_0)$, si $\exists p \in S$ tel que $M(p) \geq min_{p^\bullet}$, avec $min_{p^\bullet} = \min\{W(p,t) | \forall t \in p^\bullet\}$.*

Définition 38. *Un siphon est dit min-contrôlé, s'il est min-marqué pour chacun de ses marquages accessibles. Lorsque tous les siphons d'un Réseau de Petri $(\mathcal{N}, M)$ sont min-marqués on dit alors que le RdP satisfait la propriété de Min-cs (Min controled siphon).*

Propriété 15. *Si un Réseau de Petri $(\mathcal{N}, M_0)$ est vivant, il satisfait forcément la propriété Min-cs.*

Remarque 21. *Un siphon max-contrôlé est aussi min-contrôlé.*

Des conditions structurelles permettent de garantir qu'un RdP généralisé respecte les propriétés Min-cs ou Max-cs.

Propriété 16. *Soit un Réseau de Petri marqué $(\mathcal{N}, M_0)$ et S un siphon de $\mathcal{N}$. Si une des deux propriétés suivantes est vérifiée alors S est min-marqué :*

1. *Il existe un piège Q inclus dans S tel que : Q est min-marqué à M_0 et que le sous-Réseau induit par $(Q, Q^\bullet)$ est de pondération non-bloquante.*
2. *Il existe un invariant de places porté par un P-flot $X \in P_{\circlearrowleft}(\mathcal{N})$ tel que $\forall p \in (\|X\|^- \cap S), max_{p^\bullet} = 1, \|X\|^+ \subseteq S$ et $X^T.M_0 > \sum_{p \in P}[X(p).(min_{p^\bullet} - 1)]$. Avec $\|X\|^+ = \{p \in P | X(p) > 0\}$ et $\|X\|^- = \{p \in P | X(p) < 0\}$.*

Propriété 17. *Soit un Réseau de Petri marqué $(\mathcal{N}, M_0)$ et S un siphon de $\mathcal{N}$. Si une des deux propriétés suivantes est vérifiée alors S est max-marqué :*

1. *Il existe un piège Q inclus dans S tel que : Q est max-marqué à M_0 et que le sous-Réseau induit par $(Q, Q^\bullet)$ est de pondération fortement non-bloquante.*
2. *Il existe un invariant de places porté par un P-flot $X \in P_{\circlearrowleft}(\mathcal{N})$ tel que $\forall p \in (\|X\|^- \cap S), min_{p^\bullet} = 1, \|X\|^+ \subseteq S$ et $X^T.M_0 > \sum_{p \in P}[X(p).(max_{p^\bullet} - 1)]$. Avec $\|X\|^+ = \{p \in P | X(p) > 0\}$ et $\|X\|^- = \{p \in P | X(p) < 0\}$.*

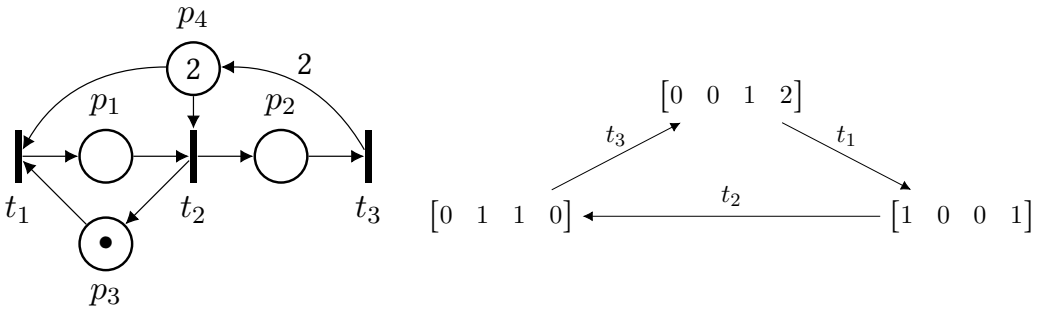


Figure 3.17 – Un Réseau de Petri $(\mathcal{N}, M_0)$ et son graphe des marquages accessibles

Exemple Dans la figure 3.17 sont présentés un Réseau de Petri marqué $(\mathcal{N}, M_0)$ et son graphe des marquages accessibles. $\mathcal{N}$ comporte un siphon strict $S = \{p_2, p_4\}$ et deux invariants élémentaires : $\forall M \in \mathcal{R}(\mathcal{N}, M_0)$, $M^T.X_1 = 1$ et $M^T.X_2 = 2$, avec

$X_1 = \begin{bmatrix}1 & 0 & 1 & 0\end{bmatrix}^T$, et $X_2 = \begin{bmatrix}1 & 2 & 0 & 1\end{bmatrix}^T$. Par combinaison de $M^T.X_1 = 1$ et $M^T.X_2 = 2$ on trouve l'invariant $X'^T.M = 1$ avec $X' = X_2 - X_1 = [0; 2; -1; 1]^T$. Pour cet invariant $\|X'\|^- = \{p_3\}$, on a donc $\|X'\|^- \cap S = \varnothing$. $\|X'\|^+ = \{p_2, p_4\}$ est compris dans le siphon $S \supseteq \|X\|^+$. On a $max_{p_2^\bullet} = 1$, $max_{p_3^\bullet} = 1$, $max_{p_4^\bullet} = 1$ et donc $\sum_{p\in P}[X(p).(max_{p^\bullet} - 1)] = 0$. Sachant que $X'^T.M_0 = 1$, la relation $X'^T.M_0 > \sum_{p\in P}[X(p).(max_{p^\bullet} - 1)]$ du cas 2 de la propriété 17 est donc vérifiée. Le siphon S est donc bien contrôlé par l'invariant $X'.M^T = 1$. On peut vérifier sur le graphe de marquages accessibles que $\forall M \in \mathcal{R}(\mathcal{N}, M_0)$, on a $M(P_2) + M(P_4) > 0$ et donc le siphon ne se vide jamais. Dans le cas où on modifie la valeur de $M_0(p_3) = 2$, $X'^T.M_0 = 2$ et la relation $X'^T.M_0 > \sum_{p\in P}[X(p).(max_{p^\bullet} - 1)]$ n'est plus vérifiée. S devient vide pour un état $M = 2p_1$ tel que $M_0[t_1t_1\rangle M$.

3.3.2 Siphons élémentaires et dépendants

Les siphons élémentaires ont été introduits dans [LZ04] pour une application aux Systèmes Flexibles de Production Manufacturière (FMS : *Flexible Manufacturing Systems*), et sont d'une manière générale applicables aux Réseaux de Petri ordinaires. Le vocabulaire développé dans [LZ04] est par la suite revu en [LZ06] et c'est finalement sur ces notions que le reste de la littérature s'appuie. L'idée des siphons élémentaires part de l'observation que beaucoup de méthodes de contrôle basées sur l'analyse des siphons génèrent plus de places de contrôles que nécessaires. Pour réduire le nombre de places de contrôle, les siphons sont séparés en deux ensembles distincts : les siphons *élémentaires* et les siphons dépendants. Les siphons élémentaires constituent une base génératrice de l'ensemble des siphons d'un RdP. Les siphons dépendants sont donc obtenus par composition de siphons élémentaires. Leur contrôle est donc indirectement assuré par le contrôle des siphons élémentaires.

On considère un Réseau de Petri ordinaire $\mathcal{N} = (P, T, C)$ caractérisé par sa matrice d'incidence C.

Définition 39. *Le T-vecteur caractéristique d'un ensemble de places S est noté η_S avec :*

$$\eta_S^\intercal = \|S\|^\intercal.C$$

Le T-vecteur caractéristique d'un ensemble de place S, sert à mesurer l'impact du tir d'une transition sur le marquage de cet ensemble de places. Ainsi pour un vecteur de tir $\vec{t} \in \mathbb{N}^{|T|}$ relatif au tir d'une transition t amenant d'un état M à M' i.e $M[t\rangle M'$, le marquage de S aura varié de $\vec{t}^\intercal.\eta_S$ i.e $M'(S) = \vec{t}^\intercal.\eta_S + M(S)$

Définition 40. *On appelle siphon élémentaire, un siphon $S_0 \in \Pi(\mathcal{N})$ si :*

$$\nexists S_1, S_2, ..., S_n \in \Pi(\mathcal{N}), \text{ tel que } \quad \eta_{S_0} = \sum_{i=1}^{n} a_i.\eta_{S_i}$$

Où a_1, a_2, ... , $a_n \in \mathbb{N}$ et $\forall i \in \{1, 2, ..., n\}, S_i \neq S_0$

L'ensemble des siphons élémentaires de $\mathcal{N}$ est noté $\Pi_E(\mathcal{N})$.

Définition 41. *On appelle siphon dépendant, un siphon $S_0 \in \Pi(\mathcal{N})$ si :*

$$\exists S_1, S_2, ..., S_n \in \Pi(\mathcal{N}), \text{ tel que } \quad \eta_{S_0} = \sum_{i=1}^{n} a_i.\eta_{S_i}$$

Où $a_1, a_2, \ldots, a_n \in \mathbb{N}$ *et* $\forall i \in \{1, 2, \ldots, n\}, S_i \neq S_0$ *L'ensemble des siphons dépendants de* $\mathcal{N}$ *est noté* $\Pi_D(\mathcal{N})$.

Définition 42. *Soit* $S_0 \in \Pi_D(\mathcal{N})$ *un siphon et* $S_1, S_2, \ldots, S_n \in \Pi_E$ *les siphons élémentaires d'un RdP.* S_0 *est appelé un* ***siphon fortement dépendant*** *vis-à-vis de* $S_1, S_2, \ldots$ *et* S_n *s'il est possible d'exprimer* η_{S_0} *par une somme de T-vecteurs caractéristiques de siphons élémentaires pondérés par des coefficients* ***positifs*** *:*

$$\eta_{S_0} = \sum_{i=1}^{n} a_i.\eta_{S_i}$$

Où $a_1, a_2, \ldots, a_n \in \mathbb{N}^+$.

L'ensemble des siphons fortement dépendants est noté Π_{D^+}.

Définition 43. *Soit* $S_0 \in \Pi_D(\mathcal{N})$ *un siphon et* $S_1, S_2, \ldots, S_n, S_{n+1}, \ldots, S_{m+n} \in \Pi_E$ *les siphons élémentaires d'un RdP.* S_0 *est appelé un* ***siphon faiblement dépendant*** *si :*

$$\eta_{S_0} = \sum_{i=1}^{n} a_i.\eta_{S_i} - \sum_{i=n+1}^{m} a_i.\eta_{S_i}$$

Où $a_1, a_2, \ldots, a_{n+m} \in \mathbb{N}^+$.

L'ensemble des siphons faiblement dépendants est noté Π_{D^-}*, et l'ensemble des siphons fortement dépendants et faiblement dépendants forment l'ensemble des siphons dépendants* $\Pi_D = \Pi_{D^+} \cup \Pi_{D^-}$

Définition 44. *On définit la matrice* $[\Pi] \in \mathbb{N}^{k \times |P|}$*, avec* $k = |\Pi(\mathcal{N})|$ *constituée de l'ensemble des vecteurs porteurs des siphons de* $\mathcal{N}$.

$$[\Pi] = [||S_1|| \; ||S_2|| \; \ldots ||S_k||]$$

De la même manière on définit la matrice $[\eta_\Pi] \in \mathbb{N}^{k \times |T|}$ *constituée des T-vecteurs caractéristiques de chaque siphon de* $\mathcal{N}$.

$$[\eta_\Pi] = [\eta_{S_1} \, \eta_{S_2} \ldots \eta_{S_k}]$$

Propriété 18. *Le rang de la matrice* $[\eta_\Pi]$ *est égal au nombre de siphons élémentaires de* $\mathcal{N}$.

$$rang([\eta_\Pi]) = |\Pi_E(\mathcal{N})|$$

Théorème 4. *Il existe une limite structurelle du nombre de siphons élémentaires dans un Réseau de Petri ordinaire* $\mathcal{N}$ *:*

$$|\Pi_E(\mathcal{N})| \leq rang(C) \leq \min(|P|, |T|)$$

Remarque 22. *Les siphons relatifs aux colonnes linéairement indépendantes de la matrice* $[\eta_\Pi]$ *constituent donc l'ensemble des siphons élémentaires* $\Pi_E(\mathcal{N})$*. Les bases d'une matrice pouvant être multiples, pour un même RdP, il est possible de définir plusieurs couples* $(\Pi_E(\mathcal{N}), \Pi_D(\mathcal{N}))$ *d'ensembles de siphons élémentaires et dépendants corrects.*

La figure de 3.18 représente un Réseau de Petri ordinaire $\mathcal{N}$. Le théorème 4 indique que le nombre maximum de siphons élémentaires dans un RdP est borné par

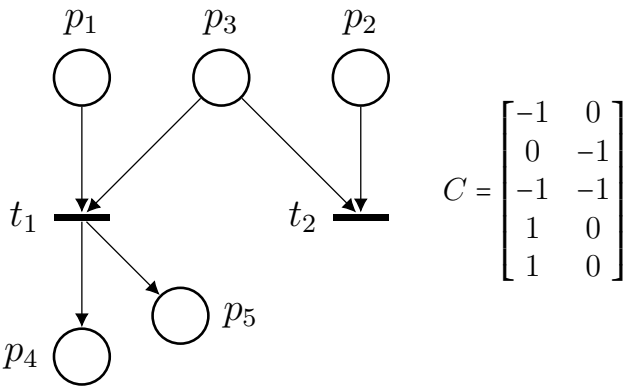


FIGURE 3.18 – Un Réseau de Petri ($\mathcal{N}$) et sa matrice d'incidence C

$\min(|P|, |T|)$. Nous choisissons donc ici de prendre comme siphons élémentaires $\Pi_E(\mathcal{N}) = \{S_1, S_2\}$ avec $S_1 = \{p_1\}$ et $S_2 = \{p_2\}$. $\eta_{S_1} = \begin{bmatrix} -1 & 0 \end{bmatrix}^\top$ et $\eta_{S_2} = \begin{bmatrix} 0 & -1 \end{bmatrix}^\top$ comme η_{S_1} et η_{S_2} sont linéairement indépendants et $\min(|P|, |T|) = 2$, $\{S_1, S_2\}$ est un ensemble correct de siphons élémentaires. $S_3 = \{p_3\}$ est un siphon avec $\eta_{S_3} = \begin{bmatrix} -1 & -1 \end{bmatrix}^\top$. S_3 est un siphon fortement dépendant car $\eta_{S_3} = \eta_{S_1} + \eta_{S_2}$, i.e $\eta_{S_3} \in \Pi_{D^+}(\mathcal{N})$. $S_4 = \{p_3, p_4, p_5\}$ avec $\eta_{S_4} = \begin{bmatrix} 2 & -1 \end{bmatrix}^\top$ est un siphon faiblement dépendant. En effet, son T-vecteur caractéristique s'exprime de la manière suivante $\eta_{S_4} = \eta_{S_2} - 2\eta_{S_1}$. S_4 est donc un siphon faiblement dépendant i.e $S_4 \in \Pi_{D^-}(\mathcal{N})$.

Définition 45. *Deux siphons S et S' sont équivalents si leurs T-vecteurs caractéristiques sont égaux. On définit la classe $S_{\cong}$ des siphons équivalents de $S \in \Pi(\mathcal{N})$:*

$$S_{\cong} = \{S' \in \Pi(\mathcal{N}) \mid \eta_S = \eta_{S'}\}$$

On utilisera la notation $S \cong S'$ si $\eta_S = \eta_{S'}$.

Définition 46. *Un siphon $S' \in S_{\cong}$ avec $S \in \Pi(\mathcal{N})$ est dit jeton-pauvre dans $S_{\cong}$ si :*

$$\forall S'' \in S_{\cong},\ S' \leq S''$$

A l'inverse un siphon $S' \in S_{\cong}$ avec $S \in \Pi(\mathcal{N})$ est dit jeton-riche si :

$$\exists S'' \in S_{\cong} \mid S'' < S'$$

Propriété 19. *Un siphon $S' \in S_{\cong}$ avec $S \in \Pi(\mathcal{N})$ est jeton-riche, alors S' est contrôlé.*

Propriété 20. *Si les siphons $S', S'' \in S_{\cong}$ avec $S \in \Pi(\mathcal{N})$ sont jetons-pauvres, alors S est contrôlé.*

On considère les valeurs suivantes qui correspondent au nombre maximal et minimal de jetons qu'un siphon S peut contenir :

$$M^{\max}(S) = \max_{M \in \mathcal{R}(\mathcal{N}, M_0)} M(S)$$
$$M^{\min}(S) = \min_{M \in \mathcal{R}(\mathcal{N}, M_0)} M(S)$$

Théorème 5. *Soit S un siphon fortement dépendant dans un Réseau $(\mathcal{N}, m_0)$ avec $\eta_S = \sum_{i=1}^{n} a_i.\eta_{S_i}$ et $\forall i \in \{1...n\}$, $S_i \in \Pi_E(\mathcal{N})$. S est contrôlé si :*

$$M_0(S) > \sum_{i=1}^{n} a_i(M_0(S_i) - M\cdot\underline{\min}(S_i))$$

Théorème 6. *Soit S un siphon faiblement dépendant dans un Réseau $(\mathcal{N}, m_0)$ avec $\eta_S = \sum_{i=1}^{n} a_i.\eta_{S_i} - \sum_{i=n+1}^{m} a_i.\eta_{S_i}$ et $\forall i \in \{1...m\}$, $S_i \in \Pi_E(\mathcal{N})$. S est contrôlé si :*

$$M_0(S) > \sum_{i=1}^{n} a_i(M_0(S_i) - M\cdot\underline{\min}(S_i)) - \sum_{i=n+1}^{m} a_i(M_0(S_i) - M\cdot\underline{\max}(S_i))$$

Corollaire 3. *Soit un siphon S fortement ou faiblement dépendant alors si $M_0(S) > \sum_{i=1}^{n} a_i(M_0(S_i) - M\cdot\underline{\min}(S_i))$ alors S est contrôlé.*

Le calcul de $M\cdot\underline{\max}$ et $M\cdot\underline{\min}$ nécessite l'exploration de $\mathcal{R}(\mathcal{N}, M_0)$ face à cette problématique, deux approximations basées sur $\vec{\mathcal{R}}(\mathcal{N}, M_0)$ ont été proposées. On définit donc les valeurs suivantes, utiles pour la suite :

$$\vec{M}\cdot\underline{\max} = \max_{M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)} M(S)$$

$$\vec{M}\cdot\underline{\min} = \min_{M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)} M(S)$$

$$D_1 = \min_{M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)} \sum_{i=1}^{n} a_i.S_i(M)$$

$$D_2 = \max_{M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)} \sum_{i=n+1}^{m} a_i.S_i(M)$$

Corollaire 4. *Soit S un siphon fortement dépendant dans un RdP $(\mathcal{N}, m_0)$, avec $\eta_S = \sum_{i=1}^{n} a_i.\eta_{S_i}$ et $\forall i \in \{1...n\}$, $S_i \in \Pi_E(\mathcal{N})$. S est contrôlé si :*

$$M_0(S) > \sum_{i=1}^{n} a_i(M_0(S_i) - \vec{M}\cdot\underline{\min}(S_i))$$

Corollaire 5. *Soit S un siphon faiblement dépendant dans un RdP $(\mathcal{N}, m_0)$ avec $\eta_S = \sum_{i=1}^{n} a_i.\eta_{S_i} - \sum_{i=n+1}^{m} a_i.\eta_{S_i}$ et $\forall i \in \{1...m\}$, $S_i \in \Pi_E(\mathcal{N})$. S est contrôlé si :*

$$M_0(S) > \sum_{i=1}^{n} a_i(M_0(S_i) - \vec{M}\cdot\underline{\min}(S_i)) - \sum_{i=n+1}^{m} a_i(M_0(S_i) - \vec{M}\cdot\underline{\max}(S_i))$$

Corollaire 6. *Soit S un siphon fortement dépendant, avec $\eta_S = \sum_{i=1}^{n} a_i.\eta_{S_i}$ et $\forall i \in \{1...n\}$, $S_i \in \Pi_E(\mathcal{N})$. S est contrôlé si :*

$$M_0(S) > \sum_{i=1}^{n} a_i(M_0(S_i)) - D_1$$

Corollaire 7. *Soit S un siphon faiblement dépendant dans un RdP$(\mathcal{N}, m_0)$ avec $\eta_S = \sum_{i=1}^{n} a_i.\eta_{S_i} - \sum_{i=n+1}^{m} a_i.\eta_{S_i}$ et $\forall i \in \{1...m\}$, $S_i \in \Pi_E(\mathcal{N})$. S est contrôlé si :*

$$M_0(S) > \sum_{i=1}^{n} a_i(M_0(S_i)) - D_1 - \sum_{i=n+1}^{m} a_i(M_0(S_i)) + D_2$$

Les corollaires 4, 5, 6 et 7 proposent des conditions de contrôles pour les siphons fortement dépendants et faiblement dépendants. Pour les siphons fortement dépendants, la relation du corollaire 4 implique forcément celle du 6. De la même manière, pour les siphons faiblement dépendants, le corollaire 5 implique le corollaire 7.

Le nombre de siphons dépendants d'un Réseau de Petri étant exponentiel vis-à-vis de la taille du RdP, les calculs associés à chacun des siphons dépendants doivent être relativement courts. Pour les corollaires 4, 5, il est uniquement nécessaire de calculer au maximum $\vec{M}_{\min}(S_i)$ et $\vec{M}_{\min}(S_i)$ pour chaque siphon élémentaire. Les conditions sont par la suite vérifiées uniquement par sommation. Pour les corollaires 6 et 7, D_1 doit être calculé pour chacun des siphons dépendants, plus D_2 pour les siphons faiblement dépendants.

Le calcul de $\vec{M}_{\max}^{\mathcal{X}}$, $\vec{M}_{\min}$, D_1 et D_2 se fait par résolution d'un problème d'optimisation linéaire. Même si ce calcul est relativement peu coûteux, le nombre exponentiel de siphons dépendants peut rendre la vérification complexe. Il est donc préférable de vérifier en premier lieu la contrôlabilité d'un siphon en utilisant les conditions des corollaires 4 et 5, puis si celles-là ne sont pas vérifiées, les conditions des corollaires 6 et 7.

Dans [LZ04], une application des siphons élémentaires pour le contrôle d'un système manufacturier flexible est présentée. Dans cet article, le contrôle des siphons élémentaires est fait via la génération d'une place monitrice mettant en oeuvre un contrôle par invariant.

3.3.3 Méthodes d'analyse de la vivacité des Réseaux de Petri

Les méthodes d'analyse de la vivacité des Réseaux de Petri sont nombreuses. Trois grandes catégories de méthodes existent :

— *Énumération des siphons* : Les méthodes d'énumérations des siphons souffrent du potentiel grand nombre de siphons qu'un RdP peut contenir. Des méthodes d'énumérations apparaissent tôt dans la littérature [Lau87, TA84]. Même si des méthodes [CFP05, WYSG15, WLWZ12, CL13, NLWZ14] se revèlent très efficaces pour effectuer l'extraction des siphons, le grand nombre des siphons, exponentiel en fonction de la taille du RdP, rend impossible l'énumération dans les systèmes de grandes tailles. Beaucoup de méthodes ont été développées pour des classes de Réseaux de Petri, où le nombre de siphons est réduit par définition des classes. Nous ne présentons pas ces méthodes dans cette section.

— *Méthodes basées sur le graphe d'accessibilité* : Le graphe d'accessibilité est évidement un outil parfait pour caractériser le comportement d'un Réseau de Petri. Cependant le nombre d'états accessibles étant potentiellement infini, ou de très grande taille pour les modèles bornés, son utilisation directe est

impossible pour des systèmes de grandes tailles. Cependant il existe des méthodes de représentations compactes de l'espace d'états, basées sur des diagrammes de décision binaire (BDD : binary decision diagram) [PCP99], sur des bases de marquages [MTLG16], ou encore basées sur des structures particulières de RdP[Kün05, CP00, MH13]. Les méthodes de contrôles qui se basent sur le graphe d'accessibilité ou ses représentations compactes, ont souvent l'avantage de pouvoir exercer un contrôle maximalement permissif. Parmi elles, nous pouvons citer [CL11] qui utilise un BDD pour représenter les marquages légaux accessibles, ou encore [BCP13].

— *Énumération partielle des siphons* : A la croisée des méthodes d'énumération et des méthodes basées sur le graphe d'accessibilité, il peut être intéressant de chercher uniquement les siphons qui sont potentiellement bloquants. Les siphons potentiellement bloquants sont les siphons tels qu'il existe un marquage accessible où le siphon peut compromettre la vivacité du Réseau de Petri. S'intéresser uniquement aux siphons potentiellement bloquants permet de réduire grandement le nombre de siphons énumérés. La méthode de [CX97] basée sur un problème de programmation en nombre mixte (Mixed Integer Programming ou MIP), a permis le développement de beaucoup de méthodes de contrôle avec des temps de calculs relativement courts. Cette méthode est détaillée dans la suite de cette section.

Extraction de siphons potentiellement vidables dans les Réseaux de Petri ordinaire [CX97]

La méthode présentée dans [CX97], recherche une situation de blocage dans un Réseau de Petri **ordinaire** et **borné**. Elle recherche à la fois un siphon S, un marquage M dans lequel ce siphon se vide et un vecteur de séquence de transitions Y qui permet d'amener de M_0 à M. Les MIP sont des problèmes NP-difficiles en théorie. Mais, les algorithmes de contrôle se basant sur cette méthode bénéficient d'un temps de calcul relativement court. De plus, il est montré que le temps de détection d'un siphon potentiellement vidable est relativement insensible à la modification du marquage initial.

Les MIP comportent un ensemble de variables booléennes et de variables réelles. Ici les variables booléennes sont constituées d'un vecteur $v \in \{0,1\}^{|P|}$ qui représente le porteur des places en dehors de S i.e $\mathbb{1}^{|P|} - ||S||$. $z \in \{0,1\}^{|T|}$ est un vecteur qui représente les transitions sans liens avec le siphon i.e $\mathbb{1}^{|T|} - ||S^\bullet||$. v et z sont donc définis de la manière suivante :

$$v(p) = \begin{cases} 1 & \text{if } p \notin S \\ 0 & \text{sinon} \end{cases} \quad \text{et } z(t) = \begin{cases} 1 & \text{if } t \notin S^\bullet \\ 0 & \text{sinon} \end{cases} \tag{3.16}$$

On définit un ensemble de contraintes à implémenter dans le MIP :

$$v(p) \geq z(t), \quad \forall (t,p) \in F \tag{3.17}$$

L'équation 3.17 génère une contrainte par arc $(T \times P) \cap F$. Ces contraintes permettent de faire respecter la relation 3.16 entre $v(p)$ et $z(t)$.

$$z(t) \geq \sum_{p \in {}^\bullet t} v(p) - |{}^\bullet t| + 1 \tag{3.18}$$

La contrainte 3.18 impose à $v(p)$ d'être un siphon strict i.e ${}^\bullet S \subset S^\bullet$. L'équation 3.18 peut être traduite comme : si $z(t) = 0$ i.e $t \in S^\bullet$ alors $\sum_{p \in {}^\bullet t} v(p) \geq |{}^\bullet t| - 1$ ce qui force au moins une place de ${}^\bullet t$ à être dans S ($v(p) = 0$).

$$M = M_0 + CY, \quad M \geq 0,\ Y \geq 0 \tag{3.19}$$

L'équation 3.19 traduit l'équation d'état.

$$v(p) \geq M(p)/SB(p), \quad \forall p \in P \tag{3.20}$$

L'équation 3.20 force le siphon S à être vide pour l'état M. $SB(p)$ est la limite structurelle de la places p. Cette valeur peut être trouvée en résolvant le problème d'optimisation linéaire $SB(p) = \max\{M(p) | M = M_0 + CY, M \geq 0, Y \geq 0\}$

G^{MIP} est la fonction de minimisation :

$$G^{MIP} = \min \sum_{p \in P} v(p) \tag{3.21}$$

Comme G^{MIP} cherche à minimiser $\sum_{p \in P} v(p)$, cette méthode cherche le plus grand siphon possible. Elle renvoie donc un siphon maximal. Les SMS sont toujours préférés dans les méthodes de contrôles pour les RdP ordinaires. Ce sont eux que l'on cherche à contrôler. Dans [Cha09], l'auteur vient "revisiter" la méthode de [CX97] de façon à ce que ce soit directement un SMS obtenu à partir du MIP. La génération directe de SMS rend obsolète les méthodes d'extractions d'un SMS à partir d'un siphon maximal dans le cas des RdP ordinaires. Dans la littérature, les méthodes qui "revisitent" la méthode de [CX97] sont généralement nommées RMIP [Cha09, LL12, LAW+14]

Cette méthode se base sur l'équation d'état 3.19. Elle trouve donc un état $M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)$. Il est donc possible que le marquage trouvé n'appartienne pas à $\mathcal{R}(\mathcal{N}, M_0)$. Il est possible de venir contrôler le siphon extrait par la génération d'une place de contrôle de manière à ce siphon ne se vide pas. Dans le cas ou le marquage M n'est en réalité pas atteignable (i.e $M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)$ et $M \notin \mathcal{R}(\mathcal{N}, M_0)$) la présence de la place de contrôle n'est donc pas nécessaire.

Extraction de siphons mortellement marqués dans les Réseaux de Petri Généralisés [PR01]

Une méthode d'extraction des siphons est présentée dans [PR01]. Dans les RdPs généralisés, il n'est pas suffisant de s'intéresser uniquement au fait qu'un siphon soit vidé, les siphons pouvant atteindre un marquage critique non nul entraînant la mort de certaines de leurs transitions. Dans [PR01], les auteurs définissent la notion de marquage mortel d'un siphon maximal S, un marquage qui une fois atteint entraîne la mort de transitions dans ${}^\bullet S$. La détection de ce marquage se fait aussi par la formulation d'un MIP et reprend les notions présentées dans [CX97]. Comme dans [CX97], cette méthode n'est utilisable que pour les RdPs **bornés**.

Définition 47. *Pour un Réseau de Petri $\mathcal{N} = (P, T, F, W)$ initialement marqué par M_0, $S \subseteq P$ est un siphon mortellement marqué en $M \in \mathcal{R}(\mathcal{N}, M_0)$ si et seulement si $\forall t \in {}^\bullet S$, t n'est pas sensibilisée par une place $p \in S$.*

Un MIP est formulé de manière à ce que la solution renvoie un siphon S et un marquage M, où S est mortellement marqué. Comme dans [CX97], les deux vecteurs v et z sont utilisés avec $v \in \{0,1\}^{|P|}$ qui représente le porteur des places en dehors de S et z est un vecteur $z \in \{0,1\}^{|T|}$. Un nouveau vecteur binaire $f \in \{0,1\}^{|F^-|}$ est rajouté, associé à chaque arc allant d'une place à une transition avec F^-, l'ensemble des arcs place-transition avec $F^- = F \cap (P \times T)$. Avec $f(p,t) = 0$ si le marquage de la place $p \in S$ désensibilise la transition t. v, z et f sont donc définis de la manière suivante :

$$v(p) = \begin{cases} 1 & \text{if } p \notin S \\ 0 & \text{else} \end{cases}, z(t) = \begin{cases} 1 & \text{if } t \notin S^\bullet \\ 0 & \text{else} \end{cases} \text{ et } f(p,t) = \begin{cases} 1 & M(p) \geq W(p,t) \vee v(p) = 1 \\ 0 & \text{else} \end{cases} \tag{3.22}$$

$$f(p,t) \geq \frac{M(p) - W(p,t) + 1}{SB(p)}, \quad \forall (p,t) \in F^- \tag{3.23}$$

$$f(p,t) \geq v(p), \quad \forall (p,t) \in F^- \tag{3.24}$$

$$z(t) \geq \sum_{p \in {}^\bullet t} f(p,t) - |{}^\bullet t| + 1, \quad \forall t \in T \tag{3.25}$$

$$v(p) \geq z(t), \quad \forall (t,p) \in F^+ \tag{3.26}$$

$$M = M_0 + CY, \quad M \geq 0,\ Y \in \mathbb{Z}^+ \tag{3.27}$$

Comme G^{MIP} cherche à minimiser $\sum_{p \in P} v(p)$, cette méthode cherche le plus grand siphon possible. Elle renvoie donc un siphon maximal. Comme dans [CX97] cette formulation se base sur l'équation d'états. Le marquage M trouvé appartient à $\vec{\mathcal{R}}(\mathcal{N}, M_0)$, mais ne fait pas partie nécessairement de $\mathcal{R}(\mathcal{N}, M_0)$. Dans le cas où un marquage mortellement bloquant n'est pas accessible (i.e $M \notin \mathcal{R}(\mathcal{N}, M_0)$), une place de contrôle visant à empêcher d'atteindre ce marquage M n'a pas nécessairement d'utilité.

3.4 Conclusion

Ce chapitre présente des éléments importants de la théorie du contrôle supervisé. La première section (section 3.1) introduit la méthode historique présentée par Ramadge et Wonham sous le formalisme des automates à états finis. La suite de la thèse s'axe uniquement autour du contrôle des Réseaux de Petri. La section sur les automates permet d'établir un cadre de comparaison avec les méthodes sur les Réseaux de Petri. En effet dans le cadre de la contrôlabilité partielle, la méthode de Ramadge et Wonham permet d'obtenir un contrôle maximalement permissif. Cette méthode a l'avantage de pouvoir traiter en une seule fois le respect des spécifications et l'évitement des blocages. Une importante remarque est faite dans la sous-section

3.1.4 qui traite des systèmes sous observabilité et contrôlabilité partielle : il n'existe pas de contrôleur suprême pour ces systèmes. Cependant dans le cas ou $E_c \subseteq E_o$, le système est dit normal et induit l'existence d'un contrôleur suprême. Même si les méthodes de contrôle sur les automates donnent d'excellents résultats en termes de permissivité, leur complexité algorithmique rend leur application impossible sur des systèmes de grandes tailles, d'autant plus pour le cas de l'observabilité partielle où la réalisation d'un observateur est nécessaire (dont la complexité est exponentielle).

Les sections 3.2 et 3.3 sont consacrées aux Réseaux de Petri. La section 3.2 considère des spécifications d'évitement de marquages interdits. Ici les marquages interdits sont formulés sous forme d'une contrainte d'inégalité sur les marquages. Ces contraintes, dites GMEC, pondèrent chacune des places du Réseau de Petri et définissent si un marquage est interdit ou non en fonction du nombre de jetons dans ces places. Dans le cadre contrôlable et observable, il est possible de venir générer une place dite monitrice qui viendra appliquer un contrôle maximalement permissif au Réseau de Petri de manière à ce que celui-ci respecte la GMEC. Le cas de la contrôlabilité et observabilité partielles est plus complexe. Il est montré qu'il n'est pas toujours possible d'avoir une place de contrôle qui exerce un contrôle maximalement permissif (même dans le cas normal). Cette remarque a donné lieu à un nouveau champ d'étude : la transformation des contraintes, qui consiste à définir une conjonction/disjonction de contraintes admissibles permettant de définir les marquages interdits. À notre connaissance, la solution de ce problème, dans le cas général, n'a toujours pas été définie. À la fin de la section 3.2, une méthode de transformation de contrainte, présentée dans [MA00], est présentée et permet à partir d'une contrainte non-admissible de trouver une contrainte admissible. La méthode n'exerce pas de contrôle optimal mais est extrêmement rapide à calculer (faible complexité : résolution d'un problème d'optimisation linéaire).

La section 3.3 s'intéresse uniquement au problème d'évitement des blocages. Dans les Réseaux de Petri, les blocages sont liés à une structure particulière : les siphons. Les blocages sont des problèmes comportementaux du Réseau de Petri, qui peuvent être simplement traités par l'exploration du graphe d'accessibilité. L'étude des siphons permet de transformer ce problème comportemental en un problème structurel. La sous-section 3.3.1 présente les conditions de contrôle des siphons. Elle est divisée en deux parties : le cas des Réseaux de Petri ordinaires et celui des Réseaux de Petri généralisés. Dans le cas ordinaire, pour un siphon donné, le contrôle du siphon est direct car il suffit d'éviter que celui-ci ne se vide. La génération d'une place de contrôle exerce un contrôle maximalement permissif. Malheureusement, la place de contrôle qui interdit le vidage du siphon peut avoir des arcs pondérés qui ramènent le problème à celui du cas généralisé. Dans le cas généralisé, le problème est plus complexe. Les travaux de [BPP96] établissent un cadre à partir duquel on peut identifier des contraintes qui assurent que le siphon ne se vide pas mais ne garantissent pas que le contrôle soit maximalement permissif. Dans un Réseau de Petri le nombre de siphons peut être exponentiel en fonction du nombre de places. Si tous les siphons sont contrôlés, cela peut mener à un grand nombre de places. L'étude des siphons élémentaires permet d'identifier quels siphons sont d'intérêt à contrôler. L'identification de ces siphons permet de réduire grandement le nombre de places de contrôle. La section 3.3.2 présente les notions relatives aux siphons

élémentaires développées dans [LZ04]. La sous-section 3.3.3 présente trois philosophies liées à la détection de blocage : les méthodes basées sur le graphe d'accessibilité, l'énumération des siphons et l'extraction des siphons. Les deux premières ont un haut degré de complexité algorithmique. Dans la suite de la thèse nous nous servirons de méthodes d'extraction car elles sont plus compatibles avec des systèmes de grandes tailles. Deux méthodes d'extraction basées sur des MIP sont présentées : [CX97] qui n'est applicable qu'aux Réseaux de Petri ordinaires et [PR01] applicable à des Réseaux de Petri généralisés mais plus complexe à résoudre.

Ce chapitre présente des éléments importants de la théorie du contrôle supervisé mais ne se veut pas exhaustif. Dans le cas de la SCT, différentes approches ont été proposées pour réduire la complexité algorithmique, c'est le cas de l'approches modulaire [WR88, DQC00], hiérarchique [dCC07] ou encore décentralisée [RW95]. Pour les Réseaux de Petri, des formalismes de hauts niveaux capturent les symétries d'un Réseau de Petri généralisé. Les méthodes de synthèse basées sur ces formalismes permettent un gain notable en complexité algorithmique et en complexité structurelles [FGS03, FGS06].

La synthèse de contrôleur se trouve être étendue à d'autres paradigmes que les SED « logiques ». La prise en compte de notions fluidiques [DA10] a amenée de nouvelles méthodes de synthèse pour des formalismes qui sortent de notre cadre d'étude. Plusieurs types d'approches permettent d'aborder la problématique de la supervision dans ces formalismes : systèmes temporels [KLB09, KLB18], hybrides [KHLA98], Réseau de Petri lots (*batches Petri nets*) [DG14]...

Enfin la notion de synthèse de contrôleurs est adaptée à différents contextes : changement de modes de marches [FPN09], contrôleur robuste [Lin93], en présence de fautes [Jen03]...

Chapitre 4

Réseaux de Petri pour les systèmes d'allocations de ressources

La recherche des blocages dans les Réseaux de Petri étant complexe dans le cas général, beaucoup d'auteurs ont présenté des méthodes d'analyse et de contrôle pour des classes spécifiques de Réseaux de Petri. Beaucoup de ces classes de RdP ont été spécialement développées pour modéliser des Systèmes de Production Flexible Manufacturière (FMS : Flexible Manufacturing Systems), mais aussi des programmes informatiques, des protocoles de communication, des systèmes de transports ... Ces systèmes se trouvent inclus dans la dénomination : système d'allocations de ressources (RAS : Resource Allocation System). Un RAS est un Système à Événement Discret (SED) dans lequel un ensemble de processus évoluent en parallèle. Ces processus requièrent l'allocation de ressources qui sont partagées entre les différents processus. Une mauvaise allocation des ressources aux différents processus peut entraîner une situation de blocage. L'étude de ces situations est relativement ancienne. Elle a déjà été évoquée par Dijkstra [Dij71] à travers le classique problème des philosophes. L'étude [CES71] énumère les conditions nécessaires à l'apparition d'un blocage dans un RdP. Ces conditions peuvent être résumées de la manière suivante :

1. *Condition d'exclusion mutuelle :* Une ressource ne peut pas être utilisée par plus d'un processus à la fois.
2. *Condition de maintien et d'attente :* Un processus qui détient déjà une ressource peut demander une ressource différente.
3. *Condition de non-préemption :* Une ressource ne peux pas être retirée au processus qui la détient de manière forcée. La ressource est relâchée par l'action du processus.
4. *Condition d'attente circulaire :* Au moins deux processus forment une chaîne circulaire où chacun des processus attend qu'un autre processus libère une autre ressource.

La figure 4.1 illustre une situation de blocage dans un système d'allocation de ressources. Les 4 zones hachurées peuvent être interprétées comme des ressources. Dans le contexte de cet exemple, tous les conducteurs souhaitent aller droit devant. Les conducteurs étant extrêmement têtus, leur trajectoire respective ne peut pas être modifiée. Dans ces zones, un seul véhicule peut être présent. La condition 1 d'exclusion mutuelle est donc vraie. La condition 2 est donc respectée car pour se déplacer, les véhicules doivent accéder à la zone suivante avant de libérer leur zone. Pour qu'une voiture libère sa ressource, il faudrait qu'elle fasse marche arrière. Cela

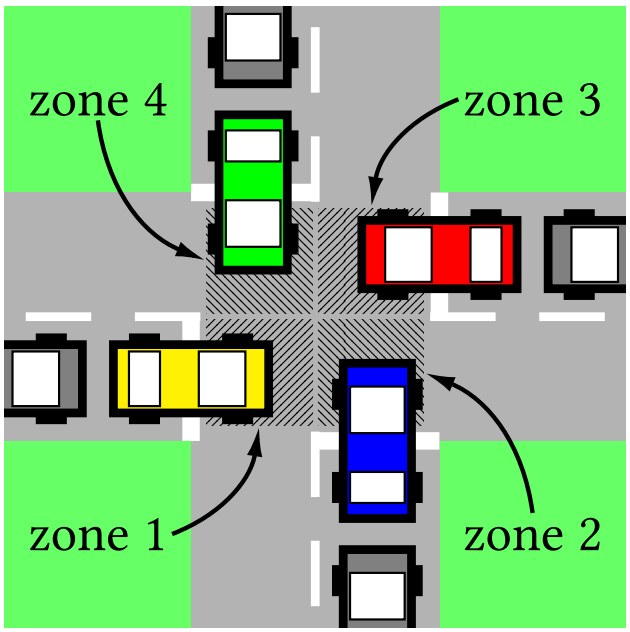


FIGURE 4.1 – Exemple d'un blocage dans un RAS

est impossible car elle est bloquée par une voiture grise. Elle pourrait également tourner à droite, mais les conducteurs sont trop têtus et ne veulent modifier leur direction sous aucun prétexte. La condition 3 est donc vraie, les ressources ne pouvant être libérées de manière impérative. Dans cette situation, il a été créé une chaîne circulaire d'attente des ressources :

— Le véhicule jaune attend que la zone 2 soit libérée par le véhicule bleu ;
— Le véhicule bleu attend que la zone 3 soit libérée par le véhicule rouge ;
— Le véhicule rouge attend que la zone 4 soit libérée par le véhicule vert ;
— Le véhicule vert attend que la zone 1 soit libérée par le véhicule jaune.

La situation de la figure 4.1 est donc bloquante.

Dans un premier temps sont présentées différentes classes de RAS-PN présentes dans la littérature. Ces classes seront décrites via les restrictions qui leur sont appliquées par rapport aux Réseaux de Petri généralisés. Un ordre partiel d'expressivité y est défini selon la restriction de chacune des classes. Sur l'ensemble de ces classes seulement trois seront utilisés dans la suite de cette thèse : les S3PR, les S3PMR et les S4PR. Ces classes se trouvent être ordonnées par restrictivité de la manière suivante : S3PR $\subseteq$ S3PMR $\subseteq$ S4PR. Par la suite, la structure de ces classes est détaillée. Les propriétés de chaque classe vis-à-vis de la vivacité et de leurs liens avec les siphons sont énumérées. Ces propriétés permettent par la suite la mise en place de stratégies de contrôle. Dans le cadre, de ce chapitre le contrôle est effectué par génération de places monitrices. La hiérarchie entre les classes induit que les propriétés des classes les moins restrictives s'appliquent aussi aux classes les plus restrictives. Lorsque c'est possible, ces propriétés et les méthodes de contrôle seront affiné aux cas des classes les plus restrictives. Les méthodes de contrôle développées pour les classes les moins restrictives sont aussi applicables aux classes les plus restrictives.

Mais les classes les plus restrictives autorisent des méthodes plus spécialisées avec des propriétés plus intéressantes.

4.1 Classes de Réseaux de Petri pour la modélisation des RAS

Les RAS ont fait l'objet de nombreuses recherches. Les Réseaux de Petri sont des outils efficaces pour modéliser, analyser et superviser les RAS. De nombreuses classes de RdP ont été développées pour modéliser différents types de RAS. La grande diversité de ces classes et les dénominations hétérogènes pour chacune d'elles rend la synthèse de l'état de l'art complexe. Plusieurs travaux de recensement, synthèse et classification ont été réalisés. [LB16] présente une étude générale des publications concernant les siphons. Dans [HB17], plusieurs classes sont présentées et leurs propriétés vis-à-vis des siphons énoncées. Dans [LWZ11], à travers la thématique du contrôle des SFPMs, l'auteur présente différentes classes de RdPs et les stratégies de contrôles qui leur sont associées. Dans [LZ09], les mêmes auteurs traitent aussi de la thématique du contrôle des SFPMs axée particulièrement autour de la notion de siphon élémentaire. Un travail de catégorisation des Réseaux de Petri pour les systèmes d'allocations de ressources a été établi dans [LGC13]. Dans cet article, les auteurs observent « l'accablante » littérature pour l'étude des blocages dans les FMS (« The disparity of Petri net models in the literature for the study of resource allocation problems in Flexible Manufacturing Systems (FMSs) is overwhelming »). Fort de cette observation, les auteurs proposent une relation d'ordre entre les différentes classes de RdP présentées dans la littérature. A partir d'une énumération de différentes restrictions structurelles, ces classes sont catégorisées. Ensuite les propriétés de chacune d'elles sont énumérées.

4.1.1 Modélisation des RAS pour les Réseaux de Petri

La modélisation des RAS repose sur deux concepts élémentaires : les *processus* et les *ressources*. Dans la grande majorité des classes de Réseaux de Petri pour la modélisation des RAS (RAS-PN : Resource Allocation System-Petri Net), l'ensemble des places est divisé en deux sous-ensembles disjoints : les places de *processus* et les places de ressources. La majorité des classes de RAS-PN permet de modéliser un ensemble de n processus dont les indices sont répertoriés dans l'ensemble $\mathcal{I}$ avec $|\mathcal{I}| = n$. Dans le cas des RAS, seules les ressources permettent de lier les processus sous un modèle unique. En absence de ressources, les différents processus sont considérés comme indépendants.

Définition 48. *Considérons un Réseau de Petri $\mathcal{N} = (P, T, F, W)$ qui modélise un RAS avec n processus indicés par les éléments de $\mathcal{I}$. Pour chaque processus, il est associé un Réseau de Petri $\mathcal{N}_i = (P_i, T_i, F_i, W_i)$ avec $i \in \mathcal{I}$, tel que le Réseau du processus en un sous-Réseau de $\mathcal{N}$ i.e $\mathcal{N}_i \subseteq \mathcal{N}$. Les processus étant indépendants, les places et transitions d'un processus ne peuvent être partagés avec un autre processus i.e $\forall i, j \in \mathcal{I}$ avec $i \neq j$, $P_i \cap P_j = \varnothing$ et $T_i \cap T_j = \varnothing$*

Dans la majorité des classes de RAS-PN, l'évolution du système correspond uniquement à l'évolution des processus. Les transitions, modélisant la dynamique des

Réseaux de Petri, sont donc forcement associées à un unique processus. Il s'ensuit donc que $\bigcup_{i\in\mathcal{I}} T_i = T$.

Définition 49. *Un Réseau de Petri $\mathcal{N} = (P, T, F, W)$ modélisant un RAS possède un ensemble P^R de places de ressources. Ces places de ressources ne sont comprises dans aucun processus. P est formé par l'union disjointe des places de processus et des places de ressources i.e $P = \biguplus_{i\in\mathcal{I}} P_i \uplus P^R$.*

La figure 4.2 représente un RAS-PN $\mathcal{N}$. Ce Réseau comporte deux processus $\mathcal{I} = \{a, b\}$. Le sous-Réseau $\mathcal{N}_a$ du processus a est formé par l'ensemble des places $P_a = \bigcup_{i=1}^{3}\{p_{ai}\} \cup \{p_a^0\}$ et des transitions $T_a = \bigcup_{i=1}^{5}\{t_{ai}\}$. Pour le sous-Réseau $\mathcal{N}_b$, l'ensemble des places est $P_b = \bigcup_{i=1}^{3}\{p_{bi}\} \cup \{p_b^0\}$ et l'ensemble des transitions est $T_b = \bigcup_{i=1}^{5}\{t_{bi}\}$. $\mathcal{N}$ comporte trois places de ressources : $P^R = \{p_1^R, p_2^R, p_3^R\}$.

4.1.2 Restriction des classes de Réseaux de Petri pour les RAS

Dans [LGC13], un travail de catégorisation des classes de RAS-PN a été effectué. Les classes RAS-PN sont définies par rapport aux restrictions supplémentaires appliquées à un RdP généralisé. Les auteurs ont identifié un ensemble de restrictions habituellement rencontrées dans les classes de RAS-PN. Ces restrictions permettent de comparer aisément deux classes.

Restrictions globales du système

1. *Réseau de Processus Unique :* Le RAS-PN ne comporte qu'un seul Réseau de processus $\mathcal{N}_i$ avec $\{i\} = \mathcal{I}$. Dans le cas de la figure 4.2 ce n'est pas le cas car il existe deux processus a et b.
2. *Réseau de Processus fermé :* Il est possible de représenter plusieurs instances d'un même processus sans dédoubler les sous-Réseaux de processus. Pour cela, il suffit d'avoir plusieurs jetons dans le même processus. Les processus intègrent une place spéciale appelée place de repos. Dans un processus $i \in \mathcal{I}$, cette place est unique et notée par p_i^0. Quand elle est marquée, cette place représente des instances de processus inactif. Son marquage initial permet de limiter le nombre d'instances d'un même processus, bornant le nombre de jetons dans le Réseau de processus $\mathcal{N}_i$. Généralement les jetons dans les places de repos ne requièrent la consommation d'aucune ressource. Lorsque les processus sont ouverts (sans places de repos), le nombre de jetons dans un processus n'est pas explicitement borné. Dans le cas de la figure 4.2, les processus a et b possèdent des places de repos respectivement p_a^0 et p_b^0. $\mathcal{N}$ a donc des processus fermés.
3. *Ressources Binaires :* Chaque type de ressource est modélisé par une place de ressource. Si on considère qu'au marquage initial il n'existe pas de processus actifs, le marquage initial des places ressources définit le nombre d'instances disponibles de chaque type de ressources. Dans le cas d'un Réseau avec ressources binaires, le nombre de ressources allouées simultanément aux processus n'excède jamais un. Dans le cas de la figure 4.2, les ressources modélisées par les places p_2^R et p_3^R sont des ressources binaires mais pas la ressource p_1^R. $\mathcal{N}$ n'est donc pas un RAS-PN avec des ressources binaires.

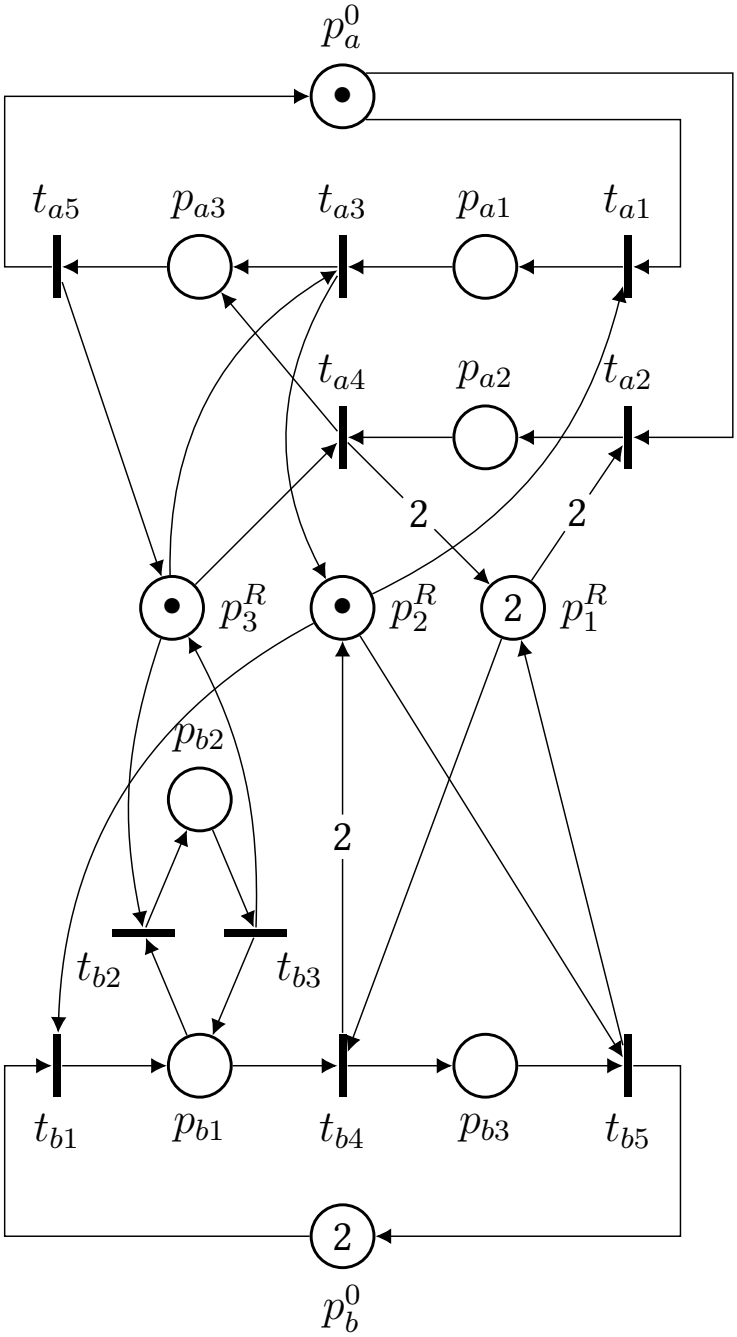


FIGURE 4.2 – Réseau de Petri modélisant un RAS

Restrictions liées aux Réseaux de processus

i *Reproductibilité des T-semiflots :* Concerne la possibilité de pouvoir répéter indéfiniment les T-flots d'un RAS-PN considérant uniquement les processus. Dans les classes de RAS-PN, les T-semiflots représentent l'exécution d'un processus ou d'un cycle interne. Dans le cas où les Réseaux de processus sont modélisés par des machines à états fortement connexes (la majorité des classes de RAS-PN) cette condition est nécessairement respectée. Les lecteurs peuvent se référer à [ER04] où une classe de RAS-PN ne respectant pas cette condition est présentée.

ii *Consistance :* Cette propriété est souvent désirée dans les RAS-PN. Elle est vraie lorsque chaque transition est incluse dans un T-semiflot. Associée à la restriction *Reproductibilité des T-semiflots*, elle garantit la répétabilité d'un processus. Dans le cas où chaque « chemin » d'un processus est un T-semiflot, alors le RAS-PN est consistant.

iii *Equivalence des T-semiflots :* Cette propriété se réfère au fait que chaque T-semiflot est exécutable, isolé de la place de repos (toujours exécutable immédiatement si la place de repos est supprimée). La place de repos ne représente pas une étape active d'un processus. Certains T-semiflots correspondent à des cycles internes dans un processus. L'exécution de ces cycles dépend de l'état courant du processus. Dans le cas de la figure 4.2, le T-semiflot $Y \in \mathcal{T}_{\circlearrowleft}(\mathcal{N}_b)$ tel que $\|Y\| = \{t_{b2}, t_{b3}\}$ correspond à un cycle interne du processus b. Pour qu'une séquence de transitions correspondant à Y puisse être tirée, il faut nécessairement que p_{b1} ou p_{b2} soit marqué. Y ne dépend donc pas uniquement du marquage de p_b^0. Le RAS-PN de la figure 4.2 ne respecte donc pas la propriété sur l' *équivalence des T-semiflots*. La présence des cycles internes dans les RAS-PN compliquent beaucoup leur analyse dans le cas général.

iv *Terminaison des processus :* Cette propriété est vérifiée si, quel que soit l'état d'un processus, son exécution peut être terminée. Il est donc toujours possible d'atteindre la place de repos.

v *Vivacité des processus :* Cette propriété est vérifiée si, considérant les processus seuls, chacun des processus est vivant. Dans beaucoup de RAS-PN, la problématique de la non-vivacité intervient uniquement lorsque sont ajoutées les places de ressources. Cette propriété est vérifiée lorsque les processus sont terminaux et le Réseau est consistant.

vi *Pas de décisions internes :* Cette propriété impose de ne pas avoir de conflit au sein d'un processus. Il est impossible dans une place d'un processus de se retrouver face à un choix pour la prochaine place du processus. Autrement dit, pour chaque place de processus p autre que la place d'attente, $|p^\bullet| = 1$.

vii *Séquentialité du processus :* Cette propriété interdit au processus de pouvoir se séparer en différentes sous-tâches et interdit aussi la synchronisation de fin de tâches. Autrement dit, $\forall t \in T$, $|t^\bullet| = 1$ et $|{}^\bullet t| = 1$. La non-séquentialité des processus rend l'analyse des RAS-PN complexe.

Restrictions liées aux Réseaux de ressources

Un Réseau de ressources est un sous-Réseau d'un RAS-PN qui rassemble une place de ressource et les places de processus qui la consomment. Un Réseau de ressource permet donc de suivre l'évolution d'une ressource de sa prise jusqu'à sa libération.

Définition 50. *On appelle $\mathcal{N}_i^R$ le sous-Réseau d'une ressource $p_i^R \in P^R$ d'un RAS-PN $\mathcal{N} = (P, T, F, W)$. $\mathcal{N}_i^R$ est un sous-Réseau de $\mathcal{N}$ i.e $\mathcal{N}_i^R \subseteq \mathcal{N}$ avec $\mathcal{N}_i^R = (P^{\mathcal{N}_i^R}, T^{\mathcal{N}_i^R}, F^{\mathcal{N}_i^R}, W^{\mathcal{N}_i^R})$. Le sous-Réseau d'une ressource p_i^R ne peut comporter qu'une seule ressource, $P^{\mathcal{N}_i^R} \cap P^R = \{p_i^R\}$.*

a *Conservation des ressources :* Cette propriété est couramment rencontrée dans les RAS-PN. Lorsque cette propriété est vérifiée, alors pour chaque ressource $r \in P^R$ d'un RAS-PN $\mathcal{N}$, il existe un P-semiflot $X \in \mathcal{P}^+_{\circlearrowleft}(\mathcal{N})$ tel que $\|X\| \cap P^R = \{r\}$. Dans le cas de la figure 4.2, la ressource modélisée par la place p_3^R dans le RAS-PN $\mathcal{N}$ est conservatrice car il existe un P-semiflot $X \in \mathcal{P}^+_{\circlearrowleft}(\mathcal{N})$ avec $X \in \{0,1\}^{|P|}$ et $\|X\| = \{p_{a3}, p_{b2}, p_3^R\}$.

Définition 51. *Le P-semiflot généré par une place de ressource p_i^R est noté $\vec{p}^R_{i\circlearrowleft}$ tel que $\vec{p}^R_{i\circlearrowleft} \in \mathcal{P}^+_{\circlearrowleft}(\mathcal{N}_i^R)$, avec $\|\vec{p}^R_{i\circlearrowleft}\| = P^{\mathcal{N}_i^R}$.*

b *Sous-Réseau de ressource ordinaire :* Le maintien d'une ou plusieurs ressources est autorisé dans une place de processus, mais il est impossible de prendre ou de libérer plusieurs ressources simultanément. D'un point de vue des RAS-PN, cela se traduit par le fait que les poids des arcs liant les ressources sont égaux a un i.e quelque soit $p^R \in P^R$ alors $\forall t \in p^{R\bullet}, W(p^R, t) = 1$ et $\forall t \in {}^\bullet p^R, W(t, p^R) = 1$.

c *P-semiflot binaire :* Cette restriction implique que les processus ne peuvent requérir plus d'une instance d'une même ressource. Cela se traduit dans les RAS-PN par des P-semiflots binaires générés par les Réseaux de ressources.

d *P-semiflot orthogonaux :* Cette propriété implique qu'une place de processus ne peut consommer qu'un seul type de ressource. Pour évoluer, un processus doit libérer la/les ressource(s) précédemment acquise(s) avant de pouvoir acquérir une ressource d'un autre type. Pour les RAS-PN cela ce traduit par des supports de P-semiflots qui ne peuvent avoir de places de processus communes i.e $\forall p_1^R, p_2^R \in P^R$ avec $p_1^R \neq p_2^R$ alors $\vec{p}^R_{1\circlearrowleft} \cap \vec{p}^R_{2\circlearrowleft} = \varnothing$.

e *Conflit interne indépendant des ressources :* Cette propriété est vérifiée lorsque les ressources n'influent pas sur les conflits internes d'un processus (voir restriction vi). Lorsque un processus se trouve dans une situation de choix, ce choix ne peux être influencé par l'état des places de ressources. Dans le cas de la figure 4.2, il existe un conflit dans le processus b, car $p_{b1}^\bullet = \{t_{b2}, t_{b4}\}$. Pour tirer la transition t_{b2} il est nécessaire d'avoir un jeton dans p_3^R. Le conflit est donc influencé par une ressource. Ce RAS-PN ne respecte donc pas cette propriété.

f *Pas de prêt de ressource :* Cette propriété est vérifiée lorsque une ressource est générée par un processus. Cette ressource peut être ensuite partagée avec un

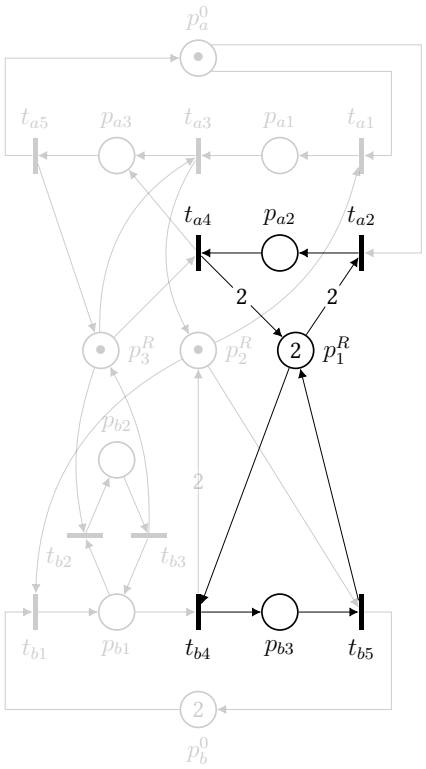


FIGURE 4.3 – Réseau de $p_1^R, \mathcal{N}_1^R$

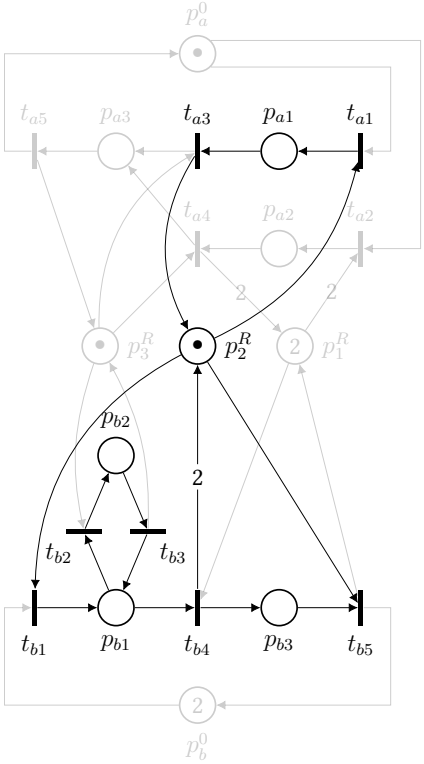


FIGURE 4.4 – Réseau de $p_2^R, \mathcal{N}_2^R$

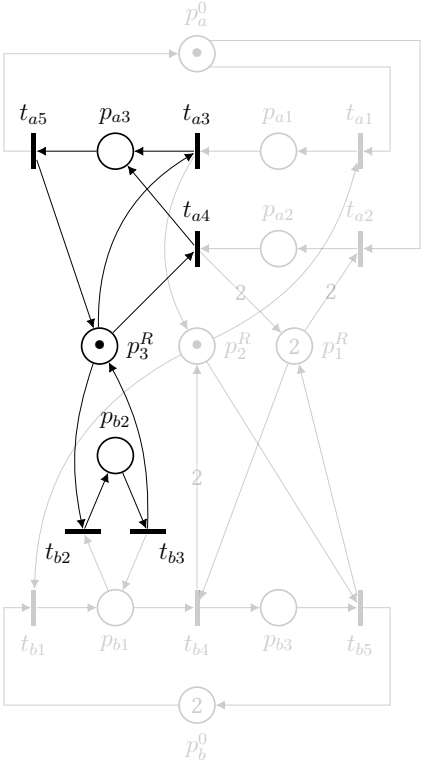


FIGURE 4.5 – Réseau de $p_3^R, \mathcal{N}_3^R$

autre processus, puis détruite par le processus qui l'a générée, de manière à ce que le Réseau de ressource génère un invariant. Dans le cas de la figure 4.2, le tir de la transition t_{b4} génère un jeton supplémentairement dans p_2^R. Ce jeton supplémentaire sera ensuite consommé au tir de t_{b5}.

g *Premier alloué, premier relâché :* Cette propriété établit un lien entre les ordres d'allocation et de libération des ressources. Dans chaque chemin de processus, les ressources qui ont été allouées en premier doivent être les premières à être relâchées.

Classes de RAS-PN dans la littérature

Les restrictions définies précédemment suffisent pour catégoriser la majeure partie des classes de RAS-PN rencontrées dans la littérature. La figure 4.6 présente un certain nombre de classes de RAS-PN, les relations d'inclusion et les possibilités de transformer une classe en une autre. Le tableau 4.1 détaille les restrictions imposées à chacune de ces classes. Une version de la figure 4.6 et du tableau 4.1 comportant plus de classes de RAS-PN est détaillée en dans [LGC13]. Dans la littérature, il est possible qu'une même classe porte des noms différents en fonction des auteurs qui ont étudié cette classe. Les classes présentées dans le tableau 4.1 peuvent être retrouvées dans les articles suivants.

— 1 : L-S3PR (Linear - System of Simple Sequential Process with Resources) [EGVC98]

— 2 : S3PR (System of Simple Sequential Process with Resources) [ECM95, BA95]

— 3 : S3PMR (System of Simple Sequential Process with Multiples Resources) [HJXC06], ES3PR [HJXC01]

— 4 : ES3PR (Extended System of Simple Sequential Process with Resources) [TGVCE98]

— 5 : S4PR [1] [TGVCE05], WS3PSR (Weighted Simple Sequential Processes with Several Resources) [TM95], S4R (Systems of Sequential Systems with Shared resources) [ZL10], S3PGR2 [PR01, Cha07] (System of Simple Sequential Processes with General Resource Requirements)

— 6 : SPQR (Systems of Processes Quarrelling over Resources) [LGC06]

— 7 : PC2R (Processes Competing for Conservative Resources)[LGC12, LGCP12]

— 8 : S*PR [2] [ETGVC02]

4.2 Définition des classes : S3PR, S3PMR et S4PR

Dans cette section, trois classes de RAS-PN sont détaillées. Elles ont en commun une même structure de processus. La distinction entre ces classes se fait via les Réseaux de ressources. Moins de restrictions sont appliquées aux Réseaux de ressources, plus la classe de RAS-PN peut modéliser un système complexe. En contrepartie, plus la structure des Réseaux de Petri est complexe, moins le RdP comporte de bonnes propriétés. Cela rend son analyse et son contrôle plus difficiles.

1. n'est pas un acronyme
2. n'est pas un acronyme

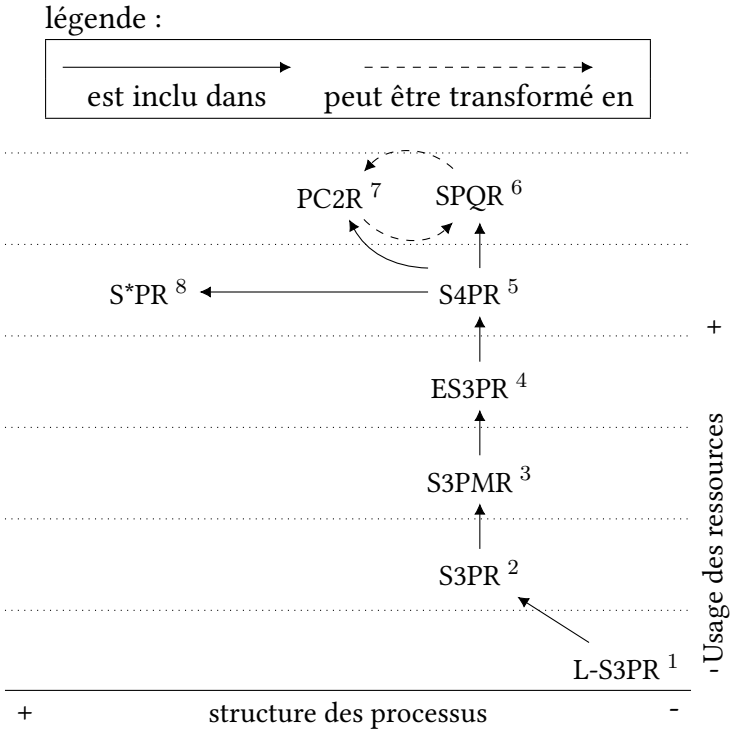


FIGURE 4.6 – Relations entre classes de RAS-PN

4.2.1 Réseaux de Processus

Pour les classes S3PR, S3PMR et S4PR, les Réseaux de processus sont définis de manière équivalente :

Définition 52. *Un Réseau de processus d'indice i est un Réseau de Petri ordinaire défini par $\mathcal{N}_i^P = (\{p_i^0\} \cup P_i^P, T_i, F_i^P)$ avec P_i^P l'ensemble des places du processus, p_i^0 sa place de repos, T_i l'ensemble de ses transitions et F_i^P ses arcs. $\mathcal{N}_i^P$ est une machine à états*[3] *fortement connexe dont tous les cycles*[4] *comportent p_i^0.*

Remarque 23. *Les transitions qui appartiennent à $p_i^{0\bullet}$ sont appelées transitions sources. Le tir d'une transition source correspond à l'instanciation d'un cycle du processus. Le tir de cette transition consomme un jeton dans la place de repos et en génère un dans une place de processus. Dans le cas d'un SFPM cela correspondrait au lancement en production d'une pièce.*

De la même manière, les transitions qui appartiennent à ${}^\bullet p_i^0$ sont appelées transitions puits. Le tir d'une transition puits correspond à la fin d'un processus. Lors du tir de ces transitions, un jeton d'une place de processus est consommé et, un jeton est généré dans la place de repos. Dans le cas d'un SFPM, le tir de ces transitions correspond le plus souvent à la fin de fabrication d'une pièce.

Définition 53. *On considère qu'à l'état initial, un Réseau de processus $\mathcal{N}_i$ a uniquement sa place de repos marquée.*

$$M_0(p_i^0) \geq 1 \text{ et } \quad \forall p \in P_i^P,\ M_0(p) = 0$$

3. voir annexe A.1.3
4. voir annexe A.1.1

réference des classes de RAS-PN	1	2	3	4	5	6	7	8
Restrictions globales du système								
Réseau de Processus Unique	✗	✗	✗	✗	✗	✗	✗	✗
Réseau de Processus fermé	✓	✓	✓	✓	✓	✗	✓	✓
Ressources Binaires	✗	✗	✗	✗	✗	✗	✗	✗
Restrictions liées aux réseaux de processus								
Reproductibilité des T-semiflots	✓	✓	✓	✓	✓	✓	✓	✓
Consistance	✓	✓	✓	✓	✓	✓	✓	✓
Equivalence des T-semiflots	✓	✓	✓	✓	✓	✓	✗	✗
Terminaison des processus	✓	✓	✓	✓	✓	✓	✓	✓
Vivacité des processus	✓	✓	✓	✓	✓	✓	✓	✓
Pas de décisions internes	✓	✗	✗	✗	✗	✗	✗	✗
Séquentialité du processus	✓	✓	✓	✓	✓	✓	✓	✓
Restrictions liées au réseau de ressource								
Conservation des ressources	✓	✓	✓	✓	✓	✓	✓	✓
Sous-réseau de ressource ordinaire	✓	✓	✓	✓	✗	✗	✗	✗
P-semiflot binaire	✓	✓	✓	✗	✗	✗	✗	✗
P-semiflot orthogonaux	✓	✓	✗	✗	✗	✗	✗	✗
Conflit interne indépendant des ressources	✓	✗	✗	✗	✗	✗	✗	✗
Pas de prêt de ressource	✓	✓	✓	✓	✓	✗	✗	✓
Premier alloué, premier relaché	✓	✓	✗	✗	✗	✗	✗	✗

TABLE 4.1 – Propriétés structurelles de différentes classes de RAS-PN

Les Réseaux de processus pour les S3PR, S3PMR et S4PR ont les propriétés suivantes :

— Reproductibilité des T-semiflots,

— Consistance,

— Equivalence des T-semiflots,

— Terminaison des processus,

— Vivacité des processus,

— Séquentialité du processus.

La *reproductibilité des T-semiflots* et la *consistance* sont assurées par le fait que les processus sont des machines à états fortement connexes. L'*équivalence des T-semiflots* est respectée car il n'existe pas de cycles internes dans le processus (tous les cycles doivent inclure p_i^0). La *terminaison des processus* est garantie car tous les cycles passent par p_i^0. Le Réseau étant fortement connecté, il n'existe pas « d'impasse ». Les jetons sont donc forcés de passer par la place de repos. Les processus sont *séquentiels* car les transitions ne pouvant avoir qu'une place prédécesseure et une place successeure, le synchronisme est rendu impossible. L'ensemble des propriétés précédentes implique qu'un processus seul est obligatoirement *vivant*.

Un exemple de Réseau de processus est donné en figure 4.7. $\mathcal{N}_i^P$ comporte 5 places de processus $P_i^P = \bigcup_{j=1}^{5}\{p_{j,i}\}$ et neuf transitions $T_i = \bigcup_{j=1}^{9}\{t_{j,i}\}$. $\mathcal{N}_i^P$ est bien une machine à états ($\forall t \in T_i$, $|t^\bullet| = 1$ et $|^\bullet t| = 1$). Le tir d'une transition du modèle

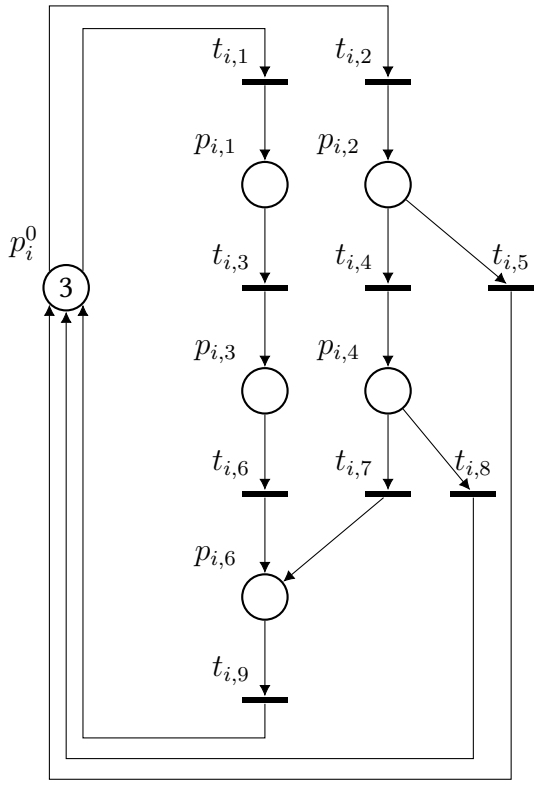


FIGURE 4.7 – Exemple d'un Réseau de Processus $\mathcal{N}_i^P$

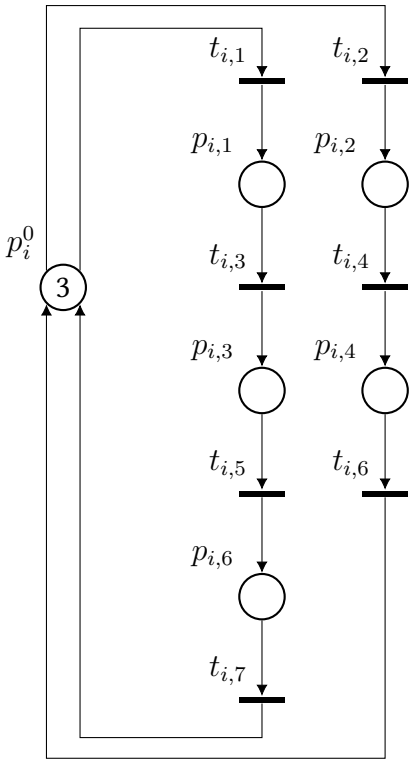


FIGURE 4.8 – Exemple d'un Réseau de Processus Linéaire $\mathcal{N}_i^P$

déplace uniquement un jeton d'une place à une autre. Tous les cycles comportent la place de repos p_i^0. Cela entraîne que le processus ne comporte pas de cycles internes.

La figure 4.8 modélise un Réseau de processus linéaire. Ce processus respecte la restriction vi, sur les décisions internes. Cette restriction implique qu'il n'est pas possible de faire de choix au sein du processus. La classe des L-S3PMR utilise ce type de processus (colonne 1 sur le tableau 4.1).

4.2.2 Réseaux de Ressources

Pour les classes S3PR, S3PMR et S4PR, seule la définition des Réseaux de ressources est différente. Ces trois classes ont une relation d'inclusion : S3PR ⊆ S3PMR⊆ S4PR . Cette sous-section traitera en premier les classes les plus englobantes, car leurs propriétés sont aussi applicables aux classes plus restrictives.

Une fois les places de ressources de P^R ajoutées, les différents Réseaux de processus se trouvent connectés pour former un Réseau unique.

Définition 54. *La structure de Réseaux de Petri qui appartiennent aux classes S4PR, S3PMR et S4PR peut être décrites de manière homogène. Un Réseau de Petri $\mathcal{N}$ qui appartient à une de ces classes sera toujours composé de $n = |\mathcal{I}|$ processus avec $n \in \mathbb{R}^+$. Ainsi $\mathcal{N}$ peut être défini par :*

$$\mathcal{N} = (P^0 \cup P^P \cup P^R, T, F, W)$$

Avec :

- $P^0 = \bigcup_{i\in\mathcal{I}}\{p_i^0\}$ *est l'ensemble des places de repos de chaque processus,*
- $P^P = \bigcup_{i\in\mathcal{I}} P_i^P$ *est l'ensemble des places de processus,*
- P^R *est l'ensemble des places de ressources,*
- $T = \bigcup_{i\in\mathcal{I}} T_i$ *est l'ensemble des transitions de chaque processus,*
- $F = \bigcup_{i\in\mathcal{I}} F_i^P \cup \bigcup_{p^R\in P^R}(p^{R\bullet} \cup {}^\bullet p^R)$ *est l'ensemble des arcs de chaque processus et arcs successeurs et prédécesseurs des ressources.*
- W *est l'application de pondération des arcs :*

$$W = \begin{cases} W(f) = 1 & si\ \exists i \in \mathcal{I}\,|\, f \in F_i^P \\ W(f) = W^{\mathcal{N}_i^R}(f) & si\ \exists p_i^R \in P^R\,|\, f \in F^{\mathcal{N}_i^R} \end{cases}$$

Remarque 24. *Les S3PR et S3PMR sont des Réseaux de Petri ordinaires. On pourra donc décrire ces Réseaux comme $\mathcal{N} = (P^0 \cup P^P \cup P^R, T, F)$. Pour les S4PR, seuls les arcs prédécesseurs et successeurs des ressources peuvent être pondérés.*

Les S3PR, S3PMR et S4PR respectent la propriété de *conservation des ressources* et de *non-prêt de ressource*. La propriété de conservation des ressources implique l'existence d'un P-semiflot $\vec{p}^R_{i\circlearrowleft}$ associé à chaque ressource $p_i^R \in P^R$ (voir définition 51). Le respect de la *conservation des ressources* implique le *non-prêt de ressource.*

Remarque 25. *Pour une place de processus $p \in P^P$ et une place de ressource $p^R \in P^R$ avec un P-semiflot associé $\vec{p}^R_{\circlearrowleft}$, la valeur de $\vec{p}^R_{\circlearrowleft}(p)$ peut être interprétée comme le nombre de jetons de p^R devant être consommé pour avoir un jeton dans p.*

Définition 55. *On définit l'ensemble détenteur par l'application $H : P^R \to 2^{P^P}$ mettant en relation une place de ressource et un ensemble de places de processus pour lesquelles la ressource est nécessaire. Pour une place de ressource p^R et son P-semiflot associé $\vec{p}^R_{\circlearrowleft}$, $H(p^R)$ est défini de la manière suivante :*

$$H(p^R) = \{p \in P^P \,|\quad \vec{p}^R_{\circlearrowleft}(p) > 0\}$$

Ressources des S4PR

Définition 56. *Pour chaque ressource d'un S4PR $\mathcal{N} = (P^0 \cup P^P \cup P^R, T, F, W)$, un P-semiflot $\vec{p}^R_{\circlearrowleft}$ est associé à sa place de ressource p^R tel que :*

$$H(p^R) \cap P^0 = \varnothing$$
$$\vec{p}^R_{\circlearrowleft}(p^R) = 1$$

Théorème 7. *Pour une place de ressource p^R d'un S4PR $\mathcal{N}$ qui génère un P-semiflot $\vec{p}^R_{\circlearrowleft}$ les transitions successeurs et prédécesseurs de la place de ressource sont définies comme suit :*

$$p^{R\bullet} = \{t \in T \,|\, \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P) > \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P)\}$$
$$\textit{avec } \forall t \in p^{R\bullet},\quad W(p^R, t) = \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P) - \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P)$$
$${}^\bullet p^R = \{t \in T \,|\, \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P) > \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P)\}$$
$$\textit{avec } \forall t \in {}^\bullet p^R,\quad W(p^R, t) = \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P) - \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P)$$

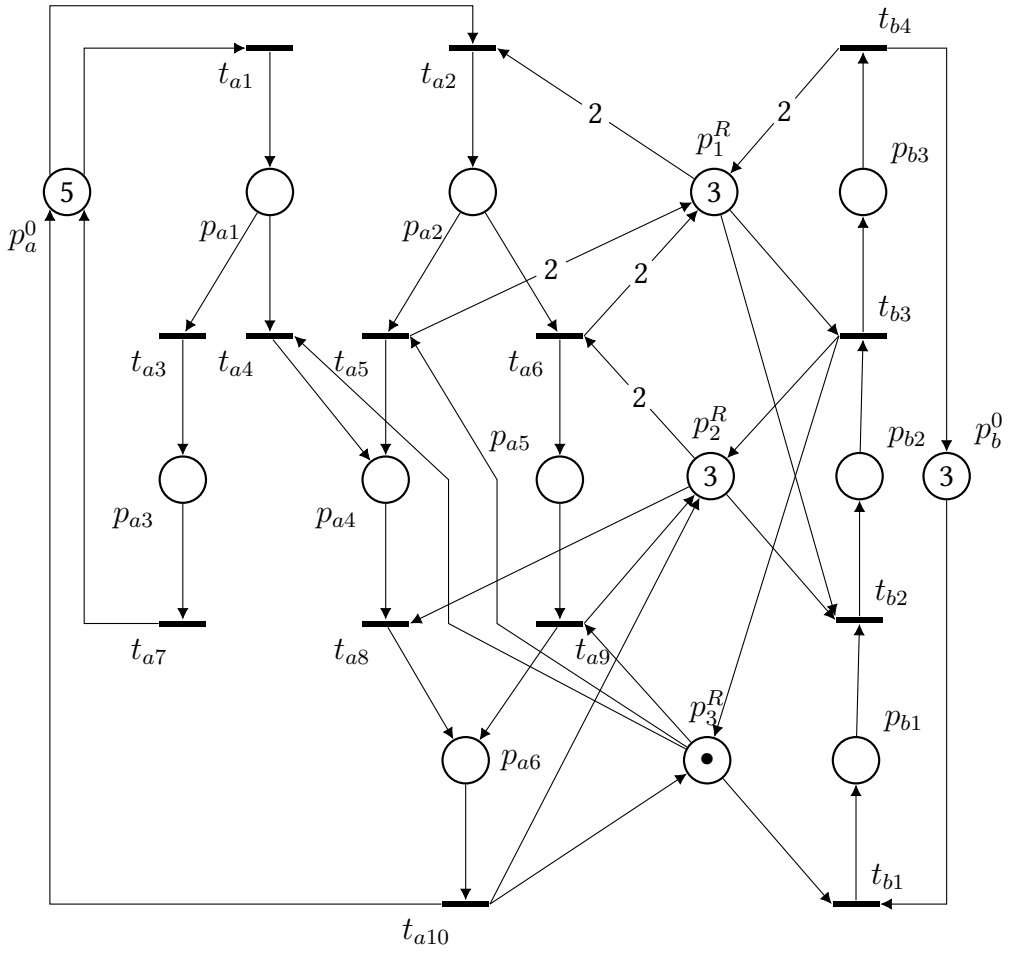


FIGURE 4.9 – Exemple d'un S4PR convenablement marqué $(\mathcal{N}, M_0)$

Démonstration. Les Réseaux de processus sont des machines à états fortement connectées impliquant que pour chaque transition $t \in T, |t^\bullet \cap P^P| = |{}^\bullet t \cap P^P| = 1$. Considérons le tir d'une transition $t \in T$ tel que $M[t\rangle M'$ alors :

$$\forall p \in P^P \quad M'(p) = \begin{cases} \text{si } p \in {}^\bullet t & M'(p) = M(p) - 1 \\ \text{si } p \in t^\bullet & M'(p) = M(p) + 1 \\ \text{sinon} & M'(p) = M(p) \end{cases}$$

L'existence d'un P-semiflot $\vec{p}^R_{\circlearrowleft}$ implique un invariant $M.\vec{p}^R_{\circlearrowleft} = M'.\vec{p}^R_{\circlearrowleft}$. Comme p^R est la seule place de ressources comprise dans l'invariant alors : $M'(p^R) = M(p^R) + \vec{p}^R_{\circlearrowleft}({}^\bullet t) - \vec{p}^R_{\circlearrowleft}(t^\bullet)$. Si $M'(p^R) \neq M(p^R)$, la variation de marquage implique donc l'existence un arc f entre p^R et t, avec $f = (p^R, t)$ si $\vec{p}^R_{\circlearrowleft}({}^\bullet t) - \vec{p}^R_{\circlearrowleft}(t^\bullet) < 0$ et $W(f) = \vec{p}^R_{\circlearrowleft}({}^\bullet t) - \vec{p}^R_{\circlearrowleft}(t^\bullet)$ ou $f = (p^R, t)$ si $\vec{p}^R_{\circlearrowleft}({}^\bullet t) - \vec{p}^R_{\circlearrowleft}(t^\bullet) > 0$ avec $\vec{p}^R_{\circlearrowleft}(t^\bullet) - \vec{p}^R_{\circlearrowleft}({}^\bullet t)$. □

Définition 57. *Le Réseau de ressource $\mathcal{N}_i^R$ d'une place $p_i^R \in P^R$ dans un S4PR $\mathcal{N}$ est défini comme l'ensemble des places et des transitions nécessaires à suivre l'évolution d'une ressource, de sa prise à sa libération. $\mathcal{N}_i^R = (\{p_i^R\} \cup H(p_i^R), T_i^R, F_i^R, W_i^R)$ est un sous-Réseau de $\mathcal{N}$ i.e $\mathcal{N}_i^R \subseteq \mathcal{N}$ avec :*

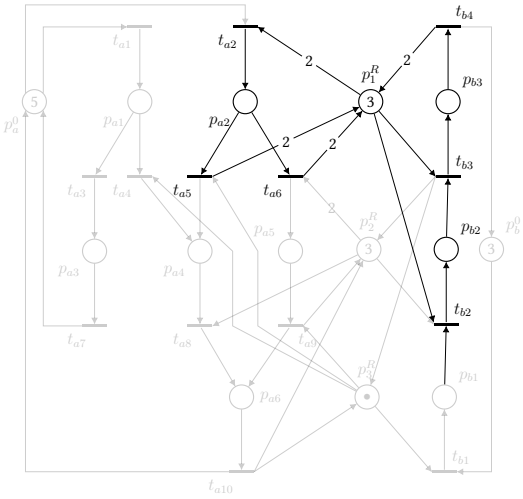


FIGURE 4.10 – Réseau de p_1^R du S4PR $\mathcal{N}$

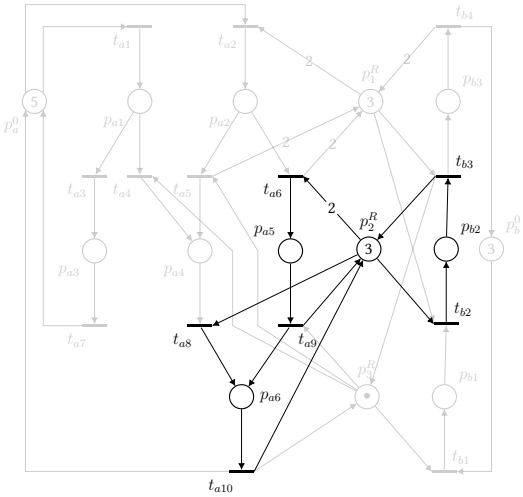


FIGURE 4.11 – Réseau de p_2^R du S4PR $\mathcal{N}$

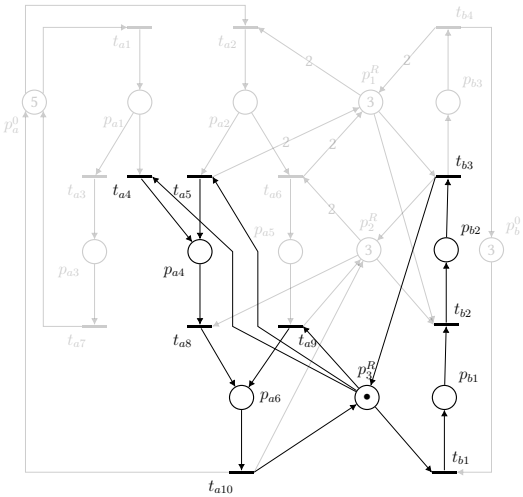


FIGURE 4.12 – Réseau de p_3^R du S4PR $\mathcal{N}$

— $T_i^R = {}^\bullet H(p_i^R) \cup H(p_i^R)^\bullet$ *l'ensemble des transitions qui modifient le marquage de* $H(p_i^R)$
— $F_i^R = \{f = (x,y) \in F \mid (x,y) \in (p_i^R \cup H(p_i^R) \times T_i^R) \cup (T_i^R \times p_i^R \cup H(p_i^R))\}$
— W_i^R *est défini par le théorème 7 dans le cas des arcs liés à la place de ressource, sinon* $W_i^R(f) = 1$ *pour les arcs* f *internes aux processus.*

Définition 58. *Pour un S4PR, le marquage initial des places de ressources est dit convenable si :*

$$\forall p_i^R \in P^R,\ M_0(p_i^R) \geq \max(\vec{p}_{i\circlearrowleft}^R)$$

Remarque 26. *Une place de ressource qui n'est pas convenablement marquée implique qu'une ou plusieurs transitions du Réseau de ressources sont mortes. En effet, s'il existe une place* $p \in P^P$ *et une place de ressource* p^R *telle que* $\vec{p}_{i\circlearrowleft}^R(p) > M_0(p^R)$ *alors* p *nécessite pour avoir un jeton, plus de jetons que la place de ressource ne peut en donner.*

Le Réseau de la figure 4.9 représente un S4PR $\mathcal{N}$ composé de deux processus identifiés par $\mathcal{I} = \{a, b\}$. Le Réseau comporte trois places de ressources $P^R = \{p_1^R, p_2^R, p_3^R\}$. Les Réseaux des ressources $\mathcal{N}_1^R, \mathcal{N}_2^R, \mathcal{N}_3^R$ sont représentés respectivement en figures 4.10, 4.11 et 4.12. Le tableau 4.2 synthétise les informations relatives à chaque ressource. $\mathcal{N}$ est correctement marqué.

Place de processus	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p_{a5}	p_{a6}	p_{b1}	p_{b2}	p_{b3}
$\vec{p}_{1\circlearrowleft}^R$	0	2	0	0	0	0	0	1	2
$H(p_1^R)$	✗	✓	✗	✗	✗	✗	✗	✓	✓
$\vec{p}_{2\circlearrowleft}^R$	0	0	0	0	2	1	0	1	0
$H(p_2^R)$	✗	✗	✗	✗	✓	✓	✗	✓	✗
$\vec{p}_{3\circlearrowleft}^R$	0	0	0	1	0	1	1	1	0
$H(p_3^R)$	✗	✗	✗	✓	✗	✓	✓	✓	✗

TABLE 4.2 – Tableau détaillant les ensembles détenteurs et les P-semiflots associés aux places de ressources p_1^R, p_2^R et p_3^R du S4PR $\mathcal{N}$

Ressources des S3PMR

En plus des restrictions appliquées aux S4PR, les S3PMR sont soumis à la restriction *P-semiflot binaire* (Restriction c). Cette restriction implique que pour chaque place de ressource $p^R \in P^R$, le P-semiflot $\vec{p}_{\circlearrowleft}^R$ associé à p^R ne peut prendre que des valeurs binaires :

$$\vec{p}_{\circlearrowleft}^R : P^R \to \{0, 1\}$$

Cette restriction implique que le P-semiflot $\vec{p}_{i\circlearrowleft}^R$ associé à p_i^R n'apporte pas plus d'informations que l'ensemble détenteur $H(p_i^R)$. Pour les S3PMR, $H(p_i^R)$ est suffisant pour caractériser le Réseau de ressource $\mathcal{N}_i^R$.

Définition 59. *Pour chaque ressource d'un S3PMR $\mathcal{N}$, un P-semiflot $\vec{p}^R_{\circlearrowleft}$ est associé à la place de ressource p^R telle que :*

$$\vec{p}^R_{\circlearrowleft} : P^R \to \{0,1\}$$
$$H(p^R) \cap P^0 = \varnothing$$

Théorème 8. *Pour une place de ressource p^R d'un S3PMR $\mathcal{N}$ avec un ensemble détenteur $H(p^R)$ associé, les transitions entrantes et sortantes de la place de ressource sont définies comme suit :*

$$p^{R\bullet} = {}^\circ H(p^R)$$
$${}^\bullet p^R = H(p^R)^\circ$$

Démonstration. Ce théorème est une simplification du théorème 7. Pour les transitions $t \in p^{R\bullet}$ telles que $\vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P) > \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P)$, cela induit que $\vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P) = 1$ et que $\vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P) = 0$ car $(t^\bullet \cap P^P) \in H(p^R)$ et $({}^\bullet t \cap P^P) \notin H(p^R)$ ce qui implique que $t \in {}^\circ H(p^R)$. Un raisonnement semblable est appliqué aux transitions telles que $\vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P) > \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P)$. □

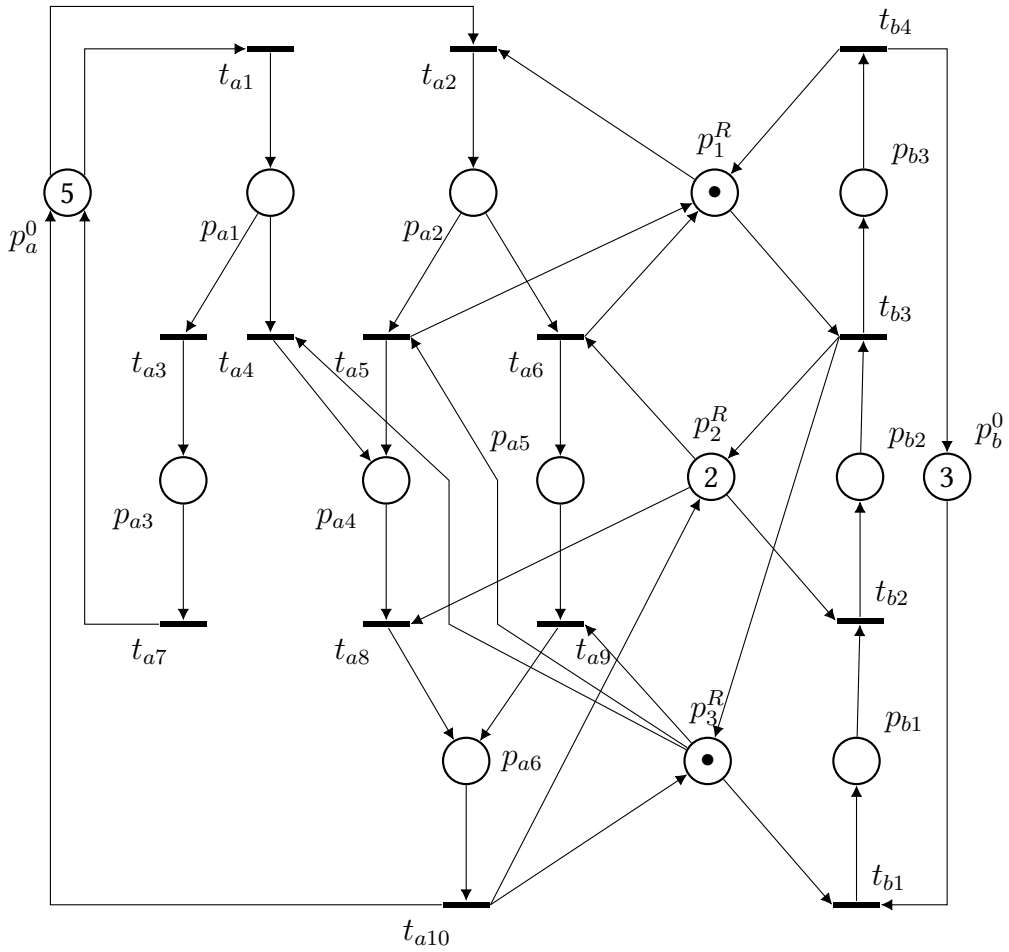


FIGURE 4.13 – Exemple d'un S3PMR convenablement marqué $(\mathcal{N}, M_0)$

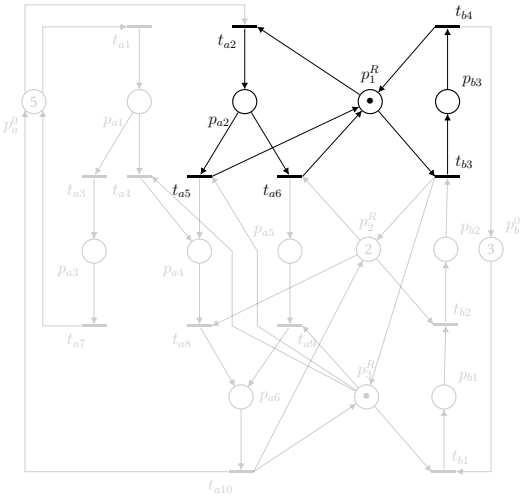


FIGURE 4.14 – Réseau $\mathcal{N}_1^R$ de p_1^R du S3PMR $\mathcal{N}$

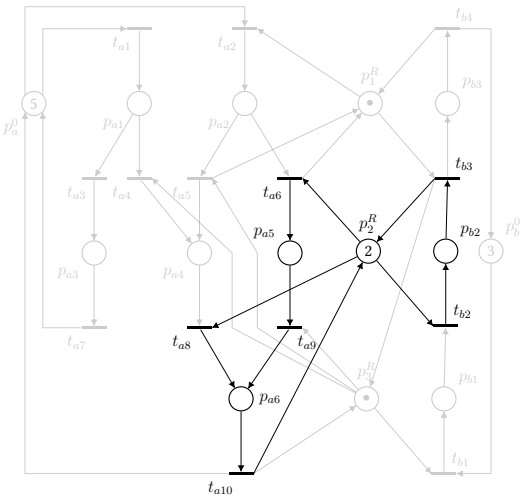


FIGURE 4.15 – Réseau $\mathcal{N}_2^R$ de p_2^R du S3PMR $\mathcal{N}$

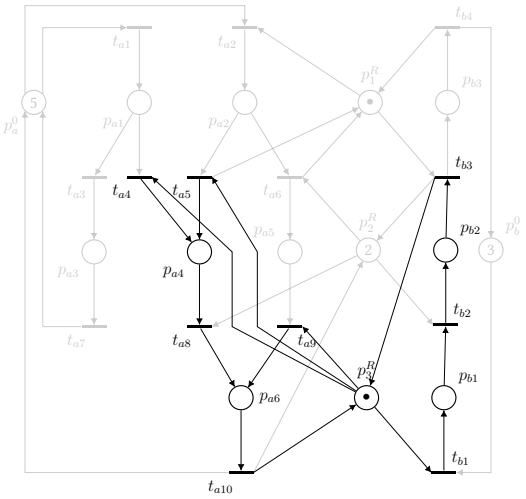


FIGURE 4.16 – Réseau $\mathcal{N}_3^R$ de p_3^R du S3PMR $\mathcal{N}$

Remarque 27. *La restriction P-semiflot binaire (Restriction c) implique forcément un sous-Réseau de ressource ordinaire (voir restriction b). Un S3PMR $\mathcal{N}$ peut donc simplement être défini comme un Réseau de Petri ordinaires $\mathcal{N}_i^R = (\{p_i^R\} \cup H(p_i^R), T_i^R, F_i^R)$.*

Définition 60. *Le Réseau de ressource $\mathcal{N}_i^R$ d'une place $p_i^R \in P^R$ dans un S3PMR $\mathcal{N}_i^R = (\{p_i^R\} \cup H(p_i^R), T_i^R, F_i^R)$ est un sous-Réseau de $\mathcal{N}$ i.e $\mathcal{N}_i^R \subseteq \mathcal{N}$ avec :*

- $T_i^R = {}^\bullet H(p_i^R) \cup H(p_i^R)^\bullet$ *l'ensemble des transitions qui modifient le marquage de $H(p_i^R)$,*
- $F_i^R = \{f = (x, y) \in F \mid (x, y) \in (p_i^R \cup H(p_i^R) \times T_i^R) \cup (T_i^R \times p_i^R \cup H(p_i^R))\}$

Remarque 28. *Le Réseau d'une place de ressource p^R d'un S3PMR est une machine à états fortement connexe dont tous les cycles passent par p^R.*

Définition 61. *Pour un S3PMR, le marquage initial des places de ressource est dit convenable si :*

$$\forall p_i^R \in P^R,\ M_0(p_i^R) \geq 1$$

Remarque 29. *Pour les S3PMR, les places de ressource nécessitent au moins un jeton pour pouvoir tirer les arcs des transitions de $p_i^{R\bullet}$. Pour avoir un jeton dans une place de processus, jamais plus d'un jeton de ressource ne sera consommé (restriction sur les P-semiflots binaires c).*

Le Réseau de la figure 4.13 modélise un S3PMR $\mathcal{N}$ composé de deux processus identifiés par $\mathcal{I} = \{a, b\}$. Le Réseau comporte trois ressources $P^R = \{p_1^R, p_2^R, p_3^R\}$. Les Réseaux des ressources $\mathcal{N}_1^R$, $\mathcal{N}_2^R$, $\mathcal{N}_3^R$ sont représentés respectivement en figures 4.14, 4.15 et 4.16. Le tableau 4.3 synthétise les informations relatives à chaque ressource. $\mathcal{N}$ est correctement marqué.

Place de processus	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p_{a5}	p_{a6}	p_{b1}	p_{b2}	p_{b3}
$H(p_1^R)$	✗	✓	✗	✗	✗	✗	✗	✗	✓
$H(p_2^R)$	✗	✗	✗	✗	✓	✓	✗	✓	✗
$H(p_3^R)$	✗	✗	✗	✓	✗	✓	✓	✓	✗

TABLE 4.3 – Tableau détaillant les ensembles détenteurs associés aux places de ressources p_1^R, p_2^R et p_3^R du S3MPR $\mathcal{N}$

Ressources des S3PR

En plus des restrictions appliquées aux S4PR et aux S3PMR, les S3PR sont soumis à la restriction *P-semiflot orthogonaux* (Restriction d). Cette restriction implique qu'une place de processus $p \in P^P$ ne puisse pas consommer deux ressources différentes. Une place de processus ne peut donc pas faire partie de deux Réseaux de ressources différents.

Une restriction supplémentaire, non répertoriée dans la sous-section 4.1.2, est appliquée aux S3PR. Cette restriction nommée *Réseaux de ressources non-adjacents* implique que deux places de processus ne peuvent pas faire partie du même Réseau de ressource si ces deux places sont adjacentes, i.e :

$$\forall p^R \in P^R, \quad {}^{\bullet\bullet}H(p^R) \cap H(p^R)^{\bullet\bullet} \cap P^P = \varnothing$$

La restriction *Réseau de ressources non-adjacentes* implique nécessairement la restriction *Premier alloué, premier relâché* (voir restriction g)

Définition 62. *Pour chaque ressource r d'un S3PR $\mathcal{N}$, un P-semiflot $\vec{p}^R_{\circlearrowleft}$ est associé à la ressource p^R telle que :*

$$\vec{p}^R_{\circlearrowleft} : P^R \to \{0,1\}$$
$$H(p^R) \cap P^0 = \varnothing$$
$$^{\bullet\bullet}H(p^R) \cap H(p^R)^{\bullet\bullet} \cap P^P = \varnothing$$
$$\forall p^{R\prime} \in P^R \text{ avec } p^{R\prime} \neq p^R \quad H(p^R) \cap H(p^{R\prime}) = \varnothing$$

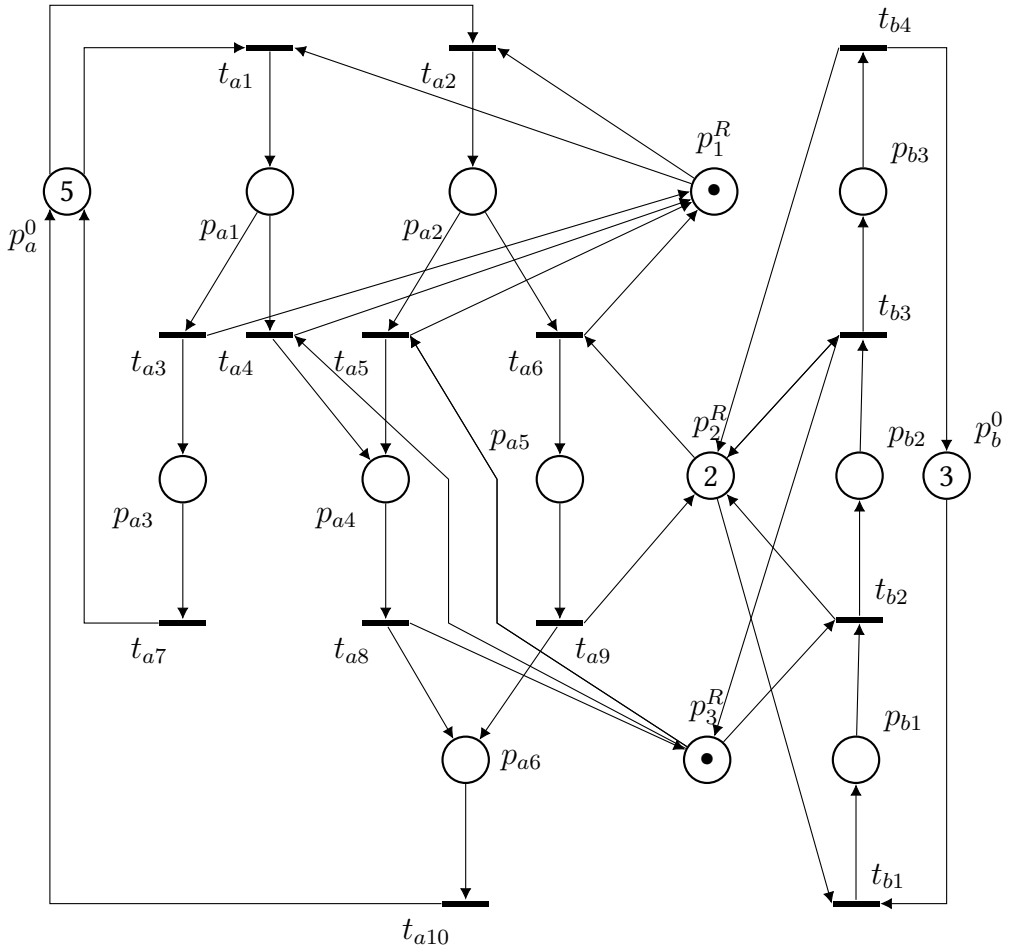


FIGURE 4.17 – Exemple d'un S3PR convenablement marqué $(\mathcal{N}, M_0)$

Théorème 9. *Étant donné une place de ressource p^R d'un S3PR $\mathcal{N}$ et son ensemble détenteur $H(p^R)$ associé, ses transitions entrantes et sortantes sont définies comme suit :*

$$p^{R\bullet} = {}^{\bullet}H(p^R)$$
$$^{\bullet}p^R = H(p^R)^{\bullet}$$

Démonstration. Ce théorème est une simplification du théorème 8. Comme $^{\bullet\bullet}H(p^R) \cap H(p^R)^{\bullet\bullet} \cap P^P = \varnothing$, $H(p^R)^{\bullet} \cap {}^{\bullet}H(p^R) \cap P^P = \varnothing$ donc $H(p^R)^{\circ} = H(p^R)^{\bullet}$ et ${}^{\circ}H(p^R) = {}^{\bullet}H(p^R)$. □

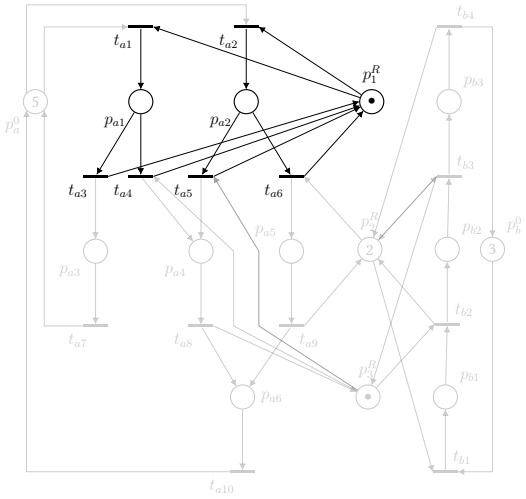


FIGURE 4.18 – Réseau $\mathcal{N}_1^R$ de p_1^R du S3PR $\mathcal{N}$

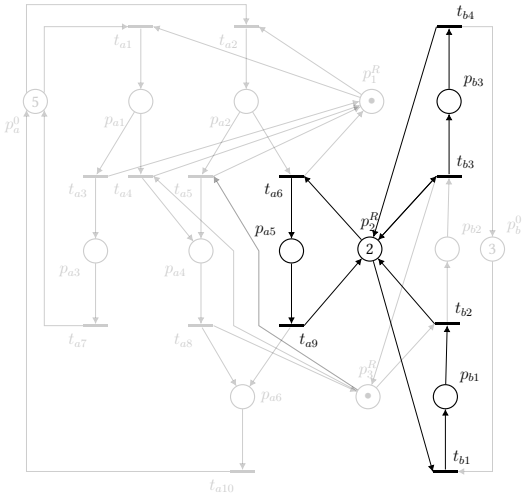


FIGURE 4.19 – Réseau $\mathcal{N}_2^R$ de p_2^R du S3PR $\mathcal{N}$

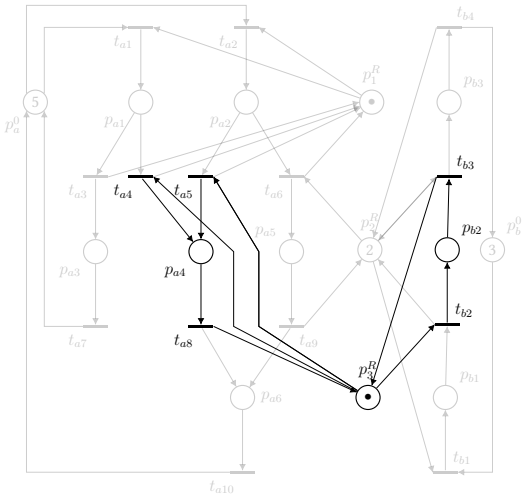


FIGURE 4.20 – Réseau $\mathcal{N}_3^R$ de p_3^R du S3PR $\mathcal{N}$

Définition 63. *Le Réseau de ressource $\mathcal{N}_i^R$ d'une place $p_i^R \in P^R$ dans un S3PR $\mathcal{N}_i^R = (p_i^R \cup H(p_i^R), T_i^R, F_i^R)$ est un sous-Réseau de $\mathcal{N}$ i.e $\mathcal{N}_i^R \subseteq \mathcal{N}$ avec :*

- $T_i^R = {}^\bullet H(p_i^R) \cup H(p_i^R)^\bullet$ *l'ensemble des transitions qui modifient le marquage de $H(p_i^R)$*
- $F_i^R = \{f = (x,y) \in F \mid (x,y) \in (p_i^R \cup H(p_i^R) \times T_i^R) \cup (T_i^R \times p_i^R \cup H(p_i^R))\}$

Définition 64. *Pour un S3PR, le marquage initial des places de ressource est dit convenable si :*

$$\forall p_i^R \in P^R,\ M_0(p_i^R) \geq 1$$

Le Réseau de la figure 4.17 représente un S3PR $\mathcal{N}$, composé de deux processus identifiés par $\mathcal{I} = \{a, b\}$. Le Réseau comporte trois ressources $P^R = \{p_1^R, p_2^R, p_3^R\}$. Les Réseaux des ressources $\mathcal{N}_1^R$, $\mathcal{N}_2^R$, $\mathcal{N}_3^R$ sont représentés respectivement en figures 4.18, 4.19 et 4.20. Le tableau 4.4 synthétise les informations relatives à chaque ressource. $\mathcal{N}$ est correctement marqué.

Place de processus	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p_{a5}	p_{a6}	p_{b1}	p_{b2}	p_{b3}
$H(p_1^R)$	✓	✓	✗	✗	✗	✗	✗	✗	✗
$H(p_2^R)$	✗	✗	✗	✗	✓	✗	✓	✗	✓
$H(p_3^R)$	✗	✗	✗	✓	✗	✗	✗	✓	✗

Table 4.4 – Tableau détaillant les ensembles détenteurs associés aux places de ressources p_1^R, p_2^R et p_3^R du S3PR $\mathcal{N}$

4.3 Analyse de vivacité et propriétés des RAS-PN

Les structures particulières des RAS-PN entraînent des propriétés intéressantes pour le développement de méthodes de contrôle. Ces propriétés sont dites structurelles car elles ne nécessitent pas d'exploration du graphe d'état. La caractérisation des siphons permet de faire un lien entre une particularité structurelle d'un RAS-PN et l'existence d'un état accessible bloquant. Plus les classes de RAS-PN ont une structure complexe, plus la caractérisation des siphons est complexe. Dans cette section nous nous contenterons uniquement de détailler les propriétés des S3PR, S3PMR et S4PR. Les lecteurs peuvent se référer à [LGC12] pour des propriétés sur un nombre plus important de classes de RAS-PN.

4.3.1 Propriétés Structurelles

La structure des processus des S3PR, S3PMR et S4PR étant identique, la propriété suivante est valable pour ces trois classes .

Propriété 21. *Étant donné un S3PR, un S3PMR un ou un S4PR $(\mathcal{N}, M_0)$ convenablement marqué, chaque processus forme un invariant caractérisé comme suit :*

$$\sum_{p \in (\{p_i^0\} \cup P_i^P)} M(p) = M_0(p_i^0) \quad \forall i \in \mathcal{I}, \forall M \in \mathcal{R}(\mathcal{N}, M_0)$$

On note $\vec{p}^P_{i\circlearrowleft}$ avec $i \in \mathcal{I}$ le vecteur générant l'invariant relatif au processus i. On a donc :

$$\forall p \in P, \quad \vec{p}^P_{i\circlearrowleft}(p) = \begin{cases} 1 & \textit{si } p \in \{p_i^0\} \cup P_i^P \\ 0 & \textit{sinon} \end{cases}$$

L'invariant du processus $i \in \mathcal{I}$ peut donc être écrit sous la forme :

$$M^\top.\vec{p}^P_{i\circlearrowleft} = M_0(p_i^0) \quad \forall M \in \mathcal{R}(\mathcal{N}, M_0)$$

Démonstration. Tous les Réseaux de processus sont des machines à états fortement connexes. □

Propriété 22. *Étant donné un S3PR, un S3PMR un ou un S4PR $(\mathcal{N}, M_0)$ convenablement marqué, chaque ressource forme un invariant caractérisé comme suit :*

$$M^\top.\vec{p}^R_{\circlearrowleft} = M_0(p^R) \quad \forall p^R \in P^R, \forall M \in \mathcal{R}(\mathcal{N}, M_0)$$

Propriété 23. *Un S3PR, un S3PMR ou un S4PR $\mathcal{N}$ convenablement marqué est potentiellement-vivant.*

Démonstration. $\mathcal{N}$ est pseudo-vivant si $\forall t \in T, \exists \sigma \in T^* \mid$ et $M_0[\sigma t\rangle$. Autrement dit, pour n'importe quelle transition t il est possible de trouver une séquence de transitions qui depuis M_0 amène au tir de t. Considérons un Réseau de processus $\mathcal{N}_i^P = (\{p_i^0\} \cup P_i^P, T_i, F_i)$ tel que $t \in T_i$ et $i \in \mathcal{I}$ et une séquence de transitions $\sigma = \sigma' t$ avec $\max(\vec{\sigma}) = 1$ (les transitions faisant partie de σ ne sont tirées qu'une seule fois) et $\forall t' \in \sigma$, $t' \in T_i$, $\mathcal{N}_i$ étant une machine à état connexe, σ est réalisable si on considère le processus seul (i.e $\sigma = t_1 t_2 ... t_n$ avec $\forall i \in [1..n-1]$, $t_{i+1} \in t_i^{\bullet\bullet} \cap T_i$). Lors du tir de σ, il y a au maximum un seul jeton dans toutes les places de processus : $\forall \sigma'' \in \bar{\sigma}, M_0[\sigma''\rangle M$ avec $\sum_{p \in P_i^P} M(p) \leq 1$.[5] Or, $\mathcal{N}$ est un Réseau correctement marqué, ce qui implique que les Réseaux de processus doivent toujours pouvoir autoriser d'avoir un jeton dans les places de processus du Réseau de ressource. Aucun tir de transition de σ n'est empêché par un manque de jeton dans une place de ressource, donc σ existe.

□

La distinction des S3PR, S3PMR et S4PR n'est faite que via les restrictions appliquées aux Réseaux de ressources. L'inclusion des classes $S3PR \subseteq S3PMR \subseteq S4PR$ implique que les propriétés s'appliquant aux S4PR s'appliquent aussi aux S3PMR et aux S3PR et, celles des S3PMR s'appliquent aussi aux S3PR.

Lemme 1. *Soit un S4PR $\mathcal{N}$ alors $\mathcal{P}_{\circlearrowleft}(\mathcal{N})$ admet comme base :*

$$\mathcal{B} = (\vec{p}^P_{\alpha\circlearrowleft}, ..., \vec{p}^P_{\omega\circlearrowleft}, \vec{p}^R_{1\circlearrowleft}, ..., \vec{p}^R_{m\circlearrowleft})$$

avec $\{\alpha, ..., \omega\} = \mathcal{I}$, l'ensemble des indices des processus, $m = |P^R|$ le nombre de places de ressources.

5. La notation $\bar{x}$ correspond aux préfixes de x

Démonstration. Les vecteurs $\{\vec{p}^R_{1\circlearrowleft}, ..., \vec{p}^R_{m\circlearrowleft}\}$ font partie de $\mathcal{P}_{\circlearrowleft}(\mathcal{N})$ par définition des places de ressources (définition 51). Les vecteurs $\{\vec{p}^P_{\alpha\circlearrowleft}, ..., \vec{p}^P_{\omega\circlearrowleft}\}$ font aussi partie de $\mathcal{P}_{\circlearrowleft}(\mathcal{N})$ (voir propriété 21). Ces vecteurs sont linéairement indépendants puisque seul le vecteur $\vec{p}^P_{i\circlearrowleft}$ d'un processus i inclut p^0_i, i.e $p^0_i \in \|\vec{p}^P_{i\circlearrowleft}\|$ et que seul le vecteur $\vec{p}^R_{i\circlearrowleft}$ inclut la place de ressource p^R_i, i.e $p^R_i \in \|\vec{p}^R_{i\circlearrowleft}\|$. Il est maintenant nécessaire de prouver qu'il n'existe pas d'autre vecteur dans $\mathcal{B}$. Les combinaisons linéaires des vecteurs de $\mathcal{B}$ constituent des P-flots. Considérons $X, X' \in \mathcal{P}_{\circlearrowleft}(\mathcal{N})$, deux P-flot de $\mathcal{N}$ liés par la relation :

$$X' = X - \sum_{p^0_i \in \|X\|} X(p^0_i).\vec{p}^P_{i\circlearrowleft} - \sum_{p^R_i \in \|X\|} X(p^R_i)\vec{p}^R_{i\circlearrowleft}$$

Le support de X' ne comprend donc aucune place de ressource, ni de place de repos i.e $\|X'\| \subseteq P^P$. Si on considère uniquement les places d'un processus $i \in \mathcal{I}$ les places comprises dans $\|X'\| \cap P^P_i$ ont forcément une transition source, i.e. ${}^\circ(\|X'\| \cap P^P_i) \neq \phi$, car tous les cycles des processus passent par les places de repos. Une transition source "pure" (i.e. sans place en amont) dans une machine à états empêche la possibilité d'un invariant, $X' \notin \mathcal{P}_{\circlearrowleft}(\mathcal{N})$ donc $X \notin \mathcal{B}$, $\mathcal{B}$ est donc complète comme définie. □

Propriété 24. *[LGC12] Un S3PR, S3PMR ou S4PR $\mathcal{N}$ est vivant si et seulement si M_0 est un état d'accueil (i.e. le système est réversible).*

Démonstration.

$\Longrightarrow$ Nous allons pour ce sens démontrer la contraposée. Supposons que M_0 ne soit pas un état d'accueil i.e. $\exists M' \in \mathcal{R}(\mathcal{N}, M_0)$ tel que $M_0 \notin \mathcal{R}(\mathcal{N}, M')$. Soit M un état atteint depuis M' obtenu en avançant au maximum les processus actifs (sans tir de transition sources $P^{0\bullet}$) jusqu'à ce qu'il ne soit plus possible de tirer de transitions. Donc le fait que M_0 ne soit pas un état d'accueil entraîne un état bloquant, donc le Réseau n'est pas vivant.

$\Longleftarrow$ Si M_0 est un état d'accueil, comme d'après la propriété 23, le Réseau est pseudo-vivant, cela signifie que $\forall M \in \mathcal{R}(\mathcal{N}, M_0)$, $\forall t \in T$, il existe $\sigma \in T^*|M[\sigma.t\rangle$. Donc cela entraîne que le Réseau est vivant.

□

4.3.2 Caractérisation des siphons

Définition 65. *On appelle ensemble des transitions M-processus-sensibilisées, $T^P_{[\rangle}(M)$: $\mathbb{N}^{|P|} \to T$, les transitions dont le tir est autorisé pour un marquage M donné du point de vue des processus :*

$$T^P_{[\rangle}(M) = \{t \in T \mid \forall p \in ({}^\bullet t \cap P^P),\ M(p) \geq W(p,t)\}$$

A l'inverse, les transitions M-processus-désensibilisées sont les transitions $T^P_{\not[\rangle}(M)$ dont le marquage des processus ne permet pas le tir :

$$T^P_{\not[\rangle}(M) = \{t \in T \mid \exists p \in ({}^\bullet t \cap P^P,\ M(p) < W(p,t)\}$$

Définition 66. *On appelle ensemble des transitions M-ressources-sensibilisées, $T^R_{[\rangle}(M)$: $\mathbb{N}^{|P|} \to T$, les transitions dont le tir est autorisé pour un marquage M donné uniquement vis-à-vis du marquage des ressources :*

$$T^R_{[\rangle}(M) = \{t \in T \mid \forall p \in ({}^\bullet t \cap P^R),\ M(p) \geq W(p,t)\}$$

A l'inverse, les transitions M-ressources-désensibilisées sont les transitions $T^R_{[\not\rangle}(M)$ dont le marquage des ressources ne permet pas le tir :

$$T^R_{[\not\rangle}(M) = \{t \in T \mid \exists p \in ({}^\bullet t \cap P^R),\ M(p) < W(p,t)\}$$

Lemme 2. *Considérons un S4PR convenablement marqué $(\mathcal{N}, M_0)$. Si $M, M' \in \mathcal{R}(\mathcal{N}, M_0)$ tel que $\forall p \in P^P$, $M'(p) \geq M(p)$ alors, $\forall p^R \in P^R$, $M'(p^R) \leq M(p^R)$*

Démonstration. Directement obtenue de la propriété d'invariant de ressources (propriété 22). □

Théorème 10. *[Tri03] Étant donné un S4PR convenablement marqué $(\mathcal{N}, M_0)$. $(\mathcal{N}, M_0)$ n'est pas vivant si et seulement si il existe un marquage accessible tel que l'ensemble des transitions M-processus-sensibilisées soit non-vide et soit inclus dans l'ensemble des transitions M-ressource-désensibilisées :*

$$\exists M \in \mathcal{R}(\mathcal{N}, M_0) \mid T^P_{[\rangle}(M) \neq \varnothing \wedge T^P_{[\rangle}(M) \cap T^R_{[\rangle}(M) = \varnothing$$

Démonstration.

$\Longrightarrow$ Pour M seulement, les transitions de $P^{0\bullet}$ sont potentiellement tirables. Nous allons prouver par induction que peu importe une séquence de transition $\sigma \in T^*$ l'état $M[\sigma\rangle M''$ aura les conditions suivantes :

$$(||M|| \cap P^P)^\bullet \cap ||\sigma|| = \varnothing$$
$$\forall p \in (||M|| \cap P^P), \quad M''(p) \geq M(p)$$

Autrement dit, les transitions bloquées à M ne peuvent être tirées au cours de σ et le marquage des places de processus de M ne peut pas diminuer.

- cas $\sigma = t : t$ est forcément une transition source $t \in P^{0\bullet}$. On identifie deux cas : si $t \in {}^\bullet(||M|| \cap P^P)$ dans ce cas $t \notin (||M|| \cap P^P)^\bullet$ et $M''(t^\bullet \cap P^P) = M(t^\bullet \cap P^P) + 1$ et le marquage de toutes les autres places de processus est inchangé entre M'' et M. Si $t \notin {}^\bullet(||M|| \cap P^P)$ dans ce cas $t \notin (||M|| \cap P^P)^\bullet$ et $M''(t^\bullet \cap P^P) = 1$. Dans les deux cas les deux conditions sont maintenues.
- cas $M[\sigma\rangle M'''[t\rangle M''$: par induction M''' vérifie les deux conditions. M''' vérifie donc que $\forall p \in (||M|| \cap P^P), M'''(p) \geq M(p)$, en appliquant le lemme 2 et en prenant en compte que $\forall p \in P^P, M'''(p) \geq M(p)$, t ne peut être dans les transitions bloquées à M i.e $t \notin (||M|| \cap P^P)^\bullet$. Donc le marquage de $(||M|| \cap P^P)$ ne diminue pas.

$\Longleftarrow$ $(\mathcal{N}, M_0)$ n'est pas vivant s'il existe un marquage accessible qui, une fois atteint, ne permet plus le tir d'au moins une transition. Si on considère un état

$M' \in \mathcal{R}(\mathcal{N}, M_0)$ à partir duquel une transition t est morte, M est le marquage obtenu si on tire toutes les transitions dans $T \backslash P^{0\bullet}$ jusqu'à ce que les jetons présents dans les processus rejoignent si possible les places de repos. Si $M_0 = M$ alors t n'est pas morte en M' car $\mathcal{R}(\mathcal{N}, M_0)$ est potentiellement-vivant par définition, ce qui contredit notre hypothèse. Donc à M, des places de processus $\|M\| \cap P^P$ sont non-vides et $(\|M\| \cap P^P)^\bullet = T^R_{[\rangle}(M)$ et $T^P_{[\rangle}(M) \cap T^R_{[\rangle}(M) = \varnothing$, ce qui conclut la preuve.

□

Théorème 11. *[Tri03] Étant donné un S4PR convenablement marqué $(\mathcal{N}, M_0)$. $(\mathcal{N}, M_0)$ n'est pas vivant si et seulement si $\exists M \in \mathcal{R}(\mathcal{N}, M_0)$ et un siphon S tels que :*

1. *Il existe au moins une transition sensibilisée par une place de processus : $t \in {}^\bullet S$ tel que $\{p\} = {}^\bullet t \cap P^P$ et $M(p) \geq 1$,*
2. *Les places de ressource de S désensibilisent toutes les transitions de $T^P_{[\rangle}(M)$: $\forall t \in T^P_{[\rangle}(M),\ \exists p^R \in P^R \cap S \,|\, W(p^R, t) > M(p^R)$,*
3. *Les places de processus sont vides pour M : $\forall p \in S \cap P^P,\ M(p) = 0$.*

Démonstration.

$\Longrightarrow$ Selon le théorème 10, $(\mathcal{N}, M_0)$ n'est pas vivant si et seulement s'il existe un marquage accessible $M \in \mathcal{R}(\mathcal{N}, M_0)$ tel que toutes les transitions M-processus-sensibilisées soient aussi des transitions M-ressource-désensibilisées. On notera S^R les places de ressource de S, i.e. $S^R = S \cap P^R$ et S^P les places de processus de S, i.e $S^P = S \cap P^P$. S se construit comme suit :

- Soit $S^R = \{p^R \in P^R | \exists t \in p^{R\bullet}$ avec $M(p^R) < W(r,t) \wedge M(p) > 0$ avec $\{p\} = {}^\bullet t \cap P^P\}$ $S^R \neq \varnothing$ car il existe au moins une transition M-processus-sensibilisée.
- Soit $S^P = \{p \in \bigcup_{p^R \in S^R} H(p^R) \,|\, M(p) = 0\}$, l'ensemble des places détentrices de S^R non marquées par M.

Nous allons prouver que $S^P \neq \varnothing$ et $S^P \subset \bigcup_{p^R \in S^R} H(p^R)$. Supposons que $S^P = \varnothing$. Considérons un processus de $\mathcal{N}^P_i$ avec $i \in \mathcal{I}$ du S4PR et un chemin $\lambda \in \Lambda(\mathcal{N}^P_i)$ avec $\lambda = p^0_i, t_1, p_1, ..., t_k, p^0_i$ un cycle démarrant de la place de repos p^0_i et y revenant. On note $t_l \in T_i$ avec $l \in [1..k]$, la dernière transition du chemin λ telle qu'il existe une ressource avec $p^R \in S^R$ avec $t_l \in {}^\bullet p^R$. Dit de manière informelle, un jeton parcourant le chemin λ consommera plus de ressources de S^R, une fois passé t_l.

Cela implique que $\exists p^R \in S^R$ telle que ${}^\bullet t_l \cap P^P \in H(p^R)$. Comme t_l est la dernière transition de λ tel que $t_l \in {}^\bullet S^R$, cela implique que $t_l \notin S^{R\bullet}$ (les places de ressources n'ayant pas d'arcs aller-retours sur les transitions). Ainsi t_l n'est pas une transition M-ressource-désensibilisée par S^R. Or, par définition de M, si $t_l \notin T^R_{[\rangle} \implies t_l \notin T^P_{[\rangle}$. La place $\{p\} = {}^\bullet t_l \cap P^P$ appartient bien à S^P car $p \in H(S^R)$ et $M(p) = 0$.

Il est maintenant nécessaire de prouver que S est bien un siphon. Pour chaque transition $t \in {}^\bullet S$ on prouvera que $t \in S^\bullet$. Deux cas doivent être vérifiés :

- $t \in {}^\bullet S^R$ avec $p^R \in S^R$ tel que $t \in {}^\bullet p^R$. Il existe $p \in H(p^R) \cap {}^\bullet t$

- Si $M(p) = 0$ alors $p \in S^P$ donc $t \in S^\bullet$
- Si $M(p) > 0$ comme t n'est pas sensibilisée il existe une place $p^{R'} \in P^R \cap {}^\bullet t$ telle que $M(p^{R'}) < W(p^{R'}, t)$. Donc $p^{R'} \in S^R$ donc $t \in S^{R\bullet}$.

— $t \notin {}^\bullet S^R$. Alors, il existe $p \in S^P$ tel que $t \in {}^\bullet p$ et $\exists p^{R'} \in S^R$ tel que $p \in H(p^{R'})$. Si $\exists p^R \in ({}^\bullet t \cap S^R)$, $t \in S^\bullet$.

On suppose maintenant que ${}^\bullet t \cap S^R = \varnothing$. Dans ce cas, t ne peut pas être M-processus sensibilisé. Si c'est le cas, par le théorème 10, t doit être M-ressource-désensibilisée et implique l'existence de $p^R \in {}^\bullet t \cap S^R$.

Soit $\{q\} = {}^\bullet t \cap P^P$ (cette place existe car $p \in H(p^{R'})$ et ${}^\bullet t \cap S^R = \varnothing$). Comme t n'est pas une transition M-processus-sensibilisée, $M(q) = 0$.

De plus, comme $p \in H(p^{R'})$ appartient au P-semiflot $\vec{p}^R_{\circlearrowleft}{}'$, cela implique que $q \in H(p^{R\prime})$. Par conséquent $q \in S^R$ (q n'est pas marqué), et $t \in S^{P\bullet}$.

On remarque donc que le tir de chaque transition M-processus-sensibilisée est empêché par une place de ressource appartenant à S.

$\Longleftarrow$ Si chaque transition M-processus-sensibilisée est empêchée par un ensemble de places de ressources (appartenant à S), le théorème 10 nous permet de conclure.

□

Définition 67. *[TGVCE05] Soit un S4PR convenablement marqué $(\mathcal{N}, M_0)$ et un siphon $S \in \Pi(\mathcal{N})$. $M \in \mathcal{R}(\mathcal{N}, M_0)$ est dit un mauvais marquage de S si :*

— $S \cap P^R = \{p^R \in P^R \cap S \mid \exists t \in p^{R\bullet}$ *avec* $M(p^R) < W(p^R, t) \wedge M({}^\bullet t \cap P^P) > 0\}$

— $S \cap P^P = \{p \in P^P \cap S \mid \exists p^R \in S \cap P^R$ *avec* $p \in H(p^R) \wedge M(p) = 0\}$

Avec $S \cap P^R \neq \varnothing$ et $S \cap P^P \neq \varnothing$. Cette définition correspond au marquage du théorème 11.

Propriété 25. *[LGC12] Il existe un S4PR non-vivant, convenablement marqué, tel que chaque siphon caractérisant la non-vivacité est non-minimal. C'est-à-dire que les siphons minimaux ne sont pas suffisants pour caractériser la vivacité d'un S4PR.*

Théorème 12. *[LGC12]*

Dans le cas d'un S3PMR (ou S3PR) convenablement marqué $(\mathcal{N}, M_0)$. $(\mathcal{N}, M_0)$ est non-vivant si et seulement si $\exists M \in \mathcal{R}(\mathcal{N}, M_0)$ et un siphon <u>minimal</u> S tel que $M(S) = 0$.

Démonstration. Cette preuve se base sur le théorème 11. Un S3PMR étant un S4PR, si $\mathcal{N}$ est non-vivant, il existe un siphon et un marquage M pour lesquels un ensemble de transitions sont mortes. Les S3PMR étant des Réseaux de Petri ordinaires, les arcs ne sont pas pondérés. Aussi, les transitions mortes telles que $\forall t \in T^P_{\not\circlearrowleft}(M) \cap T^R_{\not\circlearrowleft}(M)$ impliquent que le marquage des places de ressource est vide, i.e. $\forall p \in ({}^\bullet t \cap S^R)$, $M(p) = 0$. On a alors $M(S) = 0$. Si S est minimal, alors le théorème est correct. Sinon, alors il existe un siphon $S' \subset S$ avec $M(S') = 0$, ce qui conclut la preuve. □

La figure 4.21 montre un S4PR $\mathcal{N}$ composé de deux Réseaux de processus $\mathcal{I} = \{a, b\}$ et de deux places de ressource $\{p^R_1, p^R_2\} = P^R$. Les P-semiflots générés par les places de ressource sont explicités dans le tableau 4.5. La figure 4.21 représente le

S4PR $\mathcal{N}$ dans un état M' où les transitions t_{a3}, t_{a4}, t_{a5}, t_{b1} et t_{b2} sont mortes. Cet état correspond à l'état M' de la preuve du théorème 10. Le marquage initial M_0 de $\mathcal{N}$ est donné par la figure 4.22). On atteint M' par la séquence de transitions $\sigma = t_{b1}, t_{b2}, t_{b3}, t_{a1}, t_{a3}, t_{b1}$. Dans l'état M', même si les transitions t_{a5} et t_{b2} sont mortes, il est toujours possible de tirer des transitions M'-processus-sensibilisées. On applique la stratégie qui consiste à "vider" toutes les places de processus de leurs jetons. Après le tir de t_{b4}, on obtient le marquage M i.e $M'[t_{b4}\rangle M$. Dans l'état M, à part les transitions sources, il n'est plus possible de tirer aucune transition. M est un état bloquant qui implique la non-vivacité du Réseau $\mathcal{N}$ selon le théorème 10. Les places de processus marquées à M ne peuvent plus se vider de leurs jetons et, les transitions successeures de ces places sont donc mortes. Ce marquage implique l'existence d'un siphon S. A M les transitions t_{a5} et t_{b2} sont M-processus sensibilisées et M-ressources-désensibilisées, selon la preuve du théorème 11, $S = S^R \cup S^P$ avec :

$$S^R = \{p^R \in P^R \mid \exists t \in p^{R\bullet} \text{ et } M \in \mathcal{R}(\mathcal{N}, M_0) \text{ avec } t \in T^P_{[\rangle}(M)\} = \{p^R_1, p^R_2\}$$
$$S^P = \{p \in \bigcup_{p^R \in S^R} H(p^R) \mid M(p^R) = 0\} = \{p_{a1}, p_{a4}, p_{b2}, p_{b3}\}$$

Comme le marquage M entraîne la mort des transitions $t_{a3}, t_{a4}, t_{a5}, t_{b1}$ et t_{b2}, S n'est pas max-marqué en M mais n'est pas cependant min-marqué puisque $M(p^R_1) = 1$. On dit alors que S est insuffisamment marqué par M.

	p^0_a	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p^0_b	p_{b1}	p_{b2}	p_{b3}	p^R_1	p^R_2
$\vec{p}^R_{\circlearrowleft 1}$	0	1	0	2	0	0	0	2	1	1	0
$\vec{p}^R_{\circlearrowleft 2}$	0	0	0	1	2	0	1	1	0	0	1
S	✗	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓
M_0	4	0	0	0	0	3	0	0	0	3	2
M	3	0	0	1	0	2	1	0	0	1	0
M'	3	0	0	1	0	1	1	0	1	0	0

TABLE 4.5 – Informations relatives au S4PR $\mathcal{N}$ et aux états M_0, M et M'

Propriété 26. *Dans un S3PMR, un Strict Minimal Siphon (SMS) comporte au moins deux places de ressources.*

Démonstration. Considérons un SMS S non vide.

Si S ne comporte pas de places de ressources ($S \cap P^R = \varnothing$) alors S est forcément composé d'un ou plusieurs processus complets. En effet, si une place $p \in P^P_i \cap \{p^0_i\}$ d'un processus $i \in \mathcal{I}$ appartient à S alors nécessairement ses places prédécesseures $^{\bullet\bullet}S$, sont nécessairement incluses dans S puisque $^\circ S = \varnothing$ et $|^\bullet t \cap P^P_i|$. Comme les Réseaux de processus sont fortement connexes, l'opération recommence jusqu'à ce que toutes les places de $P^P_i \cap p^0_i$ finissent par être incluses dans S. S étant une machine à états fortement connexe, S n'est pas un siphon strict.

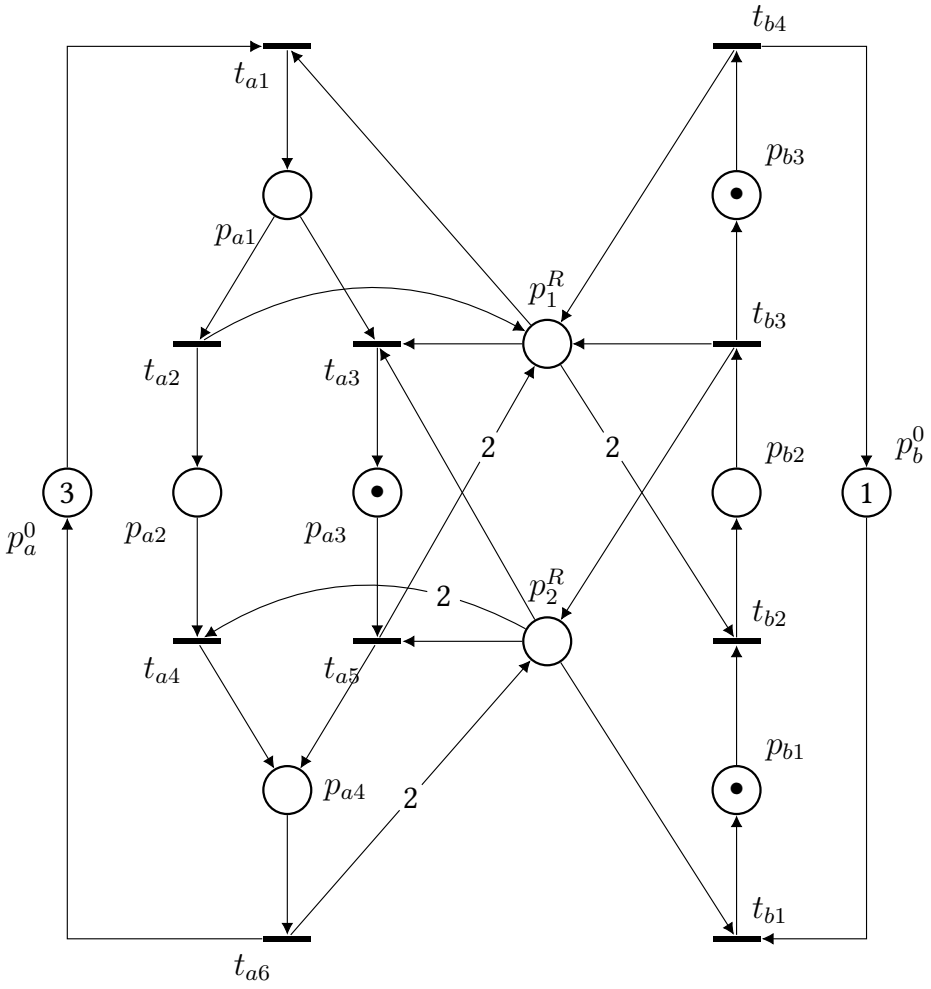


FIGURE 4.21 – Exemple d'un S4PR $\mathcal{N}$ avec un marquage M' entraînant des transitions mortes

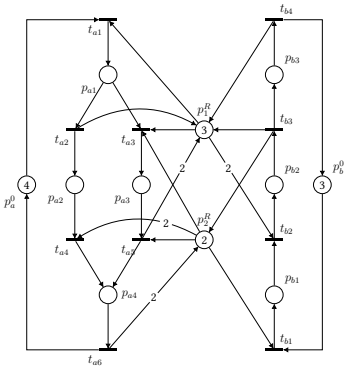


FIGURE 4.22 – S4PR $\mathcal{N}$ dans l'état M_0

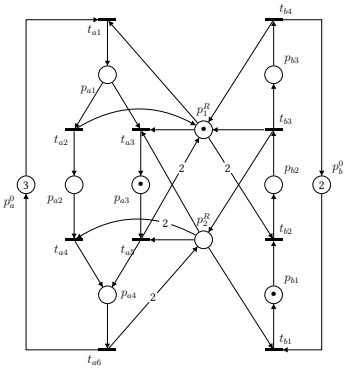


FIGURE 4.23 – S4PR $\mathcal{N}$ dans l'état M

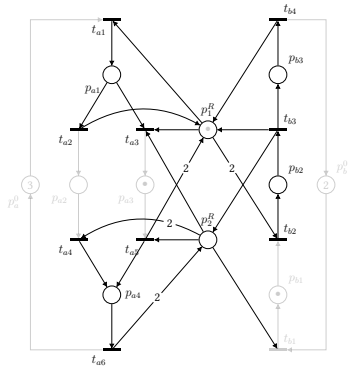


FIGURE 4.24 – Siphon S empêchant le tir de transitions

Si S comporte une seule place de ressources telle que $S \cap P^R = \{p^r\}$, alors S peut être construit de manière récursive comme suit :

$$\begin{aligned} S_0 &= ({}^{\bullet\bullet}p_r \cap P^P) \cup \{p^r\} \\ S_i &= S_{i-1} \cup ({}^{\bullet\bullet}S \cap P^P) \end{aligned}$$

On définit S comme étant l'ensemble convergent de S_i (tel que $S_i = S_{i-1}$) qui garantit ainsi que S est un siphon comportant p^r. Ainsi construit, S correspond aux places du Réseau de ressource de p^r. En effet, les places précédentes du processus sont ajoutées jusqu'à arriver à $p^{r\bullet\bullet}$. On a donc $S = H(p^r) \cap \{p^r\}$. Le Réseau de ressource de p^r étant un Réseau fortement connexe, il n'existe pas de transition $t \in S^\circ$. S n'est donc pas un siphon strict, ce qui conclut la preuve. □

Remarque 30. *La propriété 26 ne s'applique pas aux S4PR. En effet, l'exemple de la figure 4.26 montre un S4PR $\mathcal{N}$, dans un état initial convenable M_0. La figure 4.26 montre un SMS $S = \{p_{a2}, p_{b2}, p^r\}$ dans $\mathcal{N}$ ne comportant qu'une ressource et dans un état accessible $M \in \mathcal{R}(\mathcal{N}, M_0)$ bloquant.*

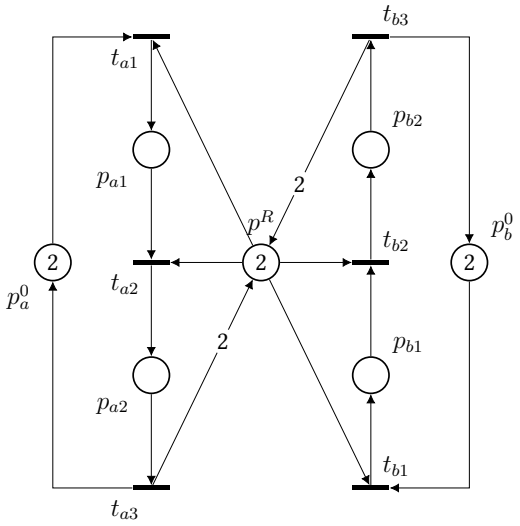


FIGURE 4.25 – Un S4PR $\mathcal{N}$ convenablement marqué M_0

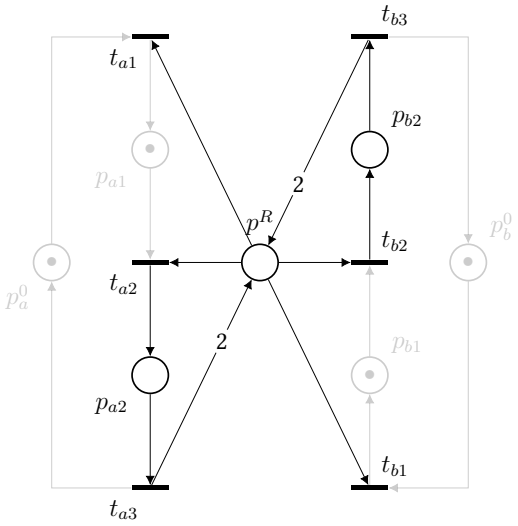


FIGURE 4.26 – SMS S dans $\mathcal{N}$ comportant une seule ressource

4.4 Contrôle des RAS-PN

Lorsque qu'un siphon pouvant entrainer une situation de blocage a été identifié, une stratégie de contrôle peut être mise en place de manière à ce que des mauvais marquages associés à ce siphon ne puissent plus être atteints.

4.4.1 Définitions et propriétés liées au contrôle des siphons

Définition 68. *On appelle $[S]$, l'ensemble complémentaire d'un siphon, les places qui consomment les jetons d'un siphon S. Avec $[S]$ défini par :*

$$[S] = \bigcup_{p^r \in P^R \cap S} H(p^r) \backslash S$$

Remarque 31. *Dans la partie 3.3.1 ont été présentées les conditions de contrôle d'un siphon. La définition 33 propose une solution pour le contrôle des siphons visant à fixer un nombre minimal de jetons dans le siphon. Dans le cas des méthodes de contrôle pour les RAS-PN, on cherche généralement les ensembles de places qui viennent "voler" les jetons du siphon, pour limiter le nombre maximal de jetons dans ces places. L'ensemble complémentaire dans le cas des S3PR, S3PMR et S4PR sont ces ensembles de places qui viennent "voler" des jetons au siphon.*

Définition 69. *On appelle $k_{[S]} : [S] \to \mathbb{N}^+$, l'application qui associe à chaque place de $[S]$ le nombre de jetons de ressources à consommer pour obtenir un jeton dans cette place. Ainsi $k_{[S]}(p)$ avec $p \in [S]$ est définie comme :*

$$k_{[S]}(p) = \sum_{p_i^R \in P^R \cap S | p \in H(p_i^R)} \vec{p}_{i\circlearrowleft}^R(p)$$

La notion de siphon max'-contrôlé apparait pour la première fois dans [Cha07] pour le développement d'une méthode de contrôle des S4PR (appelé S3PGR2 dans l'article). Cette notion est ensuite formalisée dans [ZL10] pour les WS3PR (Weighted Systems of Simple Sequential process) et pour les S4PR (appelés S4R dans l'article). La notion de siphon max'-contrôlé permet une description plus fine des conditions de contrôle d'un siphon que celle de siphon max-contrôlé.

Définition 70. *Dans un S4PR $\mathcal{N}$ s'il existe un siphon $S \in \Pi(\mathcal{N})$ alors il est possible de définir le Réseau du siphon $\mathcal{N}_S$ comme un sous-Réseau de $\mathcal{N}$ i.e $\mathcal{N}_S \subseteq \mathcal{N}$ et $\mathcal{N}_S = (S, S^\bullet, F_S, W_S)$.*

Définition 71. *Étant donné un S4PR $(\mathcal{N}, M_0)$ convenablement marqué et un siphon $S \in \Pi(\mathcal{N})$, un état $M \in \mathcal{R}(\mathcal{N}, M_0)$ est dit M-max'-marqué si :*

$$\begin{array}{lll} \exists p \in S \cap P^P & \textit{tel que} & M(p) \geq 1 \\ & \textit{ou} & \\ \exists p^R \in S \cap P^R & \textit{tel que} & M(p^R) \geq \max\limits_{t \in (p^{R\bullet} \cap [S]^\bullet)} W(p^R, t) \end{array}$$

De manière informelle la notion de Max'-contrôle divise le problème en deux sous-parties :

- La partie Réseau de Petri Ordinaire : Elle concerne les places de processus. Ces Réseaux étant ordinaires, la présence d'un jeton dans une place de processus garantit que l'état n'est pas bloquant.
- La partie Réseau de Petri Généralisé : Pour un siphon S dans un état de blocage, les jetons sont "coincés" dans les places de $[S]$. C'est donc les transitions

de $[S]^\bullet$ dont il faut s'assurer de la vivacité. [Cha07] prouve qu'il est uniquement nécessaire d'avoir une seule place de ressource maximalement contrôlée pour s'assurer qu'un siphon ne soit pas bloquant dans un état M.

Définition 72. *Dans un S4PR $(\mathcal{N}, M_0)$ convenablement marqué, on considère un siphon $S \in \Pi(\mathcal{N})$. S est dit max'-contrôlé si S est M-max'-marqué pour tout $M \in \mathcal{R}(\mathcal{N}, M_0)$. Lorsque tous les siphons de $\mathcal{N}$ sont max'-contrôlés on dit alors que $(\mathcal{N}, M_0)$ est max'-contrôlé.*

Théorème 13. *[ZL10] Étant donné un S4PR $(\mathcal{N}, M_0)$ convenablement marqué et, $t \in T$ une transition morte dans un état $M \in \mathcal{R}(\mathcal{N}, M_0)$, alors il existe un état $M' \in \mathcal{R}(\mathcal{N}, M)$ et un siphon $S \in \Pi(\mathcal{N})$ tels que S n'est pas M'-max'-marqué.*

De manière informelle, ce théorème se base sur le théorème 10 qui stipule qu'un S4PR $\mathcal{N}$ est non-vivant si et seulement s'il existe un marquage M ou toutes les transitions M-processus-sensibilisées sont aussi des transitions M-ressources-désensibilisés. Dans ce marquage, toutes les places de processus marquées garderont leurs jetons pour tous marquages de $\mathcal{R}(\mathcal{N}, M)$. Ce marquage induit l'existence d'un siphon S selon le théorème 11. À M les places de processus du siphon ($M(P^P \cap S) = 0$) ne sont pas marquées alors $\nexists p \in S \cap P^P$ tel que $M(p) \geq 1$. Selon la définition 67, une place de ressource est incluse dans un siphon seulement si elle empêche le tir d'une transition M-processus-sensibilisée, ce qui est impliqué par (mais n'est pas équivalent à) la relation suivante $\nexists p^R \in S \cap P^R$ tel que $M(p^R) \geq \max_{t \in (p^{R\bullet} \cap [S]^\bullet)} W(p^R, t)$.

Théorème 14. *[Cha07] Un S4PR $(\mathcal{N}, M_0)$ convenablement marqué est vivant si chaque siphon $S \in \Pi(\mathcal{N})$ est max'-contrôlé.*

Le théorème 14, permet une caractérisation des siphons potentiellement bloquants, plus fine que celles des siphons max-cs (voir propriété 14), qui s'applique aux Réseaux de Petri Généralisés. Cependant, la condition de max'-contrôle n'est qu'une condition suffisante. Il est possible d'avoir un S4PR qui ne soit pas max'-contrôlé et qui soit cependant vivant.

Si on prend l'exemple, (présenté dans [LLZ10]) de la figure 4.27 représentant un S4PR $(\mathcal{N}, M_0)$ convenablement marqué et **vivant**. À l'état initial, seules les places de repos et les places de ressources sont marquées telles que $M_0(p_a^0) = 5, M_0(p_b^0) = 5, M_0(p_c^0) = 5, M_0(p_1^R) = 3$ et $M_0(p_2^R) = 3$. L'état M est obtenu par la séquence $\sigma = t_{a1}t_{b1}t_{c1}$ i.e $M_0[\sigma\rangle M$ et est représenté par la figure 4.27. $\mathcal{N}$ comporte un siphon $S = \{p_{a2}, p_{b2}, p_{c2}, p_1^R, p_2^R\}$ et son complémentaire $[S] = \{p_{a1}, p_{b1}, p_{c1}\}$ qui n'est pas M-max'-marqué puisque $\forall p \in S \cap P^P, M(p) = 0$ et $M(p_1^R) = 0 < W(p_1^R, t_{c2})$ et $M(p_2^R) = 1 < max\{W(p_2^R, t_{a2}), W(p_2^R, t_{b2})\} = 2$. Le Réseau $(\mathcal{N}, M_0)$ est vivant mais l'état $M \in \mathcal{R}(\mathcal{N}, M_0)$ n'est pas M-max'-marqué. La condition de max'-contrôle est donc une condition suffisante mais n'est pas nécessaire.

La condition de max'-contrôle n'étant pas suffisante pour caractériser si un siphon peut engendrer une situation de blocage au sein d'un S4PR, la condition de **max"-contrôle** a été développée pour caractériser de manière plus fine les conditions suffisantes à l'évitement de blocages. La notion de **max"-contrôle** est développée dans [LLZ10] mais comporte des erreurs au niveau de la rédaction. Il est donc nécessaire de consulter en parallèle les corrections faites en [LLZ13].

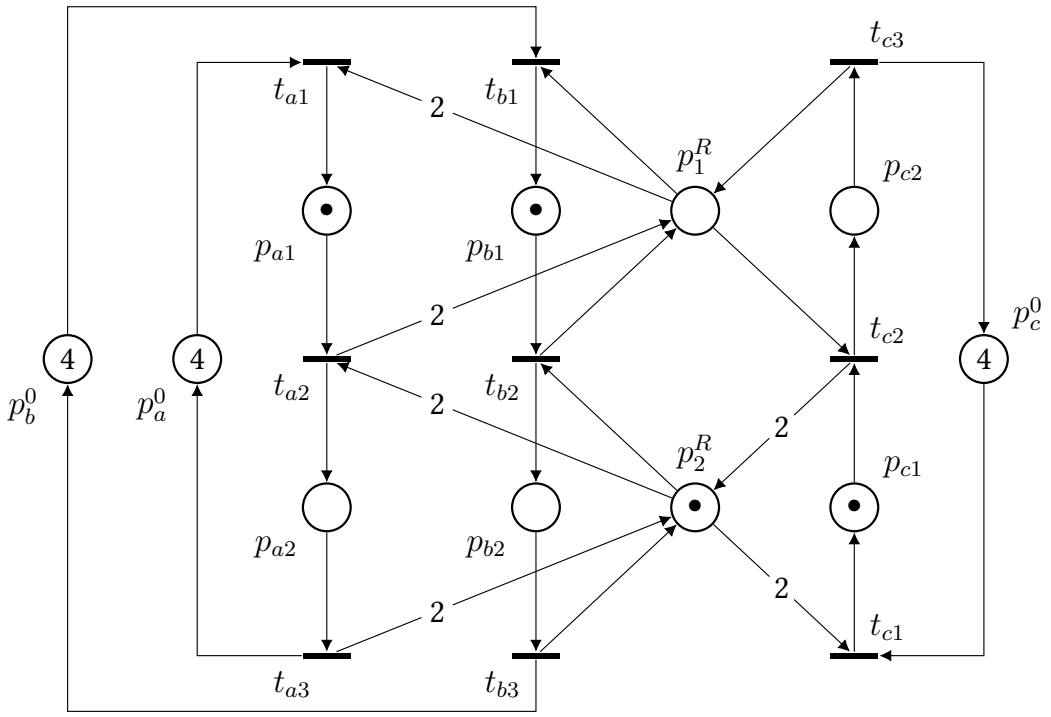


FIGURE 4.27 – Un S4PR vivant $(\mathcal{N}, M)$ avec un siphon non M-max'-contrôlé

Définition 73. *Étant donné un S4PR convenablement marqué et un siphon $S \in \Pi(\mathcal{N})$. S est dit M-max"-marqué pour un état $M \in \mathcal{R}(\mathcal{N}, M_0)$ si **une** des conditions suivantes est vérifiée :*

1. $M = M_0$
   ou
2. $\exists p \in P^P \cap S$ *tel que* $M(p) \geq 1$
   ou
3. $\exists p^R \in P^R \cap S$ *alors :*

$$\min \sum_{t \in T'} \alpha_t . W(t, p^R) + M(p^R) \geq \max_{t' \in p^{R\bullet} \cap [S]^\bullet} \{W(p^R, t')\}$$

avec
$T' = \{t \,|\, t \in {}^\bullet p^R \cap [S]^\bullet,\ \forall p^{R\prime} \in {}^\bullet t \cap P^R,\ M(p^{R\prime}) \geq W(p^{R\prime}, t),\ M(P^P \cap {}^\bullet t) \geq 1\}$.
α_t *est le nombre de fois que la transition* t *peut être tirée depuis* M. *La valeur de* $\min \sum_{t \in T'} \alpha_t . W(t, p^R)$ *peut être obtenue en résolvant le MIP suivant :*

$$\min \sum_{t \in T'} \alpha_t . W(t, p^R)$$

$$p \in {}^\bullet t \cap P^P,\ M(p) \geq 1, t_x \in p^\bullet \cap T'$$
$$\sum \alpha_{t_x} \leq M(p)$$
$$p^{R\prime} \in {}^\bullet t \cap P^R, t_y \in p^{R\prime\bullet} \cap T'$$
$$\sum \alpha_{t_y} . W(p^{R\prime}, t_y) \leq M(p^{R\prime})$$

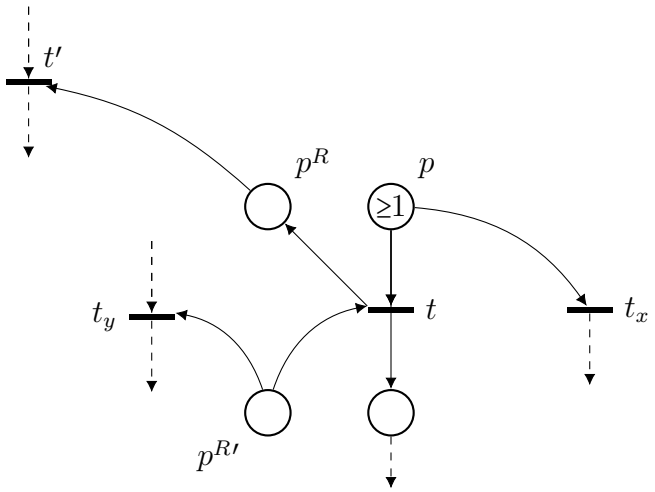


FIGURE 4.28 – Illustration des relations des différentes places et transitions du MIP de la définition 73

$$t \in {}^\bullet p^R \cap [S]^\bullet$$
$$\min\left\{\frac{M(p^{R\prime}) - \sum \alpha_{t_y}.W(p^{R\prime}, t_y)}{W(p^{R\prime}, t)}, M(p) - \sum \alpha_{t_x}\right\} < 1$$
$$\alpha_t \in \mathbb{N}$$

On retrouve les termes $M(p^R) \geq \max_{t' \in {p^R}^\bullet \cap [S]^\bullet}\{W(p^R, t')\}$ du max'-contrôle auquel est ajouté $\min \sum_{t \in T'} \alpha_t.W(t, p^R)$. De manière informelle, T' représente l'ensemble des transitions, qui, si tirées, vont générer des jetons dans p^R. On cherche donc α_t, le nombre minimal de fois que l'on sera obligatoirement amené à tirer une transition $t \in T'$. Le tir de t est conditionné par le nombre de jetons dans sa place de processus prédécesseurs ($p = {}^\bullet t \cap P^P$) et par les places de ressources prédécesseurs (${}^\bullet t \cap P^R$). Le MIP vient déterminer la valeur minimale de $\min \sum_{t \in T'} \alpha_t.W(t, p^R)$ en prenant en compte les tirs des transitions t_y pouvant vider les places de ressources prédécesseures ou le tir des transitions t_x pouvant vider les places de processus prédécesseures. Une illustration du MIP pour une transition t est donnée en figure 4.28, où il n'existe qu'une instance de $p^{R\prime}$, t_x et t_y.

Définition 74. *Étant donné un S4PR convenablement marqué $(\mathcal{N}, M_0)$ contenant un siphon $S \in \Pi(\mathcal{N})$. S est dit max"-contrôlé si S est M-max"-marqué pour tout $M \in \mathcal{R}(\mathcal{N}, M_0)$*

Théorème 15. *Étant donné un S4PR convenablement marqué $(\mathcal{N}, M_0)$. $(\mathcal{N}, M_0)$ est vivant si tous ses siphons $S \in \Pi(\mathcal{N})$ sont max"-contrôlés.*

Dans [LL10], l'auteur propose un MIP non-linéaire qui est en mesure de trouver un siphon S et un marquage M qui n'est pas M-max"-contrôlé. [CCY13] critique cette formulation de MIP non-linéaire car ayant une complexité proche de l'accessibilité. Dans [CCY13], les auteurs proposent une autre condition suffisante pour la contrôlabilité des siphons, plus fine que celle du max'-contrôle. Ils introduisent la notion de marquage pivot. L'existence d'un tel marquage est une condition nécessaire et suffisante à l'existence d'une situation de blocage.

4.4.2 Génération de places monitrices

La génération de places monitrices est une des méthodes classiques du contrôle des RAS-PN. Dans les S4PR, elle ne permet cependant pas d'obtenir un contrôle maximalement permissif. Toutefois, elle permet d'avoir un contrôle exprimé sous forme de Réseau de Petri et dans le cas des S4PR, que le système contrôlé soit aussi un S4PR. Les figures 4.29, 4.30, 4.31 présentées dans [LZ09] sont respectivement un S4PR convenablement marqué $(\mathcal{N}, M_0)$, $(\mathcal{N}, M_0)$ contrôlé maximalement, et $(\mathcal{N}, M_0)$ contrôlé par une place monitrice. Dans la figure 4.31, le contrôle représenté est le meilleur contrôle qu'il soit possible d'obtenir par ajout de places monitrices (utilisant la méthode de [CLZ12]). Cependant ce contrôle exercé par p_1^V n'est pas optimal. On remarque que pour avoir un contrôle optimal, l'ajout de places non-pures est nécessaire, ces places sont donc des places de contrôles mais pas des places monitrices. La figure 4.32 représente le graphe d'accessibilité du Réseau de Petri de la figure 4.29. Le contrôle optimal (figure 4.30) supprime 4 états qui entraînent la non-vivacité du Réseau de Petri. Le contrôle par place monitrice supprime 6 états du graphe d'accessibilité. On remarque que le contrôle par place monitrice favorise le processus a par rapport au processus b interdisant d'avoir plus d'un jeton dans p_{b1}. Un contrôle d'une permissivité équivalente favorisant le processus b au lieu du a aurait aussi bien pu être présenté. Les méthodes générant des places monitrices dans les S4PR ne cherchent donc jamais à avoir un contrôle optimal mais plutôt à s'en approcher.

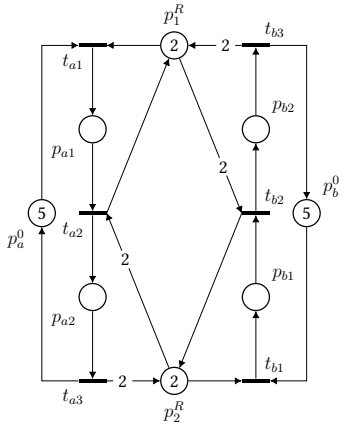


FIGURE 4.29 – Marquage initial M_0 d'un S4PR $\mathcal{N}$ convenablement marqué

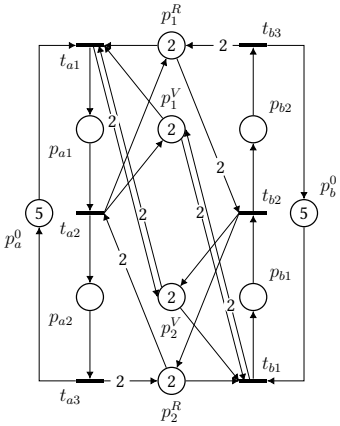


FIGURE 4.30 – SMS S dans $\mathcal{N}$ maximalement contrôlé

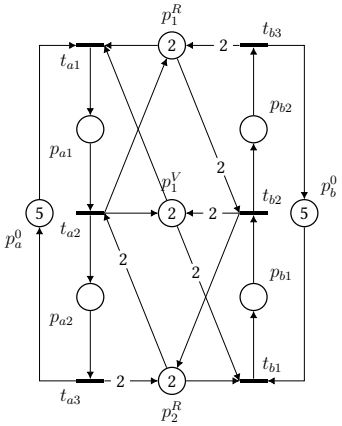


FIGURE 4.31 – SMS S dans $\mathcal{N}$ contrôlé par une place monitrice

Définition 75. *Considérons un siphon S et un mauvais marquage M d'un S4PR convenablement marqué $(\mathcal{N}, M_0)$ avec $\mathcal{N} = (P^P \cup P^R \cup P^0, T, F, W)$. Il est possible d'ajouter une place monitrice p^V à $(\mathcal{N}, M_0)$ pour obtenir $(\mathcal{N}^C, M_0^C)$ tel que $\nexists M' \in \mathcal{R}(\mathcal{N}^C, M_0^C)$ avec $M|_{P^P} = M'|_{P^P}$* [6]*. p^V est une place de ressources de $(\mathcal{N}, M_0)$*

6. Considérons une application $X : E \to \mathbb{N}$ sur un ensemble E et $S \subset E$, $X|_S$ est la projection de X uniquement sur les éléments de S.

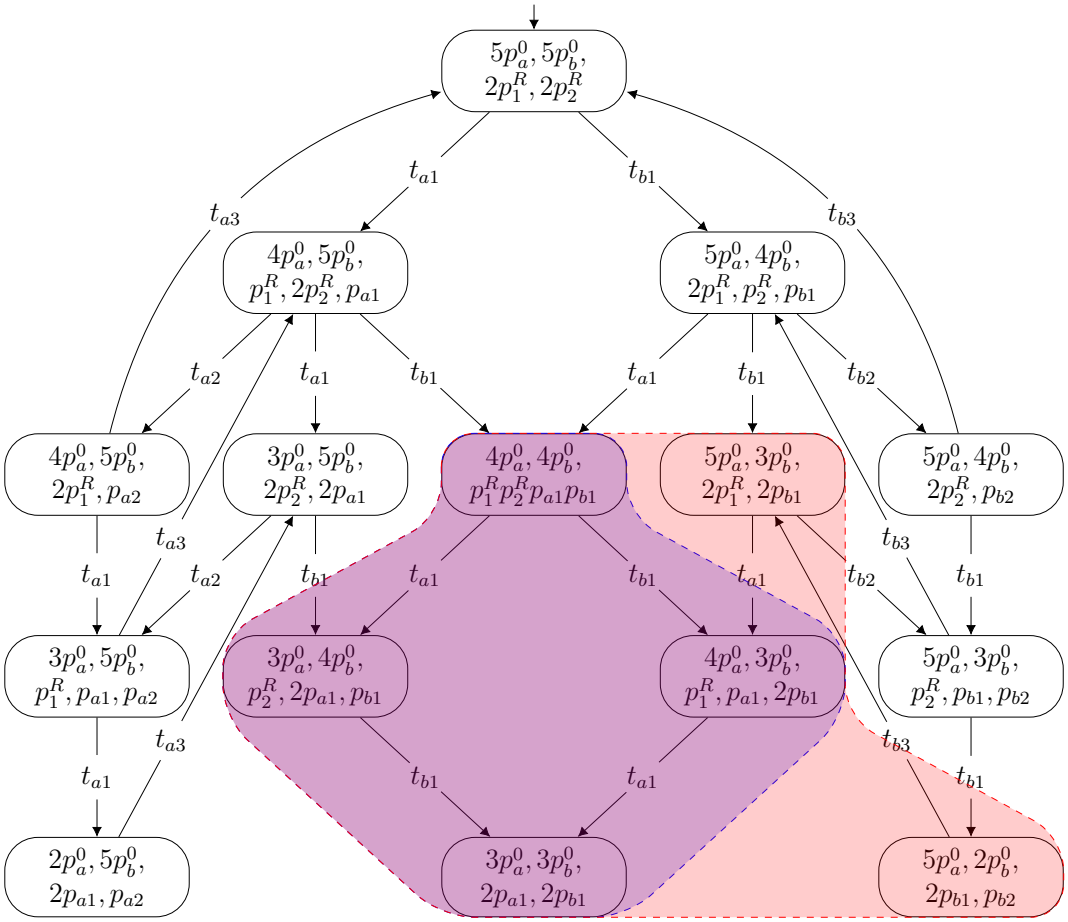


FIGURE 4.32 – Graphe d'accessibilité du Réseau de Petri de la figure 4.29

dont l'invariant est défini par :

$$\vec{p}^R_{V\circlearrowleft} = \begin{cases} \forall p \in [S], & k_{[S]}(p) \\ \forall p \in P^P \backslash [S], & 0 \end{cases}$$

Et le marquage initial :

$$M_0^C(p^V) = M_0(S) - M(S) - 1$$

Le Réseau contrôlé est donc aussi un S4PR avec $\mathcal{N}^C = (P^P \cup P^{R\prime} \cup P^0, T, F', W')$ *avec* $P^{R\prime} = \{p^V\} \cup P^R$.

Remarque 32. *L'ajout de places monitrices dans un S4PR suivant la méthode de la définition 75 peut créer de nouveaux siphons contenant cette place. Les méthodes itératives utilisant cette technique peuvent amener à un grand nombre de places.*

Définition 76. *Dans un S4PR $\mathcal{N} = (P^P \cup P^R \cup P^0, T, F, W)$ composé de $n = |\mathcal{I}|$ processus, l'ensemble des places avales $p^\downarrow$ d'une place p, est l'ensemble des places d'un même processus i qui peuvent être atteintes par un jeton de p sans que ce jeton ne passe par p_i^0.*

$$p^\downarrow = \{p' \in P_i^P \,|\, p \in P_i^P,\ \exists \lambda = p \ldots p' \in \Lambda(\mathcal{N}_i) \text{ avec } p_i^0 \notin \lambda\}$$

Définition 77. *Considérons un siphon S et un mauvais marquage M d'un S4PR convenablement marqué $(\mathcal{N}, M_0)$ avec $\mathcal{N} = (P^P \cup P^R \cup P^0, T, F, W)$. Il est possible d'ajouter une place monitrice p^W à $(\mathcal{N}, M_0)$ pour obtenir un Réseau contrôlé $(\mathcal{N}^C, M_0^C)$ tel que $\nexists M' \in \mathcal{R}(\mathcal{N}^C, M_0^C)$ avec $M|_{P^P} = M'|_{P^P}$. p^W est une place de ressources de $(\mathcal{N}, M_0)$ qui ne génère pas de nouveaux siphons. Son invariant est défini par :*

$$\vec{p}^R_{W\circlearrowleft} = \begin{cases} \forall p \in |k^\downarrow_{[S]}|, & k^\downarrow_{[S]}(p) \\ \forall p \in P^P \backslash |k^\downarrow_{[S]}|, & 0 \end{cases}$$

avec

$$k^\downarrow_{[S]}(p) = \max_{p' \in p^\downarrow} k_{[S]}(p') \quad \forall p \in P^P$$

Et le marquage initial :

$$M_0^C(p^W) = M_0(S) - M(S) - 1$$

Le Réseau contrôlé est donc aussi un S4PR avec $\mathcal{N}^C = (P^P \cup P^{R\prime} \cup P^0, T, F', W')$ avec $P^{R\prime} = \{p^W\} \cup P^R$.

Propriété 27. *Une place de contrôle p^W respectant la définition 77 a tous ses arcs sortants allant sur des transitions sources i.e $p^{W\bullet} \subseteq P^{0\bullet}$.*

Démonstration. Selon la définition 7, les transitions de sortie d'une place de ressources p^R d'un S4PR sont définies par

$$p^{R\bullet} = \{t \in T \,|\, \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P) > \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P)\}$$

Dans le cas de p^W, cette situation peut s'appliquer uniquement aux transitions de $P^{0\bullet}$ car $\forall t \in T \backslash P^{0\bullet}$, $\{p\} = {}^\bullet t \cap P^P$ et $\{p'\} = t^\bullet \cap P^P$ et $k^\downarrow_{[S]}(p') \leq k^\downarrow_{[S]}(p)$. □

Propriété 28. *Une place monitrice p^W respectant la définition 77 ne peut être incluse dans un siphon avec un mauvais marquage.*

Démonstration. Selon la définition 67, étant donné un siphon S et un mauvais marquage M, les places de ressources du siphon sont caractérisées par $S \cap P^R = \{p^R \in P^R \,|\, \exists t \in p^{R\bullet}$ avec $M(p^R) < W(p^R, t) \wedge M({}^\bullet t \cap P^P) > 0\}$. Or p^W ne peut correspondre à cette situation puisque les transitions de sortie de p^W sont des transitions sources i.e $\forall t \in p^{W\bullet} \,|\, {}^\bullet t \cap P^P = \varnothing$. □

Remarque 33. *Dans le cas de la génération de places monitrices pour un S3PMR, les places monitrices peuvent avoir des arcs pondérés. Le Réseau résultant n'est donc plus un S3PMR, mais un S4PR. Cependant dans le cas d'un siphon S dont le coût associé de l'ensemble complémentaire $k_{[S]}$ n'excède jamais 1 (i.e $\forall p \in [S]$, $k_{[S]}(p) = 1$) la place monitrice obtenue est ordinaire. Le Réseau contrôlé reste alors un S3PMR.*

Remarque 34. *Dans le cas de la génération de places monitrices pour un S3PR, le coût associé de l'ensemble complémentaire $k_{[S]}$ d'un siphon S ne dépasse jamais 1 (voir propriété d, les places de processus ne consomment qu'un seul type de ressource). La place de contrôle obtenue est donc ordinaire mais ne respecte pas nécessairement la propriété d. Le Réseau contrôlé obtenu peut donc devenir un S3PMR.*

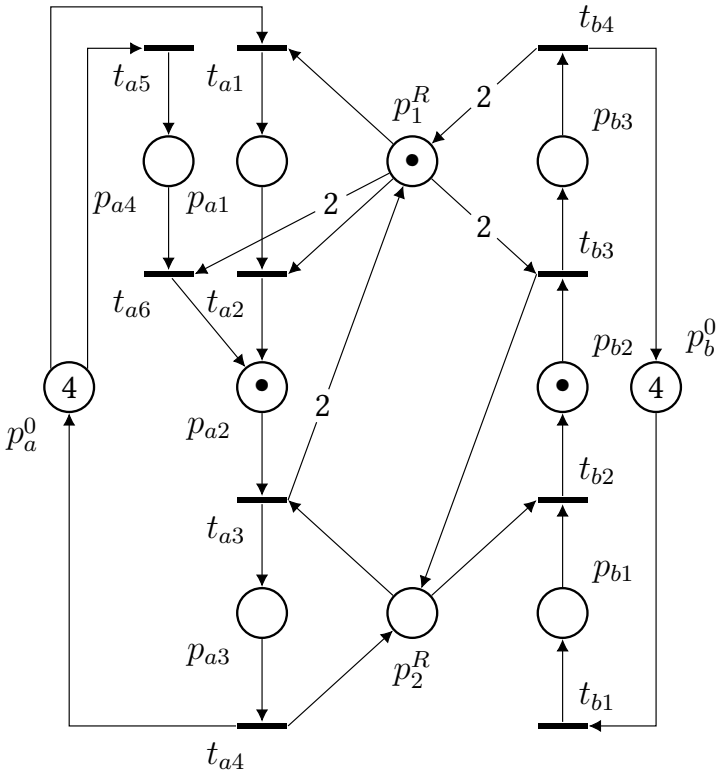


FIGURE 4.33 – Un S4PR $\mathcal{N}$ dans un mauvais marquage M relatif à un siphon S

La figure 4.33 présente un S4PR $\mathcal{N}$ dans un état M. M est un état accessible depuis M_0 i.e $M \in \mathcal{R}(\mathcal{N}, M_0)$ ou M_0 est un état initial convenable. L'état M est un mauvais état vis-à-vis de $S = \{p_{a1}, p_{a3}, p_{b3}, p_1^R, p_2^R\}$. A l'état M, des jetons se retrouvent bloqués dans $[S] = \{p_{a2}, p_{b2}\}$. Pour éviter cette situation, une place monitrice peut être générée. La figure 4.34 montre le Réseau obtenu après l'ajout de la place monitrice p^V suivant la définition 75 générant une place de ressource dont l'invariant s'étend sur $k_{[S]}$. La figure 4.35 montre quant à elle, la génération d'une place monitrice p^W suivant la définition 77 générant une place de ressources dont l'invariant s'étend sur $k^{\downarrow}_{[S]}$.

On remarque que p^V peut engendrer de nouveaux blocages. En effet, le marquage $M'(P^P) = 2p_{a1} + p_{b2}$ crée un nouveau mauvais marquage vis-à-vis de $S' = \{p_{a2}, p_{b2}, p_1^R, p^V\}$ un siphon englobant p^V. La place monitrice p^W, quant à elle, ne crée pas de nouveaux siphons, mais son contrôle peut être trop restrictif. Par exemple, les places de processus p_{b1} et p_{a4} ne sont pas comprises dans des Réseaux de ressources. Prendre en compte leur marquage dans la stratégie de contrôle se trouve donc être excessivement restrictif.

	p_a^0	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p_b^0	p_{b1}	p_{b2}	p_{b3}	p_1^R	p_2^R
$\vec{p}^R_{\circlearrowleft 1}$	0	1	2	0	0	0	0	0	2	1	0
$\vec{p}^R_{\circlearrowleft 2}$	0	0	0	1	0	0	0	1	0	0	1
M_0	5	0	0	0	0	5	0	0	0	3	2
M	4	0	1	0	0	4	0	1	0	0	0
S	✗	✓	✗	✓	✗	✗	✗	✗	✓	✓	✓
$k_{[S]}$	✗	0	2	0	0	✗	0	1	0	✗	✗
$k^{\downarrow}_{[S]}$	✗	2	2	0	2	✗	1	1	0	✗	✗

TABLE 4.6 – Information relative au S4PR $\mathcal{N}$, M_0, M, S, $k_{[S]}$ et $k^{\downarrow}_{[S]}$

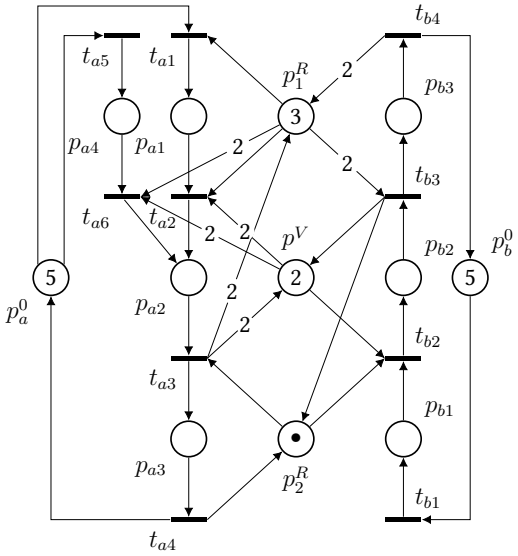


FIGURE 4.34 – Place monitrice p^V générée à partir du siphon S et du mauvais marquage M

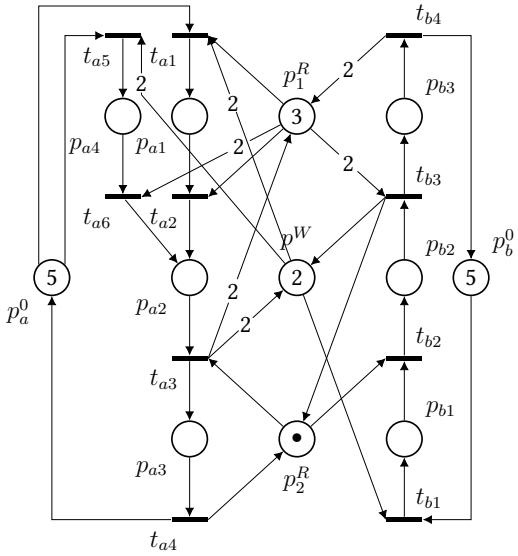


FIGURE 4.35 – Place monitrice p^W générée à partir du siphon S et du mauvais marquage M

4.4.3 Extraction des siphons

L'extraction des siphons joue un rôle important dans les méthodes d'évitement des blocages. À partir d'un RAS-PN, un siphon est extrait, la plupart du temps avec un mauvais marquage associé. Les méthodes d'extraction sont souvent intégrées dans des algorithmes de contrôles itératifs, où un couple extraction/contrôle du siphon est répété jusqu'à obtention d'un Réseau vivant. Les méthodes d'extractions cherchent à retourner des siphons possédant des caractéristiques intéressantes d'un point de vue de la génération de places monitrices, c'est-à-dire des siphons liés à un nombre important de mauvais-marquages. La sélection d'une "bonne" méthode d'extraction influe directement sur le nombre de places monitrices et la permissivité de la méthode de contrôle.

Dans [TGVCE05], une méthode d'extraction est développée pour l'extraction d'un siphon et d'un mauvais marquage associé dans un S4PR. Cette méthode se base sur un MIP, dérivé de la méthode de [CX97]. [ZDWN18, ZDW+19] montrent que cette méthode d'extraction de [TGVCE05] comporte des erreurs de formulation,

et montrent que le marquage associé n'est pas nécessairement mauvais. [ZDWN18, ZDW+19] reformule alors le MIP de [TGVCE05] de manière à ce que le marquage soit nécessairement mauvais. Cette méthode est présentée dans la suite de cette section.

Considérant un S4PR convenablement marqué $(\mathcal{N}, M_0)$, si $(\mathcal{N}, M_0)$ n'est pas vivant alors il existe un siphon S et un mauvais marquage M correspondant à la définition 67. [ZDW+19] propose une formulation de MIP qui permet de trouver un tel siphon avec son mauvais marquage associé.

Définition 78. *Trois vecteurs de variables binaires sont définis :*

— $v \in \{0,1\}^{|P^R \cup P^P|}$ *définit les places du siphon* S*. Une variable binaire est associée par places de processus et de ressource.*

$$\forall p \in P \backslash P^0, \quad v(p) = \begin{cases} 0 & \text{si } p \in S \\ 1 & \text{si } p \notin S \end{cases}$$

— $\tau \in \{0,1\}^{T \backslash P^{0\bullet}}$ *correspond à l'ensemble des transitions qui sont M-processus sensibilisées. Les transitions sources ne pouvant être Processus-sensibilisées ne sont donc pas considérées.*

$$\forall t \in T \backslash P^{0\bullet}, \quad \tau(t) = \begin{cases} 1 & \text{si } t \in M^P_{[\rangle} \\ 0 & \text{si } t \notin M^P_{[\rangle} \end{cases}$$

— $f \in \{0,1\}^{|F^{R,Tin}|}$ *avec* $F^{R,Tin} = (P^R \times T \backslash P^{0\bullet}) \cap F$*, une variable binaire associée à chacun des arcs entre les places de ressources et les transitions non-sources. Pour une place de ressources* $p^R \in P^R$ *et une transition* $t \in T \backslash P^{0\bullet}$*,* $f(p^R, t)$ *vaut 1 si la transition est M-ressources désensibilisée par* p^R*.*

$$\forall p^R \in P^R, \ \forall t \in T \backslash P^{0\bullet}, \quad f(p^R, t) = \begin{cases} 1 & \text{si } W(p^R, t) \leq M(p^R) \\ 0 & \text{si } W(p^R, t) > M(p^R) \end{cases}$$

Deux vecteurs de variables réelles sont définis :

— $M \in \mathbb{R}^{|P|}$ *correspond au mauvais marquage obtenu.*

— $Y \in \mathbb{R}^{|T|}$ *correspond au vecteur de tir pour obtenir* M*. Comme* M *est obtenu par l'équation d'états, le marquage* M *n'appartient pas nécessairement à* $\mathcal{R}(\mathcal{N}, M_0)$ *mais appartient à* $\vec{\mathcal{R}}(\mathcal{N}, M_0)$*.*

$$\forall p \in P\backslash P^0, \quad \forall t \in {}^\bullet p, \qquad v(p) \geq \sum_{q \in {}^\bullet t} v(q) - |{}^\bullet t| + 1 \tag{4.1}$$

$$\sum_{p \in P\backslash P^0} v(p) < |P\backslash P^0| \tag{4.2}$$

$$M(P^P) > 0 \tag{4.3}$$

$$\forall t \in T\backslash P^{0\bullet},\ \exists p \in {}^\bullet t \cap P^P,$$

$$M(p) \geq \tau(t) \tag{4.4}$$

$$\tau(t) \geq M(p)/SB(p) \tag{4.5}$$

$$\forall p^R \in P^R,\ \forall t \in p^{R\bullet}\backslash P^{0\bullet},$$

$$f(p^R, t) \geq \frac{M(p^R) - W(p^R, t) + 1}{M_0(p^R) - W(p^R, t) + 1} \tag{4.6}$$

$$|\tau(t) - f(p^R, t) - 1| \geq v(p^R) \tag{4.7}$$

$$\forall p^R \in P^R,$$

$$\sum_{t \in p^{R\bullet}\backslash P^{0\bullet}} f(p^R, t) - |p^{R\bullet}\backslash P^{0\bullet}| + 1 \leq v(p^R) \tag{4.8}$$

$$\sum_{t \in p^{R\bullet}\backslash P^{0\bullet}} \tau(t) \geq 1 - v(p^R) \tag{4.9}$$

$$\forall t \in T\backslash P^{0\bullet} \qquad \sum_{p^R \in {}^\bullet t \cap P^R} f(p^R, t) < |{}^\bullet t \cap P^R| + 1 - \tau(t) \tag{4.10}$$

$$M = M_0 + C.Y,\ M \geq 0, Y \geq 0 \tag{4.11}$$

$$\max \sum_{p \in P\backslash P^0} v(p).|P| + \sum_{p^R \in P^R} v(p^R) \tag{4.12}$$

La contrainte 4.1 assure que S est bien un siphon en s'assurant que ${}^{\bullet\bullet}S \in S$.

$$4.1 :\ \forall p \in S \implies {}^{\bullet\bullet}p \subseteq S$$

La contrainte 4.2 assure que S n'est pas un siphon englobant toutes les places, force à faire échouer la recherche lorsqu'un siphon et un mauvais marquage associé ne peuvent être trouvés. La contrainte 4.3 force le mauvais marquage à avoir des places de processus marquées conformément à la définition d'un mauvais marquage. Les contraintes 4.4 et 4.5 appliquent la définition de τ pour les transitions non sensibilisées.

$$4.4 :\ \forall t \in T\backslash P^{0\bullet}, \{p\} = P^P \cap {}^\bullet t \text{ si } M(p) = 0 \implies \tau(t) = 0$$

$$4.5 :\ \forall t \in T\backslash P^{0\bullet}, \{p\} = P^P \cap {}^\bullet t \text{ si } M(p) \neq 0 \implies \tau(t) = 1$$

La contrainte 4.6 applique la définition de f pour les arcs (p^R, t) avec $p^R \in P^R$ et $t \in T\backslash P^{0\bullet}$ qui désensibilisent la transition t.

$$4.6 :\ \forall p^R \in P^R,\ \forall t \in p^{R\bullet}\backslash P^{0\bullet},\ \text{si } M(p^R) < W(p^R, t) \implies f(p^R, t) = 1$$

La contrainte 4.7 implique que si une transition est M-ressource-désensibilisée par p^R et M-processus-sensibilisée, p^R doit être inclus dans S

$$4.7 : \forall p^R \in P^R,\ \forall t \in p^{R\bullet} \backslash P^{0\bullet},\ \text{si } f(p^R, t) = 0 \wedge \tau(t) = 1 \implies p^R \in S$$

La contrainte 4.8 implique que les places de ressources de S doivent avoir au moins un arc qui désensibilise une transition.

$$4.8 : \forall p^R \in P^R, \text{ si } p^R \in S \implies \exists t \in p^{R\bullet} \mid f(p^R, t) = 0$$

La contrainte 4.9 force les places de ressources du siphon qu'au moins une transition de sortie soit M-processus sensibilisée.

$$4.9 : \forall p^R \in P^R, \text{ si } p^R \in S \implies \exists t \in p^{R\bullet} \mid \tau(t) = 1$$

La contrainte 4.10 oblige que toutes les transitions M-processus-sensibilisées soient aussi M-ressources-désensibilisées.

$$4.10 : \forall t \in T \backslash P^{0\bullet} \text{ si } \forall p^R \in t \cap P^R,\ f(p^R, t) = 1 \implies \tau(t) = 0$$

La fonction objectif $\max \sum_{p \in P^P} v(p).|P| + \sum_{p^R \in P^R} v(p^R)$ cherche à minimiser le nombre de places de ressources et de places de processus dans le siphon. Ainsi, plus le nombre de places de ressources et de places de processus est faible, plus la fonction objectif est élevée. Le coefficient $|P|$ est ajouté devant les places de processus de façon à ce que, pour deux siphons S et S', qui correspondent aux contraintes et comportent le même nombre de places mais dont S dispose d'une place de ressources de plus que S', alors la fonction objectif du MIP retournera S. $|P|$ a été choisi comme coefficient pondérant les places de processus. Ainsi, si deux siphons S et S' qui correspondent aux contraintes avec $|S| > |S'|$, alors c'est le siphon S qui sera retourné par le MIP. Et cela, peu importe le nombre de places ou la distribution des places de processus et de ressources dans S et S'.

Définition 79. *La résolution du MIP sous les contraintes 4.1 à 4.11 avec la fonction objectif 4.12 permet de trouver un couple (S, M) ou $S \in \Pi(\mathcal{N})$ et $M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)$, un mauvais-marquage relatif au siphon S. Nous noterons la résolution de ce MIP sous la fonction $MIP^1(\mathcal{N}, M_0) = (S, M)$. Dans le cas où le Réseau est vivant $MIP^1(\mathcal{N}, M_0)$ n'aboutit pas à un couple (S, M). Par convention nous noterons $MIP^1(\mathcal{N}, M_0) = \varnothing$*

Si $MIP^1(\mathcal{N}, M_0) = (S, M)$, le couple (S, M) justifie qu'il est d'intérêt de contrôler le siphon S. Il est maintenant nécessaire de trouver une borne maximale du marquage de S en dessous de laquelle un mauvais marquage est atteint. Pour cela, l'ensemble des contraintes 4.13-4.22 est défini. Connaissant un siphon S, tout marquage qui vérifie ces contraintes est un mauvais marquage.

On retrouve dans ces contraintes les vecteurs de variables τ, f, M et Y, définis de la même manière que le MIP précédant selon la définition 78. La majorité des contraintes est semblables à celles du précédant MIP où la variable v est supprimée. En effet, S est déjà connu. Ici, la fonction objectif cherche à maximiser le marquage dans les places de ressources. La valeur M_S^{max} nous donne une borne maximale du

$$M'(P^P) > 0 \quad (4.13)$$

$\forall t \in T \backslash P^{0\bullet}, \exists p \in {}^\bullet t \cap P^P$

$$M'(p) \geq \tau(t) \quad (4.14)$$

$$\tau(t) \geq M'(p)/SB(p) \quad (4.15)$$

$\forall p^R \in S \cap P^R, \forall t \in p^{R\bullet} \backslash P^{0\bullet}$

$$\frac{M'(p^R)}{W(p^R,t)} \geq f(p^R,t) \quad (4.16)$$

$$f(p^R,t) \geq \frac{M(p^R) - W(r,t) + 1}{M_0(p^R) - W(r,t) + 1} \quad (4.17)$$

$\forall p^R \in S \cap P^R,$

$$\sum_{t \in p^{R\bullet} \backslash P^{0\bullet}} \tau(t) \geq 1 \quad (4.18)$$

$\forall p^R \in P^R \backslash S, \forall t \in p^{R\bullet} \backslash P^{0\bullet}$

$$f(p^R,t) = 1 \quad (4.19)$$

$\forall t \in T \backslash P^{0\bullet},$

$$\sum_{p^R \in {}^\bullet t \cap P^R} f(p^R,t) < |{}^\bullet t \cap P^R| + 1 - \tau(t) \quad (4.20)$$

$$M' = M_0 + C.Y,\ M' \geq 0, Y \geq 0 \quad (4.21)$$

$$M'(P^P/[S]) = 0 \quad (4.22)$$

$$M_S^{max} = \max_{p^R \in P^R} M'(p^R) \quad (4.23)$$

nombre de jetons dans les places de ressources à partir de laquelle le siphon peut atteindre un mauvais marquage.

Définition 80. *Considérons un S4PR convenablement marqué $(\mathcal{N}, M_0)$ et un siphon $S \in \Pi(\mathcal{N})$. La résolution du MIP sous les contraintes 4.13-4.22 et sous la fonction objectif 4.23 retourne une valeur $M_S^{max} \in \mathcal{N}$. On notera la résolution de ce MIP sous la forme $MIP^2(\mathcal{N}, M_0, S) = M_S^{max}$.*

Définition 81. *Pour un S4PR convenablement marqué $(\mathcal{N}, M_0)$, la résolution successive de $MIP^1(\mathcal{N}, M_0) = (S, M)$ et $MIP^2(\mathcal{N}, M_0, S) = M_S^{max}$ permet d'obtenir un couple (S, M_S^{max}), soit un siphon $S \in \Pi(\mathcal{N})$ et le marquage maximal des places de ressources à partir duquel un mauvais-marquage peut survenir. On notera $\mathcal{E}(\mathcal{N}, M_0) = (S, M_S^{max})$, l'exécution successive des deux MIP qui extrait un siphon et sa borne maximale.*

De la même manière que la méthode de [PR01] et de [CX97], ce MIP se base sur l'équation d'états. Il est donc possible que la borne maximale trouvée pour un marquage $M' \in \mathcal{R}(\vec{\mathcal{N}}, M_0)$ soit plus faible. Il est même possible d'extraire un siphon qui ne peut atteindre un mauvais-marquage.

Remarque 35. *Pour les S3PR et S3PMR, il est seulement intéressant de chercher les SMS et un marquage associé qui les vident. Le temps de calcul des MIP est fortement corrélé au nombre de variables binaires du MIP. La formulation du MIP présentée dans [Cha09], qui permet d'extraire un SMS potentiellement vidable, comporte $|T| + |P|$ variables binaires. La méthode de [PR01] comporte $|T \backslash P^{0\bullet}| + |P \backslash P^0| + |F^{R,Tin}|$ variables binaires. Pour les S3PR et S3PMR, la méthode de Chao [Cha09] est donc à préférer.*

Il peut être intéressant d'extraire plusieurs siphons consécutivement. Si l'opération d'extraction (la résolution du MIP) est exécutée plusieurs fois consécutivement, le siphon et le marquage retournés seront identiques. Pour éviter d'obtenir le même résultat, il est possible de rajouter des contraintes sur le marquage de manière à interdire au MIP d'atteindre un mauvais marquage pour un siphon déjà trouvé.

On considère $\mathcal{X}$ une méthode d'extraction quelconque obtenue par un MIP dérivé de la formulation de [CX97]. $\mathcal{X}$ trouve un siphon et un mauvais-marquage associé. La fonction $\mathcal{X}_1(\mathcal{N}, M_0)$ retourne un couple (S_1, M_1).

Pour que la fonction d'extraction $\mathcal{X}_i(\mathcal{N}, M_0) = (S_i, M_i)$ ne retrouve pas le siphon S_i une contrainte supplémentaire c_i est ajoutée à $\mathcal{X}_i$:

$$c_i : \quad M(S_i) > M_i(S) \tag{4.24}$$

Pour une méthode d'extraction $\mathcal{X}_i = (S_i, M_i)$ une fois la contrainte 4.24 ajoutée, la méthode d'extraction se note $\mathcal{X}_{i+1}$. Si $M_i(S)$ est une valeur maximale pour laquelle le siphon S_i atteint un mauvais marquage (i.e $\nexists M' \in \vec{\mathcal{R}}(\mathcal{N}, M_0)$ avec $M'(S) > M_i(S)$ et M' un mauvais marquage de S) alors on est assuré qu'aucune exécution de $\mathcal{X}_j$ avec $j > i$ ne trouve le siphon S_i. Lorsque $\mathcal{X}_{k+1}$ n'aboutit pas à une solution, alors il ne reste plus de siphons qui puissent être vidés dans $\vec{\mathcal{R}}(\mathcal{N}, M_0)$ sous les contraintes $c_1, ..., c_k$.

Cette méthode d'extraction itérative de siphon est différente de l'énumération des siphons. Ici les siphons $S_1, ..., S_k$ forment un ensemble de siphons d'intérêt pour la mise en place d'une politique de contrôle, mais ils ne constituent pas l'ensemble des siphons de $\mathcal{N}$.

4.4.4 Méthode de contrôle

Dans [ECM95] est présenté pour la première fois une classe de RAS-PN : la classe des S3PR. Dans ce papier est présenté le Réseau de la figure 4.36. Aujourd'hui, ce Réseau de Petri est un classique pour comparer les différentes méthodes de contrôle des RAS-PN. Comme S3PR ⊆ S3PMR ⊆ S4PR, il est possible de comparer les méthodes de ces trois classes de RAS-PN sur ce Réseau. Le tableau 4.7 énumère différentes méthodes de contrôles rencontrées dans la littérature.

[PR01] propose une méthode de contrôle quasi-directe (polynomiale) se basant sur les Réseaux de ressources. Mais elle engendre un contrôle très restrictif.

Les méthodes [ECM95, HJXC01, TGVCE05], sont aujourd'hui obsolètes d'un point de vue technique. Néanmoins, [HJXC01, TGVCE05] permettent tout de même d'avoir des contrôles avec relativement peu de places monitrices (Bonne complexité structurelle). Les méthodes [PCF08, CLKM10] permettent d'avoir un contrôle optimal. La méthode de [UZ07] est très proche d'un contrôle optimal. Ces méthodes sont basées sur le graphe d'accessibilité ou sur l'énumération des siphons. Elles sont de complexité exponentielle. Même si des progrès ont été faits pour réduire le temps de calcul, ces méthodes supportent généralement mal d'être appliquées à des systèmes de grandes tailles.

[HHJ+15] propose une méthode d'extraction des siphons max'-contrôlés. Elle applique une méthode de réduction de places monitrices basée sur les siphons élémentaires augmentés. Cette méthode offre un bon compromis entre la complexité structurelle (le nombre de places monitrices) et la permissivité du contrôle.

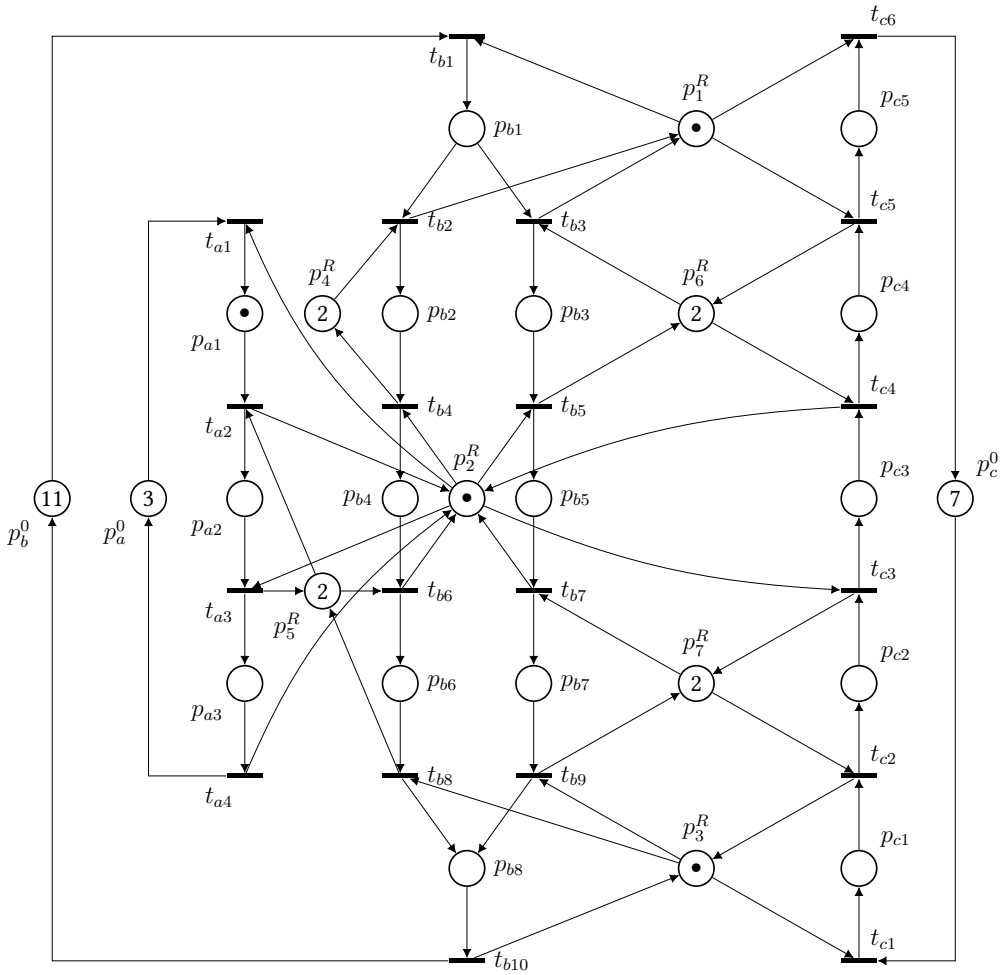


FIGURE 4.36 – Un S3PR $\mathcal{N}$ présenté dans [ECM95]

La méthode de [ZDW+19] utilise la méthode d'extraction présentée dans la sous-section 4.4.3. Elle offre de très bons résultats en ce qui concerne la permissivité du contrôle mais n'est cependant pas optimale (ce que l'on pourrait croire en regardant le nombre d'états atteints).

4.5 Conclusion

Ce chapitre présente les Réseaux de Petri pour les systèmes d'allocations de ressources (RAS-PN).

La section 4.1 donne une vue d'ensemble des différentes classes de RAS-PN. D'une manière générale, les RAS-PN sont composés d'un ensemble de Réseaux de processus évoluant en parallèle, seules des ressources partagées viennent lier ces processus. La définition d'un ensemble de restrictions liées à la structure des Réseaux processus et des Réseaux de ressources permet de différencier de manière claire les classes de RAS-PN de la littérature.

La section 4.2 se concentre sur trois classes de RAS-PN : les S3PR, les S3MPR et les S4PR. Ces trois classes comportent des restrictions identiques sur les Réseaux de processus, seules les restrictions sur les Réseaux de ressources sont différentes et

Référence	États	Places monitrices	Énumération des siphons	Complexité	Classe
[ECM95]	6287	18	oui	Exponentielle	S3PR
[PR01]	2480	7	oui	Polynomiale	S4PR
[CLKM10]	6287	6	oui	Exponentielle	S4PR
[TGVCE05]	14850	8	non	NP-difficile	S4PR
[HJXC01]	12656	16	non	NP-difficile	S3PMR
[LHW07]	16636	7	non	NP-difficile	S3PR
[HHJ$^+$15]	20444	9	non	NP-difficile	S3PR
[WZW15]	21562	15	non	NP-difficile	S3PR
[UZ07]	21562	17	non	Exponentielle	RDP bornés
[CLKM10]	21581	17	non	Exponentielle	RDP bornés
[PCF08]	21581	13	oui	Exponentielle	RDP ordinaires
[ZDW$^+$19]	21581	15	non	NP-difficile	S4PR

Table 4.7 – Comparaison des performances de diverses méthodes de la littérature

induisent la hiérarchie suivante : S3PR ⊆ S3PMR ⊆ S4PR. Les structures de ces trois classes sont détaillées dans la section 4.2, par la suite uniquement ces trois classes seront utilisées.

Les RAS-PN sont par nature des systèmes entraînant les blocages. La section 4.3 détaille les propriétés des S3PR, des S3MPR et des S4PR vis-à-vis de leurs relations aux blocages. Les S4PR étant la classe la plus englobante, les propriétés des S4PR sont aussi valables pour les S3PMR et S3PR, dans certains cas ces même propriétés sont affinées pour prendre en compte les structures plus restreintes des S3PMR et des S3PR. Dans un premier temps, la lumière est mise sur l'existence de "mauvais marquages" qui induisent l'existence de blocages dans le Réseau. Ce marquage est caractérisé puis associé à la présence de siphons, qui permet de faire le lien entre une propriété comportementale (mauvais marquage) et structurelle (siphon).

Lorsqu'un siphon pouvant entraîner une situation de blocage a été identifié, une stratégie de contrôle peut être mise en place de manière à ce que des mauvais marquages associés à ce siphon ne puissent plus être atteints. La section 4.4 se concentre sur l'aspect du contrôle des siphons. Pour un siphon donné, on identifie l'ensemble des places qu'il est intéressant de contrôler. Cet ensemble de places est nommé "ensemble complémentaire" du siphon et correspond aux places dans lesquelles les jetons viennent se coincer lorsque le siphon est dans un mauvais marquage. Cet ensemble de places peut être contrôlé grâce à une place monitrice. Pour trouver un siphon entraînant un blocage, des méthodes d'extraction peuvent être utilisées. Une méthode d'extraction introduit dans [ZDW$^+$19] est présentée. Cette méthode développée pour les S4PR permet de trouver un siphon intéressant du point de vue du contrôle par place monitrice. Cette méthode exécute deux MIP successifs : le premier renvoie un siphon et le deuxième une borne maximale du nombre de jetons en dessous de laquelle le siphon peut engendrer des situations de blocages. Finalement, une comparaison des performances des différentes méthodes de contrôles de la littérature est présentée.

Chapitre 5

Méthodes itératives de synthèse de contrôleurs pour les RAS-PN sous contrôlabilité et observabilité partielles

La majorité des méthodes d'évitement de blocage dans les RAS-PN ont été développées pour des applications manufacturières, au point que la notion de Système Flexible de Production Manufacturière (SFPM) soit devenue dans la littérature, synonyme de systèmes d'allocations de ressources (RAS). Dans les systèmes manufacturiers, les ressources (machines, robots, stocks ...) sont souvent associées à des opérations. Le nombre de jetons des places de ressources décrit le nombre d'exemplaires de chaque type de ressources dans le système. Ces modèles ont généralement un niveau d'abstraction élevé et le modèle du traitement d'une opération n'est pas intégré dans les modèles des SFPM. Cette abstraction, qui permet d'obtenir des modèles de haut-niveau, est parfaitement justifiée dans ce cadre, et permet de se concentrer uniquement sur la problématique de l'évitement des blocages. Cependant, la théorie de la supervisions introduite par Ramadge et Wonham [RW87] traite à la fois des problèmes d'évitement des blocages tout en prenant en compte l'existence de transitions incontrôlables et même inobservables [CDFV88].

Les RAS-PN permettent cependant de modéliser d'autres systèmes que les SFPM. Par exemple, les systèmes ferroviaires peuvent être modélisés par de tels Réseaux de Petri (RdP). Dans les systèmes ferroviaires, il est indispensable de prendre en compte l'existence d'évènements incontrôlables et inobservables liés à l'instrumentation si l'on veut que le modèle soit proche de la réalité.

La majorité des méthodes de synthèse pour les RAS-PN présentées dans la littérature ne prennent pas en compte les transitions inobservables et incontrôlables [CKT20, BCZ97, PR02, QLAA15, LS08]. Ce chapitre propose des méthodes de synthèse de contrôleurs pour les S4PR, S3PMR et S3PR sous contrôlabilité et observabilité partielles (COP). L'objectif est de garantir la vivacité de ces contrôleurs par synthèse.

Dans ce chapitre, en complément du chapitre précédent, nous commencerons par une section relative à la génération d'une place monitrice à partir d'une contrainte inadmissible dans un RAS-PN sous COP. Dans une seconde section, nous nous intéresserons au contrôle des siphons dans les RAS-PN sous COP. Dans la troisième section, nous proposerons deux techniques de synthèse. La première appelée H2'

s'applique aux S3PMR. La deuxième technique appelée Z2' s'applique aux S4PR. Dans la section suivante, nous analysons les performances de ces deux algorithmes en les comparant à des algorithmes représentatifs de la littérature. L'objectif de cette analyse de performance est de montrer que la proposition d'une technique de synthèse, est un compromis entre les critères de performances et l'étendue des classes de modèles auxquels la méthode peut être appliquée. Le chapitre se terminera par une conclusion nous permettant de faire une synthèse des propositions et enseignements de ce chapitre, dans l'optique du chapitre suivant.

5.1 Transformation des contraintes dans les RAS-PN sous COP

Dans cette section, nous allons proposer un certain nombre de résultats permettant de définir des conditions nécessaires, voire suffisantes, pour la génération de places monitrices pour les S4PR sous COP. L'idée est d'affiner la notion de GMEC que nous avons introduite dans le chapitre 3 dans le cadre général des Réseaux de Petri aux RAS-PN, et plus précisément, de transposer les conditions d'admissibilité définies dans la section 3.2.4 en termes de conditions sur les paramètres du GMEC (w, b) de manière à ce que la place monitrice corresponde à une place de ressource. Nous proposerons à la fin de la section un algorithme permettant de calculer des vecteurs w qui soient admissibles pour la synthèse de contrôleurs.
Dans le cas des RAS-PN, la génération de places monitrices est équivalente à l'ajout de places de ressources si l'ensemble des places contrôlées ne comporte que des places de processus.

Hypothèse 1. *Considérons un S4PR $\mathcal{N} = (P^P \cup P^0 \cup P^R, T, F, W)$ sous contrôlabilité et observabilité partielles. Il existe un ensemble de transitions incontrôlables $T_{uc} \subset T$ et inobservables $T_{uo} \subset T$. On suppose que propriété suivante est vérifiée :*

- $P^{0\bullet} \cap T_{uc} = \varnothing$ *et* $P^{0\bullet} \cap T_{uo} = \varnothing$
- ${}^\bullet P^0 \cap T_{uo} = \varnothing$

Remarque 36. *L'hypothèse 1 est justifiée dans beaucoup d'exemples pratiques ; manufacturiers, ferroviaires ... En effet, les transitions sources correspondent au lancement d'un processus qui est généralement un événement de grande importance. De ce fait, il est rendu contrôlable par l'architecture du système. Les transitions de sortie modélisent quant à elles la fin des processus. Cette fin est généralement un événement observable. Cette propriété assure aussi de pouvoir générer un contrôleur : l'entrée des jetons étant contrôlable et la sortie d'un jeton observable, ces propriétés garantissent qu'il existe une politique de contrôle simple consistant à avoir uniquement un jeton dans l'ensemble des places de processus.*

Propriété 29. *Considérant un S4PR $\mathcal{N} = (P^P \cup P^0 \cup P^R, T, F, W)$, une contrainte GMEC (w, b) génère une place de ressources p^V si les deux conditions suivantes sont vérifiées :*

- $\|w\| \subseteq P^P$
- $w \geq 0$

Démonstration. Une place monitrice p^V appliquant un GMEC (w, b) crée un P-semiflot $X \in \mathcal{P}^+_{\circlearrowleft}(\mathcal{N})$ dans le Réseau contrôlé $\mathcal{N}^C = \mathcal{N} \oplus p^V$ avec $\forall p \in P^P \cup P^0 \cup P^R, X(p) = w(p)$ et $X(p^V) = 1$. D'autre part, une place de ressource p^R est associée à son P-semiflot induit $\vec{p}^R_{\circlearrowleft}$ (voir la définition 51 et la définition 56 avec $|\vec{p}^R_{\circlearrowleft}| \cap P^R = \{p^R\}$, $\|\vec{p}^R_{\circlearrowleft}\| \backslash \{p^R\} \subseteq P^P$ et $\forall M \in \mathcal{R}(\mathcal{N}, M_0)\ \vec{p}^R_{\circlearrowleft}(p^R) = 1$. Dans le cas ou $\|w\| \subseteq P^P$ et $w \geq 0$, on a donc les conditions pour dire que $X = \vec{p}^V_{\circlearrowleft}$. On peut donc en déduire que p^V est une place de ressources.

□

Dans la suite de cette section, les places monitrices pour les S4PR, S3PMR et S3PR seront implémentées en respectant la propriété 29. Par souci de lisibilité, nous nous permettrons de noter les contraintes w comme des applications $w : P^P \to \mathbb{N}^{|P^P|}$ ou de manière équivalente par un vecteur $w \in \mathbb{N}^{|P^P|}$.

Propriété 30. *Étant donné un S4PR sous COP, une contrainte GMEC (w, b) est admissible, si elle respecte les conditions suivantes :*

- $\forall t \in T_{uc}$, $\{p'\} = t^\bullet \cap P^P$ *et* $\{p\} = {}^\bullet t \cap P^P$, $w(p') \leq w(p)$
- $\forall t \in T_{uo}$, $\{p'\} = t^\bullet \cap P^P$ *et* $\{p\} = {}^\bullet t \cap P^P$, $w(p') = w(p)$

Démonstration. Le théorème 7 montre comment sont définies les transitions successeurs et prédécesseurs d'une place de ressources p^R par rapport à son invariant :

$$p^{R\bullet} = \{t \in T \,|\, \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P) > \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P)\}$$

$${}^\bullet p^R = \{t \in T \,|\, \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P) > \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P)\}$$

La définition des transitions incontrôlables et inobservables de la sous-section 3.2.3 implique la relation suivante sur les arcs de la place monitrice p^V :

- $\forall t \in T_{uc}$, $t \notin p^{V\bullet}$
- $\forall t \in T_{uo}$, $t \notin p^{V\bullet} \cup {}^\bullet p^V$

Ici, la place de contrôle est aussi une place de ressource. Pour que le w soit admissible vis-à-vis de la contrôlabilité il faut donc que :

$$\forall t \in T_{uc},\ \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P) \leq \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P)$$

Et pour que le w soit admissible vis-à-vis d'observabilité il faut donc que :

$$\forall t \in T_{uo},\ \vec{p}^R_{\circlearrowleft}(t^\bullet \cap P^P) = \vec{p}^R_{\circlearrowleft}({}^\bullet t \cap P^P)$$

Ce qui conclut la preuve. □

Définition 82. *On définit une relation d'ordre partielle $>$ sur les places de P^P. Elle permet d'établir un ordre entre des places de processus liées par un chemin de transitions incontrôlables. Les places amont d'un processus se trouvent être supérieures au sens de $>$.*

$$\forall p, p' \in P^P,\ \textit{si } \exists \lambda \in \Lambda(\mathcal{N}) \textit{ avec } \lambda = p' \ldots p,\ \lambda \cap T \subseteq T_{uc} \textit{ et } \lambda \cap P \subseteq P^P \textit{ alors } p' > p$$

$\succ$ étant une relation d'ordre partielle, il n'est pas toujours possible d'établir une relation entre deux places $p, p' \in P^P$. L'absence de cycles dans les processus et les transitions sources contrôlables font que $\succ$ est correctement définie.

Propriété 31. *Considérant une contrainte admissible (w, b) appliquée à un S4PR $\mathcal{N}$, s'il existe deux places $p, p' \in P^P$ telles que $p' \succ p$ alors $w(p') \geq w(p)$.*

Démonstration. $p' \succ p$ implique l'existence d'un chemin $\lambda \in \Lambda(\mathcal{N})$ avec $\lambda = p_1 t_1 p_2 t_2 ... t_{n-1} p_n$, $\lambda \cap P \subseteq P^P$ et $\lambda \cap T \subseteq T_{uc}$. L'application de la propriété 30, on obtient, $\forall i \in [1..n]$, $t_i \in T_{uc}$, $\{p_i\} = {}^\bullet t \cap P^P$, $\{p_{i+1}\} = t^\bullet \cap P^P$, $w(p_i) \geq w(p_{i+1})$. Cela conclut la preuve.

□

Définition 83. *Dans un S4PR $\mathcal{N}$ sous COP, un ensemble de places constitue un ensemble connexe inobservable maximal noté $\theta \subseteq P^P$ si :*

1. *$\forall p \in \theta, \exists p' \in \theta$ tel que $\exists \lambda \in \Lambda(\mathcal{N})$ avec $\lambda = p' ... p$, $\lambda \cap P \subseteq P^P$ et $\lambda \cap T \subseteq T_{uo}$*
2. *Le sous-Réseau de $\mathcal{N}$, $\mathcal{N}^\theta = (\theta, T^\theta, F^\theta, W^\theta)$ avec $\mathcal{N}^\theta \subset \mathcal{N}$ est faiblement connexe.*
3. *$\theta^\circ \cap T_{uo} = \varnothing$ et ${}^\circ\theta \cap T_{uo} = \varnothing$*

L'ensemble de tous les ensembles connexes inobservables maximaux de $\mathcal{N}$ est noté $\Theta(\mathcal{N})$.

La définition 83 définit les ensembles connexes inobservables maximaux comme des ensembles de places liés par des transitions inobservables (point 1). Le point 3, stipule qu'il ne peut pas y avoir de transition successeur ou prédécesseur stricte inobservable. L'ensemble est donc maximal, dans le sens où il ne peut être plus grand, sans violer le point 2.

Propriété 32. *Dans un S4PR $\mathcal{N}$, les ensembles connexes inobservables maximaux sont disjoints :*

$$\forall \theta, \theta' \in \Theta(\mathcal{N}) \,|\, \theta \cap \theta' = \varnothing$$

Démonstration. Elle est directe puisque les éléments de $\Theta(\mathcal{N})$ sont maximaux et connexes selon les points 2 et 3 de la définition 83. □

Propriété 33. *Si on considère une contrainte GMEC (w, b) d'un S4PR $\mathcal{N}$, la condition suivante doit être nécessairement respectée pour que (w, b) soit admissible vis-à-vis de la COP :*

$$\forall \theta \in \Theta(\mathcal{N}), \forall p, p' \in \theta,\ w(p) = w(p')$$

Démonstration. D'après la propriété 30, $\forall t \in T_{uo}$, $\{p'\} = t^\bullet \cap P^P$ et $\{p\} = {}^\bullet t \cap P^P$, $w(p') = w(p)$, i.e. deux places consécutives d'un RdP appartenant à un même ensemble θ ont le même coefficient dans w. En utilisant le point 2 de la définition 83, par récurrence on étend la propriété à toutes les places de l'ensemble θ. □

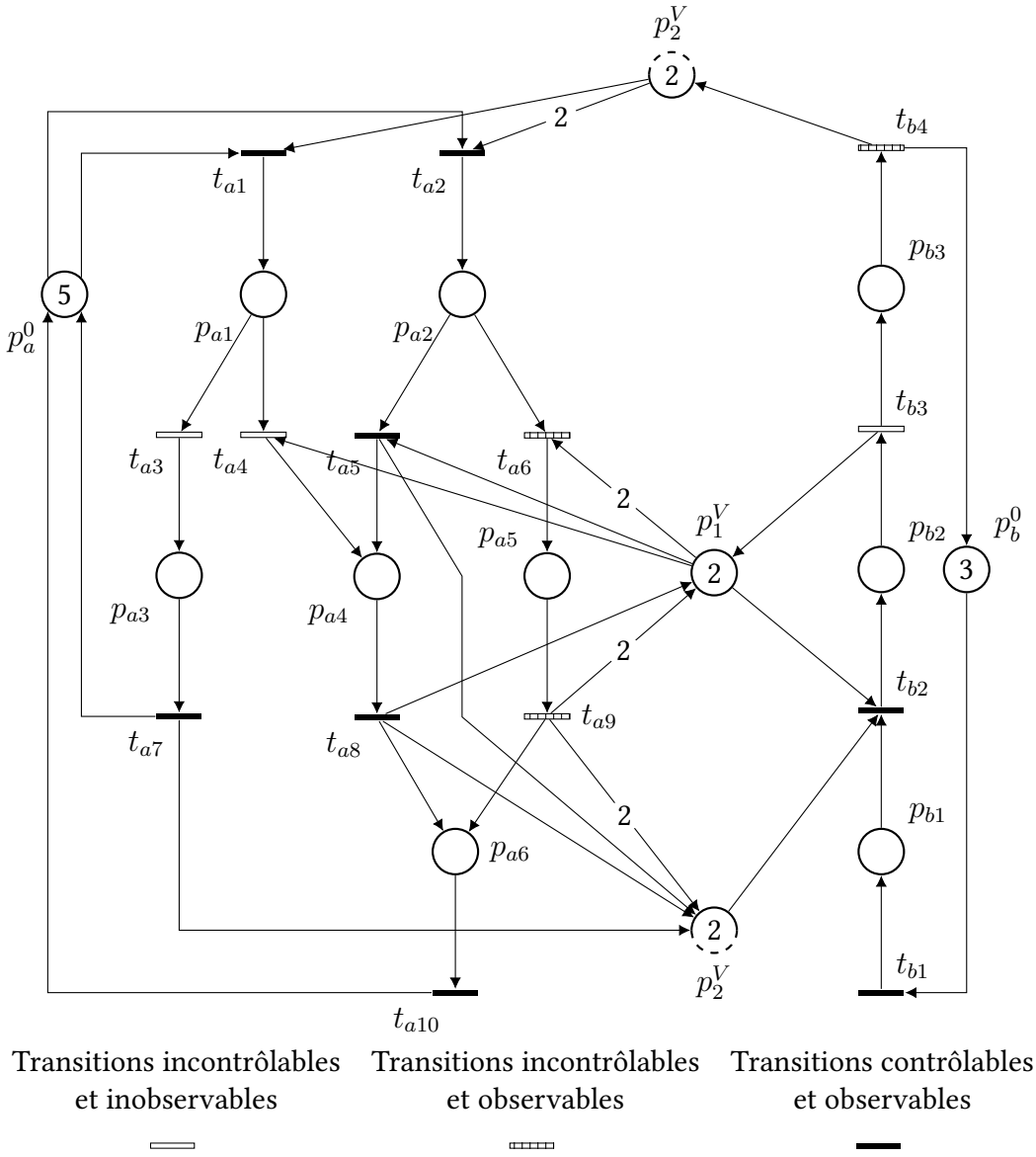


FIGURE 5.1 – Génération de places monitrices pour deux processus sous COP

La figure 5.1 montre deux Réseaux de processus $\mathcal{I} = \{a, b\}$ qui forment un Réseau de Petri $\mathcal{N}$. Une contrainte non-admissible (w_1, b) et une contrainte admissible (w_2, b) sont détaillées dans le tableau 5.1 avec $b = 2$. Les places monitrices p_1^V et p_2^V font respecter respectivement (w_1, b) et (w_2, b), formant le Réseau contrôlé $\mathcal{N}^{\mathcal{C}} = \mathcal{N} \oplus p_1^V \oplus p_2^V$. Par mesure de lisibilité, la place p_2^V a été dupliquée. Uniquement trois places peuvent être comparées avec l'opérateur $\succ$: $p_{a2} \succ p_{a5} \succ p_{a6}$. $\mathcal{N}$ comporte deux ensembles connexes inobservables maximaux $\Theta(\mathcal{N}) = \{\theta_1, \theta_2\}$ avec $\theta_1 = \{p_{a1}, p_{a3}, p_{a4}\}$ et $\theta_2 = \{p_{b2}, p_{b3}\}$. Le vecteur de contrainte w_1 ne respecte pas la propriété 30 définissant les contraintes admissibles vis-à-vis de la COP. En effet, pour que w_1 soit admissible, il faudrait que $w_1(p_{b2}) = w_1(p_{b3})$ et $w_1(p_{a2}) \geq w_1(p_{a5})$. On peut directement vérifier sur la figure 5.1 que la place p_1^V a des arcs liés à des transitions inobservables (t_{a4} et t_{b3}) et un arc lié à une transition incontrôlable (t_{a6}),

et donc que le contrôle qu'elle met en œuvre n'est pas admissible. La place p_2^V issue de la contrainte admissible (w_2, b) respecte les conditions de contrôlabilité et d'observabilité. La propriété 31 est donc bien vérifiée. En effet, comme $p_{a2} \succ p_{a5} \succ p_{a6}$, on a bien $w_2(p_{a2}) \geq w_2(p_{a5}) \geq w_2(p_{a6})$. On remarque que $\forall p \in \theta_1$, $w_2(p) = 1$ et $\forall p \in \theta_2$, $w_2(p) = 1$ conformément à la propriété 33.

	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p_{a5}	p_{a6}	p_{b1}	p_{b2}	p_{b3}
w_1	0	0	0	1	2	0	0	1	0
w_2	1	2	1	1	2	0	0	1	1

TABLE 5.1 – Tableau des vecteurs de contrainte w_1 et w_2

Une contrainte (w, b) quelconque ne peut pas être directement sélectionnée pour la génération d'une place monitrice dans le cas d'un Réseau de Petri sous COP. La section 3.2 a proposé un rappel de l'état de l'art en ce qui concerne la transformation de contraintes non-admissibles en contraintes admissibles dans le cas général des RdPs. Dans le cas des sous-classes de RdP étudiées dans cette section et des restrictions appliquées aux contraintes (propriété 29), la transformation de contraintes peut être faite plus simplement que dans le cas général des RdPs. Comme nous l'avons vu au début de cette section, les conditions pour qu'une contrainte soit admissible concernent le vecteur de contrainte w. La borne b, quant à elle, sert uniquement au calcul du nombre de jetons à mettre dans la place monitrice. Aussi, pour la transformation d'une contrainte non-admissible, nous allons définir différentes classes de vecteurs de contraintes en fonction de caractéristiques structurelles de sous-Réseaux du modèle étudié. Un vecteur de contrainte sera admissible s'il appartient à l'ensemble de ces classes.
Trois grandes classes de vecteurs de contrainte sont ainsi définies pour aider à caractériser un vecteur de contrainte admissible w' à partir d'un vecteur de contrainte non-admissible w.

Définition 84. *Pour un S4PR $\mathcal{N}$, la classe $\mathcal{C}^C$ regroupe l'ensemble des vecteurs de contrainte qui sont admissibles vis-à-vis des transitions incontrôlables.*

$$\mathcal{C}^C = \{w \in \mathbb{N}^{|P^P|} \mid \forall p, p' \in P^P \text{ tel que } p' \succ p, w(p') \geq w(p)\}$$

Définition 85. *Pour un S4PR $\mathcal{N}$, la classe $\mathcal{C}^O$ regroupe l'ensemble des vecteurs de contrainte qui sont admissibles vis-à-vis des transitions inobservables.*

$$\mathcal{C}^O = \{w \in \mathbb{N}^{|P^P|} \mid \forall \theta \in \Theta(\mathcal{N}), \forall p, p' \in \theta, w(p') = w(p)\}$$

Propriété 34. *Si un vecteur de contrainte $w \in \mathbb{N}^{|P^P|}$ dans un S4PR $\mathcal{N}$ est inclus dans les classes $\mathcal{C}^O$ et $\mathcal{C}^C$ i.e $w \in \mathcal{C}^O \cap \mathcal{C}^C$, alors w est admissible.*

Démonstration. Directe à partir des propriétés 31, 33 et 30. □

Définition 86. *Pour un S4PR $\mathcal{N}$ et, un vecteur de contrainte $w \in \mathbb{N}^{|P^P|}$, la classe $\mathcal{C}_w$ regroupe l'ensemble des vecteurs de contrainte qui sont plus restrictives que w.*

$$\mathcal{C}_w = \{w' \in \mathbb{N}^{|P^P|} \mid w \leq w'\}$$

Étant donné une place $p \in P^P$, $w(p)$ correspond au nombre de jetons issus de la place monitrice correspondante, nécessaire pour avoir un jeton dans la place de processus p. Considérer un vecteur w', tel que $w' \geq w$, signifie que $\forall p \in P^P, w'(p) \geq w(p)$. Dans le cas où $w(p) = 0$ et $w'(p) > 0$, cela signifie que la place monitrice va concerner d'avantage de places de processus. Donc le vecteur de contrainte w' est plus restrictif car plus de places auront leur marquage borné par celui de la place monitrice. Dans le cas $w'(p) > w(p) > 0$, cela signifie que la place p, une fois marquée, aura « capturé » plus de jetons du marquage initial de la place monitrice, cela au détriment d'autres places. On réduit ainsi d'autant les marquages accessibles. Ces deux cas rendent le vecteur de contrainte plus restrictif.
Nous allons à présent définir des fonctions qui nous permettrons algorithmiquement, de rechercher un vecteur de contrainte admissible optimal w' à partir d'un vecteur de contrainte non-admissible w.

Définition 87. *On note l'application $\mathcal{F}^C : \mathbb{N}^{|P^P|} \to \mathbb{N}^{|P^P|}$, la transformation d'un vecteur de contrainte $w \in \mathbb{N}^{|P^P|}$ qui permet à partir de w, d'obtenir le plus petit vecteur de $\mathcal{C}_w \cap \mathcal{C}^C$. Considérant $w^C = \mathcal{F}^C(w)$, w^C est définie par :*
$\forall p \in P^P$:

$$w^C(p) = \begin{cases} \textit{si } \exists p' \in P^P \textit{ avec } p \succ p', & w^C(p) = \max w(p'') \ \forall p'' \in P^P \textit{ tel que } p \succ p'' \\ \textit{si } \nexists p' \in P^P \textit{ avec } p \succ p', & w^C(p) = w(p) \end{cases}$$

L'application $\mathcal{F}^C$ recherche le vecteur de contrainte admissible qui soit le moins restrictif (afin de garantir une bonne permissivité) et qui respecte la propriété 31. Du point de vue de la génération de places monitrices cela revient à remonter l'arc $f = (p^V, t_{uc})$ avec $t_{uc} \in T_{uc}$ jusqu'à la/les première(s) transition(s) contrôlable(s) en remontant vers l'amont du processus.
De manière analogue on définit l'application $\mathcal{F}^O$.

Définition 88. *On note l'application $\mathcal{F}^O : \mathbb{N}^{|P^P|} \to \mathbb{N}^{|P^P|}$, la transformation d'un vecteur de contrainte $w \in \mathbb{N}^{|P^P|}$ qui permet à partir de w d'obtenir le plus petit vecteur de $\mathcal{C}_w \cap \mathcal{C}^O$. Considérant $w^O = \mathcal{F}^O(w)$, w^O est définie par :*
$\forall p \in P^P$:

$$w^O(p) = \begin{cases} \textit{si } \exists \theta \in \Theta(\mathcal{N}) \textit{ avec } p \in \theta, & w^O(p) = \max w(p'') \ \forall p'' \in \theta \\ \textit{si } \nexists \theta \in \Theta(\mathcal{N}) \textit{ avec } p \in \theta, & w^O(p) = w(p) \end{cases}$$

L'application $\mathcal{F}^O$ considère chaque sous-réseau de transitions non-observables défini par un θ donné. Elle recherche la valeur maximale des $w(p)$ d'un ensemble θ pour l'appliquer comme valeur $w'(p)$ de tous les éléments de θ. Cela permet de rendre w' admissible puisqu'elle vérifie la propriété 33.

Définition 89. *Si on considère un S4PR $\mathcal{N}$ et une contrainte (w, b), la classe des contraintes $\mathcal{C}_w^A$ constitue l'ensemble des vecteurs de contraintes qui garantissent que les états protégés par (w, b) ne seront pas atteints et que $\forall \bar{w} \in \mathcal{C}_w^A$, $\bar{w}$ est admissible vis-à-vis de la COP.*

Définition 90. *On note $\breve{w}$ la valeur minimale de $\mathcal{C}_w^A$:*

$$\breve{w} = \min_{\bar{w} \in \mathcal{C}_w^A} (\bar{w})$$

Algorithme 4 Algorithme pour le calcul de $\breve{w}$

Entrée : Un S4PR $\mathcal{N}$ et un vecteur de contrainte w
Sortie : $\breve{w}$

$\breve{w} \leftarrow w$
$Continuer \leftarrow Vrai$
tant que $Continuer = Vrai$ **faire**
  si $\breve{w} \notin \mathcal{C}^C$ **alors**
    $\breve{w} \leftarrow \mathcal{F}^C(\breve{w})$
  sinon si $\breve{w} \notin \mathcal{C}^O$ **alors**
    $\breve{w} \leftarrow \mathcal{F}^O(\breve{w})$
  sinon
    $Continuer \leftarrow Faux$
  fin si
fin tant que
retourner $\breve{w}$

L'algorithme 4 permet d'obtenir $\breve{w}$ de la définition 90. Il itère les opérations $\breve{w} \leftarrow \mathcal{F}^C(\breve{w})$ et $\breve{w} \leftarrow \mathcal{F}^O(\breve{w})$ jusqu'à ce que $\breve{w} \in \mathcal{C}^C \cap \mathcal{C}^O$. Comme $\mathcal{F}^C$ et $\mathcal{F}^O$ sont des fonctions croissantes, par rapport à $\geq$ et que $\breve{w}$ est initialisé à w, $\breve{w} \in \mathcal{C}_w^A$ à la fin de l'algorithme.

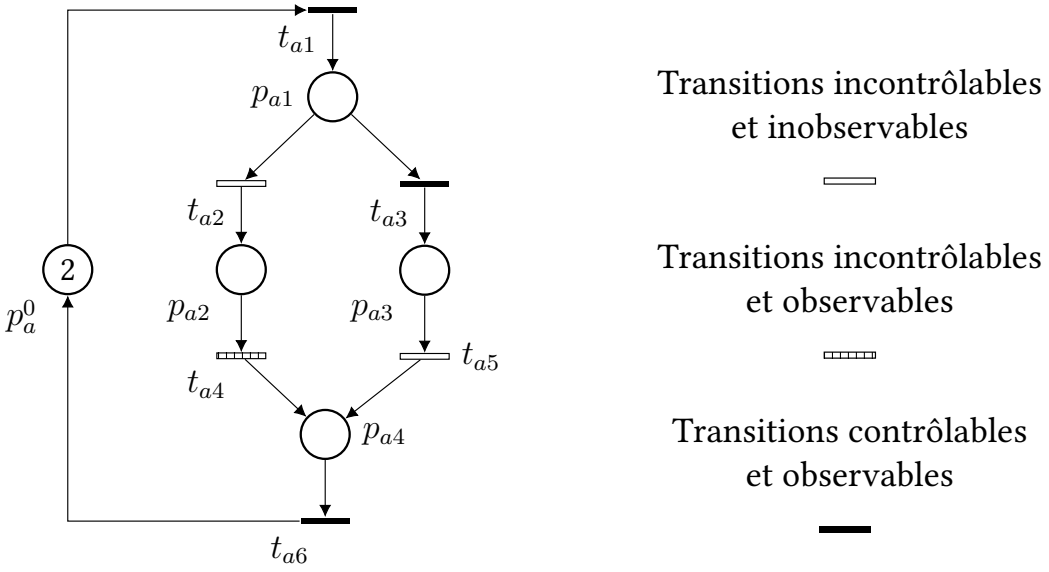


FIGURE 5.2 – Réseau de processus sous COP

La figure 5.2 montre un Réseau de processus $\mathcal{N}^P$ sous COP. $\mathcal{N}^P$ comporte deux ensembles connexes inobservables maximaux $\Theta(\mathcal{N}^P) = \{\theta_1, \theta_2\}$ avec $\theta_1 = \{p_{a3}, p_{a4}\}$ et $\theta_2 = \{p_{a1}, p_{a2}\}$ et deux places p_{a2} et p_{a4} comparables par $>$ tel que $p_{a2} > p_{a4}$. On considère une contrainte w décrite en première ligne du tableau 5.2. Le vecteur de contraintes initiales w est tel que $w(p_{a3}) = 1$. Cette contrainte consiste donc à limiter le nombre de jetons dans la place p_{a3}. Normalement, il suffirait de construire une place monitrice p^V, telle que $Pre(p^V, t_{a3}) = 1$ et $Post(p^V, t_{a5}) = 1$. Mais la

transition t_{a5} est non observable. Il n'est donc pas possible d'avoir un arc postérieur entre cette transition et la place monitrice p^V. Le tableau 5.2 montre les détails de la transformation de w via l'algorithme 4 de manière à obtenir la contrainte admissible $\breve{w}$. La première itération correspond à la 3e ligne du tableau. À l'initialisation, on a $\breve{w}(p_{a2}) = \breve{w}(p_{a4}) = 0$, comme $p_{a2} \succ p_{a4}$ on a donc $\breve{w} \in \mathcal{C}^C$. Par contre $\breve{w}(p_{a3}) \neq \breve{w}(p_{a4})$, ce qui implique que $\breve{w} \notin \mathcal{C}^O$. Par conséquent, la première itération (ligne 3 du tableau) fait appel à la fonction $\mathcal{F}^O$ (étape 7 de l'algorithme) afin de calculer un nouveau vecteur contrainte $\breve{w} = [0, 0, 1, 1]^T$). Ce vecteur contrainte impliquerait, si on essayait de générer une place de monitrice, d'avoir t_{a6} comme seule transition prédécesseur de p^V. Comme la transition t_{a6} est observable $\breve{w} \in \mathcal{C}^O$. Lors de la 2e itération, p_{a2} et p_{a4} ne respectent plus la condition de contrôlabilité. En effet, $\breve{w}(p_{a2}) = 0$ et $\breve{w}(p_{a4}) = 1$, comme $p_{a2} \succ p_{a4}$, cela induit que $\breve{w} \notin \mathcal{C}^C$. La 2e itération fait donc appel à la fonction $\mathcal{F}^C$ (étape 5 de l'algorithme) afin de calculer un nouveau vecteur contrainte $\breve{w} = [0, 1, 1, 1]^T$). Lors de la 3e itération, il est de nouveau fait appel à la fonction $\mathcal{F}^O$ afin d'ajouter la place p_{a1} dans l'ensemble des places contraintes car $p^{a4} \in \theta$. A la 4e itération, l'algorithme retourne le nouveau vecteur contrainte admissible $\breve{w} \in \mathcal{C}_w^A$ qui va permettre de générer la place monitrice p^V avec $p^{V\bullet} = \{t_{a1}\}$ et ${}^\bullet p^V = \{t_{a6}\}$.

	p_{a1}	p_{a2}	p_{a3}	p_{a4}	
w	0	0	1	0	
$\breve{w}$	0	0	1	0	initialisation
$\breve{w}$	0	0	1	1	$\mathcal{F}^O$
$\breve{w}$	0	1	1	1	$\mathcal{F}^C$
$\breve{w}$	1	1	1	1	$\mathcal{F}^O$

Table 5.2 – Exemple de l'utilisation de l'algorithme 4 pour le calcul d'un vecteur contrainte admissible

Notre objectif est à présent d'analyser si la solution donne un contrôleur maximalement permissif. Pour cela, il est nécessaire de comparer notre solution avec celle obtenue en appliquant la SCT de Ramadge et Wonham. La figure 5.3 montre le graphe des marquages accessibles du Réseau de Petri de la figure 5.2. Les marquages qui respectent la contraintes $\breve{w}$ sont entourés en bleu. Seul l'état $M_{11} : 2 * p_{a3}$ (en rouge) viole la contrainte $(w, 1)$. Sous la contrainte de contrôlabilité et observabilité partielle, la réalisation d'un observateur est nécessaire si on souhaite trouver un contrôleur maximalement permissif, en appliquant la SCT. L'observateur du graphe d'accessibilité est représenté en figure 5.4. Cet observateur comprend 9 états. L'observateur comprend un état interdit $\{M_{11}, M_{13}, M_{14}\}$ (état en rouge). La synthèse du contrôleur depuis l'observateur est directe puisque cet état ne peut être atteint que par le tir de t_3 qui est contrôlable. Le superviseur optimal comporte donc huit états, alors que $\breve{w}$ n'autorise que quatre états. Cette observation nous permet de conclure que cette transformation des contraintes ne permet pas d'exercer un contrôle maximalement permissif.

Il est démontré que la génération d'une place monitrice à partir d'un vecteur transformé n'exerce pas un contrôle optimal. Un résultat similaire a été préalablement mis en évidence dans la sous-section 3.2.4, où il est montré que dans le cas général des Réseaux de Petri sous COP, une place motrice n'est pas toujours en

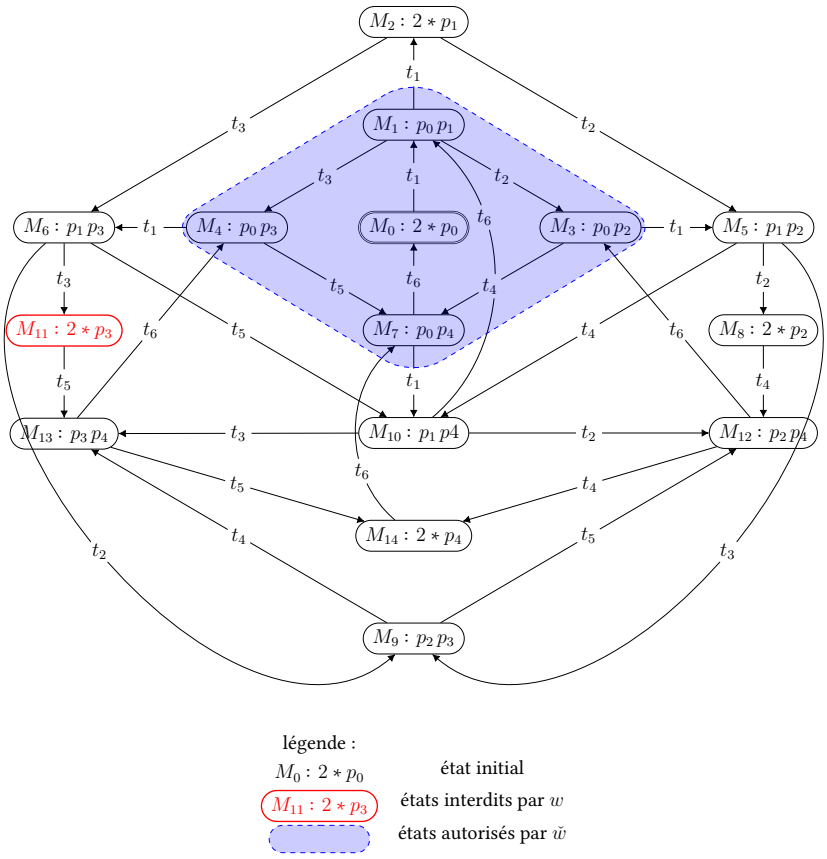


FIGURE 5.3 – Graphe des marquages accessibles du Réseau de Petri de la figure 5.2

mesure d'appliquer un contrôle optimal. Cependant les Réseaux de processus sont des machines à état, ce qui implique qu'il ne peut y avoir de synchronisme dans un Réseau de Processus i.e $\forall t \in T, \in |^\bullet t| = 1$. Le synchronisme, dans le cas des réseaux sous COP, induit que l'espace des marquages accessibles n'est pas nécessairement convexe (voir la remarque sur la figure 3.11) et donc qu'une place monitrice ne peut pas engendrer un contrôle optimal. Dans l'exemple de la figure 5.2, malgré les propriétés d'un Réseau de processus, le contrôle par places monitrices n'est pas nécessairement optimal au sens de la SCT. Cependant, cette méthode de contrôle présente des propriétés très intéressantes d'un point de vue de la mise en place d'un contrôle itératif. En effet, la propriété 29 stipule qu'une place de contrôle telle que définie dans cette section a les même propriétés qu'une place de ressource. Un RAS-PN auquel on ajoute une place de contrôle reste donc une RAS-PN. Il est donc possible de mettre en place une stratégie de contrôle ou itérativement un siphon est extrait puis contrôlé.

5.2 Contrôles des siphons dans les RAS-PN sous COP par places monitrices

Sous COP, le contrôle des siphons ne peut être fait directement comme présenté dans 4.4.2. Ici il est nécessaire de prendre en compte les transitions contrôlables

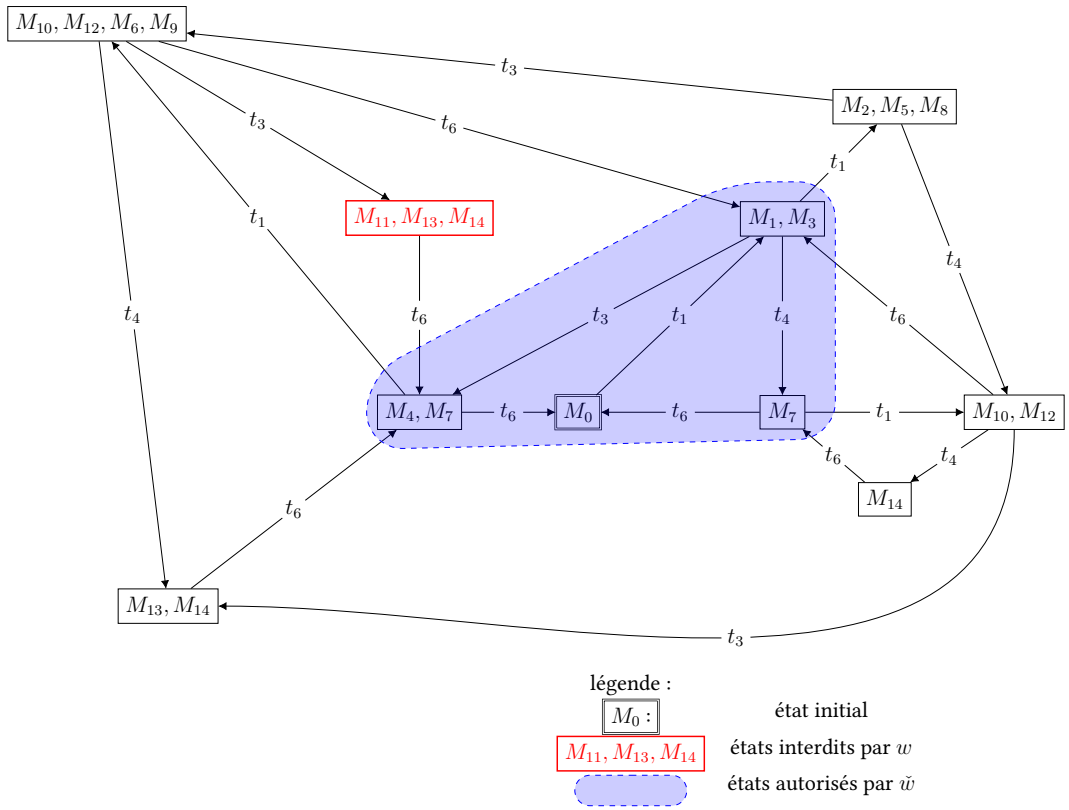


FIGURE 5.4 – Observateur du graphe des marquages accessibles de la figure 5.3

et observables avant de générer une place monitrice. Nous présenterons en quoi la COP influe sur la définition 75 pour la génération d'une place monitrice p^V (contrôle de l'ensemble complémentaire d'un siphon S) et la définition 77 pour la génération d'une place monitrice p^W (contrôle de l'ensemble complémentaire d'un siphon S avec les arcs sur les transitions sources).

Définition 91. *Considérons un siphon S et un mauvais marquage M d'un S4PR convenablement marqué $(\mathcal{N}, M_0)$ avec $\mathcal{N} = (P^P \cup P^R \cup P^0, T, F, W)$. Il est possible d'ajouter une place monitrice p^V à $(\mathcal{N}, M_0)$ pour obtenir $(\mathcal{N}^C, M_0^C)$ tel que $\nexists M' \in \mathcal{R}(\mathcal{N}^C, M_0^C)$ avec $M|_{P^P} = M'|_{P^P}$, et que p^V respecte les conditions de contrôlabilité et d'observabilité.*

On définit le vecteur de contrainte admissible $\breve{w}_{[S]}$ qui permet de générer la place monitrice p^V. $\breve{w}_{[S]}$ est obtenu à partir du coût en ressource $k_{[S]}$ associé au siphon complémentaire $[S]$. En considérant $k_{[S]}$ comme un vecteur de contrainte, $\breve{w}_{[S]}$ est obtenu directement en appliquant l'algorithme 4 à $k_{[S]}$.

Et le marquage initial de p^V est défini comme :

$$M_0^C(p^V) = M_0(S) - M(S) - 1$$

Le Réseau contrôlé est donc aussi un S4PR avec $\mathcal{N}^C = (P^P \cup P^{R\prime} \cup P^0, T, F', W')$ avec $P^{R\prime} = \{p^V\} \cup P^R$.

Propriété 35. *L'ajout de la place p^V n'entraine pas la mort de transitions.*

Démonstration. L'algorithme 4 applique de manière itérative les opérations $\mathcal{F}^C$ et $\mathcal{F}^O$. Ces opérations égalisent les valeurs du vecteur de contrainte sur des ensembles de places de processus, ainsi $\max(\check{w}) = \max(w)$. $(\mathcal{N}^C, M_0^C)$ est donc convenablement marqué, ce qui conclut la preuve. □

Dans le cas d'une place monitrice p^W (de la définition 77), ses arcs sortants sont dirigés vers les transitions sources. Il n'est plus nécessaire de prendre en compte la notion de contrôlabilité. En effet, il est interdit d'avoir des arcs allant de places monitrices vers des transitions incontrôlables. Hors, la propriété 1 stipule que les transitions sources sont contrôlables. Une place monitrice p^W respecte donc naturellement la condition de contrôlabilité. Les transitions inobservables restent cependant à prendre en compte.

Définition 92. *On note l'application $\mathcal{F}^{\downarrow} : \mathbb{N}^{|P^P|} \to \mathbb{N}^{|P^P|}$, la transformation d'un vecteur contrainte $w \in \mathbb{N}^{|P^P|}$ qui permet à partir de w d'obtenir le plus petit vecteur $w^{\downarrow}$ tel que la place monitrice obtenue à partir de $w^{\downarrow}$ a tous ses arcs redirigés vers les transitions sources. $\mathcal{F}^{\downarrow}(w) = w^{\downarrow}$ est définie par :*

$$w^{\downarrow}(p) = \max_{p' \in p^{\downarrow}} w(p') \quad \forall p \in P^P$$

Algorithme 5 Algorithme pour le calcul de $\check{w}^{\downarrow}$

Entrée : Un S4PR $\mathcal{N}$ et un vecteur de contrainte w
Sortie : $\check{w}^{\downarrow}$

1: $\check{w}^{\downarrow} \leftarrow w$
2: $Continuer \leftarrow Vrai$
3: **tant que** Continuer=Vrai **faire**
4: $\quad \check{w}^{\downarrow} \leftarrow \mathcal{F}^O(\check{w}^{\downarrow})$
5: $\quad \check{w}^{\downarrow} \leftarrow \mathcal{F}^{\downarrow}(\check{w}^{\downarrow})$
6: $\quad$ **si** $\check{w}^{\downarrow} \in \mathcal{C}^O$ **alors**
7: $\quad\quad Continuer \leftarrow Faux$
8: $\quad$ **fin si**
9: **fin tant que**
10: retourner $\check{w}^{\downarrow}$

L'algorithme 5 permet d'obtenir à partir d'une contrainte w, une contrainte $\check{w}^{\downarrow}$ admissible vis-à-vis de la COP. Une place monitrice p^W générée à partir de $\check{w}^{\downarrow}$ a ses arcs dirigés vers les transitions sources. L'algorithme 5 applique itérativement les opérations $\mathcal{F}^O$ puis $\mathcal{F}^{\downarrow}$. Cet algorithme s'arrête, si après l'application de $\mathcal{F}^{\downarrow}$, la contrainte obtenue respecte la COP (i.e $\check{w}^{\downarrow} \in \mathcal{C}^O \cap \mathcal{C}^C$).

Définition 93. *Considérons un siphon S et un mauvais marquage M d'un S4PR sous COP convenablement marqué $(\mathcal{N}, M_0)$, avec $\mathcal{N} = (P^P \cup P^R \cup P^0, T, F, W)$. Il est possible d'ajouter une place monitrice p^W à $(\mathcal{N}, M_0)$ pour obtenir $(\mathcal{N}^C, M_0^C)$ tel que $\nexists M' \in \mathcal{R}(\mathcal{N}^C, M_0^C)$ avec $M|_{P^P} = M'|_{P^P}$ et que p^W ne puisse être compris dans un autre siphon de $\mathcal{N}^C$.*

p^W est générée à partir du vecteur contrainte $w^{\downarrow}_{[S]}$ obtenue en appliquant l'algorithme 5 sur $k_{[S]}$. Et le marquage initial :

$$M_0^C(p^W) = M_0(S) - M(S) - 1$$

Le Réseau contrôlé est donc aussi un S4PR avec $\mathcal{N}^{\mathcal{C}} = (P^P \cup P^{R\prime} \cup P^0, T, F', W')$ *avec* $P^{R\prime} = \{p^W\} \cup P^R$.

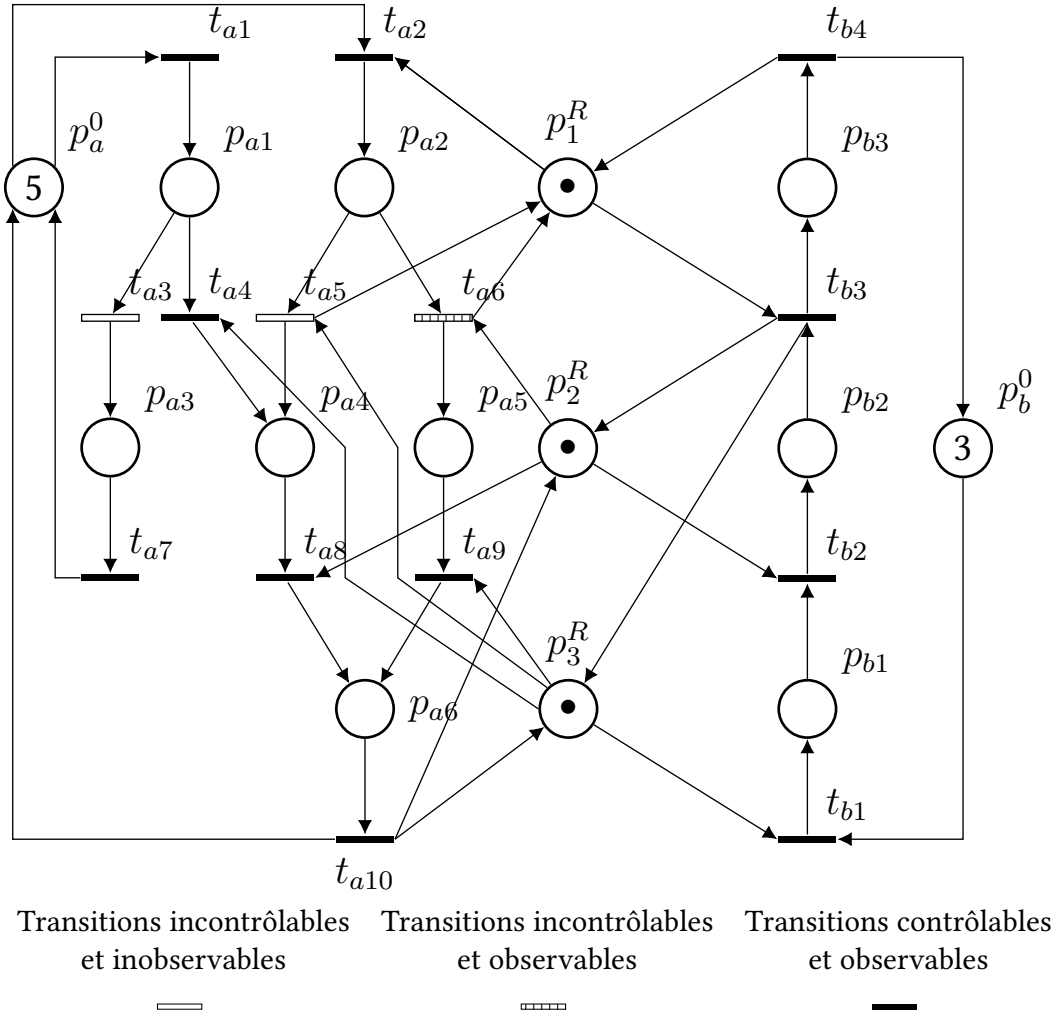


FIGURE 5.5 – S3PMR $\mathcal{N}$ sous COP contenant des siphons S_1 et S_2

La figure 5.5 représente un S3PMR $\mathcal{N}$ qui comporte deux processus $\mathcal{I} = \{a, b\}$ et trois places de ressources $P^R = \{p_1^R, p_2^R, p_3^R\}$. Les places de ressources font partie intégrante du modèle et ne sont pas considérées comme des places monitrices. Ces places ne sont pas nécessairement obligées de respecter les conditions d'admissibilité vis-à-vis de la COP. $\mathcal{N}$ contient deux siphons S_1 et S_2. S_1 et S_2 entraîne des blocages respectivement pour les marquages M_1 et M_2 tel que $M_1(S_1) = 0$ et $M_2(S_2) = 0$. $\mathcal{N}$ étant un S3PMR, il est uniquement nécessaire de s'assurer que les siphons S_1 et S_2 ne se vident pas. Les informations relatives aux siphons S_1 et S_2 sont récapitulées dans le tableau 5.3. On souhaite contrôler le siphon S_1 par une place de contrôle p^V suivant la définition 93, réduisant ainsi le nombre de jetons dans l'ensemble complémentaire $[S_1]$. Pour le siphon S_2, une place de contrôle p^W suivant la définition 93 est générée de façon à ce qu'elle ne crée pas de nouveaux siphons. Les étapes intermédiaires de calcul de $\check{w}_{[S_1]}$ sont détaillées dans le tableau 5.4 et ceux de $\check{w}^{\downarrow}_{[S_2]}$ dans le tableau 5.5. Le Réseau contrôlé, après ajout de p^V et p^W, est représenté en figure 5.6, sans les places de ressources par mesure de lisibilité. Les deux places

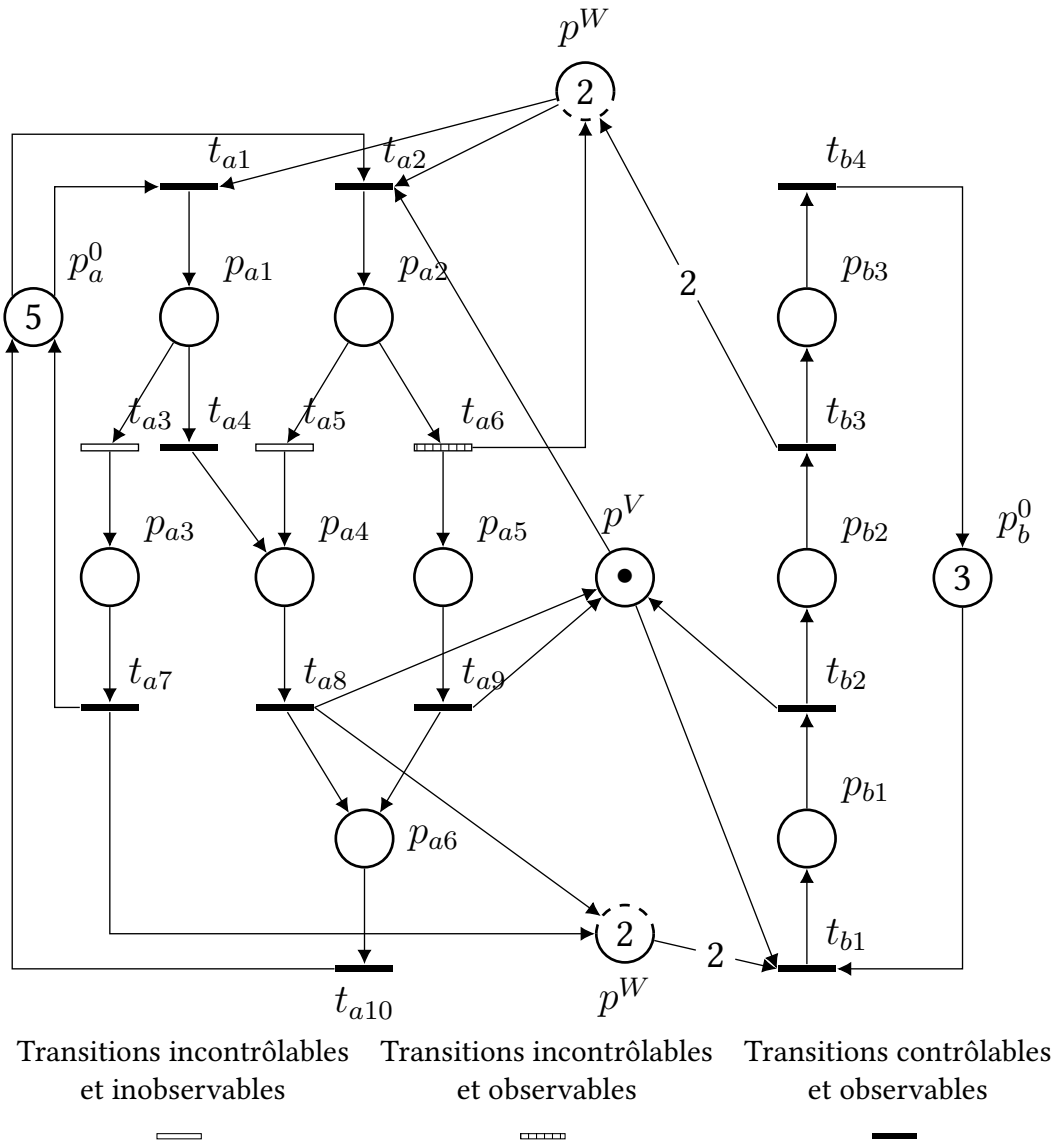


FIGURE 5.6 – S3PMR $\mathcal{N}^C$ sans les places de ressources

de contrôle p^V et p^W sont admissibles vis-à-vis de la COP. Le marquage des places est donc obtenu par la formule suivante $M_0^C(p^V) = M_0(S_1) - M_1(S_1) - 1 = 1$ et $M_0^C(p^W) = M_0(S_2) - M_2(S_2) - 1 = 2$. On remarque que la place p^W redirige ces arcs sortants vers les transitions sources, de manière à ce qu'aucun siphon de $\mathcal{N}^C$ n'inclut p^W.

5.3 Algorithmes de contrôle pour les RAS-PN sous COP

Dans cette section, nous présenterons deux algorithmes de contrôle pour les RAS-PN sous COP. Le premier algorithme est inspiré des travaux de [HJXC06]. Il est applicable uniquement aux S3PMR et aux S3PR sous COP. Le second algorithme s'inspire des travaux de [ZDW+19]. Il est plus général car il s'applique aux S4PR

	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p_{a5}	p_{a6}	p_{b1}	p_{b2}	p_{b3}	p_1^R	p_2^R	p_3^R
S_1	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✓	✓
$[S_1]$	✗	✗	✗	✗	✓	✗	✓	✗	✗			
$k_{[S_1]}$	0	0	0	0	1	0	1	0	0			
$\check{w}_{[S_1]}$	0	1	0	1	1	0	1	0	0			
S_2	✗	✗	✗	✓	✗	✓	✓	✗	✓	✓	✓	✓
$[S_2]$	✗	✓	✗	✗	✗	✗	✗	✓	✗			
$k_{[S_2]}$	0	1	0	0	0	0	0	2	0			
$\check{w}^{\downarrow}_{[S_2]}$	1	1	1	1	0	0	2	2	0			

TABLE 5.3 – Informations relatives aux siphons S_1 et S_2

	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p_{a5}	p_{a6}	p_{b1}	p_{b2}	p_{b3}	
$k_{[S_1]}$	0	0	0	0	1	0	1	0	0	
$\check{w}_{[S_1]}$	0	0	0	0	1	0	1	0	0	⤸ initialisation
$\check{w}_{[S_1]}$	0	1	0	0	1	0	1	0	0	⤸ $\mathcal{F}^C$
$\check{w}_{[S_1]}$	0	1	0	1	1	0	1	0	0	⤸ $\mathcal{F}^O$

TABLE 5.4 – Application de l'algorithme 4 pour l'obtention de $\check{w}_{[S_1]}$

	p_{a1}	p_{a2}	p_{a3}	p_{a4}	p_{a5}	p_{a6}	p_{b1}	p_{b2}	p_{b3}	
$k_{[S_2]}$	0	1	0	0	0	0	0	2	0	
$\check{w}^{\downarrow}_{[S_2]}$	0	1	0	0	0	0	0	2	0	⤸ initialisation
$\check{w}^{\downarrow}_{[S_2]}$	0	1	0	0	0	0	2	2	0	⤸ $\mathcal{F}^{\downarrow}$
$\check{w}^{\downarrow}_{[S_2]}$	0	1	0	1	0	0	2	2	0	⤸ $\mathcal{F}^O$
$\check{w}^{\downarrow}_{[S_2]}$	1	1	0	1	0	0	2	2	0	⤸ $\mathcal{F}^{\downarrow}$
$\check{w}^{\downarrow}_{[S_2]}$	1	1	1	1	0	0	2	2	0	⤸ $\mathcal{F}^O$

TABLE 5.5 – Application de l'algorithme 5 pour l'obtention de $\check{w}^{\downarrow}_{[S_1]}$

sous COP. Dans la littérature, l'algorithme issu de [HJXC06] est nommé H2 et celui de [ZDW+19] nommé Z2. Nous proposons des algorithmes qui en sont dérivés, nommés respectivement H2' et Z2'.

5.3.1 Algorithme H2'

L'algorithme H2' s'applique aux S3PMR sous COP. Il est inspiré de l'algorithme H2 présenté dans [HJXC06]. H2 propose une solution de génération de places monitrices de manière itérative, de façon à éviter des situations de blocages dans un S3PMR sous contrôlabilité et observabilité totale. Cet algorithme se base sur la méthode d'extraction présentée dans [Cha09] qui permet d'extraire d'un Réseau de Petri ordinaire un siphon strict minimal potentiellement vidable. H2 distingue deux types de siphons extraits d'un S3PMR $\mathcal{N}$:

— Le premier type de siphon, nommé siphon ordinaire et noté S^O est caractérisé par $\max_{p\in[S^O]} k_{[S^0]}(p) = 1$. Lorsque un siphon ordinaire S^O est extrait, H2 génère une place monitrice p^V qui limite le nombre de jetons dans $[S^O]$ suivant la définition 75. De par les propriétés de S^O, la place monitrice p^V

est ordinaire (pas de pondération de ses arcs, i.e $\forall t \in p^{V\bullet}, W(p^V, t) = 1$ et $\forall t \in {}^\bullet p^V, W(t, p^V) = 1$). Lorsque p^V est ajoutée à $\mathcal{N}$, le Réseau contrôlé $\mathcal{N}^\mathcal{C} = \mathcal{N} \oplus p^V$ reste un S3PMR.

— Le deuxième type de siphon, nommé siphon pondéré et noté S^W est caractérisé par $\max_{p\in[S^W]} k_{[S^W]}(p) > 1$. Lorsque un siphon pondéré S^W est extrait, H2 génère une place monitrice p^W qui limite le nombre de jetons dans l'ensemble des places en amont de S^W suivant la définition 77. La place p^W ainsi générée peut avoir des arcs pondérés. Le Réseau contrôlé $\mathcal{N}^\mathcal{C} = \mathcal{N} \oplus p^W$, ainsi obtenu par l'ajout de p^W n'est donc plus un S3PMR. Par contre, les propriétés de p^W impliquent qu'il n'existe pas de siphon potentiellement vidable dans $\mathcal{N}^C$ qui comprenne p^W (i.e $\nexists S \in \Pi(\mathcal{N}^C) \,|\, p^W \in S \wedge M \in \mathcal{R}(\mathcal{N}^C, M_0) \,|M$ est un mauvais marquage de$S$).

H2 applique itérativement des couples extraction/contrôle de siphons jusqu'à l'obtention d'un Réseau de Petri vivant. Un siphon S est extrait en utilisant la méthode de [Cha07], puis, selon que le siphon soit ordinaire ou pondéré, une place p^V (définition 75) ou p^W (définition 77) est générée. Si un siphon S^W est extrait d'un S3PMR $\mathcal{N}$ avec $\max_{p\in[S^W]} k_{[S^W]}(p) > 1$, le Réseau $\mathcal{N}^\mathcal{C} = \mathcal{N} \oplus p^W$ est pondéré. Il est donc impossible de venir extraire un siphon de $\mathcal{N}^\mathcal{C}$ en utilisation la méthode de [Cha09] qui ne fonctionne que sur les Réseaux de Petri ordinaires. Les places générées suivant la définition 77 ont la propriété de ne pas pouvoir générer de situations de blocage. Il est donc possible d'ignorer l'influence structurelle de p^W dans la formulation du MIP de [Cha09] et de prendre en compte uniquement son influence sur les marquages accessibles (voir remarque 35). Les places monitrices issues d'un siphon pondéré sont prises en compte sous forme d'une contrainte sur le marquage dans la formulation du MIP. On note $SMS(\mathcal{N}, W, b)$ la méthode d'extraction de [Cha09] pour un Réseau de Petri $\mathcal{N}$ sous contraintes de $W \in \mathbb{N}^{|P|\times n}$ et $b \in \mathbb{N}^n$ où n est le nombre de contraintes. La méthode retourne ainsi un siphon $S \in \Pi(\mathcal{N})$ et un marquage $M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)$ tel que $W^\intercal.M \leq b$.

L'algorithme H2 fonctionne sous contrôlabilité et observabilité totale. Il est donc nécessaire de prendre en compte l'influence des transitions incontrôlables et inobservables dans la génération des places monitrices. En remplaçant la génération de places monitrices des définitions 75 pour p^V et 77 pour p^W par leur équivalent pour les RAS-PN sous COP, respectivement 91 et 93, la COP est prise compte.

Lorsque qu'aucun siphon potentiellement vidable ne peut être extrait (i.e la méthode de [Cha09] n'est pas en mesure de trouver une solution), les places de contrôles peuvent être ajoutées pour obtenir un S4PR. H2' est synthétisé dans l'algorithme 6.

5.3.2 Algorithme Z2'

L'algorithme Z2' s'applique aux S4PR sous COP. Il exploite les méthodes d'extraction présentées dans [ZDW+19]. Les siphons sont extraits itérativement en résolvant le MIP^1 présenté dans la sous-section 4.4.3 qui permet d'extraire un siphon S potentiellement bloquant. À partir de S, une place de contrôle p^V est calculée suivant la définition 91. Ensuite le problème MIP^2 est résolu pour trouver un marquage M_S^{max} qui permet de calculer le marquage initial de p^V, suffisant pour assurer que le siphon S n'entraine plus la mort de transitions dans le Réseau contrôlé. Des

Algorithme 6 Algorithme H2'

Entrée : Un S3PMR $\mathcal{N}$ sous COP
Sortie : Un S4PR $\mathcal{N}^C$ vivant

1: $\mathcal{N}^C \leftarrow \mathcal{N}$
2: $i = 0$; ▷ nombre de places de contrôle ordinaire générées
3: $j = 0$; ▷ nombre de places de contrôle pondérées générées
4: $W = 0^{|P|\times 0}$; ▷ Matrice de contraintes vide
5: $b = 0^{0\times 1}$; ▷ Vecteur de bornes de contraintes vide
6: $S \leftarrow SMS(\mathcal{N}^C, W, b)$ ▷ Extraction d'un siphon S
7: **si** S existe **alors**
8: **si** $\max_{p\in[S]} k_{[S]}(p)$ =1 **alors** ▷ Cas d'un siphon ordinaire
9: $i \leftarrow i + 1$
10: Calcul de p_i^V suivant la définition 91
11: $\mathcal{N}^C \leftarrow \mathcal{N}^C \oplus p_i^V$
12: **sinon** ▷ Cas d'un siphon pondéré
13: $j \leftarrow j + 1$
14: Calcul de la place p_j^W et du vecteur de contrainte $\check{w}_{[S]}^{\downarrow}$ suivant la définition 93
15: $W \leftarrow [W, \check{w}_{[S]}^{\downarrow}]$
16: $b \leftarrow [b^\top, M_0(p_j^W)]^\top$
17: **fin si**
18: Retour à l'étape 6
19: **sinon**
20: **si** $j > 0$ **alors** ▷ Ajout des places p_k^W avec $k = [1...j]$ à $\mathcal{N}^C$
21: **pour** $k \in [1...j]$ **faire**
22: $\mathcal{N}^C \leftarrow \mathcal{N}^C \oplus p_k^W$
23: **fin pour**
24: **fin si**
25: Retourner $\mathcal{N}^C$
26: **fin si**

places de contrôles sont ainsi générées jusqu'à l'obtention d'un S4PR vivant. Dans l'algorithme 7 on note $S \leftarrow MIP^1(\mathcal{N})$ l'extraction d'un siphon S utilisant par la résolution de MIP^1 construit à partir du S4PR $\mathcal{N}$. De manière analogue, on note $M_S^{max} \leftarrow MIP^2(\mathcal{N}, S)$, l'obtention du marquage M_S^{max} obtenue par la résolution de MIP^2 à partir du S4PR $\mathcal{N}$ et d'un siphon S.

5.4 Comparaison des techniques de synthèse de RAS-PN sous COP

Dans cette section, sont présentées différentes techniques de synthèse des RAS-PN sous COP de la littérature.

Algorithme 7 Algorithme Z2'

Entrée : Un S4PR $\mathcal{N}$ sous COP
Sortie : Un S4PR $\mathcal{N}^C$ vivant

1: $\mathcal{N}^C \leftarrow \mathcal{N}$
2: $i = 0$; ▷ nombre de places de contrôle générées
3: $S \leftarrow MIP^1(\mathcal{N}^C)$
4: **si** S existe **alors**
5: $i \leftarrow i + 1$
6: $M_S^{max} \leftarrow MIP^2(\mathcal{N}, S)$
7: Calcul de p_i^V et $M_0(p_i^V)$ à partir de S et M_S^{max} suivant la définition 91
8: $\mathcal{N}^C \leftarrow \mathcal{N}^C \oplus p_i^V$
9: Retour à l'étape 3
10: **fin si**
11: Retourner $\mathcal{N}^C$

5.4.1 État de l'art

La liste des publications présentée ici ne se veut pas exhaustive. Ces publications concernent uniquement des méthodes comparables dans l'esprit aux nôtres, c'est à dire des méthodes qui n'énumèrent pas l'ensemble des états de l'espace d'états.

[PR02] Dérivée de la politique de contrôle **G-RUN** (Generalised Ressource Upstream Neighbourhood) présentée dans [Par00], cette méthode ne se base pas sur l'étude des siphons. Elle définit un ordre arbitraire entre les ressources et un ordre partiel entre les places de processus selon leurs positions (les places amonts ayant un ordre plus élevé). À partir de ces ordonnancements, un ensemble de contraintes est défini, et un problème d'optimisation (LP) peut être formulé. La résolution de ce problème d'optimisation permet d'étendre le contrôle des places de ressources, donc redéfinir des places de ressources qui vont venir remplacer les places de ressources originales du S4PR. Cette technique de contrôle permet d'obtenir un Réseau contrôlé en un temps relativement faible comparé aux méthodes se basant sur l'extraction des siphons (Résolution d'un LP au lieu d'un MIP), avec une complexité structurelle extrêmement faible puisque aucune place de contrôle n'a été ajoutée. Ces performances sont obtenues au détriment d'une permissivité qui est très réduite. La prise en compte des transitions incontrôlables se fait directement dans la résolution du problème d'optimisation en rajoutant des contraintes supplémentaires.

[QLAA15] Cette technique de contrôle se base sur la théorie des siphons élémentaires. Elle s'applique aux S3PR sous COP. Dans un premier temps, l'ensemble des siphons stricts minimaux potentiellement vidables sont extraits itérativement en utilisant la méthode [CX97]. À partir de cet ensemble de siphons, un ensemble de siphons élémentaires est calculé. Pour chaque siphon élémentaire S, un ensemble complémentaire étendu $[S]_e$ est calculé en prenant en compte l'influence des transitions inobservables (i.e $[S]_e = \{p \,|\, \mathcal{F}^O(k_{[S]})(p) > 0\}$). Pour chaque siphon S une place de contrôle p^V est générée, permettant de limiter le nombre de jetons dans $[S]_e$, créant ainsi un S3PMR $\mathcal{N}^C$. Pour chaque

transition de $t \in p^{V\bullet}$, un problème d'optimisation est résolu pour vérifier s'il existe un marquage $M \in \vec{\mathcal{R}}(\mathcal{N}, M_0)$ tel que $M(p^V) = 0$ et $M({}^\bullet t \cap P^P) > 0$. Si un tel marquage existe, la transition est dite "risky" car la place monitrice peut être amenée à empêcher le tir de la transition. Si ce n'est pas le cas, la transition est dite "secure" car la place monitrice n'empêchera jamais le tir de cette transition. L'algorithme redirige ensuite les arcs sortants, dits "risky, des places monitrices vers les transitions sources jusqu'à ce que le Réseau soit vivant (test par la méthode de [CX97]). La redirection des arcs sortants est faite itérativement et la sélection d'un arc est faite aléatoirement. Après chaque redirection, on teste si le réseau contrôlé est devenu vivant. L'algorithme finit nécessairement par converger vers un Réseau contrôlé vivant puisque dans le pire des cas, tous les arcs sortants de chaque place de contrôle sont redirigés vers les transitions sources. Cette méthode permet d'améliorer la permissivité, car elle ne reporte pas nécessairement tous les arcs vers les transitions sources.

[LS08] Cette technique de synthèse s'applique aux S3PR comportant des transitions incontrôlables. Elle a une approche relativement proche de celle de [HJXC01] car la technique de synthèse comprend également deux phases. Lors de la première phase, tous les siphons stricts minimaux du S3PR $\mathcal{N}$ sont énumérés et un ensemble de siphons élémentaires en est extrait. Comme le réseau est un S3PR, les siphons extraits respectent nécessairement la condition suivante $\max_{p\in[S]} k_{[S]}(p) = 1$ (voir remarque 34). Pour chaque siphon élémentaire S_e, une place monitrice est générée contrôlant les places p telles que $\mathcal{F}^C k_{[S_e]}(p) = 1$, les places monitrices sont donc ordinaires. Le Réseau obtenu $\mathcal{N}'$ est un S3PMR. L'ajout des places monitrices supprime des états bloquants mais leur présence crée de nouveaux siphons dans $\mathcal{N}'$. La deuxième phase est basée sur le même principe. Elle commence par calculer l'ensemble des siphons stricts minimaux de $\mathcal{N}'$. Ensuite, elle extrait de ces SMS, un ensemble de siphons élémentaires. Pour chaque siphon élémentaire, une place monitrice p^W est générée selon la définition 77. Le Réseau obtenu est un S4PR vivant et présente une relativement faible complexité structurelle.

Technique de synthèse	Type de RAS-PN	Transitions inobservables	Approche utilisée
[PR02]	S4PR	non	LP
[QLAA15]	S3PR	oui	MIP
[LS08]	S3PR	non	Enumération des siphons
H2'	S3PMR	oui	MIP
Z2'	S4PR	oui	MIP

Table 5.6 – Tableau récapitulatif des techniques de synthèse de RAS-PN sous COP

Le tableau 5.6 récapitule les techniques de synthèse présentées dans cette section. La première colonne rappelle sur quel type de RAS-PN chaque méthode de contrôle peut-être appliquée. La seconde colonne synthétise si les transitions inobservables peuvent faire partie du modèle (toutes les méthodes considèrent par

contre des transitions incontrôlables). La troisième colonne rappelle par quelle approche sont identifiés les siphons en vue de leur contrôle. Les méthodes d'identifications des siphons sont fortement corrélées avec la complexité de la technique de synthèse.

5.4.2 Comparaison des performances

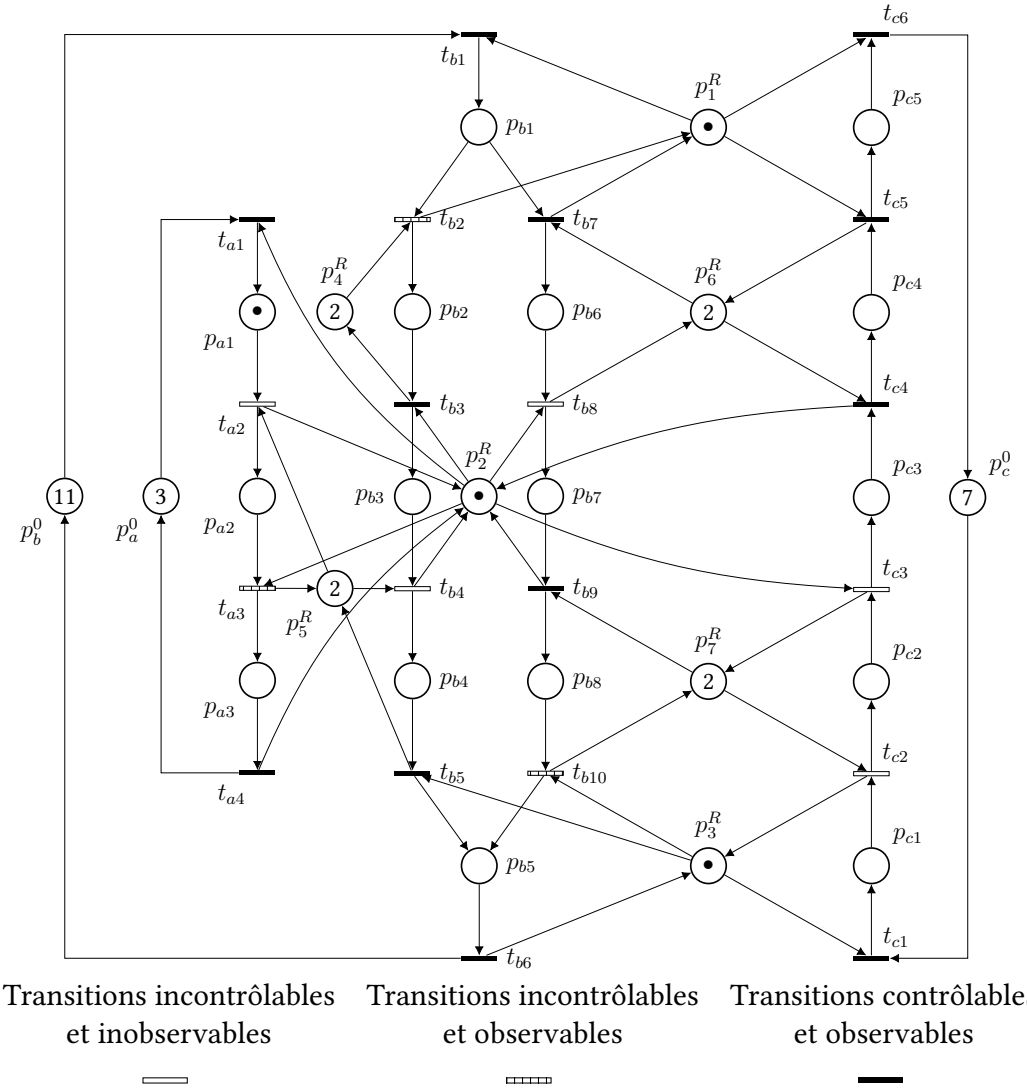


FIGURE 5.7 – Un S3PR $\mathcal{N}$ sous COP

Nous proposons de comparer et d'analyser les performances de trois techniques de synthèse de contrôleurs applicables aux S3PR sous COP : ([QLAA15], H2', Z2'). Ces trois techniques ont été appliquées au même Réseau de la figure 5.7, car ces trois méthodes considèrent des transitions inobservables et incontrôlables. Les résultats obtenus pour chacune des méthodes sont récapitulés dans les tableaux 5.7, 5.8 et 5.9 qui recensent les places monitrices générées respectivement par les méthodes Z2', H2' et [QLAA15]. Les techniques de synthèse Z2' et H2' extraient et contrôlent itérativement des siphons. Dans 5.7 et 5.8, les siphons sont présentés par ordre d'extraction. Pour la technique de synthèse de [QLAA15], il n'a pas été nécessaire de

réorienter les arcs vers les transitions sources. En effet, le Réseau obtenu après les ajouts des places monitrices contrôlant les ensembles complémentaires étendus des siphons élémentaires est vivant. De plus, aucune transition n'a été identifiée comme "secure".

Dans le tableau 5.8, les places sont nommées p^V lorsque le siphon est ordinaire et p^W lorsque il est pondéré.

Dans la littérature, classiquement trois critères sont utilisés pour comparer les performances :

- **Complexité structurelle :** Elle mesure l'intelligibilité de la structure du contrôle. Ici les trois méthodes exercent le contrôle via la génération de places monitrices. Ainsi, la mesure de la complexité structurelle se fait simplement par le décompte des places de contrôles ajoutées. Plus de places de contrôles sont ajoutées, plus la technique de synthèse est structurellement complexe.
- **Permissivité :** Elle mesure à quel point le contrôle appliqué est restrictif. Ici le contrôle est fait par l'ajout de places monitrices qui vont empêcher le système d'atteindre les états interdits. Si les places monitrices du modèle en boucle fermé ne "supprime" que les états interdits, le contrôleur est dit maximalement permissif. Mais s'il empêche d'atteindre des états qui ne devraient pas être interdits, le contrôle a une permissivité réduite. La mesure de cette permissivité peut être faite de manière simple par le décompte du nombre d'états accessibles du Réseau contrôlé.
- **Complexité de calcul :** Les trois méthodes se basent sur la résolution d'un MIP qui est NP-difficile en théorie. Cependant, la formulation du MIP et, le nombre de fois que le MIP est résolu, influent directement sur la durée des algorithmes de synthèse. Les trois méthodes étant exécutées sur le même ordinateur, les temps de calculs sont donc comparables.

i	S_i	${}^\bullet p_i^V$	$p_i^{V\bullet}$	$M_0(p_i^V)$
1	$p_{b6}, p_{c3}, p_3^R, p_7^R$	t_{b10}, t_{c4}	t_{b9}, t_{c1}	2
2	$p_{a4}, p_{b5}, p_{b8}, p_{c4}, p_2^R, p_5^R$	t_{a3}, t_{b5}	t_{a1}, t_{b3}	2
3	$p_{a2}, p_{a4}, p_{b4}, p_{b8}, p_{c5}, p_2^R, p_6^R$	t_{b9}, t_{c4}	t_{b7}, t_{c1}	2
4	$p_{a2}, p_{a4}, p_{b4}, p_{b8}, p_{c6}, p_1^R, p_2^R, p_4^R, p_6^R$	t_{b3}, t_{b9}, t_{c5}	t_{b1}, t_{c1}	5

TABLE 5.7 – Places monitrices obtenues par application de la technique de synthèse Z2' au S3PR de la figure 5.7

Le tableau 5.10 compare les politiques de contrôles selon les critères énoncés ci-dessus. Les 3 politiques de contrôles ont des temps de calcul relativement proche. Ceci est le résultat de deux facteurs : la méthode d'extraction et le nombre d'extractions. H2' et [QLAA15] utilisent la même méthode d'extraction (celle de [CX97]) et est répétée dans un même ordre de grandeur : 11 générations de places de contrôle pour H2' et 18 extractions de siphons avant détermination des siphons élémentaires pour [QLAA15]. Il faut noter que la résolution du MIP dans H2' devient de plus en plus complexe au fur et à mesure que des places monitrices sont ajoutées (26 places pour la première extraction et 37 lors de la dernière). Dans le cas de Z2' la résolution du MIP utilisé est plus complexe. La complexité des MIP dépend fortement du nombre de variables binaire $|P|+|T|$ pour [CX97] et $\approx |P|+|T|+|p^{R\bullet}|$ pour [ZDW+19].

p	S	$^\bullet p$	$p^\bullet$	$M_0(p)$
p_1^V	$p_{a4}, p_{b6}, p_{c6}, p_1^R, p_2^R, p_3^R, p_4^R, p_5^R, p_6^R, p_7^R$	$t_{a3}, t_{b5}, t_{b10}, t_{c5}$	t_{a1}, t_{b1}, t_{c1}	10
p_2^V	$p_{a4}, p_{b5}, p_{b9}, p_{c6}, p_1^R, p_2^R, p_4^R, p_5^R, p_6^R, p_7^R$	$t_{a3}, t_{b5}, t_{b9}, t_{c5}$	t_{a1}, t_{b1}, t_{c1}	9
p_3^V	$p_{a4}, p_{b5}, p_{b9}, p_{c5}, p_2^R, p_5^R, p_6^R, p_7^R$	$t_{a3}, t_{b5}, t_{b9}, t_{c4}$	$t_{a1}, t_{b3}, t_{b7}, t_{c1}$	6
p_4^V	$p_{a4}, p_{b6}, p_{c4}, p_2^R, p_3^R, p_5^R, p_7^R$	$t_{a3}, t_{b5}, t_{b10}, t_{c4}$	$t_{a1}, t_{b3}, t_{b7}, t_{c1}$	5
p_5^V	$p_{b6}, p_{c3}, p_3^R, p_7^R$	t_{b10}, t_{c4}	t_{b9}, t_{c1}	2
p_6^V	$p_{a4}, p_{b5}, p_{b9}, p_{c4}, p_2^R, p_5^R, p_5^V$	$t_{a3}, t_{b5}, t_{b9}, t_{c4}$	$t_{a1}, t_{b3}, t_{b7}, t_{c1}$	4
p_7^V	$p_{a4}, p_{b5}, p_{b8}, p_{c6}, p_1^R, p_2^R, p_4^R, p_5^R, p_6^R$	$t_{a3}, t_{b5}, t_{b9}, t_{c5}$	t_{a1}, t_{b1}, t_{c1}	7
p_8^V	$p_{a4}, p_{b5}, p_{b8}, p_{c4}, p_2^R, p_5^R$	t_{a3}, t_{b5}	t_{a1}, t_{b3}	2
p_9^V	$p_{a2}, p_{a4}, p_{b4}, p_{b9}, p_{c4}, p_2^R, p_5^V$	t_{b9}, t_{c4}	t_{b7}, t_{c1}	2
p_1^W	$p_{a2}, p_{a4}, p_{b4}, p_{b9}, p_{c6}, p_1^R, p_2^R, p_4^R, p_6^R, p_5^V$	$t_{b3}, t_{b9}, t_{c4}, t_{c5}$	$t_{b1}, 2t_{c1}$	7
p_{10}^V	$p_{a2}, p_{a4}, p_{b4}, p_{b8}, p_{c6}, p_1^R, p_2^R, p_4^R, p_6^R$	t_{b3}, t_{b9}, t_{c5}	t_{b1}, t_{c1}	5

Table 5.8 – Places monitrices obtenues par application de la technique de synthèse H2' au S3PR de la figure 5.7

i	S_i	$^\bullet p_i^{V\prime}$	$p_i^{V\prime\bullet}$	$M_0(p_i^V)$
1	$p_{a4}, p_{b5}, p_{b8}, p_{c4}, p_2^R, p_5^R$	t_{a3}, t_{b5}	t_{a1}, t_{b3}	2
2	$p_{a2}, p_{a4}, p_{b4}, p_{b6}, p_{c4}, p_2^R, p_3^R, p_7^R$	t_{b10}, t_{c4}	t_{b7}, t_{c1}	2
3	$p_{a2}, p_{a4}, p_{b4}, p_{b8}, p_{c5}, p_2^R, p_6^R$	t_{b9}, t_{c4}	t_{b7}, t_{c1}	1
4	$p_{a2}, p_{a4}, p_{b4}, p_{b9}, p_{c5}, p_2^R, p_6^R, p_7^R$	t_{b9}, t_{c4}	t_{b7}, t_{c1}	3
5	$p_{a2}, p_{a4}, p_{b4}, p_{b8}, p_{c6}, p_1^R, p_2^R, p_4^R, p_6^R$	t_{b3}, t_{b9}, t_{c5}	t_{b1}, t_{c1}	5
6	$p_{a4}, p_{b6}, p_{c4}, p_2^R, p_3^R, p_5^R, p_7^R$	$t_{a3}, t_{b5}, t_{b10}, t_{c4}$	$t_{a1}, t_{b3}, t_{b7}, t_{c1}$	5

Table 5.9 – Places de contrôle obtenues par application de la technique de synthèse de [QLAA15] le S3PR de la figure 5.7

Comme Z2' ne génère que 4 places monitrices, moins d'extractions sont répétées, ce qui vient harmoniser les temps de calcul entre les différentes méthodes. Il est complexe de mesurer la borne maximale de la permissivité. Dans le cas des RAS-PN sous contrôlabilité et observabilité totales, on compare le nombre d'états du graphe d'accessibilité du modèle obtenu par une technique de synthèse donnée, à celui obtenu par la SCT (la méthode de Ramadge et Wonham). Or, dans le cas de systèmes sous COP, la SCT nécessite la réalisation d'un observateur. Cet observateur peut avoir plus d'états que l'automate observé ce qui rend la comparaison difficile. La valeur optimale n'est donc pas connue.

Pour notre exemple comparatif, la permissivité de Z2' et H2' sont très proches, cela s'explique par le fait qu'une seule place de contrôle p^W a été généré par H2'. Hormis cette place de contrôle, H2' n'a généré que des places p^V qui ont un contrôle proche de l'optimale. La permissivité des deux méthodes est donc quasiment identique. Cependant on observe que H2' génère beaucoup plus de places monitrices. Ceci peut être expliqué par les différents comportements des méthodes d'extraction de Z2' et H2'. La méthode d'extraction de [ZDW+19] utilisée dans Z2' cherche des siphons ayant des propriétés intéressantes du point de vue du contrôle (voir la fonction d'optimisation de MIP^1). Dans le cas de la méthode [CX97], utilisée dans H2', un siphon maximal est extrait.Ensuite, un siphon minimal est extrait du siphon maximal. Dans le cas de H2', il n'est pas recherché en priorité des siphons

"d'intérêt" du point de vue du contrôle. En regardant de plus près, on voit que les quatre places monitrices de Z2' sont aussi générées par H2'(p_5^V, p_8^V, p_9^V, p_{10}^V), les autre places de contrôle de H2' sont donc ici redondantes. Une piste d'amélioration de H2'consisterait à modifier la méthode d'extraction de manière à ce que, de la même manière que dans [ZDW+19], soit extrait en priorité des siphons d'intérêt du point de vue du contrôle.

Dans le cas de [QLAA15], la faible permissivité (6262 états) est complexe à justifier. Cette méthode d'une manière générale à une permissivité faible à cause du report des arcs sur les transitions sources (cette opération réduit fortement la permissivité). Cependant, dans le cas de l'application à la figure 5.7 aucun arc n'a été reporté car le réseau était vivant après ajout des places monitrices.

Remarque 37. *La méthode de [QLAA15], ne définit pas de manière optimale les siphons élémentaires. Les siphons élémentaires (voir sous-section 3.3.2) sont un sous-ensemble d'un ensemble de siphon tel que s'ils sont contrôlés par des places monitrices, ces dernières n'ont pas de contrôles redondants. L'identification des siphons élémentaires se fait via leurs T-vecteurs caractéristiques, qui permettent d'identifier les transitions entrantes ou sortantes de la place monitrice du siphon. Dans [QLAA15], les siphons élémentaires sont identifiés de manière classique puis pour chacun d'entre eux une place monitrice vient contrôler l'ensemble complémentaire étendu (prise en compte de la COP). Cependant pour deux siphons élémentaires différents il est possible que leurs ensembles complémentaires étendus soient identiques, ce qui implique que deux places monitrices puissent exercer le même contrôle (ce que l'identification des siphons élémentaire vise à empêcher). Dans le cas de l'application à la figure 5.7, c'est le cas des places 3 et 4 qui ont une structure identique. Une piste d'amélioration consisterait à prendre en compte la COP dans la définition des vecteurs T-caractéristiques de chaque siphon extrait. Ainsi en définissant des vecteurs T-caractéristiques admissibles vis-à-vis de la COP, il serait possible à partir de la base de T-caractéristiques admissibles, de générer des places monitrices uniques, sans contrôle redondant et respectant la COP. À notre connaissance aucune théorie n'étend la notion de siphon élémentaire dans un Réseau de Petri sous COP.*

Les méthodes itératives créent généralement un nombre important de places monitrices. Certaines places peuvent avoir des contrôles redondants. Il est possible de trouver des places de contrôle "inutiles" en venant supprimer la place puis en analysant si le Réseau est devenu bloquant. Cette méthode nécessite de faire itérativement un test d'extraction de siphon par place de contrôle et ne garantit pas de supprimer la meilleure combinaison de place de contrôle. Cependant cette méthode peut venir grandement augmenter le nombre de places de contrôle. La politique de [QLAA15] étant basée sur la théorie des siphons élémentaires, il est normal que cette politique génère moins de place monitrices.

5.5 Conclusion

Ce chapitre vise à présenter des méthodes itératives pour la synthèse de contrôleurs pour les RAS-PN sous COP. Il reprend les notions présentées dans le chapitre précédant pour y incorporer les notions relatives à la contrôlabilité et l'observabilité partielles. Dans la section 5.1, les notions relatives à la transformation des

Politique de Contrôle	Nombre de places monitrices	États accessibles	Temps de Calcul
[QLAA15]	6	6262	5,94s
H2'	11	11999	6,61s
Z2'	4	12167	6,70s

Table 5.10 – Tableau comparatif des performances de différentes techniques de synthèse

GMECs, présentées dans la section 3.2.4, sont adaptées aux cas des RAS-PN. Dans le cas d'une contrainte de type GMEC seul le vecteur de contrainte vient impacter la structure de la place monitrice générée, la borne n'ayant que de l'influence sur le marquage initial. La spécificité de la définition d'un vecteur dans le cadre des RAS-PN (par opposition au cadre général) concerne les places processus. En effet, on souhaite que les vecteurs de contrainte ne viennent exercer leur contrôle que sur les places de processus. Cette hypothèse garantit que la place monitrice générée à partir d'un GMEC ait les mêmes propriétés qu'une place de ressource. C'est une propriété nécessaire pour appliquer un algorithme de type itératif, où des cycles d'extraction d'un siphon, puis la génération d'une place monitrice sont répétés jusqu'à l'obtention d'un Réseau contrôlé vivant. Effectivement, les méthodes d'extraction (dans le cas de Z2') où la génération des places monitrices (dans le cas de H2'), sont spécifiques aux RAS-PN. Il est donc obligatoire que l'ajout d'une place monitrice ne vienne pas modifier les propriétés du Réseau contrôlés de manière à ce qu'il reste un RAS-PN. À partir d'un vecteur contrainte non admissible une méthode de transformation est proposée. Cette méthode permet de trouver un nouveau vecteur contrainte admissible.

La section 5.2 concerne le contrôle des siphons. Pour un siphon donné on cherchera à définir une place monitrice qui vient contrôler ce siphon tout en respectant les conditions définies dans la section prétendante (la place monitrice est une place de ressources et respecte la COP). Dans la section 4.4.2 deux types de places monitrices ont été définies. Ici on étend leur définition pour prendre en compte les transitions incontrôlables et inobservables du RAS-PN. Le premier type de place de contrôle, noté p^V, cherche à limiter le nombre de jetons dans l'ensemble complémentaire du siphon. Le contrôle appliqué par p^V est relativement permissif, mais cette place peut potentiellement créer de nouveaux siphons dans le Réseau contrôlé. Le second type de place monitrice, notée p^W, exerce un contrôle sur toutes les places amonts de l'ensemble complémentaire. Le contrôle exercé par p^W et plus restrictif que celui exercé par p^V. Cependant, lors de l'ajout d'une place p^W, il est garanti qu'aucun siphon du Réseau contrôlé n'inclue p^W, p^W n'entraine donc pas de situation de blocage.

La section 5.3, présente deux algorithmes de contrôle. Ces deux algorithmes sont itératifs, ils répètent des couples d'opérations : extraction d'un siphon/génération d'une place de contrôle. Le premier $H2'$ est inspiré des travaux présentés dans [HJXC06], il concerne les S3PMR et génère, selon le siphon extrait, des places p^V ou p^W. Le second algorithme s'applique aux S4PR et génère itérativement des places de type p^V.

La section 5.4 compare les performances des méthodes de synthèse de RAS-PN

sous COP. Dans un premier temps, un aperçu des méthodes de la littérature est donné. Uniquement trois méthodes de contrôles sont présentées, les méthodes de contrôles de RAS-PN sous COP ayant fait l'objet que de peu d'études. Ensuite sur un S3PR donné les algorithmes H2', Z2' et l'algorithme de [QLAA15] y sont appliqués. Les performances sont comparées sur trois critères : permissivité (via le nombre d'états atteignables), complexité (via le temps de calcul) et complexité structurelle (via le nombre de places monitrices). Cette étude ne vise pas à caractériser complètement les performances de chacune des méthodes mais plutôt d'en avoir un aperçu.

Chapitre 6

Contrôle des systèmes ferroviaires

Les systèmes ferroviaires sont composés de lignes reliant des nœuds ferroviaires. Les nœuds ferroviaires correspondent à des gares ou à des zones du système ferroviaire où l'on réalise l'interconnexion de plusieurs lignes ferroviaires. Les nœuds ferroviaires sont donc des composants essentiels d'un système ferroviaire car leur fonctionnement va avoir un impact sur la gestion du trafic ferroviaire d'un système. Ces nœuds sont classiquement gérés par des aiguilleurs situés dans des salles d'aiguillages. La gestion est semi-automatisée dans le sens où les aiguilleurs sont assistés par des calculateurs pour décider de l'affectation d'une route à un train entrant dans le nœud. Cette route doit être sans risque de collisions avec d'autres trains. Pour garantir cette sécurité, l'ensemble des aiguillages et sections de voie de la route est affecté à la route établie pour un train donné, jusqu'à ce qu'il sorte du nœud. Dans [LLER13], l'auteur critique ce routage des trains, qui est utilisé dans le monde entier depuis plus de 50 ans. Aujourd'hui, avec la forte augmentation du trafic ferroviaire, cette stratégie de gestion montre ses limites. En effet, les nœuds ferroviaires des zones denses en trafic (cas des grandes métropoles européennes ou asiatiques), arrivent à saturation avec cette politique de contrôle. Par conséquent, il n'est plus possible d'augmenter le trafic des trains dans ces zones, ce qui entraîne des retards et des consommations énergétiques supplémentaires pour les trains devant s'arrêter à l'entrée d'un nœud. D'autre part, en cas de dysfonctionnement dans un nœud, il est très difficile d'organiser un routage des trains en mode dégradé du fait de la rigidité du contrôle. Cela a pour conséquence des retards pouvant modifier fortement l'ordonnancement prévisionnel et impactant fortement la gestion de trafic de nœuds ferroviaires au voisinage de la défaillance (propagation des conséquences d'une défaillance).

Les Réseaux de Petri ont été utilisés dans de nombreux types d'applications parmi lesquels les systèmes ferroviaires. Dans les années 1990 et au début des années 2000, ils ont surtout été utilisés dans la conception des systèmes de production manufacturiers flexibles. Toutefois, dès les années 1980, on trouve des publications sur l'étude des systèmes ferroviaires modélisés par Réseaux de Petri [Lau85]. Mais l'essentiel des travaux concernent la modélisation formelle des enclenchements afin de garantir la sécurité [DS09]. Cette étude s'intéresse au routage dynamique des trains dans les nœuds ferroviaires. Dans ce cadre, la sécurité et plus particulièrement l'évitement des collisions des trains, demeurent une problématique essentielle. Une étude intéressante est celle de [GS08] qui propose une méthode basée sur le concept de contraintes générales d'exclusion mutuelle (GMEC) pour éviter tout risque de collision. Mais les GMECs n'empêchant pas les blocages, les auteurs proposent de synthétiser de nouvelles places de contrôles pour l'évitement de blocages. Dans

[FGS06], les auteurs proposent une loi de contrôle basée sur les Réseaux de Petri colorés et exploitant la technique des places monitrices pour l'évitement de collision. L'approche proposée garantit un contrôle maximalement permissif mais s'intéresse également à l'évitement des situations de blocage. Le blocage est un problème classique dans les systèmes comportant plusieurs sous-systèmes évoluant en parallèle avec partage des ressources. Il a donc été énormément étudié ces deux dernières décennies, notamment dans le cadre du contrôle des systèmes manufacturiers. La résolution des blocages est une problématique qui est au cœur de cette étude. Nous nous sommes donc intéressés aux approches qui ont été proposées dans la littérature. Dans [Pac07], deux méthodes de résolution de blocages sont présentées : une analyse des conséquences d'un mouvement donné vérifie les conséquences futures du mouvement d'un train. La seconde méthode est la réservation dynamique de l'itinéraire d'un train. Dans cette méthode, l'ensemble des tronçons traversés par le train est réservés simultanément. Une méthode analogue a été utilisée dans [XKBT17] permettant de réserver les aiguillages et les sections de voie nécessaires pour établir la route d'un train avant qu'il ne s'engage dans un nœud ferroviaire.

Notre objectif est de contribuer à la proposition d'une nouvelle technique de routage dynamique des trains dans un nœud [CKT19a, CKT19b, CKBT19], permettant de réduire la saturation actuelle des nœuds. Ce travail emprunte des idées qui ont été développées pour le contrôle des systèmes manufacturiers. Dans un premier temps nous proposons une approche de modélisation par Réseaux de Petri de la traversée d'un nœud ferroviaire par un train, permettant un routage dynamique. Pour cela, nous définissons le concept de traversée qui a pour objectif de permettre le routage des trains. Ensuite, nous utilisons la technique de synthèse de places monitrices pour construire un contrôleur permettant de garantir l'absence de possibilités de collisions entre trains dans le nœud. Le modèle obtenu correspond à un S3PMR. Nous proposons une méthode de contrôle pour l'évitement des blocages, adaptée aux S3PMR contenant des transitions incontrôlables et non observables.

6.1 Infrastructure d'un nœud ferroviaire

6.1.1 Composant d'un nœud ferroviaire

Les infrastructures ferroviaires sont essentiellement constituées de deux types de composants ferroviaires : les sections de voie et les aiguillages. Une section de voie est une portion de rail. Pour des raisons de sécurité, elle est souvent monodirectionnelle (Figure 6.1). Mais elle peut également être bidirectionnelle (Figure 6.2). La direction est directement représentée sur les sections grâce au symbole $\rangle$ pour une section pouvant être uniquement traversée de gauche à droite ($\langle$ pour droite à gauche) et $\langle\rangle$ dans le cas d'une section bidirectionnelle. Dans un système ferroviaire, les trains circulent traditionnellement à gauche. Aussi, les sections de voie bidirectionnelles doivent avoir une instrumentation double pour le contrôle et la signalisation ferroviaire. Chaque section de voie est délimitée par des connecteurs gauche x et droite (y représentés par des barres verticales hachurées sur les figures 6.1 et 6.2. En pratique, ces connecteurs sont des isolants, permettant d'isoler électriquement chaque section de voie.

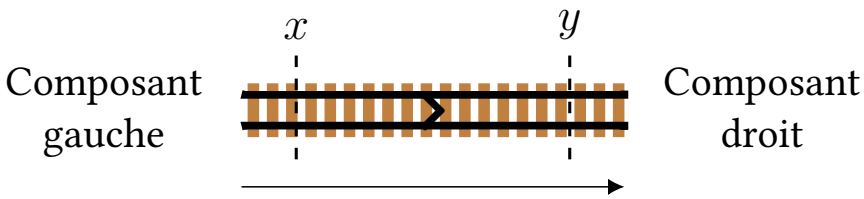


FIGURE 6.1 – Schéma d'une section de voie monodirectionnelle

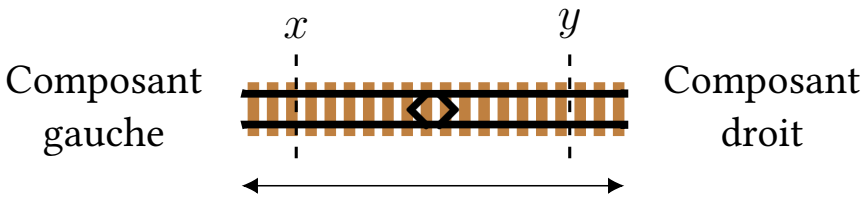


FIGURE 6.2 – Schéma d'une section de voie bidirectionnelle

Il existe en pratique différents types d'aiguillages. Mais, dans cette étude, nous nous intéressons au type le plus courant que sont les aiguillages simples 3 voies (Figure 6.3). C'est un équipement qui permet d'interconnecter 3 autres composants. En fonction de la nature des composants qui lui sont connectés, un aiguillage peut également être soit monodirectionnel, soit bidirectionnel. Pour chaque branche de l'aiguillage, la direction empruntable sur cette branche est notée par les notations `>`, `<` et `<>` identique à celle des sections. Dans le cas la figure 6.3 cela implique que les composants 1, 2 et 3 soient bidirectionnels. Il est à noter que les autres types d'aiguillages peuvent être décrits à partir de l'aiguillage simple 3 voies. Donc notre étude n'est pas limitée par cette restriction.

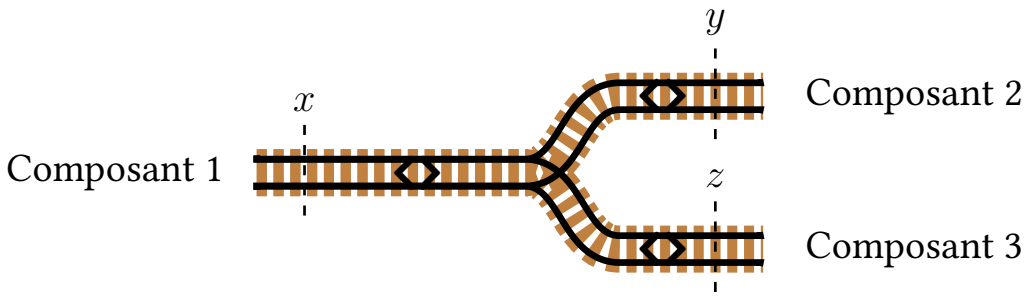


FIGURE 6.3 – Schéma d'un aiguillage

6.1.2 Instrumentation associée à l'infrastructure

Le rôle de l'instrumentation associée aux équipements de l'infrastructure est de permettre le contrôle et la signalisation. Nous allons introduire ici l'instrumentation classique des infrastructures ferroviaires actuelles non automatisées. Ensuite nous indiquerons comment cette instrumentation est prise en compte dans le cadre d'une automatisation. L'un des premiers rôles de l'instrumentation est de permettre la localisation des trains sur les voies. Pour cela plusieurs types d'équipements ont été

développés pour la détection des trains : les circuits de voie et les compteurs d'essieux. Les circuits de voie sont les détecteurs les plus couramment utilisés. Ils sont basés sur l'isolation électrique de chaque section de voie. Quand un essieu du train se trouve dans une section de voie ou un aiguillage, il fonctionne comme un shunt électrique entrainant la baisse de l'intensité du courant sur une partie de la section de voie. La mesure de cette baisse d'intensité permet de détecter la présence du train sur cette section de voie. L'autre alternative, est le compteur d'essieux qui compte le nombre d'essieux d'un train entrant dans la section puis qui en sort. Son fonctionnement est événementiel alors que celui du circuit de voie est continu. Quelle que soit la technologie utilisée, nous représentons graphiquement un détecteur de train selon la convention donnée par la Figure 6.4. L'absence de détecteur pour une connexion sera modélisée selon la convention de la Figure 6.6. Dans le fonctionnement traditionnel, des feux tricolores permettent de donner au mécanicien du train l'autorisation de mouvement nécessaire afin de lui permettre de contrôler l'avance ou l'arrêt du train. Dans les systèmes automatisés, ces feux tricolores sont remplacés par des balises localisées dans le sol qui permettent de transmettre de manière événementielle les autorisations de mouvements aux trains. Par convention, dans les figures de cette étude, nous représentons par des feux tricolores les lieux de transmissions aux trains des autorisations de mouvement (Figure 6.5).

Remarque : Il est important de noter que l'instrumentation d'un composant bidirectionnel n'est pas la même dans les deux sens. Il faut la doubler pour obtenir les mêmes niveaux de contrôle et de signalisation dans les deux sens.

FIGURE 6.4 – Détecteur de train

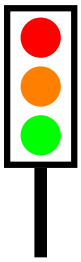


FIGURE 6.5 – Autorisation de mouvement

FIGURE 6.6 – Connexion sans instrumentation

6.1.3 Construction d'un nœud ferroviaire

La figure 6.7 illustre un nœud ferroviaire constitué de sept sections de voies notées Si avec $i \in [1..7]$. Ces sections sont toutes bidirectionnelles exceptée la section $S7$ qui est monodirectionnelle. Le nœud comporte aussi trois aiguillages notés SWi avec $i \in [1..3]$. Pour un nœud donné on note $\mathfrak{C}$ l'ensemble des composants de ce nœud, ici $\mathfrak{C} = \bigcup_{i=1}^{7}\{Si\} \cup \bigcup_{i=1}^{3}\{SWi\}$. La direction de passage des trains au sein d'un aiguillage doit être conforme aux sens de passage des composants voisins, les branches de ces aiguillages sont donc bidirectionnelles, exceptée la branche de $SW3$ liée à $S7$ qu'il est possible de traverser de gauche à droite. On note l'ensemble des composants pouvant être traversés de haut vers le bas (respectivement

bas vers le haut) par $\overrightarrow{\mathfrak{C}}$ (respectivement $\overleftarrow{\mathfrak{C}}$), ici $\overrightarrow{\mathfrak{C}} = \bigcup_{i=1}^{7}\{Si\} \cup \bigcup_{i=1}^{3}\{SWi\}$ et $\overleftarrow{\mathfrak{C}} = \bigcup_{i=1}^{6}\{Si\} \cup \bigcup_{i=1}^{3}\{SWi\}$

L'entrée des trains peut se faire par les composants $S1$ et $S6$ et la sortie par les composants $S1$, $S6$ et $S7$ (notés par les flèches ⬇ et ⬆ selon le sens de passage des trains).

Chaque connexion est numérotée, on note que les connexions avec un composant et une entrée/sortie sont aussi comptabilisées. Dans le cas de la figure 6.7, on comptabilise 13 connexions. On notera une connexion $\mathfrak{j}_i$ ou i est le numéro de la connexion et l'ensemble des connexions noté $\mathfrak{J}$ dans le cadre de ce nœud $\mathfrak{J} = \bigcup_{i=1}^{13}\{\mathfrak{j}_i\}$. Avec la même approche que les composants on distingue les connexions selon leurs sens de traversée. On note l'ensemble des connexions pouvant être traversées de haut en bas par $\overrightarrow{\mathfrak{J}} = \bigcup_{i=1}^{13}\{\mathfrak{j}_i\}$ et $\overleftarrow{\mathfrak{J}} = \bigcup_{i=1}^{11}\{\mathfrak{j}_i\}$.

L'instrumentation étant différente selon le sens de passage, on trouve deux instances d'instrumentation pour chaque connecteur de composants. L'instrumentation se trouve toujours placée à gauche du sens de circulation des trains.

Pour faire référence à l'instrumentation droite (respectivement gauche) d'une jonction nous utiliserons l'application $\overrightarrow{\mathfrak{Ins}} : \mathfrak{J} \rightarrow \{$🚦, 🔴, $\bullet\}$ (respectivement $\overleftarrow{\mathfrak{Ins}} : \mathfrak{J} \rightarrow \{$🚦, 🔴, $\bullet\}$).

6.1.4 Concept de traversées

Dans le domaine ferroviaire, un concept de base utilisé pour le routage des trains est la notion de route. Une route est un chemin permettant de traverser un nœud. Ici, nous proposons une extension de ce concept de route par celui de traversée.

Définition 94. *Une traversée est un ensemble de routes issues d'une même entrée d'un nœud ferroviaire et permettant de rejoindre une ou plusieurs sortie du nœud ferroviaire. Une traversée Γ_i est définie par un point d'entrée et un ou plusieurs points de sortie du nœud ferroviaire.*

Bien que nous n'allons pas l'illustrer dans cette étude, nous avons défini ce concept de traversée afin de gérer des problématiques de reconfiguration du routage d'un train durant sa traversée d'un nœud. En cas de défaillance d'une ressource de la route courante, le système affecte au train une autre route de la même traversée permettant d'aller vers la destination finale du train.

Afin d'illustrer ce concept de traversée, considérons le système représenté par les figure 6.8 et figure 6.9. Cette traversée correspond aux quatre routes issues de l'entrée $S1$ et allant vers l'une des sorties représentées par $S6$ ou $S7$. Ces quatre routes sont données ici sous la forme de la liste des composants utilisés par chaque route :

- $S1 - SW1 - S2 - S3 - SW2 - SW3 - S6$
- $S1 - SW1 - S4 - S5 - SW2 - SW3 - S6$
- $S1 - SW1 - S2 - S3 - SW2 - SW3 - S7$
- $S1 - SW1 - S4 - S5 - SW2 - SW3 - S7$

De même, la traversée Γ_2 correspond aux routes issues de l'entrée $S6$ et dirigées vers la sortie S_1 :

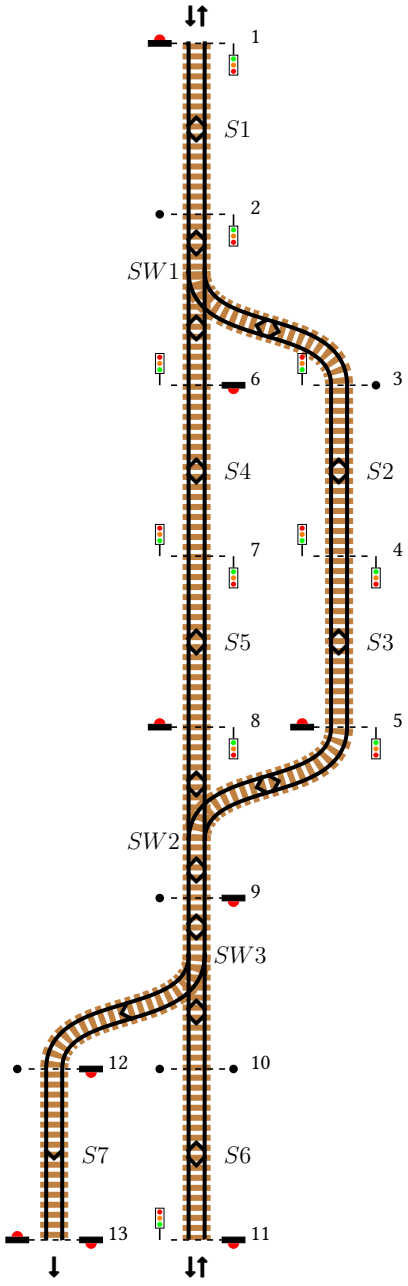


FIGURE 6.7 – Un exemple d'un nœud ferroviaire

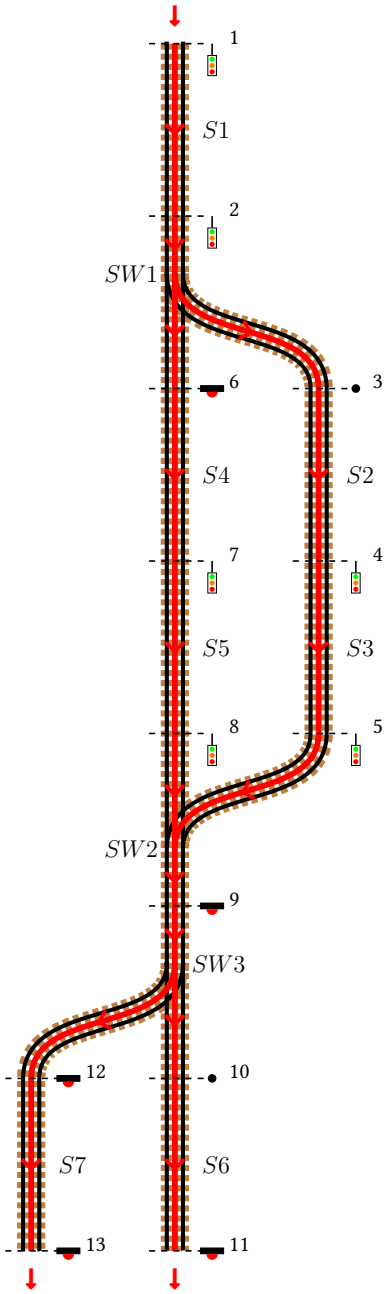


FIGURE 6.8 – Traversée Γ_1 d'un nœud ferroviaire de la figure 6.7 avec l'instrumentation correspondante

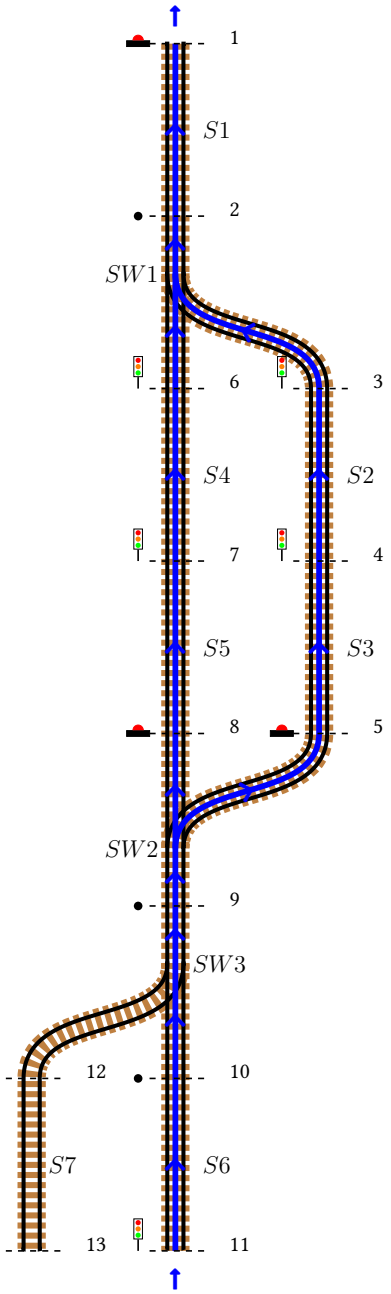


FIGURE 6.9 – Traversée Γ_2 d'un nœud ferroviaire de la figure 6.7 avec l'instrumentation correspondante

— $S6 - SW3 - SW2 - S3 - S2 - SW1 - S1$

— $S6 - SW3 - SW2 - S5 - S4 - SW1 - S1$

Comme pour le réseau nous ferons référence aux composants d'une traversée Γ par $\mathfrak{C}_\Gamma$ et les connexions par $\mathfrak{J}_\Gamma$, ces deux notions pouvant être étendues par $\rightarrow$ et $\leftarrow$.

Dans le cas de la traversée Γ_1 on a donc :

— $\overrightarrow{\mathfrak{C}}_\Gamma = \bigcup_{i=1}^{7} S_i \cup \bigcup_{i=1}^{3} SW_i$

— $\overleftarrow{\mathfrak{C}}_\Gamma = \varnothing$

— $\overrightarrow{\mathfrak{J}}_\Gamma = \bigcup_{i=1}^{13} \mathfrak{j}_i$

— $\overleftarrow{\mathfrak{J}}_\Gamma = \varnothing$

Et dans le cas de la traversée Γ_2 :

— $\overrightarrow{\mathfrak{C}}_\Gamma = \varnothing$

— $\overleftarrow{\mathfrak{C}}_\Gamma = \bigcup_{i=1}^{6} S_i \cup \bigcup_{i=1}^{3} SW_i$

— $\overrightarrow{\mathfrak{J}}_\Gamma = \varnothing$

— $\overleftarrow{\mathfrak{J}}_\Gamma = \bigcup_{i=1}^{11} \mathfrak{j}_i$

6.2 Modélisation

Cette section présente comment obtenir un modèle Réseau de Petri à partir de l'infrastructure d'un nœud ferroviaire, le concept sera ensuite étendu pour prendre en compte la notion de traversée.

6.2.1 Modélisation des composants

Pour pouvoir obtenir de manière générique un modèle RdP à partir de la structure d'un nœud ferroviaire, nous commençons par définir des modèles génériques pour chaque composant de l'infrastructure. Chaque composant est vu comme une zone géographique de l'infrastructure. L'idée est donc de représenter cette zone par une ou deux places selon qu'il s'agisse d'un composant monodirectionnel ou bidirectionnel. De même, le franchissement de chaque connecteur est vu comme un événement. Nous associons donc une transition à chaque franchissement de connecteur.

Considérons le schéma d'une section de voie monodirectionnelle donnée par la Figure 6.1. Pour cette section de voie, nous supposons que le sens de déplacement est de la gauche vers la droite. Nous représentons alors la section de voie par la place notée $\overrightarrow{p}$. Dans cette notation, le sens de la flèche schématise le sens du déplacement de la gauche vers la droite. Au connecteur x, nous associons une transition d'entrée notée $\overrightarrow{t_x}$. De même au connecteur de sortie, nous associons une transition de sortie de la section notée $\overleftarrow{t_x}$. Il en résulte le modèle de la Figure 6.10 pour une section de voie monodirectionnelle. Nous en déduisons le modèle de la Figure 6.11 pour une section de voie bidirectionnelle.

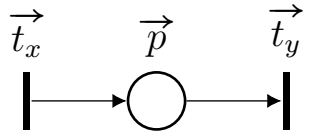


FIGURE 6.10 – Modèle Réseau de Petri d'une section de voie monodirectionnelle

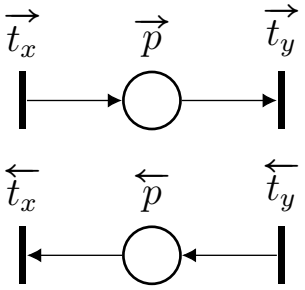


FIGURE 6.11 – Modèle Réseau de Petri section de voie bidirectionnelle

Avec une approche similaire, nous pouvons modéliser un aiguillage simple avec 3 voies bidirectionnelles 6.12. Nous pouvons constater qu'il y a en fait deux modèles Réseaux de Petri, un pour chaque sens d'utilisation de l'aiguillage. Le rôle des connecteurs change selon le sens de parcours. Ainsi au connecteur x, correspond deux transitions : $\overleftarrow{t}_x$ pour les traversées de la gauche vers la droite et $\overleftarrow{t}_x$ pour les circulations de la droite vers la gauche. La flèche au niveau du nom de chaque transition indique le sens de parcours et l'indice le nom de la connexion franchie.

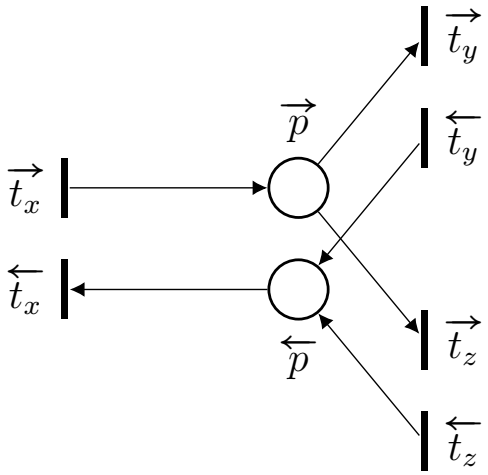


FIGURE 6.12 – Modèle Réseau de Petri d'un aiguillage

6.2.2 Modélisation de l'instrumentation

Le franchissement d'un train d'un composant à un autre composant est représenté par le tir d'une transition. L'instrumentation de la connexion entre les deux composants influe sur la nature de la transition, plus précisément sur sa relation vis-à-vis du contrôle. Pour une connexion j_i donnée, le nombre de transitions dépend du nombre de sens de passage que la transition autorise créant deux transitions $\overrightarrow{t_i}$

et $\overleftarrow{t_i}$ si $j_i \in \overrightarrow{\mathfrak{J}} \cap \overleftarrow{\mathfrak{J}}$ ou une seule : $\overrightarrow{t_i}$ ou $\overleftarrow{t_i}$ selon si $j_i \in \overrightarrow{\mathfrak{J}}$ ou $j_i \in \overleftarrow{\mathfrak{J}}$. L'instrumentation dépend du sens de passage pour une transition $\overrightarrow{t_i}$ (respectivement $\overleftarrow{t_i}$) donnée l'instrumentation dépendra directement de $\overrightarrow{\mathfrak{Ins}}(j_i)$ (respectivement $\overleftarrow{\mathfrak{Ins}}(j_i)$)

- Une instrumentation de type **autorisation de mouvement** (🚦) , permet d'interdire le passage du train d'un composant à un autre. De plus, l'hypothèse est faite que le contrôleur a l'information du passage du train si celui-ci traverse la connexion. La transition associée est donc **contrôlable et observable**.
- Une instrumentation de type **détecteur de trains** (🚄), permet au contrôleur d'avoir l'information du passage d'un train sur la connexion. La transition associée est donc **incontrôlable et observable**.
- Dans le cas d'une connexion **sans instrumentation** (•), le contrôleur ne peut ni empêcher le passage du train ni avoir l'information de son passage. La transition associée est donc **incontrôlable et inobservable**.

6.2.3 Modèle d'un nœud ferroviaire

Pour construire le modèle RdP d'un nœud ferroviaire, il est juste nécessaire d'assembler les modèles RdP des composants des différentes traversées en mettant en relation les transitions de sortie et d'entrée de deux composants adjacents. L'exemple des figures 6.13, 6.14 et 6.15 montre un agencement de deux composants, un segment S et un aiguillage SW. S et SW partagent une connexion en commun, la connexion 2. Ce partage de connexions implique que les deux composants sont donc adjacents comme montré dans la figure 6.15.

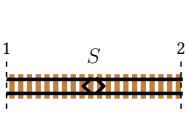


FIGURE 6.13 – Segment S

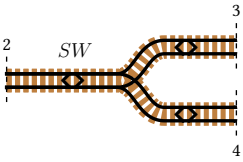


FIGURE 6.14 – Aiguillage SW

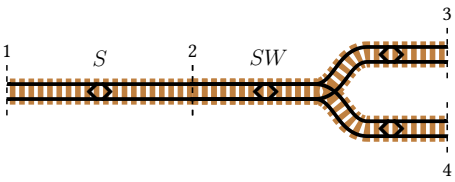


FIGURE 6.15 – Fusion de S et SW

On retrouve en figure 6.16 (respectivement 6.17), le Réseau de Petri modélisant le segment S (respectivement SW) est noté $\mathcal{N}_S$ (respectivement $\mathcal{N}_{SW}$). On remarque que les places comportent l'indice $_S$ (respectivement $_{SW}$) et que les transitions comportent en indice la connexion qu'elles réfèrent. Le réseau $\mathcal{N} = \mathcal{N}_S \cup \mathcal{N}_{SW}$ est obtenu par union de $\mathcal{N}_S$ et $\mathcal{N}_{SW}$ suivant la définition 95. Ce raisonnement est extensible à un nombre quelconque de composants.

Définition 95. *On note $\mathcal{N} = \cup_{i=1}^{n} \mathcal{N}_i$ le réseau obtenu par l'union d'un ensemble de réseau de Petri $\{\mathcal{N}_1, \mathcal{N}_2 ... \mathcal{N}_n\}$ avec $\forall i \in [1..n], \mathcal{N}_i = (P_i, T_i, F_i, W_i)$. $\mathcal{N} = (P, T, F, W)$ est définie comme :*

- $P = \bigcup_{i=1}^{n} P_i$
- $T = \bigcup_{i=1}^{n} T_i$
- $F = \bigcup_{i=1}^{n} F_i$

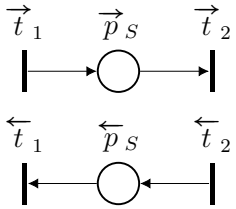


FIGURE 6.16 – Réseau de Petri modélisant $\mathcal{N}_S$ le segment S

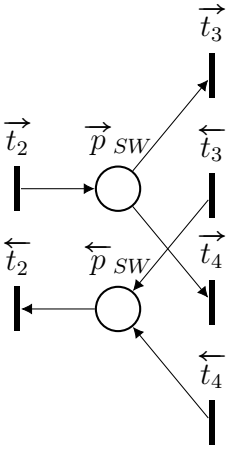


FIGURE 6.17 – Réseau de Petri $\mathcal{N}_{SW}$ modélisant le l'aiguillage SW

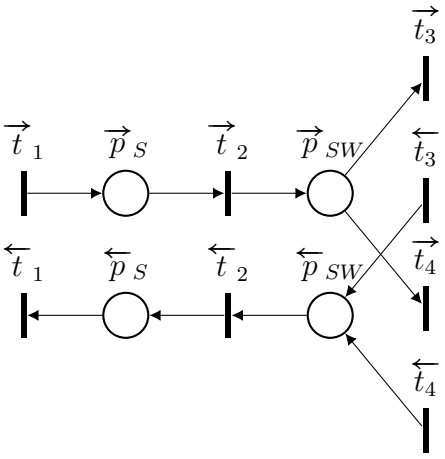


FIGURE 6.18 – Réseau de Pétri $\mathcal{N}$ obtenu par l'union des réseaux $\mathcal{N}_S$ et $\mathcal{N}_{SW}$

— $\forall i \in [1..n], \forall f \in F_i, W(f) = W_i(f)$

Définition 96. *On note l'ensemble des transitions sources du système* $T_{in} = \{t \in T \,|\, {}^\bullet t = \varnothing\}$ *dont le tir modélise l'entrée d'un train dans le nœud. De la même manière on note* $T_{out} = \{t \in T \,|\, t^\bullet = \varnothing\}$ *les transitions puits du système, qui modélisent la sortie d'un train du nœud.*

En appliquant cette procédure au nœud ferroviaire présenté en figure 6.7 on obtient le Réseau de Petri illustré en figure 6.19. Le réseau est obtenu par l'union de tous les composants, les Réseaux de Petri de chaque composant ne sont pas détaillés ici.

On remarque que le réseau comporte deux transitions sources $T_{in} = \{\overrightarrow{t_1}, \overleftarrow{t_{11}}\}$ et comporte trois transitions puits $T_{out} = \{\overleftarrow{t_1}, \overrightarrow{t_{11}}, \overrightarrow{t_{13}}\}$. Dans le cas du nœud de la figure 6.7 le sens de circulation des trains se fait verticalement, ici nous adoptons la notation $\overrightarrow{p_x}$ pour la présence d'un train dans un composant x d'un train allant de haut en bas et $\overleftarrow{p_x}$ pour allant de bas en haut.

6.2.4 Modèle des traversées

La notion de traversée constitue une restriction du comportement libre du système. Les restrictions du fonctionnement du système sont classiquement associées à la notion de contrôle. Ce n'est pas le cas de notre approche, les modèles des traversées ne sont pas obtenus par une méthode de synthèse de contrôleurs mais par une extraction directe d'un sous-réseau du système libre, et le réseau obtenu considéré comme modèle.

En prenant en compte les composants que chaque traversée comprend et le sens de parcours de ces composants et leurs connexions, il est possible d'extraire les places et les transitions du réseau de Petri libre.

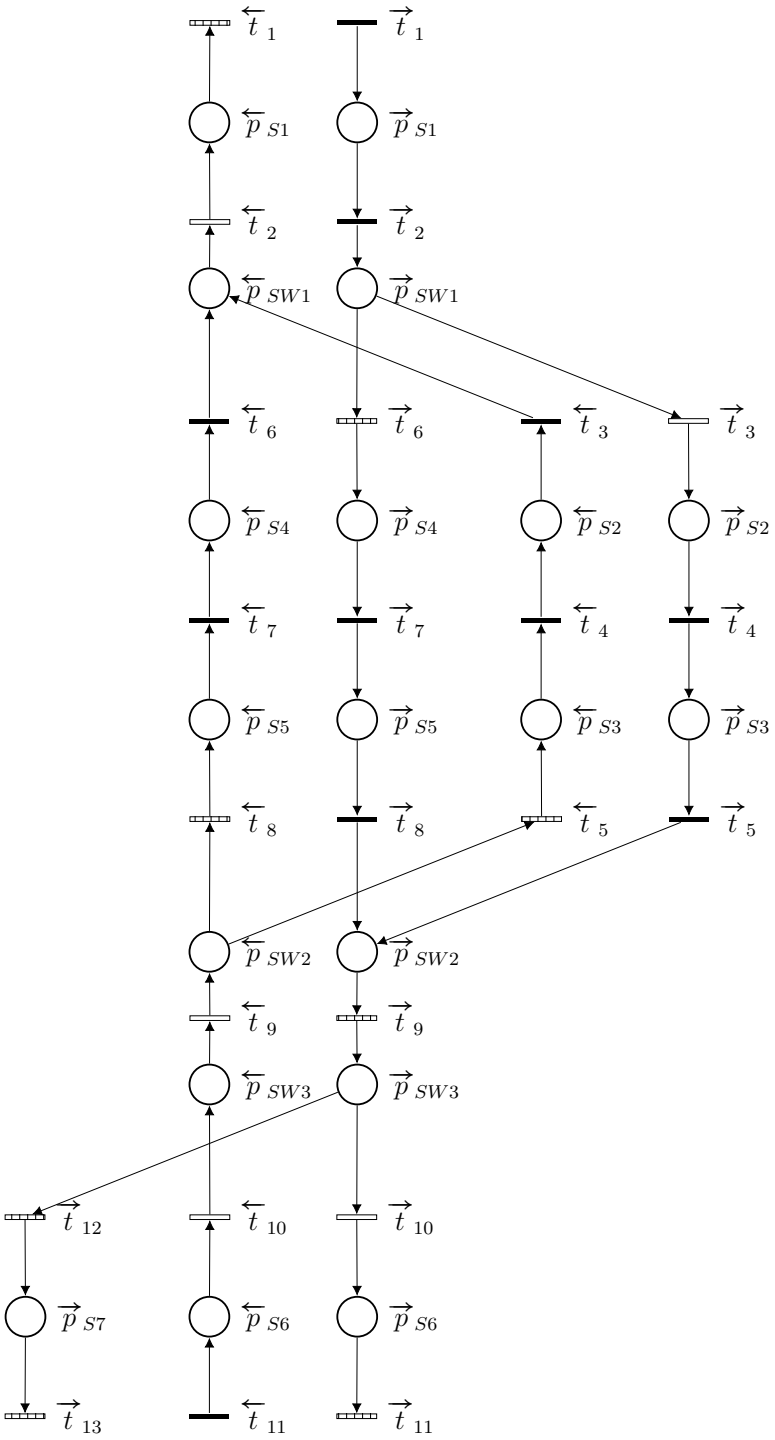


FIGURE 6.19 – Réseau de Petri $\mathcal{N}$ obtenu par l'union de tous les Réseaux de Petri des composants du nœud de la figure 6.7

Définition 97. *Considérons une traversée quelconque Γ, le réseaux de Petri $\mathcal{N}_\Gamma$ est décrit de la manière suivante : $\mathcal{N}_\Gamma = (\{p^0_\Gamma\} \cup P^P_\Gamma, T^\Gamma, F_\Gamma)$ avec*

- *$p^{0,\Gamma}$ est la place d'attente de $\mathcal{N}_\Gamma$. Cette place d'attente est ajoutée dans le but de borner le nombre de jetons dans le $\mathcal{N}_\Gamma$. Ses arcs de sortie sont liés aux transitions qui modélisent l'entrée d'un train dans le réseau et les arcs d'entrée aux transitions modélisant la sortie des trains du nœud.*
- *P^P_Γ sont les places de processus de $\mathcal{N}_\Gamma$. Les places de P^P_Γ modélisent les composants du nœud à travers lesquels la traversée passe. P^P_Γ est définie par :*

$$P^P_\Gamma = \bigcup_{\mathfrak{c} \in \overrightarrow{\mathfrak{C}}_\Gamma} \overrightarrow{p}^\Gamma_\mathfrak{c} \cup \bigcup_{\mathfrak{c} \in \overleftarrow{\mathfrak{C}}_\Gamma} \overleftarrow{p}^\Gamma_\mathfrak{c}$$

- *T_Γ sont les transitions de $\mathcal{N}_\Gamma$. Les places de T_Γ modélisent les connexions du nœud par lesquelles la traversée passe. T_Γ est définie par :*

$$T_\Gamma = \bigcup_{\mathfrak{j} \in \overrightarrow{\mathfrak{J}}_\Gamma} \overrightarrow{t}^\Gamma_\mathfrak{j} \cup \bigcup_{\mathfrak{j} \in \overleftarrow{\mathfrak{J}}_\Gamma} \overleftarrow{t}^\Gamma_\mathfrak{j}$$

- *F_Γ sont les arcs de $\mathcal{N}_\Gamma$. Avec*

$$F_\Gamma = \{f_\Gamma | f = (x^\Gamma, y^\Gamma) \in T^\Gamma \times P^\Gamma \cup P^\Gamma \times T^\Gamma \wedge \exists f = (x, y) \in F\}$$
$$\cup \{f | \forall t \in T_{in}, (p^{0,\Gamma}, t^\Gamma)\} \cup \{f | \forall t \in T_{out}, (t^\Gamma, p^{0,\Gamma})\}$$

À l'état initial M^Γ_0, seules les places d'attentes sont marquées. A l'état initial aucun train ne suit la traversée Γ. $p^{0,\Gamma}$ comporte autant de jetons que la traversée à de composants car nous verrons par la suite que le nœud ne peut contenir qu'un train par composant. Autrement dit :

$$\begin{array}{cc} & M^\Gamma_0(p^{0,\Gamma}) = |\mathfrak{C}_\Gamma| \\ \forall p \in P^\Gamma / p^{0,\Gamma} & M^\Gamma_0(p) = 0 \end{array}$$

Propriété 36. *Un réseau de traversée correspond à la définition 52 d'un processus si la traversée ne comprend pas de cycles. Avec $p^{0,\Gamma}$ qui est la place d'attente du processus.*

Les transitions de T^Γ gardent les même propriétés vis-à-vis de la COP que leurs homologues de T.

Les Réseaux de Petri modélisant les traversées Γ_1 et Γ_2 sont présentés dans les réseaux de Petri 6.20 et 6.21.

6.3 Contrôle : évitement des collisions et des blocages

Dans le cadre de notre approche, la définition des traversées suffit au routage des trains. Suite à l'étape de modélisation nous avons un ensemble de réseaux de Petri dont chacun correspond au modèle d'une traversée. Dans cette section une méthodologie de contrôle est mise en place pour permettre d'avoir des trains qui suivent ces différentes traversées sans que les trains puissent entrer en collision ni que cela entraine de blocages.

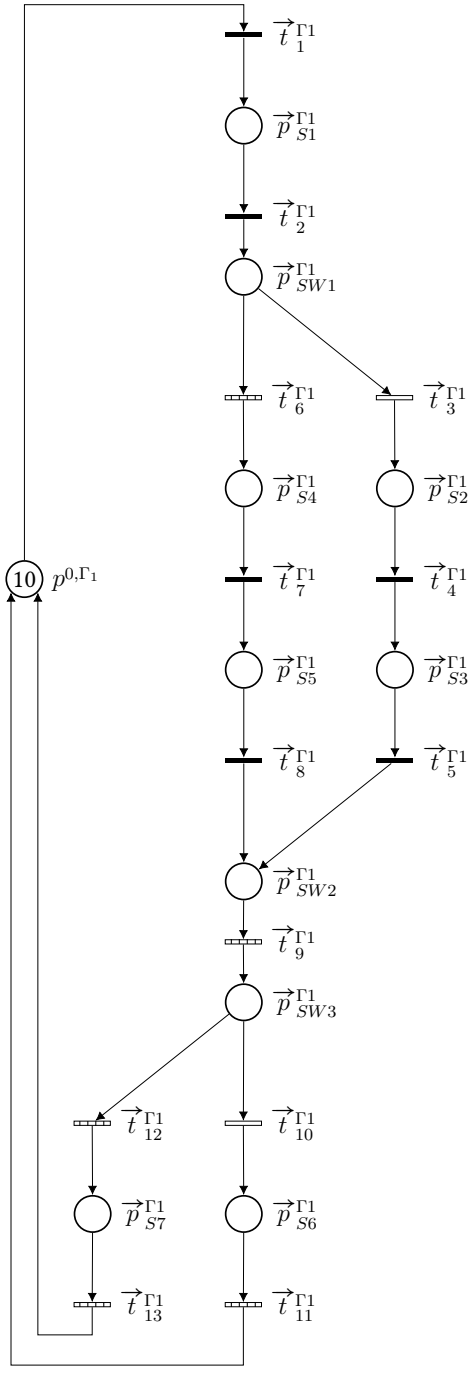


FIGURE 6.20 – Réseau de Petri modélisant la traversée Γ_1

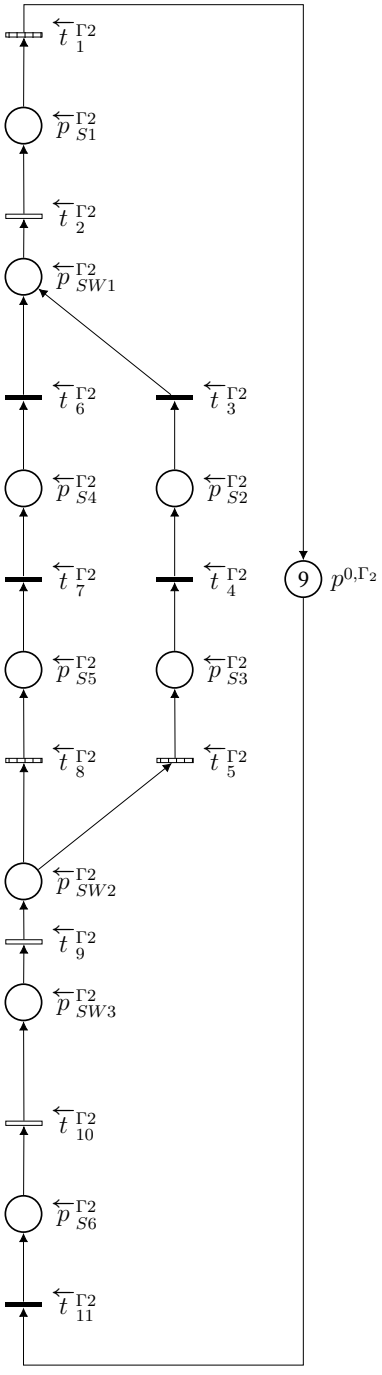


FIGURE 6.21 – Réseau de Petri modélisant la traversée Γ_2

6.3.1 Évitement des collisions

Une première étape de contrôle consiste à générer des places monitrices de manière à éviter les collisions. Ici nous considérons qu'une collision ne peut arriver si un seul train est présent par composants. De manière à ce qu'aucune collision n'arrive, un ensemble de contrainte sont définies, empêchant d'avoir plus d'un train par composant.

Définition 98. *Considérons un nœud ferroviaire dont les trains suivent un ensemble de traversées* $\{\Gamma_1, \Gamma_2, ..., \Gamma_n\}$. *On définit le Réseau de Petri* $\mathcal{N}^0$ *formé par l'union des réseaux de traversée* $\mathcal{N}^0 = \bigcup_{i=1}^{n} \mathcal{N}_{\Gamma i}$. $\mathcal{N}^0$ *peut être vu comme un ensemble de processus indépendants évoluant en parallèle et sans interaction.*

Définition 99. *Considérons un réseau* $\mathcal{N}^0$, *pour chaque composant du nœud* $\mathfrak{c} \in \mathfrak{C}$ *une contrainte GMEC* $(w_\mathfrak{c}, 1)$ *est défini par :*

$$\forall i \in [1..n], \quad si\ \mathfrak{c} \in \overrightarrow{\mathfrak{C}}_{\Gamma_i},\ w_\mathfrak{c}(\overrightarrow{p}_\mathfrak{c}^{\Gamma_i}) = 1,\ \ si\ \mathfrak{c} \in \overleftarrow{\mathfrak{C}}_{\Gamma_i},\ w_\mathfrak{c}(\overleftarrow{p}_\mathfrak{c}^{\Gamma_i}) = 1$$
$$\forall i \in [1..n], \quad \forall p \in P^{\Gamma_i} / \{\overrightarrow{p}_\mathfrak{c}^{\Gamma_i}, \overleftarrow{p}_\mathfrak{c}^{\Gamma_i}\},\ w_\mathfrak{c}(p) = 0$$

On note W *l'ensemble des vecteurs de contraintes d'un nœud.*

Il est impossible de générer directement des places de contrôles appliquant ces contraintes car elles ne respectent pas nécessairement la COP. Il est nécessaire de rendre ces contraintes admissibles. Pour cela on applique la définition 90 qui permet de trouver une contrainte admissible $\check{w}$ à partir d'une contrainte non-admissible w à l'ensemble des vecteurs de contraintes de W. L'ensemble des contraintes ainsi obtenu est noté $\check{W}$. Il est possible qu'un certain nombre de contraintes de $\check{W}$ ait des comportements redondants voire que certaines contraintes soien inutiles (moins contraignantes que d'autres). On purge alors ces contraintes $\check{W}$.

Dans le cas des deux traversées Γ_1 et Γ_2 l'ensemble des contraintes W et $\check{W}$ sont récapitulées dans le tableau 6.1. On remarque que les vecteurs de contraintes $\check{w}_{S6}$ et $\check{w}_{SW3}$ sont identiques et que le vecteur $\check{w}_{SW2}$ moins contraignant que $\check{w}_{SW3}$ et $\check{w}_{S6}$ (i.e $\mathcal{M}_i(\check{w}_{S6}, 1) = \mathcal{M}_i(\check{w}_{SW3}, 1) \supset \mathcal{M}_i(\check{w}_{SW2}, 1)$). On considerera pour la suite que $\check{w}_{S6}$ et $\check{w}_{SW2}$ sont exclus de $\check{W}$.

Définition 100. *Pour chaque contrainte* $\check{w}_\mathfrak{c}$ *de* $\check{W}$ *on génère une place monitrice* $p_\mathfrak{c}^R$. *L'ensemble de ces places monitrices est nommé* P^R

Définition 101. *On note* $\mathcal{N}^1$ *le S3PMR obtenu après ajout de* P^R *à* $\mathcal{N}^0$. *Chaque réseau de traversée correspond à un processus de* $\mathcal{N}^1$. *Les places monitrices d'évitement de collision de* P^R *sont équivalentes à des places de ressources.*

Dans le cadre de notre exemple, le S3PMR $\mathcal{N}^1$ est représenté en figure 6.22. Il modélise donc un ensemble de trains évoluant en parallèle et suivant, soit la traversée Γ_1, soit la traversée Γ_2, tout en s'assurant qu'aucun train ne puisse rentrer en collision.

6.3.2 Évitement des blocages

Le réseau $\mathcal{N}^1$ est un S3PMR, une classe de RAS-PN, qui induit donc de potentiels blocages. Cette seconde phase de contrôles aura donc pour but d'éviter les blocages

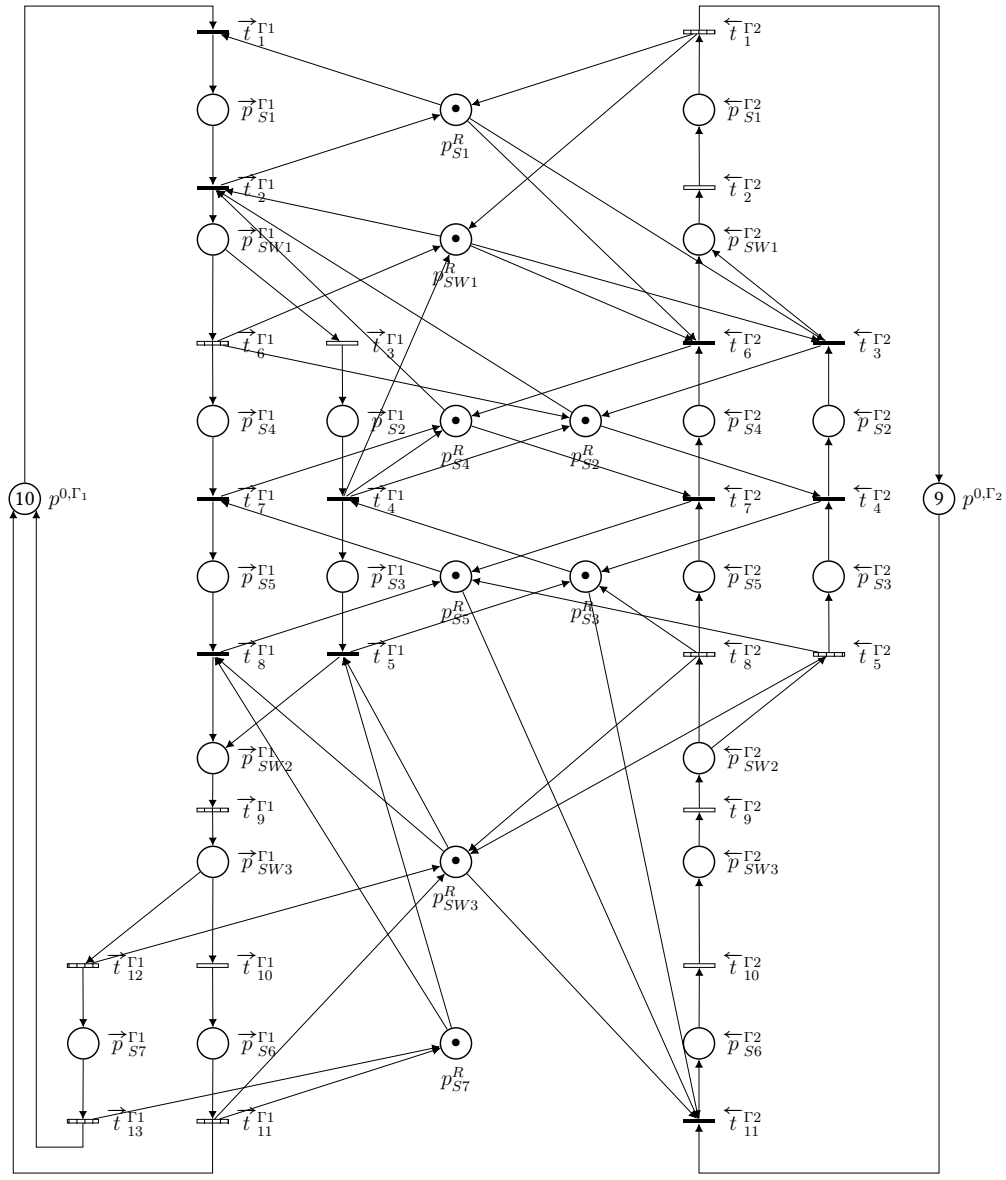


FIGURE 6.22 – S3PMR $\mathcal{N}^1$ après ajout des places monitrices d'évitement de collision

Composant $\mathfrak{c}$	Place avec $w_{\mathfrak{c}} = 1$	Place avec $\check{w}_{\mathfrak{c}} = 1$
$S1$	$\overrightarrow{p}^{\Gamma 1}_{S1}, \overleftarrow{p}^{\Gamma 2}_{S1}$	$\overrightarrow{p}^{\Gamma 1}_{S1}, \overleftarrow{p}^{\Gamma 2}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{S1}$
$S2$	$\overrightarrow{p}^{\Gamma 1}_{S2}, \overleftarrow{p}^{\Gamma 2}_{S2}$	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overleftarrow{p}^{\Gamma 2}_{S2}$
$S3$	$\overrightarrow{p}^{\Gamma 1}_{S3}, \overleftarrow{p}^{\Gamma 2}_{S3}$	$\overrightarrow{p}^{\Gamma 1}_{S3}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{S3}$
$S4$	$\overrightarrow{p}^{\Gamma 1}_{S4}, \overleftarrow{p}^{\Gamma 2}_{S4}$	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overrightarrow{p}^{\Gamma 1}_{S4}, \overleftarrow{p}^{\Gamma 2}_{S4}$
$S5$	$\overrightarrow{p}^{\Gamma 1}_{S5}, \overleftarrow{p}^{\Gamma 2}_{S5}$	$\overrightarrow{p}^{\Gamma 1}_{S5}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{S5}$
$S6$	$\overrightarrow{p}^{\Gamma 1}_{S6}, \overleftarrow{p}^{\Gamma 2}_{S6}$	$\overrightarrow{p}^{\Gamma 1}_{SW2}, \overrightarrow{p}^{\Gamma 1}_{SW3}, \overrightarrow{p}^{\Gamma 1}_{S6}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}$
$S7$	$\overrightarrow{p}^{\Gamma 1}_{S7}$	$\overrightarrow{p}^{\Gamma 1}_{SW2}, \overrightarrow{p}^{\Gamma 1}_{SW3}, \overrightarrow{p}^{\Gamma 1}_{S6}, \overrightarrow{p}^{\Gamma 1}_{S7}$
$SW1$	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{SW1}$	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overleftarrow{p}^{\Gamma 2}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{S1}$
$SW2$	$\overrightarrow{p}^{\Gamma 1}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{SW2}$	$\overrightarrow{p}^{\Gamma 1}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}$
$SW3$	$\overrightarrow{p}^{\Gamma 1}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW3}$	$\overrightarrow{p}^{\Gamma 1}_{SW2}, \overrightarrow{p}^{\Gamma 1}_{SW3}, \overrightarrow{p}^{\Gamma 1}_{S6}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}$

TABLE 6.1 – Tableau des contraintes W et $\check{W}$ pour le nœud de la figure 6.7 parcouru par les traversées Γ_1 et Γ_2

qui peuvent apparaitre dans le réseau $\mathcal{N}^1$. Le chapitre 5 présente les techniques de contrôles qui seront appliquées ici. En effet les propriétés de $\mathcal{N}^1$ (S3PMR) nous permettent d'appliquer directement méthodes de synthèse de contrôleurs de contrôles H2' et Z2'. Cette sous-section présentera donc uniquement les résultats obtenus dans le cadre de notre exemple pour les méthodes de synthèse de contrôleurs H2' et Z2' appliquées au S3PMR de la figure 6.22. Les S4PR obtenus après application d'une méthode de synthèse sont nommés $\mathcal{N}^2$, et modélisent le nœud ferroviaire dont les trains suivent les traversées sans que les trains ne puissent rentrer en collision ni être bloqués. $\mathcal{N}^2$ est vivant.

Technique de synthèse de contrôles H2'

La technique de synthèse de contrôles H2' est appliquée au S3PMR $\mathcal{N}^1$ de la figure 6.22. H2' génère quatre places monitrices. La seconde place monitrice générée p_1^W est de type généralisé et redirige donc les arcs vers les transitions sources des processus. Les trois autres places de contrôles générées, p_1^V, p_2^V et p_3^V , sont des places monitrices ordinaires. Les places monitrices sont récapitulées dans le tableau 6.2. Le graphe d'accessibilité $\mathcal{R}(\mathcal{N}^2, M_0^2)$ comporte 242 états et est générée en 2,05s.

Technique de synthèse de contrôles Z2'

La technique de synthèse de contrôles Z2' est appliquée au S3PMR $\mathcal{N}^1$ de la figure 6.22. Z2' génère sept places monitrices. Les places monitrices sont récapitulées dans le tableau 6.3. Le graphe d'accessibilité $\mathcal{R}(\mathcal{N}^2, M_0^2)$ comporte 302 états et le contrôle est généré en 6,16.

p	S	$^\bullet p_i^V$	$p_i^{V\bullet}$	$M_0(p_i^V)$
p_1^V	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S3}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{S1}, p^R_{S2}, p^R_{S3}, p^R_{SW1}$	$\overrightarrow{t}^{\Gamma 1}_{4}, \overrightarrow{t}^{\Gamma 1}_{7}, \overleftarrow{t}^{\Gamma 2}_{5}, \overleftarrow{t}^{\Gamma 2}_{6}$	$\overrightarrow{t}^{\Gamma 1}_{1}, \overleftarrow{t}^{\Gamma 2}_{11}$	2
p_1^W	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overrightarrow{p}^{\Gamma 1}_{S5}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{S1}, p^R_{S1}, p^R_{S4}, p^R_{S5}$	$2\overrightarrow{t}^{\Gamma 1}_{4}, 2\overrightarrow{t}^{\Gamma 1}_{7}, \overleftarrow{t}^{\Gamma 2}_{3}, \overleftarrow{t}^{\Gamma 2}_{8}$	$2\overrightarrow{t}^{\Gamma 1}_{1}, \overleftarrow{t}^{\Gamma 2}_{11}$	2
p_2^V	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overrightarrow{p}^{\Gamma 1}_{S4}, \overleftarrow{p}^{\Gamma 2}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{S1}, p^R_{S1}, p^R_{S4}$	$\overrightarrow{t}^{\Gamma 1}_{2}, \overleftarrow{t}^{\Gamma 2}_{6}$	$\overrightarrow{t}^{\Gamma 1}_{1}, \overleftarrow{t}^{\Gamma 2}_{7}$	1
p_3^V	$\overrightarrow{p}^{\Gamma 1}_{S6}, \overleftarrow{p}^{\Gamma 2}_{S2}, \overleftarrow{p}^{\Gamma 2}_{S4}, \overleftarrow{p}^{\Gamma 2}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{S1}, p_1^V$	$\overrightarrow{t}^{\Gamma 1}_{4}, \overrightarrow{t}^{\Gamma 1}_{7}, \overleftarrow{t}^{\Gamma 2}_{5}, \overleftarrow{t}^{\Gamma 2}_{7}$	$\overrightarrow{t}^{\Gamma 1}_{1}, \overleftarrow{t}^{\Gamma 2}_{11}$	1

Table 6.2 – Places de contrôle obtenues par application de la technique de synthèse de contrôles H2p sur le S3PMR $\mathcal{N}^1$ de la figure 6.22

i	S_i	$^\bullet p_i^V$	$p_i^{V\bullet}$	$M_0(p_i^V)$
1	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overleftarrow{p}^{\Gamma 2}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{S1}, p^R_{S1}, p^R_{S2}$	$\overrightarrow{t}^{\Gamma 1}_{2}, \overleftarrow{t}^{\Gamma 2}_{3}$	$\overrightarrow{t}^{\Gamma 1}_{1}, \overleftarrow{t}^{\Gamma 2}_{4}$	1
2	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overrightarrow{p}^{\Gamma 1}_{S4}, \overleftarrow{p}^{\Gamma 2}_{SW1}, \overleftarrow{p}^{\Gamma 2}_{S1}, p^R_{S1}, p^R_{S4}$	$\overrightarrow{t}^{\Gamma 1}_{2}, \overleftarrow{t}^{\Gamma 2}_{6}$	$\overrightarrow{t}^{\Gamma 1}_{1}, \overleftarrow{t}^{\Gamma 2}_{7}$	1
3	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S3}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{S2}, p^R_{S2}, p^R_{S3}$	$\overrightarrow{t}^{\Gamma 1}_{4}, \overrightarrow{t}^{\Gamma 1}_{7}, \overleftarrow{t}^{\Gamma 2}_{4}, \overleftarrow{t}^{\Gamma 2}_{8}$	$\overrightarrow{t}^{\Gamma 1}_{2}, \overleftarrow{t}^{\Gamma 2}_{11}$	1
4	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overrightarrow{p}^{\Gamma 1}_{S5}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{S4}, p^R_{S4}, p^R_{S5}$	$\overrightarrow{t}^{\Gamma 1}_{4}, \overrightarrow{t}^{\Gamma 1}_{7}, \overleftarrow{t}^{\Gamma 2}_{5}, \overleftarrow{t}^{\Gamma 2}_{7}$	$\overrightarrow{t}^{\Gamma 1}_{2}, \overleftarrow{t}^{\Gamma 2}_{11}$	1
5	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overrightarrow{p}^{\Gamma 1}_{S4}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{S4}, p_2^V, p_4^V$	$\overrightarrow{t}^{\Gamma 1}_{2}, \overleftarrow{t}^{\Gamma 2}_{5}, \overleftarrow{t}^{\Gamma 2}_{7}$	$\overrightarrow{t}^{\Gamma 1}_{1}, \overleftarrow{t}^{\Gamma 2}_{11}$	1
6	$\overrightarrow{p}^{\Gamma 1}_{SW1}, \overrightarrow{p}^{\Gamma 1}_{S2}, \overrightarrow{p}^{\Gamma 1}_{S4}, \overleftarrow{p}^{\Gamma 2}_{S6}, \overleftarrow{p}^{\Gamma 2}_{SW3}, \overleftarrow{p}^{\Gamma 2}_{SW2}, \overleftarrow{p}^{\Gamma 2}_{S2}, p_1^V, p_3^V$	$\overrightarrow{t}^{\Gamma 1}_{2}, \overleftarrow{t}^{\Gamma 2}_{4}, \overleftarrow{t}^{\Gamma 2}_{8}$	$\overrightarrow{t}^{\Gamma 1}_{1}, \overleftarrow{t}^{\Gamma 2}_{11}$	1

Table 6.3 – Places de contrôle obtenues par application de la technique de synthèse de contrôles Z2' sur le S3PMR $\mathcal{N}^1$ de la figure 6.22

6.4 Conclusion et remarques

Ce chapitre présente une méthode pour le contrôle des trains dans un nœud ferroviaire de type jonction, avec un routage dynamique. Le contrôle empêche les collisions entre les trains et les situations de blocages. Connaissant la topologie du réseau et les points d'entrées et de sorties des trains, un Réseau de Petri est généré.

L'instrumentation du nœud induit la présence de transitions incontrôlables et inobservables dans le Réseau de Petri. Une première phase de contrôle est appliquée à ce Réseau qui empêche les collisions entre les trains dans le nœud. Après la génération des places de contrôle de type ressource, le Réseau de Petri obtenu est un S3PMR. Pour empêcher les situations de blocages dans le S3PMR, les méthodes H2' ou Z2' peuvent être appliquées et permettent d'obtenir un S4PR vivant. La méthode proposée ici est une méthode hors ligne de synthèse, une fois que le contrôle est mis en place, il est direct. Les perspectives de ce travail concernent le développement d'une approche permettant de faire une synthèse rapide de contrôleurs pour la reconfiguration dynamique du système en cas d'apparition de défauts.

Le choix entre les méthodes H2' et Z2' influe directement sur les performances du contrôle. Pour avoir une étude fine de l'impact du choix de la méthode sur les performances, il serait souhaitable d'avoir une approche statistique. Cela nécessiterait de mesurer les performances de chaque méthode sur des nœuds ferroviaires et de traversées générées aléatoirement. Ce travail est une piste d'amélioration. Sans ce travail il semble cependant possible d'anticiper certains résultats. L'approche H2' génère des places de contrôles p^W qui viennent contraindre toutes les places de processus amont au siphon extrait. Cela veux dire que pour une traversée donnée, le nombre de trains sur une portion potentiellement étendue, peut être fortement réduit. Dans le cas où les traversées sont longues, l'utilisation d'une telle méthode peut engendrer un contrôle extrêmement restrictif. La méthode Z2' semble clairement générer un contrôle plus permissif que H2' mais c'est au détriment d'une complexité algorithmique et structurelle plus élevée. Dans le cas où ces critères ne sont pas primordiaux (la synthèse est faite hors ligne et la mise en place du contrôle peut être automatisée), Z2' est à favoriser.

Cette approche ouvre de nombreuses pistes de recherche et d'amélioration. Une des pistes d'amélioration notable au niveau de la modélisation consisterait à prendre en compte les aiguillages dans le modèle. En effet ici leurs présence est omise, et est sous-entendue dans la notion de traversées. Leur absence dans le modèle crée un non-déterminisme dans la direction que le train peut prendre, ce qui induit un contrôle plus restrictif que si le train pouvait sélectionner sa direction. Pour prendre en compte les aiguillages dans le modèle, la classe des S4PR n'est plus suffisante et il devient nécessaire de s'orienter vers des classes plus complexes autorisant des cycles internes, des arcs non-purs ou des prêts de ressources.

Chapitre 7

Conclusion et perspectives

Cette thèse s'intéresse à la problématique de la synthèse de contrôleurs. Les premières méthodes de synthèse de contrôleurs pour les systèmes à événements discrets (SED) ont été développées à la fin des années 1990, et ont mené au développement de la théorie du contrôle supervisé (SCT). La SCT garantit un contrôle optimal au prix d'une grande complexité algorithmique. En effet, cette méthode se base sur l'exploration des états du système.

Les travaux de cette thèse ont été réalisés dans l'objectif final de pouvoir générer automatiquement une loi de contrôle pour un système ferroviaire. Les systèmes ferroviaires étant des systèmes complexes et de grandes tailles, les méthodes basées sur l'exploration d'états se trouvent être incompatibles avec notre application. Ces travaux se sont naturellement tournés vers le formalisme des Réseaux Petri, qui permet l'utilisation d'outils issus de l'algèbre linéaire que ce soit pour l'analyse ou le contrôle. Même si ce formalisme permet l'utilisation d'outils puissants, la synthèse d'un contrôleur, sans recourir à l'exploration des états, est impossible dans le cas général. Pour pouvoir arriver à ce but, un cadre plus restreint est mis en place. Au lieu d'étudier les Réseaux de Petri dans le cas général, ce travail s'est placé dans le cadre des systèmes d'allocations de ressources (SAR). Les systèmes d'allocations de ressources sont une sous-classe des SED qui peuvent modéliser un grand nombre d'applications, dont parmi elles, les systèmes ferroviaires. Des classes de Réseaux de Petri ont été spécialement développées pour modéliser de tels systèmes (classes de RAS-PN). Les systèmes d'allocations de ressources sont connus pour entraîner des blocages. De nombreuses méthodes de synthèse de contrôleurs, dans le but d'éviter les blocages dans les RAS-PN ont été développées dans la littérature, et ce sujet se trouve être aujourd'hui scientifiquement très développé. Le développement de ces méthodes se place très souvent dans le cadre d'application manufacturière, où le niveau d'abstraction élevé n'oblige pas à prendre en compte, ni l'existence d'événements incontrôlables, ni l'existence d'événements inobservables. Dans le cadre d'une application ferroviaire, le modèle comporte de tels événements. Un travail d'extension de ces méthodes de contrôle a donc été mis en place. Plus particulièrement ce sont les méthodes d'extraction de siphon qui ont été étudiées. Ces méthodes permettent de détecter une situation de blocage dans les Réseaux de Petri, par la résolution d'un problème d'optimisation mixte en nombres entiers (MIP), ce qui évite de passer par l'exploration des états. De plus, la méthode retourne un siphon, une structure d'un Réseau de Petri qui est d'un grand intérêt du point de vue du contrôle. Les stratégies de contrôle développées sont itératives ; les opérations d'extraction d'un siphon puis son contrôle sont répétées jusqu'à ce que le Réseau de Petri contrôlé n'entraîne plus de blocage. Le contrôle d'un siphon se fait via la

génération une place monitrice. C'est lors de la génération de cette place monitrice que l'influence des transitions incontrôlables et inobservables est prise en compte. Les méthodes de synthèse ainsi développées sont finalement appliquées à un système ferroviaire. Une méthodologie de modélisation permet d'obtenir un Réseau de Petri qui s'inscrit dans le cadre de notre étude. Le contrôle ainsi obtenu garantit un routage dynamique des trains dans un nœud ferroviaire tout en assurant l'absence de collisions et de blocages. Les méthodes de synthèse de contrôleurs pour RAS-PN sous COP : H2', Z2' et celle présentée dans [QLAA15] ont été codées sous MATLAB.

7.1 Résumé du manuscrit

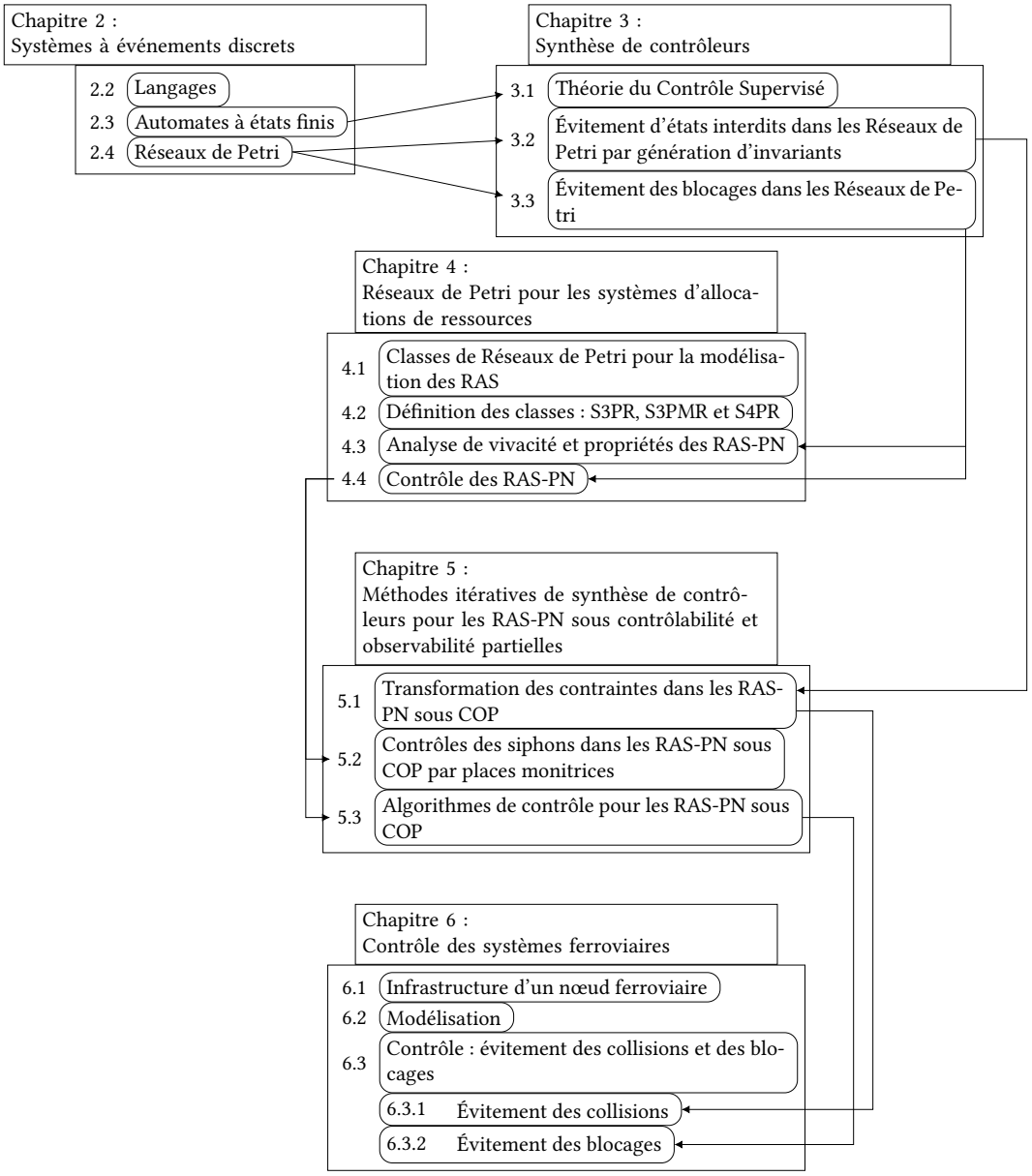


FIGURE 7.1 – Structure du manuscrit

La structure du manuscrit est rappelée dans la figure 7.1, et les dépendances entre les différentes sections et sous-sections sont illustrées.

Le chapitre 2 présente trois formalismes pour la modélisation des SED ; les langages, les automates et les Réseaux de Petri. Les automates et les langages sont des notions uniquement utiles pour la présentation de la théorie du contrôle supervisé (section 3.1). L'essentiel de la thèse se base sur le formalisme des Réseaux de Petri. La section 2.4 présente les notations, les propriétés et les méthodes d'analyse basiques relatives aux Réseaux de Petri.

Le chapitre 3 s'axe sur la thématique de la synthèse de contrôleurs. La section 3.1 présente la méthode historique de Ramadge et Woham (SCT), puis celles étendues aux systèmes partiellement contrôlables et observables. Ces méthodes sont appliquées aux automates. Le reste des travaux ne se base pas sur la SCT, ni même sur les automates mais cette section permet d'introduire la notion de contrôle optimal. La section 3.2 et la section 3.3 concernent uniquement les Réseaux de Petri. La section 3.2 traite de l'évitement d'états interdits, plus précisément la génération de places monitrices pour le contrôle de spécifications de type GMEC. Ici les transitions incontrôlables et inobservables sont considérées. Dans ce cadre sont caractérisées des contraintes dites "admissibles" engendrant des places monitrices qui respectent les conditions relatives à l'observabilité et la contrôlabilité partielles du système. La section 3.3 pose les concepts de base de la synthèse de contrôleurs pour l'évitement des blocages. Dans cette section les transitions incontrôlables et inobservables ne sont pas considérées. Cette section repose sur le concept des siphons ; une structure particulière au sein d'un Réseau de Petri fortement corrélée avec l'apparition de blocages. Ici aussi le contrôle est mis en place via des places monitrices. On cherchera donc, à partir d'un siphon donné, à définir une contrainte qui assure son contrôle. Deux cas sont identifiés : pour les Réseaux de Petri ordinaires où il suffit d'éviter qu'un siphon ne se vide pour le contrôler et dans le cas des Réseaux de Petri généralisé, les conditions nécessaires et suffisantes au contrôle d'un siphon sont beaucoup plus complexes et à notre connaissance toujours indéterminées. Finalement des méthodes d'extraction de siphon dans un Réseau de Petri ordinaire et généralisé sont présentées.

Le chapitre 4 présente des classes de Réseaux de Petri pour la modélisation des systèmes d'allocations de ressources (RAS-PN) et des méthodes de synthèse de contrôle pour l'évitement des blocages associées au RAS-PN. Les RAS-PN sont des classes qui comportent des propriétés intéressantes vis-à-vis de l'évitement des blocages et permettent de se placer dans un cadre plus favorable que dans celui des Réseaux de Petri généralisés. La section 4.1 donne une vue d'ensemble des différentes classes de RAS-PN présentes dans la littérature. Elle permet d'identifier clairement les trois classes (S3PR, S3PMR et S4PR) sur lesquelles se base le reste du manuscrit. Les structures de ces trois classes de RAS-PN sont détaillées dans la section 4.2 . La section 4.3 reprend les notions présentées dans la section 3.3 sur les propriétés relatives aux blocages et leurs relations avec les siphons. Ces notions sont adaptées à la structure particulière des RAS-PN. La section 4.4 reprend aussi des éléments de la section 3.3 pour les appliquer aux RAS-PN. On y retrouve le contrôle d'un siphon via la génération d'une place monitrice et une méthode d'extraction de siphon formulée pour les S4PR. Finalement, différentes méthodes de synthèses de contrôles pour les RAS-PN de la littérature sont comparées.

Le chapitre 5 concerne les RAS-PN et étend les méthodologies de contrôle présentées dans le chapitre précédent au cas de la contrôlabilité et observabilité partielles. La section 5.1 reprend des éléments de la partie 3.2 notamment la notion de contrainte admissible et de transformation de contrainte. Ici une méthode de transformation de contraintes GMEC est développée spécialement pour les RAS-PN. La contrainte ainsi obtenue engendre des places monitrices ayant les mêmes caractéristiques que des places de ressources. La section 5.2 s'appuie sur la section 4.4. Elle concerne l'évitement des blocages pour l'étendre au contexte de la contrôlabilité et de l'observabilité partielles. La section 5.3 s'appuie sur les résultats précédents pour étendre deux méthodes de synthèse de contrôleurs (Z2 et H2) pour des classes de RAS-PN aux cas de la contrôlabilité et l'observabilité partielles. Par la suite, sont présentées différentes méthodes de synthèse de contrôleurs pour RAS-PN sous COP.

Le chapitre 6 présente une méthode de modélisation et de contrôle pour les nœuds ferroviaires et exploite les résultats du chapitre 5. Le contrôle route dynamiquement les trains dans le nœud. Les trains peuvent adapter leur itinéraire en fonction de la saturation du noeud. La section 6.1 présente les différents composants du nœud, l'infrastructure mais aussi un concept novateur : les traversées. Les traversées sont un ensemble d'itinéraires permettant potentiellement à un train de sortir en plusieurs points d'un nœud ferroviaire pour atteindre sa destination finale, à partir d'une entrée. La notion de traversée est fortement liée au routage dynamique. Dans la section 6.2, une méthodologie de modélisation est développée. Partant de la topologie du nœud ferroviaire et des différentes traversées qu'il autorise, la méthodologie permet d'obtenir un Réseau de Petri. Pour chaque traversée un Réseau de processus est obtenu. La section 6.3 présente une méthode pour contrôler l'ensemble des Réseaux de processus obtenus. La synthèse du contrôle est divisée en deux parties, l'une concernant l'évitement des collisions, l'autre concernant l'évitement des blocages. Dans la sous-section 6.3.1, une collision dans un composant donné est ramenée à une contrainte GMEC. La génération d'une contrainte GMEC par composant du système assure l'évitement des collisions. En appliquant les résultats de la section 5.1, un ensemble de places monitrices est généré. Le Réseau contrôlé ainsi obtenu est un S3PMR. Ainsi dans la section 6.3.2, pour éviter les blocages, les algorithmes développés dans la section 5.3 sont simplement appliqués.

7.2 Perspectives

Ces travaux répondent à la problématique : « Comment appliquer la théorie du contrôle supervisé aux systèmes ferroviaires pour un contrôle automatisé garantissant une sécurité maximale ? ». Ces travaux se sont orientés vers la synthèse de contrôleurs pour le routage dynamique des trains. Dans cette optique des pistes d'amélioration persistent.

Du point de vue de la synthèse de contrôleurs, les critères : complexité structurelle, permissivité et complexité algorithmique permettent de juger de la performance d'une méthode de synthèse. Les méthodes de type itératives ont été développées au cours de ces travaux, et peuvent amener à créer des places monitrices redondantes. Ces places monitrices ont un impact direct sur la complexité structurelle. Pour réduire la complexité structurelle, il peut être intéressant de mettre en œuvre

une méthode pour supprimer ces places. Si la suppression d'une place n'affecte pas la vivacité du système alors il s'agit d'une place monitrice redondante. La vivacité du Réseau de Petri peut être testée par extraction d'un siphon. Si l'extraction réussit, le Réseau n'est pas vivant et donc la place n'est pas redondante. Une méthodologie simple consiste à tester la redondance de chaque place l'une après l'autre. Dans le cas ou une place redondante est trouvée, on la supprime immédiatement. L'ordre dans lequel les places sont sélectionnées influe directement sur le nombre de places supprimées, et une combinaison optimale des places non-redondantes n'est pas garantie. Une piste d'amélioration consisterait à développer une méthode proche des siphons élémentaires. En se basant sur les matrices d'incidences des places monitrices, il serait possible de diviser les places monitrices en deux types ; "élémentaires" et "dépendantes". Ainsi seules les places monitrices élémentaires seraient gardées, ce qui pourrait engendrer un gain notable de complexité structurelle.

L'utilisation des places monitrices, dont les arcs sont réorientés vers les transitions sources (notées p^W), peut être amené à supprimer un nombre important de marquages autorisés. La présence de ces places induit une perte de permissivité relativement importante. L'opération de réorientation des arcs vers les transitions sources permet de garantir que la présence de la place monitrice dans le Réseau contrôlé ne crée pas de nouveaux siphons avec de mauvais marquages associés. Cependant cette opération est brutale. Pour augmenter la permissivité, il peut être intéressant de ramener les arcs moins "haut" dans les processus. Une condition simple de non-génération de siphons consisterait à remonter à la première transition non-incluse dans un réseau de processus. Cette condition, implique l'absence d'attente circulaire et donc de blocages. Une méthode plus complexe peut être mise en place : itérativement un arc d'une place monitrice est réorienté sur "une transition en aval" dans le processus jusqu'à ce que le réseau contrôlé soit non-vivant. Dans ce cas-là, l'itération précédente indique la position limite de l'arc. Les itérations sont répétées pour chaque arc de chaque place monitrice. Ce type de méthode peut améliorer la permissivité au prix d'une plus grande complexité algorithmique.

La notion de traversée permet une approche novatrice pour le routage dynamique des trains. L'obtention d'un modèle Réseau de Petri à partir de la topologie d'un nœud et de la définition des traversées est directe en suivant la méthodologie présentée. Cependant, des éléments importants de l'architecture d'un nœud ferroviaire ont été omis : les aiguillages. Les aiguillages constituent des éléments importants de la partie opérative. Leur prise en compte dans le modèle est une nécessité. Plusieurs approches sont envisageables. Une première stratégie consisterait à inclure ces aiguillages dans le modèle de base. Cette stratégie nécessite une révision complète de l'approche de cette thèse. En effet la prise en compte des aiguillages peut amener à avoir des modèles de RAS-PN de classes plus complexes que ceux nous avons utilisés (moins restrictives que les S4PR). Le développement d'une méthode de synthèse de contrôle, sous contrôlabilité et observabilité partielle serait donc à refaire pour une classe de RAS-PN moins restrictive que les S4PR .

Une seconde stratégie consisterait à intégrer les aiguillages après génération des places monitrices. Dans ce cas-là, une phase de contrôle supplémentaire doit être développée de manière à ce que les aiguillages soit positionnés correctement, et de manière sécurisée. Cette phase de contrôle ne doit pas remettre en question

l'absence de blocage dans le système.

Enfin, les travaux de modélisation peuvent aussi être étendus. Dans l'état actuel, les nœuds ne permettent d'inclure que deux types de composants : les aiguillages et les sections de voie. Une poursuite logique consisterait à venir étoffer l'ensemble des composants possibles d'un nœud (passage à niveau, croisement perpendiculaire de rails...). Une extension possible serait aussi de venir considérer différents types de trains (frets et passager) qui circulent dans un système ferroviaire. Les systèmes de signalisation n'étant pas nécessairement les mêmes pour des types de trains différents, l'instrumentation devra être complexifiée.

Annexe A

Complément sur les Réseaux de Petri

A.1 Structure des Réseaux de Petri

A.1.1 Graphe

Définition 102. *Un Réseau de Petri* $\mathcal{N} = (P, T, F, W)$ *peut être transformé en un graphe orienté* $H(\mathcal{N}) = (V, E)$*. Les places et les transitions de* $\mathcal{N}$ *constituent les nœuds de* $H(\mathcal{N})$*, i.e* $V = P \cup T$ *et les arcs sont maintenus* $E = F$*.*

Définition 103. *On dit qu'il existe un chemin* $\lambda = v_1 v_2 ... v_n \in V^*$ *avec* $n \in \mathbb{R}$ *dans un Réseau de Petri* $\mathcal{N} = (P, T, F, W)$*, s'il est possible de parcourir* $H(\mathcal{N})$ *en respectant l'ordre des nœuds de* λ*.*

$$\forall v_i, v_{i+1} \in \lambda \text{ avec } i \in [1..n],\ \exists e \in E \,|\, e = (v_i, v_{i+1})$$

L'ensemble des chemins d'un Réseau de Petri $\mathcal{N}$ *est noté* $\Lambda(\mathcal{N})$*.*

Définition 104. *On dit qu'un chemin* $\lambda = v_1 v_2 ... v_n \in \Lambda(\mathcal{N})$ *est un cycle si* $v_1 = v_n$

Définition 105. *Un Réseau de Petri* $\mathcal{N}$ *est dit fortement connecté s'il existe un chemin entre tous les nœuds de son graphe associé* $H(\mathcal{N}) = (V, E)$*.*

$$\forall v_1, v_n \in V, \exists \lambda \in \Lambda(\mathcal{N}) \,|\, \lambda = v_1 v_2 ... v_n$$

A.1.2 Classes de Réseaux de Petri

A.1.3 Machine à états

Définition 106. *Un Réseau de Petri ordinaire* $\mathcal{N} = (P, T, F)$ *est une machine à états si chacune des transitions a une seule place d'entrée et une seule place de sortie i.e* $\forall t \in T, |t^\bullet| = 1$ *et* $|{}^\bullet t| = 1$*.*

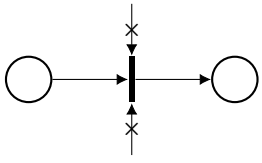


Figure A.1 – Restriction appliquée à une machine à état

Annexe B

Code MATLAB

B.1 Structure Classes

B.1.1 Petri net class

```
classdef PN
    %PN A Petri net class

    properties(SetAccess = protected)
        C    % incidence matrix
        pre  % pre matrix
        post % post matrix
        M0   % initial matrix
    end
    properties
        Tuc  % uncontrolable transition vector
        Tuo  % unobservable transition vector
    end

    properties (Dependent, SetAccess = protected)
        sizeP
        sizeT
    end

    methods
        function obj = PN(pre,post,M0)
            %PN Construct an instance of this class
            %   Construct a petri net form it pre and post matrix and it
            %   initial marking
            if(size(pre)~=size(post))
                error(" pre and post matrix must have the same size")
            end
            [sizeP, sizeT]=size(pre);
            if(size(M0)~=[sizeP, 1])
                error("M0 must be a column vector with |P| lines")
            end
            obj.pre = pre;
            obj.post = post;
            obj.M0 = M0;
            obj.C= post-pre;
            obj.Tuc= false(sizeT,1);
            obj.Tuo= false(sizeT,1);
        end

        function value = get.sizeP(obj)
            [value, ~]= size(obj.C);
        end

        function value = get.sizeT(obj)
            [~,value]= size(obj.C);
        end

        function obj=set.Tuc(obj,Tuc)
            if (~islogical(Tuc))
```

```
            error("Error Tuc must be a logical vector")
        elseif (size(Tuc)~=[obj.sizeT, 1])
            error("Tuc must be a column vector with |T| lines")
        end
        obj.Tuc=Tuc;
    end

    function obj=set.Tuo(obj,Tuo)
        if (~islogical(Tuo))
            error("Error Tuc must be a logical vector")
        elseif (size(Tuo)~=[obj.sizeT, 1])
            error("Tuc must be a column vector with |T| lines")
        end
        obj.Tuo=Tuo;
    end
    %% Function related to dynamic beavior

    function Tenabled=Tenabled(obj, M)
        % return the transitions enabled at a given marking
        Tenabled=all(M*ones(1,obj.sizeT)-obj.pre>=0)';
    end

    function reachedState=reachedState(obj, M, vectT)
        % return the state reached from M by a firing vector. The
        % function fail if the marking obtened is not positive. The
        % function do not check if the firing vector correspond to a
        % real sequence of transitions.
        reachedState=M+obj.C*vectT;
        if(any(reachedState<0))
            error("the reached state does not correspond to a possible State")
        end
    end

    function RG=RG(obj)
        % return the reachability graph as a matlab digraph

        RG=digraph();
        marking=obj.M0';
        MUntreated=table(marking);
        Name=string(int2str(obj.M0'));
        RG=RG.addnode(table(marking, Name, 'RowNames', Name));
        while(~isempty(MUntreated))
            M=MUntreated.marking(1,:)';
            MUntreated(1,:)=[];
            Tenabled=obj.Tenabled(M);
            for t=find(Tenabled)'
                vectT=zeros(obj.sizeT,1);
                vectT(t)=1;
                Mp=reachedState(obj, M, vectT);
                try(RG.Nodes(string(int2str(Mp')),:));
                catch ME
                    if(~isempty(ME))
                        marking=Mp';
                        MUntreated=[MUntreated;table(marking)];
                        Name=string(int2str(Mp'));
                        RG=RG.addnode(table(marking, Name, 'RowNames', Name));
                    end
                end
                NewEdge = table([string(int2str(M')), string(int2str(Mp'))],
                    strcat("t", string(t)), 'VariableNames',{'EndNodes','
                    transitions'});
                RG=RG.addedge(NewEdge);
            end
        end
    end

    % successor and predecessor of nodes

    function Psucc=Psucc(obj,VectP)
        if ~any(VectP)
            Psucc=false(obj.sizeT,1);
            return
```

```
            end
            Psucc=any(obj.C(VectP, :)<0,1);
        end

        function Ppred=Ppred(obj,VectP)
            if ~any(VectP)
                Ppred=false(obj.sizeT,1);
                return
            end
            Ppred=any(obj.C(VectP, :)>0,1);
        end

        function Tsucc=Tpost(obj,VectT)
            if ~any(VectT)
                Tsucc=false(obj.sizeP,1);
                return
            end
            Tsucc=any(obj.C(VectT, :)>0,2);
        end

        function Tpred=Tpred(obj,VectT)
            if ~any(VectT)
                Tpred=false(obj.sizeP,1);
                return
            end
            Tpred=any(obj.C(:, VectT)<0,2);
        end
    end
end
```

B.1.2 Process class

```
classdef Process < PN
    %PROCESS Petri Net class of a Process
    %   Process nets are petri nets states machines with all cycles going
    %   Through a special place p^0 (the net is acyclic if p^0 is deleted).
    %   A process is a Petri net and can be described with a incidence
    %   matrix
    methods

        function objProcess = Process(C,M0)
            %PROCESS Construct a process from a incidence matrix
            %   Fail if the Petri net is not a strongly
            %   connected state machine and if the cycles dont go through
            %   p^0. M0 is the initial marking of the idle place, must be
            %   an interger greater than 0.
            %   WARNING : The first line of C must be related to the idle places.
            %   Input
            %   C : incidence matrix
            %   M0: marking of the idle place

            objProcess@PN(-C.*(C<0),C.*(C>0),M0 );
            if(~isStateMachine(objProcess))
                error("C is not the incidence matrix of a State machine");
            end
            retCheckCycles = checkCycles(objProcess);
            if(retCheckCycles==0)
                error("C is not the incidence matrix of a connexe Petri net");
            elseif(retCheckCycles==-1)
                error("C is not the incidence matrix of process without inner cycles
                    ");
            end
        end

        function isStateMachine = isStateMachine(Obj)
            %finite_state Check if the Proces is a finite state machine.
            %Return 1 if the petri net is a finite state machine else 0.
            isStateMachine = 1;
            if(any(abs(Obj.C(:))>1))
```

```
                isStateMachine = 0;
                return;
            else
                pre=(Obj.C==-1);
                post=(Obj.C==1);
                if(any(sum(pre,1)>1))
                    isStateMachine = 0;
                    return;
                elseif(any(sum(post,1)>1))
                    isStateMachine = 0;
                    return;
                end
            end
        end

        function checkCycles = checkCycles(Obj)
            %finite_state Check if the Proces is a finite state machine.
            %Return 1 if the petri net is a finite state machine, 0 if not
            % connexe and -1 if is connexe witout the cycles
            %going throught p^0.
            checkCycles = 1;
            [sizeP, sizeT]=size(Obj.C);
            adjencyMat=[ zeros(sizeP, sizeP), Obj.C==-1   ;
                      (Obj.C==1)',  zeros(sizeT, sizeT) ];
             procGraph =digraph(adjencyMat);
            [~, compSize] = conncomp(procGraph);
            if(length(compSize)~=1)
                checkCycles = 0;
                return;
            end
            C_proc=Obj.C(2:sizeP,:);
            adjencyMat=[ zeros(sizeP-1, sizeP-1), C_proc==-1   ;
            (C_proc==1)',  zeros(sizeT, sizeT) ];
            procGraph =digraph(adjencyMat);
            [~, compSize] = conncomp(procGraph);
            if(length(compSize)~=sizeP-1+sizeT)
                checkCycles = -1;
            return;
            end
        end


        function [Pup] = Pup(Obj, p)
        %Pup Return the set of places in the upstream path of a place p
        % p is a process places
        % Pup is a set of places (in logical array)
        % p is inclued in Pup


            adjP=zeros(Obj.sizeP); % adjP is a adgency matrix for process places (
                transitions as vertices)
            for t=1:Obj.sizeT
              adjP(logical(Obj.pre(:,t)),:)=adjP(logical(Obj.pre(:,t)),:)+Obj.post
                 (:,t)';
            end
            adjP(:,1)=0; %edges from sink places to p^0 are deleted
            Pup=zeros(Obj.sizeP, 1);
            Pup(p)=1;
            while (~all(Pup== ((Pup+adjP*Pup)>0))) %add predecessor places util
                fixed point
                Pup= (Pup+adjP*Pup)>0;
            end
            Pup(1)=0; %p^0 is deleted
        end

    end
end
```

B.1.3 S4PR class

```
classdef Process < PN
    %PROCESS Petri Net class of a Process
    %   Process nets are petri nets states machines with all cycles going
    %   Through a special place p^0 (the net is acyclic if p^0 is deleted).
    %   A process is a Petri net and can be described with a incidence
    %   matrix

    methods

        function objProcess = Process(C,M0)
            %PROCESS Construct a process from a incidence matrix
            %   Fail if the Petri net is not a strongly
            %   connected state machine and if the cycles dont go through
            %   p^0. M0 is the initial marking of the idle place, must be
            %   an interger greater than 0.
            %   WARNING : The first line of C must be related to the idle places.
            %   Input
            %   C : incidence matrix
            %   M0: marking of the idle place

            objProcess@PN(-C.*(C<0),C.*(C>0),M0 );
            if(~isStateMachine(objProcess))
                error("C is not the incidence matrix of a State machine");
            end
            retCheckCycles = checkCycles(objProcess);
            if(retCheckCycles==0)
                error("C is not the incidence matrix of a connexe Petri net");
            elseif(retCheckCycles==-1)
                error("C is not the incidence matrix of process without inner cycles
                    ");
            end
        end

        function isStateMachine = isStateMachine(Obj)
            %finite_state Check if the Proces is a finite state machine.
            %Return 1 if the petri net is a finite state machine else 0.
            isStateMachine = 1;
            if(any(abs(Obj.C(:))>1))
                isStateMachine = 0;
                return;
            else
                pre=(Obj.C==-1);
                post=(Obj.C==1);
                if(any(sum(pre,1)>1))
                    isStateMachine = 0;
                    return;
                elseif(any(sum(post,1)>1))
                    isStateMachine = 0;
                    return;
                end
            end
        end

        function checkCycles = checkCycles(Obj)
            %finite_state Check if the Proces is a finite state machine.
            %Return 1 if the petri net is a finite state machine, 0 if not
            % connexe and -1 if is connexe witout the cycles
            %going throught p^0.
            checkCycles = 1;
            [sizeP, sizeT]=size(Obj.C);
            adjencyMat=[ zeros(sizeP, sizeP), Obj.C==-1   ;
                      (Obj.C==1)',  zeros(sizeT, sizeT) ];
             procGraph =digraph(adjencyMat);
            [~, compSize] = conncomp(procGraph);
            if(length(compSize)~=1)
                checkCycles = 0;
                return;
            end
            C_proc=Obj.C(2:sizeP,:);
            adjencyMat=[ zeros(sizeP-1, sizeP-1), C_proc==-1   ;
            (C_proc==1)',  zeros(sizeT, sizeT) ];
            procGraph =digraph(adjencyMat);
```

```
            [~, compSize] = conncomp(procGraph);
            if(length(compSize)~=sizeP-1+sizeT)
                checkCycles = -1;
            return;
            end
        end


        function [Pup] = Pup(Obj, p)
        %Pup Return the set of places in the upstream path of a place p
        % p is a process places
        % Pup is a set of places (in logical array)
        % p is inclued in Pup


            adjP=zeros(Obj.sizeP); % adjP is a adgency matrix for process places (
                transitions as vertices)
            for t=1:Obj.sizeT
              adjP(logical(Obj.pre(:,t)),:)=adjP(logical(Obj.pre(:,t)),:)+Obj.post
                  (:,t)';
            end
            adjP(:,1)=0; %edges from sink places to p^0 are deleted
            Pup=zeros(Obj.sizeP, 1);
            Pup(p)=1;
            while (~all(Pup== ((Pup+adjP*Pup)>0))) %add predecessor places util
                fixed point
                Pup= (Pup+adjP*Pup)>0;
            end
            Pup(1)=0; %p^0 is deleted
        end

    end
end
```

B.2 Problem Generation

B.2.1 MIP [CX97]

```
function[mip]=MIPSiphonOrdinary(N)
    %genetate a mip for the analysis of deadlock analysis in ordinary petri
    %net using the method from 10.1109/70.650158
    %Once solved the mip return a vector [v; z; m; y] where
    % v is a binary vector of the place ouside of the siphon
    % z is a binary vector of the transisition that are not successor of
    % that siphon
    % m is the place marking
    % y is the firing vector

    narcs_tp=0;
    for t = 1:N.sizeT
        for p = 1:N.sizeP
            if N.post(p,t)>0
                narcs_tp=narcs_tp+1;
            end
        end
    end


    %% see chu and xie paper to get the reference of  equations%%

    mip.f=[ones(1,N.sizeP),zeros(1,N.sizeP+2*N.sizeT)];
    mip.intcon=1:(N.sizeP+ N.sizeT);
    A=zeros(N.sizeT+N.sizeP+narcs_tp,2*N.sizeP+2*N.sizeT);
    B=zeros(N.sizeT+N.sizeP+narcs_tp,1);
    Aeq=zeros(N.sizeP,2*N.sizeP+2*N.sizeT);
    Beq=zeros(N.sizeP,1);
    %%equa 2  %%
    for t = 1:N.sizeT
```

```
    vecteur_id=eye(N.sizeT);
    vecteur_id=vecteur_id(:,t);
    part1=N.pre(:,t)';
    part2=-transpose(vecteur_id);
    part3=zeros(1,N.sizeP);
    part4=zeros(1,N.sizeT);
    A(t,:)=[part1,part2,part3,part4];
    B(t)=sum(N.pre(:,t)>0)-1;
end

%%equa 3 %%

cpt_arcs=0;
for t = 1:N.sizeT
    for p = 1:N.sizeP
        if N.post(p,t)>0
            cpt_arcs=cpt_arcs+1;
            vecteur_id=eye(N.sizeP);
            vecteur_id=vecteur_id(p,:);
            part1=-vecteur_id;
            vecteur_id=eye(N.sizeT);
            vecteur_id=vecteur_id(:,t);
            part2=vecteur_id';
            part3=zeros(1,N.sizeP);
            part4=zeros(1,N.sizeT);
            A(N.sizeT+cpt_arcs,:)=[part1,part2,part3,part4];
            B(N.sizeT+cpt_arcs)=0;
        end
    end
end


%%equa 5 %%
for p = 1:N.sizeP
    %%calcul de sb pour la place p%%
    vecteur_id=zeros(1,N.sizeP);
    vecteur_id(p)=1;
    vecteur_id=[vecteur_id, zeros(1,N.sizeT)];
    option=optimoptions('linprog','Display','off');
    [sb,FVAL,EXITFLAG] =linprog(-vecteur_id,[],[],[-eye(N.sizeP),N.C],-N.M0,
        zeros(N.sizeP+N.sizeT,1),[],option);
    if(EXITFLAG~=1)
       error("no solution found")
       EXITFLAG
    end
    sb_p=sb(p);

    vecteur_id=eye(N.sizeP);
    vecteur_id=vecteur_id(p,:);
    part1=-vecteur_id;
    part2=zeros(1,N.sizeT);
    part3=vecteur_id/sb_p;
    part4=zeros(1,N.sizeT);
    A(N.sizeT+narcs_tp+p,:)=[part1,part2,part3,part4];
    B(N.sizeT+narcs_tp+p,:)=0;
end

%% state equation

for p = 1:N.sizeP


    vecteur_id=eye(N.sizeP);
    vecteur_id=[vecteur_id(p,:)];
    part1=zeros(1,N.sizeP);
    part2=zeros(1,N.sizeT);
    part3=-vecteur_id;
    part4=N.C(p,:);
    Aeq(p,:)=[part1,part2,part3,part4];
end
Beq=-N.M0';
```

```
    mip.Aineq = A;
    mip.bineq = B;
    mip.Aeq = Aeq;
    mip.beq = Beq;
    mip.lb = [zeros(2*N.sizeP+2*N.sizeT,1)];
    mip.solver = 'intlinprog';
    mip.options = optimoptions('intlinprog','Display','off');
end
```

B.2.2 MIP[1] [ZDW+19]

```
function [Problem] = MIPSiphonS4PR(N)
%   MIPSiphon Generate a MIP from a S4PR to extract a siphon following
%   the method from 10.1109/ACCESS.2019.2939855
    v=1:(N.sizeP-sum(N.P0));
    p2v=zeros(N.sizeP,1);
    p2v(~N.P0)=v;

    tau=v(end)+1:v(end)+N.sizeT-sum(N.Tsources);
    t2tau=zeros(N.sizeT,1);
    t2tau(~N.Tsources)=tau;

    sizeFRTin=(N.C(N.PR,:)>0);
    sizeFRTin(:,N.Tsources)=0;
    sizeFRTin=sum(sizeFRTin(:));
    f=tau(end)+1: tau(end)+sizeFRTin;
    rt2f=zeros(size(N.C));
    rt2f_log=(N.C<0);
    rt2f_log(1:end-N.nbR,:)=0;
    rt2f_log(:,N.Tsources)=0;
    rt2f(rt2f_log)=cumsum(rt2f_log(rt2f_log));

    M=f(end)+1: f(end)+N.sizeP;
    Y=M(end)+1: M(end)+N.sizeT;

    size_var=Y(end);
    % helpful declarations for forloop


    ImatsousP0=eye(N.sizeP,'logical');
    ImatsousP0(:,N.P0)=[];

    ImatR=eye(N.sizeP,'logical');
    ImatR(:,~N.PR)=[];

    ImatsousT0=eye(N.sizeT,'logical');
    ImatsousT0(:,N.Tsources)=[];

    %%%%%% computing of places bound %%%%%%%%%%%%
    SB=zeros(1, N.sizeP);
    for i=1:N.sizeP
        pb.f=zeros(1,N.sizeP+N.sizeT);
        pb.f(i)=-1;
        pb.Aeq=[eye(N.sizeP),-N.C];
        pb.Beq=[N.M0];
        pb.Aineq=-eye(N.sizeP+N.sizeT);
        pb.Bineq=zeros(N.sizeP+N.sizeT,1);
        pb.options = optimoptions('linprog','Display','off') ;
        pb.solver='linprog';
        [~,sol]=linprog(pb);
        SB(i)=-sol;
    end
    % Dimention of matrix
    size31=0;

    for p=ImatsousP0
        size31=size31+sum( N.Ppred(p),'all');
    end
    nbeq= size31;
```

```
nbeq= nbeq +1; % 3.2
nbeq= nbeq +1; % 3.3
nbeq= nbeq + N.sizeT-sum(N.Tsources,'all'); % 3.4
nbeq= nbeq + N.sizeT-sum(N.Tsources,'all'); % 3.5
size36=0;
size37=0;
for r=ImatR
    size36=size36+sum(N.Psucc(r).*(~N.Tsources),'all');
    size37=size37+sum(N.Psucc(r).*(~N.Tsources),'all');
end
nbeq = nbeq+size36; %3.6
nbeq = nbeq+size37;%3.7
nbeq = nbeq+N.nbR; %3.8
nbeq = nbeq+N.nbR; %3.9
nbeq = nbeq + N.sizeT-sum(N.Tsources,'all'); %3.10

Problem.Aineq=zeros(nbeq,size_var);
Problem.Bineq=zeros(nbeq,1);

Problem.Aeq=zeros(N.sizeP,size_var);
Problem.Beq=zeros(N.sizeP,1);

%3.1
cptEq=1;
for p=ImatsousP0
    for t=find(N.Ppred(p))
        Problem.Aineq(cptEq,p2v(N.Tpred(t)&~N.P0))=1;
        Problem.Aineq(cptEq,p2v(p))=-1;
        Problem.Bineq(cptEq)=sum(N.Tpred(t)&~N.P0,'all')-1;
        cptEq=cptEq+1;
    end
end
%3.2
Problem.Aineq(cptEq, v)=1;
Problem.Bineq(cptEq)=sum(~N.P0,'all')-1;
cptEq=cptEq+1;
%3.3
Problem.Aineq(cptEq, M(N.PP))=-1;
Problem.Bineq(cptEq)=-1;
cptEq=cptEq+1;


%3.4
for t=ImatsousT0
    Problem.Aineq(cptEq, M(N.TPpred(t)))=-1;
    Problem.Aineq(cptEq, t2tau(t))=1;
    Problem.Bineq(cptEq)=0;
    cptEq=cptEq+1;
end
%3.5
for t=ImatsousT0
    Problem.Aineq(cptEq, M(N.TPpred(t)))=1./SB(N.TPpred(t));
    Problem.Aineq(cptEq, t2tau(t))=-1;
    Problem.Bineq(cptEq)=0;
    cptEq=cptEq+1;
end
%3.6
for r=ImatR
    for t=find((N.Psucc(r)-N.Tsources)>0)
        Problem.Aineq(cptEq,f(rt2f(r,t)))=-1;
        Problem.Aineq(cptEq,M(r))=1/(N.M0(r)-abs(N.C(r,t))+1);
        Problem.Bineq(cptEq)=(abs(N.C(r,t))-1)/(N.M0(r)-abs(N.C(r,t))+1);
        cptEq=cptEq+1;
    end
end
%3.7
for r=ImatR
    for t=find((N.Psucc(r)-N.Tsources)>0)
        Problem.Aineq(cptEq,f(rt2f(r,t)))=-1;
        Problem.Aineq(cptEq,t2tau(t))=1;
        Problem.Aineq(cptEq,p2v(r))=1;
        Problem.Bineq(cptEq)=1;
```

```
            cptEq=cptEq+1;
        end
    end
    %3.8
    for r=ImatR
        Problem.Aineq(cptEq,f(rt2f(r,(N.Psucc(r)-N.Tsources)>0)))=1;
        Problem.Aineq(cptEq,v(p2v(r)))=-1;
        Problem.Bineq(cptEq)=sum((N.Psucc(r)-N.Tsources)>0,'all')-1;
        cptEq=cptEq+1;
    end

    %3.9
    for r=ImatR
        Problem.Aineq(cptEq,t2tau((N.Psucc(r)-N.Tsources)>0))=-1;
        Problem.Aineq(cptEq,p2v(r))=-1;
        Problem.Bineq(cptEq)=-1;
        cptEq=cptEq+1;
    end

     %3.10
    for t=ImatsousT0
        Problem.Aineq(cptEq, f(rt2f(N.TRpred(t),t)))=1;
        Problem.Aineq(cptEq, t2tau(t))=1;
        Problem.Bineq(cptEq)=sum(N.TRpred(t),'all');
        cptEq=cptEq+1;
    end
     %3.11

    Problem.Aeq(:,[M,Y])=[eye(N.sizeP),-N.C];
    Problem.Beq=[N.M0];
    Problem.lb=zeros(size_var,1);
    Problem.ub=ones(size_var,1);
    Problem.ub([M,Y],1)=Inf;

    Problem.f=zeros(size_var,1);
    %Problem.f(p2v(N.PP))=-N.sizeP;
    Problem.f(p2v(N.PP))=-1;
    %Problem.f(p2v(N.PR))=-N.sizeP;
    Problem.f(p2v(N.PR))=-N.sizeP;
    %Problem.f(v)=-1;
    Problem.intcon=[v,tau,f];
    Problem.solver='intlinprog';
    Problem.options='';
end
```

B.2.3 MIP^2 [ZDW+19]

```
function [Problem] = MIPbadM(N,S)
%   MIPSiphon  Generate a MIP from a S4PR to find the highest bound of a
%   a siphon S following the method from 10.1109/ACCESS.2019.2939855 a

    tau=1:N.sizeT-sum(N.Tsources);
    t2tau=zeros(N.sizeT,1);
    t2tau(~N.Tsources)=tau;

    sizeFRTin=(N.C(N.PR,:)>0);
    sizeFRTin(:,N.Tsources)=0;
    sizeFRTin=sum(sizeFRTin,'all');
    f=tau(end)+1: tau(end)+sizeFRTin;
    rt2f=zeros(size(N.C));
    rt2f_log=(N.C<0);
    rt2f_log(1:end-N.nbR,:)=0;
    rt2f_log(:,N.Tsources)=0;
    rt2f(rt2f_log)=cumsum(rt2f_log(rt2f_log));

    M=f(end)+1: f(end)+N.sizeP;
    Y=M(end)+1: M(end)+N.sizeT;

    sizeVar=Y(end);
```

```
% helpful declarations for forloop


ImatR=eye(N.sizeP,'logical');
ImatR(:,~N.PR)=[];

ImatsousT0=eye(N.sizeT,'logical');
ImatsousT0(:,N.Tsources)=[];

cptIneq=1;
cptEq=1;

%%%%%% sizing of Aeq, Beq, AIneq and BIneq

nbEq=0;
nbIneq=0;
%3.13
nbIneq=nbIneq+1;
%3.14
nbIneq=nbIneq+sum(N.PP);
%3.15
nbIneq=nbIneq+sum(N.PP);
%3.16
for r=ImatR(:,S(N.PR))
     nbIneq= nbIneq+sum(N.Psucc(r)&(~N.Tsources),'all');
end
%3.17
for r=ImatR(:,S(N.PR))
     nbIneq= nbIneq+sum(N.Psucc(r)&(~N.Tsources),'all');
end
%3.18
nbIneq= nbIneq+sum(S&N.PR);
%3.19
for r=ImatR(:,~S(N.PR))
     nbEq= nbEq+sum(N.Psucc(r)&(~N.Tsources),'all');
end
%3.20
nbIneq= nbIneq+sum(~N.Tsources);
%3.21
nbEq= nbEq+N.sizeP;
%3.22
nbEq=nbEq+1;


Problem.Aineq=zeros(nbIneq,sizeVar);
Problem.Bineq=zeros(nbIneq,1);

Problem.Aeq=zeros(nbEq,sizeVar);
Problem.Beq=zeros(nbEq,1);



%%%%%% computing of places bound %%%%%%%%%%%%
SB=zeros(1, N.sizeP);
for i=1:N.sizeP
    pb.f=zeros(1,N.sizeP+N.sizeT);
    pb.f(i)=-1;
    pb.Aeq=[eye(N.sizeP),-N.C];
    pb.Beq=[N.M0];
    pb.Aineq=-eye(N.sizeP+N.sizeT);
    pb.Bineq=zeros(N.sizeP+N.sizeT,1);
    pb.options = optimoptions('linprog','Display','off') ;
    pb.solver='linprog';
    [~,sol]=linprog(pb);
    SB(i)=-sol;
end
%%%%%% Filling Problem matrix %%%%%%%%%%%%

%3.13 Inequation
Problem.Aineq(cptIneq, M(N.PP))=-1;
Problem.Bineq(cptIneq)=-1;
```

```
cptIneq=cptIneq+1;

%3.14 Inequation
for t=ImatsousT0
    Problem.Aineq(cptIneq, M(N.TPpred(t)))=-1;
    Problem.Aineq(cptIneq, t2tau(t))=1;
    Problem.Bineq(cptIneq)=0;
    cptIneq=cptIneq+1;
end

%3.15 Inequation
for t=ImatsousT0
    Problem.Aineq(cptIneq, M(N.TPpred(t)))=1./SB(N.TPpred(t));
    Problem.Aineq(cptIneq, t2tau(t))=-1;
    Problem.Bineq(cptIneq)=0;
    cptIneq=cptIneq+1;
end

%3.16 Inequation
for r=ImatR(:,S(N.PR))
    for t=find((N.Psucc(r)-N.Tsources)>0)
        Problem.Aineq(cptIneq,f(rt2f(r,t)))=1;
        Problem.Aineq(cptIneq,M(r))=-1/abs(N.C(r,t));
        Problem.Bineq(cptIneq)=0;
        cptIneq=cptIneq+1;
    end
end

%3.17 Inequation
for r=ImatR(:,S(N.PR))
    for t=find((N.Psucc(r)-N.Tsources)>0)
        Problem.Aineq(cptIneq,f(rt2f(r,t)))=-1;
        Problem.Aineq(cptIneq,M(r))=1/(N.M0(r)-abs(N.C(r,t))+1);
        Problem.Bineq(cptIneq)=(abs(N.C(r,t))-1)/(N.M0(r)-abs(N.C(r,t))+1);
        cptIneq=cptIneq+1;
    end
end

%3.18 Inequation
for r=ImatR(:,S(N.PR))
    Problem.Aineq(cptIneq,t2tau((N.Psucc(r)-N.Tsources)>0))=-1;
    Problem.Bineq(cptIneq)=-1;
    cptIneq=cptIneq+1;
end

%3.19 Equation
for r=ImatR(:,~S(N.PR))
    for t=find((N.Psucc(r)-N.Tsources)>0)
        Problem.Aeq(cptEq,f(rt2f(r,t)))=1;
        Problem.Beq(cptEq)=1;
        cptEq=cptEq+1;
    end
end

%3.20 Inequation
for t=ImatsousT0
    if any(N.TRpred(t))
        Problem.Aineq(cptIneq, f(rt2f(N.TRpred(t),t)))=1;
        Problem.Aineq(cptIneq, t2tau(t))=1;
        Problem.Bineq(cptIneq)=sum(N.TRpred(t),'all');
        cptIneq=cptIneq+1;
    end
end

%3.21 Equation
Problem.Aeq(cptEq:cptEq+N.sizeP-1,[M,Y])=[eye(N.sizeP),-N.C];
Problem.Beq(cptEq:cptEq+N.sizeP-1)=[N.M0];
cptEq=cptEq+N.sizeP;

%3.22 Equation
Scomp=(N.H(S&N.PR)-S)>0; %complementary set of the siphon
Problem.Aeq(cptEq,M((N.PP-Scomp)>0))=1;
```

```
    Problem.Beq(cptEq)=0;
    cptEq=cptEq+1;

    Problem.lb=zeros(sizeVar,1);
    Problem.ub=ones(sizeVar,1);
    Problem.ub([M,Y],1)=Inf;

    Problem.f=zeros(sizeVar,1);
    Problem.f(M(N.PR&S))=-1;

    Problem.intcon=[tau,f];
    Problem.solver='intlinprog';
    Problem.options='';
end
```

B.3 Control synthesis function

B.3.1 $[S]$

```
function SComp = SComp(N,S)
%SComp Compute the complementary set of siphon S in a S4PR N
SComp=(N.H(S&N.PR)-S)>0;
end
```

B.3.2 Add monitor place

```
function [NC] = addMonitorPlace(N,w,b)
%addMonitorPlace add a ressource place to a S4PR
%   Detailed explanation goes here
    if(any(w(~N.PP)))
        error('Wrong constraint vector. Only process places can be constrained');
    end
    if(~isAdmissible(N, w))
        error('The constraint is not admissible. Monitor generation failed');
    end
    controledP=w~=0;
    mapP2Proc= N.mapP2Proc;
    vectRCell={mapP2Proc(controledP,1)', mapP2Proc(controledP,2)', w(controledP)' ,b
        };
    NC=S4PR(N.vectProc, [N.vectRCell;vectRCell]);
    NC.Tuc=N.Tuc;
    NC.Tuo=N.Tuo;
end
```

B.3.3 K

```
function K=K(N,S)
%SComp Compute the ressource cost of a siphon S for a S4PR N
K=(N.Y(S&N.PR));
K(S)=0;
end
```

B.3.4 $p^{\uparrow}$

```
function [Pup] = Pup(N,p)
%PUP Return the set of places in the upstream path of a process in a S4PR
%   p is a places of a S4PR (numerotation of a S4PR)
%   N a S4PR
%   Pup a set of places in N (in logical array)
```

```
        adjP=zeros(N.sizeP); % adjP is a adgency matrix for process places (
            transitions as vertices)
        %logical pre matrix only on processes
        preP=N.pre>0;
        preP(~N.PP,:)=0;
        %logical post matrix only on processes
        postP=N.post>0;
                postP(~N.PP,:)=0;
        for t=1:N.sizeT
          adjP(logical(preP(:,t)),:)=adjP(logical(preP(:,t)),:)+postP(:,t)';
        end
        Pup=zeros(N.sizeP,1, 'logical');
        Pup(p)=1;
        while (~all(Pup== ((Pup+adjP*Pup)>0))) %add predecessor places util fixed
            point
            Pup= (Pup+adjP*Pup)>0;
        end
        Pup(1)=0; %p^0 is deleted
end
```

B.3.5 Check admissibility

```
function isAdmissible=isAdmissible(N, w)
    % test if a constraint w is admissible with respect to the
    % uncontrolable and unobservable transitions.
    isAdmissible= true;
    VC=-w'*N.C;
    if(any(VC(N.Tuc)<0) ||any(VC(N.Tuo)~=0))
        isAdmissible=false;
    end
end
```

B.3.6 $\mathcal{F}^C$

```
function w=makeWContr(N, w)
    % Return a constraint vector admissible w.r.t controlability
    % works only if N is a S4PR
    Tinadm=true;
    while(any(Tinadm))
        VC=-w'*N.C;
        Tinadm=N.Tuc & (VC<0)';
        Tinadm4loop=eye(N.sizeT, 'logical'); Tinadm4loop(:,~Tinadm)=[];
        for t= Tinadm4loop
            w(N.TPpred(t))=w(N.TPsucc(t));
        end
    end
end
```

B.3.7 $\mathcal{F}^O$

```
function w=makeWObs(N, w)
    % Return a constraint vector admissible w.r.t observability
    % works only if N is a S4PR
    Tinadm=true;
    while(any(Tinadm))
        VC=-w'*N.C;
        Tinadm=N.Tuo & (VC~=0)';
        Tinadm4loop=eye(N.sizeT, 'logical'); Tinadm4loop(:,~Tinadm)=[];
        for t= Tinadm4loop
            w(N.TPsucc(t))=max(w(N.TPpred(t)),w(N.TPsucc(t)) );
            w(N.TPpred(t))=max(w(N.TPpred(t)),w(N.TPsucc(t)) );
        end
    end
end
```

B.3.8 $\mathcal{F}^{OC}$

```
function w=makeWAdm(N, w)
% transform a constraint w into a admissible contraint w.r.t observability and
    observablility
% works only if N is a S4PR
    while(~isAdmissible(N, w))
        w=makeWObs(N, w);
        w=makeWContr(N, w);
    end
end
```

B.3.9 $\mathcal{F}^{\uparrow}$

```
%MAKEWUP  transform a constraint w into a constaint that have heavier
%weight in the places upstream in the process
% this operation garantie that a monitor place generated from this
% constraint will have arcs on sources transitions

% works only if N is a S4PR
for p=find(w)'
   w(Pup(N,p))=max(w(Pup(N,p)), w(p));
end
end
```

B.3.10 $\mathcal{F}^{\uparrow}O$

```
function w = makeWupObs(N, w)
% transform a constraint w into a admissible contraint w.r.t observability
% and observablility and that have heavier weight in the places upstream
% in the process
% works only if N is a S4PR
    while(~isAdmissible(N, w))
        w=makeWObs(N, w);
        w=makeWup(N, w);
    end
end
```

B.3.11 H2'

```
function [N,Sset] = H2p(N)
% Compute the set of monitor place necessary to obtain a deadlock free
% in a S3PMR under partial controlability and observability
% This method is inspired from https://doi.org/10.1080/00207540412331330822

    mip_N=MIPSiphonOrdinary(N);
    S=siphon_computation(mip_N,N);
    Sset=zeros(N.sizeP,0,'logical');
    constraints_vector=zeros(N.sizeP,0);
    constraint_weight=zeros(0,1);


    while(any(S))
        Sset=[Sset, S];
        if(max(K(N,S))>1)
            constraints_vector(:,end+1)=makeWupObs(N,K(N,S));
            constraint_weight(end+1)=sum(N.M0(S & N.PR))-1;
        else
            N=addMonitorPlace(N,makeWAdm(N,(K(N,S))),sum(N.M0(S & N.PR))-1);
            constraints_vector(end+1,:)=0;
            mip_N=MIPSiphonOrdinary(N);
            Sset(end+1,:)=0;
        end
```

```
        mip=add_constraint_to_mip(mip_N,constraints_vector',constraint_weight,N);
        S=siphon_computation(mip,N);
    end

    for i=1:length(constraint_weight)
        N=addMonitorPlace(N, constraints_vector(:,i), constraint_weight(i));
        constraints_vector(end+1,:)=0;
    end
end



function[mip]= add_constraint_to_mip(mip,Constraint,weight,N)

    nb_constraints=length(weight);
    if(nb_constraints>0)
        mip.Aineq(end+1:end+nb_constraints,:)=[zeros(nb_constraints,N.sizeP+N.sizeT)
            ,Constraint, zeros(nb_constraints,N.sizeT)];
        mip.bineq(end+1:end+nb_constraints)=weight;
    end
end

function[S]=siphon_computation(mip,N)
%compute one siphon from a well constructed mip
% see fonction generate_mip_from_pn

    [nplace,ntrans]=size(N.pre);
    [x, fval,  truc] = intlinprog(mip);
    siphon=find(~logical(x(1:nplace)));
    if(~isempty(siphon))
        S_min= find_min_siph(siphon,N);
            S_min_vector=zeros(nplace,1);
            S_min_vector(S_min)=1;
            siphon=find(S_min_vector);
    end
    S=zeros(N.sizeP,1, 'logical');
    S(siphon)=1;
end

function[S] = find_min_siph(S,N)
% from a siphon S and and a S4PR ordinary (S3PMR) N a minimal siphon contenained in
    S
% returned
    [nplace,ntrans]=size(N.post);
    V=[];
    Sp=S;

    while(~isempty(setdiff(S,V)))
        p=setdiff(S,V);
        p=p(1);
        V(end+1)=p;
        Sp=setdiff(S,p);
        while(any(any(N.post(Sp,:),1) & ~(any(N.pre(Sp,:),1))))
            tsource=any(N.post(Sp,:),1) & ~(any(N.pre(Sp,:),1));
            Sp=setdiff(Sp,find(N.post*tsource'>0));
        end
        if(~isempty(Sp))
            S=Sp;
        end
    end
end
```

B.3.12 Z2'

```
function [N,Sset]=Z2p(N)
    %Z2p Applies the control method from 10.1109/ACCESS.2019.2939855
    %adapted to partial observability and controlability

    Sset=false(N.sizeP,0);
```

```
    while(true)
        Problem= MIPSiphonS4PR(N);
        x=round(intlinprog(Problem));
        if isempty(x)
            return
        end
        v=1:(N.sizeP-sum(N.P0));
        S=false(N.sizeP,1);
        S(~N.P0)=~x(v);

        p2v=zeros(N.sizeP,1);
        p2v(~N.P0)=v;

        tau=v(end)+1:v(end)+N.sizeT-sum(N.Tsources);
        t2tau=zeros(N.sizeT,1);
        t2tau(~N.Tsources)=tau;

        sizeFRTin=(N.C(N.PR,:)>0);
        sizeFRTin(:,N.Tsources)=0;
        sizeFRTin=sum(sizeFRTin(:));
        f=tau(end)+1: tau(end)+sizeFRTin;
        rt2f=zeros(size(N.C));
        rt2f_log=(N.C<0);
        rt2f_log(1:end-N.nbR,:)=0;
        rt2f_log(:,N.Tsources)=0;
        rt2f(rt2f_log)=cumsum(rt2f_log(rt2f_log));

        M=f(end)+1: f(end)+N.sizeP;
        Y=M(end)+1: M(end)+N.sizeT;
        %M=x(end-N.sizeT-N.sizeP+1:end-N.sizeT);
        %Y=x(end-N.sizeT+1:end);
        [Problem] = MIPbadM(N,S);
        x=intlinprog(Problem);
        if isempty(x)
            error("internal error, max marking have not converged to a solution ")
        end
        x=round(x);
        tau=1:N.sizeT-sum(N.Tsources);
        t2tau=zeros(N.sizeT,1);
        t2tau(~N.Tsources)=tau;

        sizeFRTin=(N.C(N.PR,:)>0);
        sizeFRTin(:,N.Tsources)=0;
        sizeFRTin=sum(sizeFRTin,'all');
        f=tau(end)+1: tau(end)+sizeFRTin;
        rt2f=zeros(size(N.C));
        rt2f_log=(N.C<0);
        rt2f_log(1:end-N.nbR,:)=0;
        rt2f_log(:,N.Tsources)=0;
        rt2f(rt2f_log)=cumsum(rt2f_log(rt2f_log));

        M=f(end)+1: f(end)+N.sizeP;
        Y=M(end)+1: M(end)+N.sizeT;

        Mindex=length(x)-N.sizeT-N.sizeP+1:length(x)-N.sizeT;
        Mmax=round(x(Mindex));
        N=PV(N,S,Mmax);
        Sset=[Sset,S];
        Sset(end+1,:)=0;
    end
end
```

B.4 Scripts

B.4.1 Example chapter 6

```
    %1     2     3     4     5     6     7     8     9     10    11    12    13
CG1=[-1     0     0     0     0     0     0     0     0     0     1     0     1;%P0
      1    -1     0     0     0     0     0     0     0     0     0     0     0;%S1
      0     1    -1     0     0    -1     0     0     0     0     0     0     0;%SW1
      0     0     1    -1     0     0     0     0     0     0     0     0     0;%S2
      0     0     0     1    -1     0     0     0     0     0     0     0     0;%S3
      0     0     0     0     0     1    -1     0     0     0     0     0     0;%S4
      0     0     0     0     0     0     1    -1     0     0     0     0     0;%S5
      0     0     0     0     1     0     0     1    -1     0     0     0     0;%SW2
      0     0     0     0     0     0     0     0     1    -1     0    -1     0;%SW3
      0     0     0     0     0     0     0     0     0     1    -1     0     0;%S6
      0     0     0     0     0     0     0     0     0     0     0     1    -1];%S7

procG1=Process(CG1,10);

%    11    10     9     5     4     3     8     7     6     2     1
CG2=[-1     0     0     0     0     0     0     0     0     0     1;%P0
      1    -1     0     0     0     0     0     0     0     0     0;%S6
      0     1    -1     0     0     0     0     0     0     0     0;%SW3
      0     0     1    -1     0     0    -1     0     0     0     0;%SW2
      0     0     0     1    -1     0     0     0     0     0     0;%S3
      0     0     0     0     1    -1     0     0     0     0     0;%S2
      0     0     0     0     0     0     1    -1     0     0     0;%S5
      0     0     0     0     0     0     0     1    -1     0     0;%S4
      0     0     0     0     0     1     0     0     1    -1     0;%SW1
      0     0     0     0     0     0     0     0     0     1    -1];%S1

procG2=Process(CG2,11);

vectProc=[procG1 procG2];

RS1= {[1,2,2]      , [2,9,10]      ,  [1,1,1]       ,1    };  %S1
RS2= {[1,1,2]      , [3,4,6 ]      ,  [1,1,1]       ,1    };  %S2
RS3= {[1,2,2,2,2]  , [5,2,3,4,5]   ,  [1,1,1,1,1]   ,1    };  %S3
RS4= {[1,1,1,2]    , [3,4,6,8]     ,  [1,1,1,1]     ,1    };  %S4
RS5= {[1,2,2,2,2]  , [7,2,3,4,7]   ,  [1,1,1,1,1]   ,1    };  %S5
RS7= {[1,1,1,1]    , [8,9,10,11]   ,  [1,1,1,1]     ,1    };  %S7
RSW1={[1,1,2,2]    , [3,4,9,10]    ,  [1,1,1,1]     ,1    };  %SW1
RSW3={[1,1,1,2,2,2], [8,9,10,2,3,4],  [1,1,1,1,1,1] ,1    };  %SW3

vectRCell=[RS1 ;RS2 ;RS3 ; RS4 ;RS5 ; RS7; RSW1;RSW3];

N=S4PR(vectProc, vectRCell);

        %Gamma1  1 2 3 4 5 6 7 8 9 10 11 12 13| Gamma2 11 10 9 5 4 3 8 7 6 2 1
N.Tuc=logical([ [0 0 1 0 0 1 0 0 1 1  1  1  1],       [0  1  1 1 0 0 1 0 0 1 1]]');
N.Tuo=logical([ [0 0 1 0 0 1 0 0 0 1  0  0  0],       [0  1  1 0 0 0 0 0 0 1 0]]');

tic
[NCZ, SZset]=Z2p(N);
toc
tic
[NCH, SHset]=H2p(N);
toc
```

B.4.2 Example chapter 5

```
CA=[-1, 0, 0,1 ;
    1 ,-1, 0,0 ;
    0 ,1 ,-1,0 ;
    0 , 0, 1,-1;];

procA=Process(CA,3);

CB=[-1, 0, 0,0 , 0, 1 ,0 ,0 ,0 ,0 ;
    1 ,-1, 0,0 , 0, 0 ,-1,0 ,0 ,0 ;
    0 ,1 ,-1,0 , 0, 0 ,0 ,0 ,0 ,0 ;
    0 ,0 ,1 ,-1, 0, 0 ,0 ,0 ,0 ,0 ;
    0 ,0 , 0, 1,-1, 0 ,0 ,0 ,0 ,0 ;
```

```
    0 ,0 , 0, 0, 1,-1 ,0 ,0 ,1 ;
    0 ,0 , 0,0 , 0, 0 ,1 ,-1,0 ,0 ;
    0 ,0 ,0 ,0 , 0, 0 ,0 ,1 ,-1,0 ;
    0 ,0 ,0 ,0 , 0, 0 ,0 ,0 , 1,-1;];

procB=Process(CB,11);

CC=[-1, 0, 0,0 ,0 , 1;
    1 ,-1, 0,0 ,0 ,0 ;
    0 ,1 ,-1,0 ,0 ,0 ;
    0 ,0 ,1 ,-1,0 ,0 ;
    0 , 0, 0, 1,-1,0 ;
    0 , 0, 0,0 , 1,-1;];

procC=Process(CC,7);

vectProc=[procA procB procC];

R1={[2,3]      ,[2, 6],        [1,1],      1};
R2={[1,1,2,2,3],[2,4,4,8,4],   [1,1,1,1,1],1};
R3={[2,3],      [6,2],         [1,1],      1};
R4={[2],        [3],           [1],        2};
R5={[1,2],      [3, 5],        [1, 1],     2};
R6={[2,3],      [7, 5],        [1, 1],     2};
R7={[2,3],      [9, 3],        [1, 1],     2};

vectRCell=[R1 ;R2 ;R3 ; R4 ;R5 ; R6; R7];

N=S4PR(vectProc, vectRCell);

N.Tuc=logical([ [0 1 1 0], [0 1 0 1 0 0 0 1 0 1], [0 1 1 0 0 0]]');
N.Tuo=logical([ [0 1 0 0], [0 0 0 1 0 0 0 1 0 0], [0 1 1 0 0 0]]');

tic
[NCZ, SsetZ]=Z2p(N);
toc
tic
[NCH, SsetH]=H2p(N);
toc
```

Bibliographie

[ARX04] Zied Achour, Nidhal Rezg, and Xiaolan Xie. Supervisory controller of petri nets under partial observation. *IFAC Proceedings Volumes*, 37(18) :51–56, 2004.

[BA95] Kamel Barkaoui and I Ben Abdallah. A deadlock prevention method for a class of fms. In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, volume 5, pages 4119–4124. IEEE, 1995.

[BCG06] Francesco Basile, Pasquale Chiacchio, and Alessandro Giua. Suboptimal supervisory control of petri nets in presence of uncontrollable transitions via monitor places. *Automatica*, 42(6) :995–1004, 2006.

[BCK05] Kamel Barkaoui, Jean-Michel Couvreur, and Kais Klai. On the equivalence between liveness and deadlock-freeness in petri nets. In *International Conference on Application and Theory of Petri Nets*, pages 90–107. Springer, 2005.

[BCP13] Francesco Basile, Roberto Cordone, and Luigi Piroddi. Integrated design of optimal supervisors for the enforcement of static and behavioral specifications in petri net models. *Automatica*, 49(11) :3432–3439, 2013.

[BCZ97] Kamel Barkaoui, Alloua Chaoui, and Belhassen Zouari. Supervisory control of discrete event systems based on structure theory of petri nets. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 4, pages 3750–3755. IEEE, 1997.

[BPP96] Kamel Barkaoui and Jean-François Pradat-Peyre. On liveness and controlled siphons in petri nets. In *International Conference on Application and Theory of Petri Nets*, pages 57–72. Springer, 1996.

[CCY13] Daniel Y Chao, Jiun-Ting Chen, and Fang Yu. A novel liveness condition for s3pgr2. *Transactions of the Institute of Measurement and Control*, 35(2) :131–137, 2013.

[CDFV88] Randy Cieslak, C Desclaux, Ayman S Fawaz, and Pravin Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE transactions on automatic control*, 33(3) :249–260, 1988.

[CES71] Edward G Coffman, Melanie Elphick, and Arie Shoshani. System deadlocks. *ACM Computing Surveys (CSUR)*, 3(2) :67–78, 1971.

[CFP05] R. Cordone, L. Ferrarini, and 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*, 35(6) :844–854, 2005.

[Cha07] Daniel Yuh Chao. Max'-controlled siphons for liveness of s3pgr2. *IET Control Theory & Applications*, 1(4) :933–936, 2007.

[Cha09] Daniel Y Chao. Direct minimal empty siphon computation using mip. *The International Journal of Advanced Manufacturing Technology*, 45(3-4) :397, 2009.

[CKBT19] Paul Cazenave, Manel Khlif-Bouassida, and Armand Toguyeni. Synthèse de contrôleurs réseaux de petri pour le routage dynamique des trains dans un noeud ferroviaire. In *MSR 2019-12ème Colloque sur la Modélisation des Systèmes Réactifs, Nov 2019, Angers, France*, 2019.

[CKT19a] P. Cazenave, M. Khlif-Bouassida, and A. Toguyéni. Collisions avoidance and deadlocks prevention, for dynamic routing of trains in a railway node. In *6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1923–1928, April 2019.

[CKT19b] P. Cazenave, M. Khlif-Bouassida, and A. Toguyéni. Deadlock and collision avoidance in railway networks with dynamic routing : A petri net approach with partial controllability and observability. In *15th European Workshop on Advanced Control and Diagnosis, ACD 2019*, november 2019.

[CKT20] P. Cazenave, M. Khlif-Bouassida, and A. Toguyéni. S3pmr deadlock control with partial controllability and observability. In *15th IFAC Workshop on Discrete Event Systems, WODES 2020*, november 2020.

[CL09] Christos G Cassandras and Stephane Lafortune. *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[CL11] YuFeng Chen and ZhiWu Li. Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems. *Automatica*, 47(5) :1028–1034, 2011.

[CL13] Yufeng Chen and Gaiyun Liu. Computation of minimal siphons in petri nets by using binary decision diagrams. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(1) :1–15, 2013.

[CLKM10] YuFeng Chen, ZhiWu Li, Mohamed Khalgui, and Olfa Mosbahi. Design of a maximally permissive liveness-enforcing petri net supervisor for flexible manufacturing systems. *IEEE Transactions on automation science and engineering*, 8(2) :374–393, 2010.

[CLZ12] YuFeng Chen, ZhiWu Li, and MengChu Zhou. Most permissive liveness-enforcing petri net supervisors for flexible manufacturing systems. *International journal of production research*, 50(22) :6357–6371, 2012.

[CM89] Hangju Cho and Steven I Marcus. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Mathematics of Control, Signals and Systems*, 2(1) :47–69, 1989.

[CP00] Søren Christensen and Laure Petrucci. Modular analysis of petri nets. *The computer journal*, 43(3) :224–242, 2000.

[CX97] Feng Chu and Xiao-Lan Xie. Deadlock analysis of petri nets using siphons and mathematical programming. *IEEE Transactions on Robotics and Automation*, 13(6) :793–804, 1997.

[DA10] René David and Hassane Alla. *Discrete, continuous, and hybrid Petri nets*, volume 1. Springer, 2010.

[dCC07] Antonio Eduardo Carrilho da Cunha and JosÉ Eduardo Ribeiro Cury. Hierarchical supervisory control based on discrete event systems with flexible marking. *IEEE Transactions on Automatic Control*, 52(12) :2242–2253, 2007.

[DG14] Isabel Demongodin and Alessandro Giua. Dynamics and steady state analysis of controlled generalized batches petri nets. *Nonlinear Analysis : Hybrid Systems*, 12 :33–49, 2014.

[Dij71] Edsger W Dijkstra. Hierarchical ordering of sequential processes. In *The origin of concurrent programming*, pages 198–227. Springer, 1971.

[DQC00] Max H De Queiroz and José ER Cury. Modular supervisory control of large scale discrete event systems. In *Discrete Event Systems*, pages 103–110. Springer, 2000.

[DS09] Mustafa Seçkin Durmuş and Mehmet Turan Söylemez. Automation petri net based railway interlocking and signalization design. In *Int. Symposium on Innovations in Intelligent Systems and Applications*, pages 12–16, 2009.

[ECM95] Joaquin Ezpeleta, Jose Manuel Colom, and Javier Martinez. A petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE transactions on robotics and automation*, 11(2) :173–184, 1995.

[EGVC98] Joaquin Ezpeleta, Fernando García-Vallés, and José Manuel Colom. A class of well structured petri nets for flexible manufacturing systems. In *International Conference on Application and Theory of Petri Nets*, pages 64–83. Springer, 1998.

[ER04] Joaquin Ezpeleta and Laura Recalde. A deadlock avoidance approach for nonsequential resource allocation systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 34(1) :93–101, 2004.

[ETGVC02] Joaquin Ezpeleta, Fernando Tricas, Fernando Garcia-Valles, and José Manuel Colom. A banker's solution for deadlock avoidance in fms with flexible routing and multiresource states. *IEEE Transactions on Robotics and Automation*, 18(4) :621–625, 2002.

[FGS03] Maria Pia Fanti, Alessandro Giua, and Carla Seatzu. Generalized mutual exclusion constraints and monitors for colored petri nets. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, volume 2, pages 1860–1865. IEEE, 2003.

[FGS06] Maria Pia Fanti, Alessandro Giua, and Carla Seatzu. Monitor design for colored petri nets : An application to deadlock prevention in railway networks. *Control engineering practice*, 14(10) :1231–1247, 2006.

[FPN09] Gregory Faraut, Laurent Piétrac, and Eric Niel. Formal approach to multimodal control design : Application to mode switching. *IEEE Transactions on Industrial Informatics*, 5(4) :443–453, 2009.

[GDS92] Alessandro Giua, Frank DiCesare, and Manuel Silva. Generalized mutual exclusion contraints on nets with uncontrollable transitions. In *Systems, Man and Cybernetics, 1992., IEEE International Conference on*, pages 974–979. IEEE, 1992.

[GRX03] Asma Ghaffari, Nidhal Rezg, and Xiaolan Xie. Design of a live and maximally permissive petri net controller using the theory of regions. *IEEE transactions on robotics and Automation*, 19(1) :137–141, 2003.

[GS08] Alessandro Giua and Carla Seatzu. Modeling and supervisory control of railway networks using petri nets. *IEEE Transactions on automation science and engineering*, 5(3) :431–445, 2008.

[HB17] YiFan Hou and Kamel Barkaoui. Deadlock analysis and control based on petri nets : A siphon approach review. *Advances in Mechanical Engineering*, 9(5) :1687814017693542, 2017.

[HHJ+15] Liang Hong, YiFan Hou, JunFeng Jing, AnRong Wang, and Dmitry A Litvin. Deadlock prevention policy with behavioral optimality or suboptimality achieved by the redundancy identification of constraints and the rearrangement of monitors. *Discrete Dynamics in Nature and Society*, 2015, 2015.

[HJXC01] YiSheng Huang, MuDer Jeng, Xiaolan Xie, and ShengLuen Chung. A deadlock prevention policy for flexible manufacturing systems using siphons. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 541–546. IEEE, 2001.

[HJXC06] Y-S Huang, MuDer Jeng, Xiaolan Xie, and D-H Chung. Siphon-based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 36(6) :1248–1256, 2006.

[IA02] Marian V Iordache and Panos J Antsaklis. Synthesis of supervisors enforcing general linear vector constraints in petri nets. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, volume 1, pages 154–159. IEEE, 2002.

[IA06] Marian V Iordache and Panos J Antsaklis. Supervision based on place invariants : A survey. *Discrete Event Dynamic Systems*, 16(4) :451–492, 2006.

[Jen03] Rune M Jensen. Des controller synthesis and fault tolerant control. Technical report, Citeseer, 2003.

[KHLA98] Xenofon D Koutsoukos, Kevin X He, Michael D Lemmon, and Panos J Antsaklis. Timed petri nets in hybrid systems : Stability and supervisory control. *Discrete Event Dynamic Systems*, 8(2) :137–173, 1998.

[KLB09] Jan Komenda, Sébastien Lahaye, and Jean-Louis Boimond. Supervisory control of (max,+) automata : A behavioral approach. *Discrete Event Dynamic Systems*, 19(4) :525, 2009.

[KLB18] J Komenda, S Lahaye, and JL Boimond. (max,+)-automata with partial observations. *IFAC-PapersOnLine*, 51(7) :192–197, 2018.

[Kün05] Peep Küngas. Petri net reachability checking is polynomial with optimal abstraction hierarchies. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 149–164. Springer, 2005.

[Lau85] Peter E Lauer. A simple railway system. In *The Analysis of Concurrent Systems*, pages 271–292. Springer, 1985.

[Lau87] Kurt Lautenbach. Linear algebraic calculation of deadlocks and traps. In *Concurrency and nets*, pages 315–336. Springer, 1987.

[LAW+14] Shaoyong Li, Aimin An, Hongmei Wu, Caiqin Hou, Ying Cai, Xilian Han, and Ying Wang. Policy to cope with deadlocks and livelocks for flexible manufacturing systems using the max'-controlled new smart siphons. *IET Control Theory & Applications*, 8(16) :1607–1616, 2014.

[LB16] GaiYun Liu and Kamel Barkaoui. A survey of siphons in petri nets. *Information Sciences*, 363 :198–220, 2016.

[LGC06] Juan-Pablo López-Grao and José-Manuel Colom. Lender processes competing for shared resources : Beyond the s4pr paradigm. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3052–3059. IEEE, 2006.

[LGC12] Juan-Pablo López-Grao and José-Manue Colom. A petri net perspective on the resource allocation problem in software engineering. In *Transactions on Petri Nets and Other Models of Concurrency V*, pages 181–200. Springer, 2012.

[LGC13] Juan-Pablo López-Grao and José-Manuel Colom. Structural methods for the control of discrete event dynamic systems–the case of the resource allocation problem. In *Control of Discrete-Event Systems*, pages 257–278. Springer, 2013.

[LGCP12] Juan Pablo López Grao and José Manuel Colom Piazuelo. *Contributions to the deadlock problem in multithreaded software applications observed as Resource Allocation Systems*. PhD thesis, Universidad de Zaragoza, Prensas de la Universidad, 2012.

[LHW07] Zhi Wu Li, He Suan Hu, and An Rong Wang. Design of liveness-enforcing supervisors for flexible manufacturing systems using petri nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(4) :517–526, 2007.

[Lin93] Feng Lin. Robust and adaptive supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, 38(12) :1848–1852, 1993.

[LL10] G Liu and Z Li. General mixed integer programming-based liveness test for system of sequential systems with shared resources nets. *IET control theory & applications*, 4(12) :2867–2878, 2010.

[LL12] Shaoyong Li and Zhiwu Li. Solving siphons with the minimal cardinality in petri nets and its applications to deadlock control. *International Journal of Production Research*, 50(22) :6203–6218, 2012.

[LLER13] Richard M Lusby, Jesper Larsen, Matthias Ehrgott, and David M Ryan. A set packing inspired method for real-time junction train routing. *Computers & Operations Research*, 40(3) :713–724, 2013.

[LLZ10] G Liu, Z Li, and C Zhong. New controllability condition for siphons in a class of generalised petri nets. *IET Control Theory & Applications*, 4(5) :854–864, 2010.

[LLZ13] Gaiyun Liu, Zhiwu Li, and Chunfu Zhong. Correction to'new controllability condition for siphons in a class of generalised petri nets'. *IET Control Theory & Applications*, 7(4) :632–633, 2013.

[LS08] Zhiwu Li and Moshe Shpitalni. A deadlock prevention policy for fms using elementary siphons of petri nets with uncontrollable transitions. In *2008 IEEE International Conference on Networking, Sensing and Control*, pages 67–72. IEEE, 2008.

[LWZ11] ZhiWu Li, NaiQi Wu, and MengChu Zhou. Deadlock control of automated manufacturing systems based on petri nets—a literature review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4) :437–462, 2011.

[LZ04] ZhiWu Li and MengChu 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*, 34(1) :38–51, 2004.

[LZ06] ZhiWu Li and MengChu Zhou. Clarifications on the definitions of elementary siphons in petri nets. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 36(6) :1227–1229, 2006.

[LZ09] ZhiWu Li and MengChu Zhou. *Deadlock resolution in automated manufacturing systems : a novel Petri net approach*. Springer Science & Business Media, 2009.

[MA00] John O Moody and Panos J Antsaklis. Petri net supervisors for des with uncontrollable and unobservable transitions. *IEEE Transactions on Automatic Control*, 45(3) :462–476, 2000.

[MH13] Toshiyuki Miyamoto and Kyota Horiguchi. Modular reachability analysis of petri nets for multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 43(6) :1411–1423, 2013.

[MTLG16] Ziyue Ma, Yin Tong, Zhiwu Li, and Alessandro Giua. Basis marking representation of petri net reachability spaces and its application to the reachability problem. *IEEE Transactions on Automatic Control*, 62(3) :1078–1093, 2016.

[NLWZ14] Fan Ning, Xingxing Li, Shouguang Wang, and Qiaoli Zhuang. A method based on depth-first search to compute all minimal siphons. *JSW*, 9(8) :2148–2153, 2014.

[Pac07] Jörn Pachl. Avoiding deadlocks in synchronous railway simulations. In *2nd international seminar on railway operations modelling and analysis*, pages 1–10, 2007.

[Par00] Jonghun Park. *Structural analysis and control of resource allocation systems using Petri nets*. Georgia Institute of Technology, 2000.

[PCF08] Luigi Piroddi, Roberto Cordone, and Ivano Fumagalli. Selective siphon control for deadlock prevention in petri nets. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 38(6) :1337–1348, 2008.

[PCP99] Enric Pastor, Jordi Cortadella, and Marco A Peña. Structural methods to improve the symbolic analysis of petri nets. In *International Conference on Application and Theory of Petri Nets*, pages 26–45. Springer, 1999.

[Pet62] Carl Adam Petri. Kommunikation mit automaten. *PhD, University of Bonn, West Germany*, 1962.

[PR01] Jonghun Park and Spyros A Reveliotis. Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings. *IEEE Transactions on Automatic Control*, 46(10) :1572–1583, 2001.

[PR02] Jonghun Park and Spyros A Reveliotis. Liveness-enforcing supervision for resource allocation systems with uncontrollable behavior and forbidden states. *IEEE Transactions on Robotics and Automation*, 18(2) :234–240, 2002.

[QLAA15] Meng Qin, ZhiWu Li, and Abdulrahman M Al-Ahmari. Elementary-siphon-based control policy for flexible manufacturing systems with partial observability and controllability of transitions. *Asian Journal of Control*, 17(1) :327–342, 2015.

[RAR17] Sadok Rezig, Zied Achour, and Nidhal Rezg. Theory of regions for control synthesis without computing reachability graph. *Applied Sciences*, 7(3) :270, 2017.

[RW87] Peter J Ramadge and W Murray Wonham. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1) :206–230, 1987.

[RW95] Karen Rudie and Jan C Willems. The computational complexity of decentralized discrete-event control problems. *IEEE Transactions on Automatic Control*, 40(7) :1313–1319, 1995.

[TA84] JM Toudic and H Alaiwan. Recherche des semi-flots des verrous et des trappes dans les réseaux de petri. *Technique et Science Informatique (TSI)*, 4(1), 1984.

[TGVCE98] F Tricas, F Garcia-Valles, JM Colom, and J Ezpeleta. A structural approach to the problem of deadlock prevention in processes with resources. In *Proceedings of the 4th workshop on discrete event systems*, pages 273–278, 1998.

[TGVCE05] Fernando Tricas, Fernando Garcia-Valles, José Manuel Colom, and Joaquin Ezpeleta. A petri net structure-based deadlock prevention solution for sequential resource allocation systems. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 271–277. IEEE, 2005.

[TM95] Fernando Tricas and Javier Martinez. An extension of the liveness theory for concurrent sequential processes competing for shared resources. In *1995 IEEE International Conference on Systems, Man and*

Cybernetics. Intelligent Systems for the 21st Century, volume 4, pages 3035–3040. IEEE, 1995.

[Tri03] Fernando Tricas. Deadlock analysis, prevention and avoidance in sequential resource allocation systems. *Doctoral disserataion*, 2003.

[TU03] Shigemasa Takai and Toshimitsu Ushio. Effective computation of an lm (g)-closed, controllable, and observable sublanguage arising in supervisory control. *Systems & Control Letters*, 49(3) :191–200, 2003.

[UZ07] Murat Uzam and MengChu 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*, 37(3) :362–371, 2007.

[WLWZ12] SG Wang, Yue Li, CY Wang, and MC Zhou. Computation of all minimal siphons in petri nets. In *Proceedings of 2012 9th IEEE International Conference on Networking, Sensing and Control*, pages 46–51. IEEE, 2012.

[WR88] W Murray Wonham and Peter J Ramadge. Modular supervisory control of discrete-event systems. *Mathematics of control, signals and systems*, 1(1) :13–30, 1988.

[WYSG15] ShouGuang Wang, Dan You, Carla Seatzu, and Alessandro Giua. Complete enumeration of minimal siphons in ordinary petri nets based on problem partitioning. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 356–361. IEEE, 2015.

[WZW15] ShouGuang Wang, MengChu Zhou, and WenHui Wu. Design of a maximally permissive liveness-enforcing supervisor with reduced complexity for automated manufacturing systems. *Asian Journal of Control*, 17(1) :190–201, 2015.

[XKBT17] Yuchen Xie, Manel Khlif-Bouassida, and Armand Toguyéni. Modèles génériques en réseaux de petri bien formés pour le contrôle discret des trains autonomes. *Modelisation des Systèmes Réactifs*, 2017.

[ZDW+19] Qiaoli Zhuang, Wenzhan Dai, Shouguang Wang, Jingjing Du, and Qiuhong Tian. An mip-based deadlock prevention policy for siphon control. *IEEE Access*, 7 :153782–153790, 2019.

[ZDWN18] Qiaoli Zhuang, Wenzhan Dai, Shouguang Wang, and Fan Ning. Deadlock prevention policy for s 4 pr nets based on siphon. *IEEE Access*, 6 :50648–50658, 2018.

[ZL10] C Zhong and Z Li. Self-liveness of a class of petri net models for flexible manufacturing systems. *IET control theory & applications*, 4(3) :403–410, 2010.

Titre : Synthèse de contrôleurs pour des classes de Réseau de Petri à contrôlabilité et observabilité partielles : Application au contrôle automatisé des trains

Les collisions et les blocages sont des situations que l'on souhaite éviter dans beaucoup d'application de transport et notamment dans le contrôle automatisé des systèmes ferroviaires. Les méthodes de synthèse de contrôleurs peuvent garantir qu'un système en boucle fermé ne puisse pas atteindre de telles situations. Les méthodes issues de l'approche historique de Ramadge et Wonham permettent de traiter cette problématique mais font face à l'explosion combinatoire, qui empêche leurs applications sur les systèmes de grandes tailles. Dans le cadre de l'évitement des blocages, la grande majorité des approches font l'hypothèse que le système est sous contrôlabilité et observabilité totales. Cependant, cette hypothèse n'est pas vérifiée pour tous les systèmes. Cette thèse propose des méthodes de synthèses de contrôleurs applicables sur des systèmes sous observabilité et contrôlabilité partielles sans souffrir de l'explosion combinatoire. Ces méthodes se basent sur le formalisme des Réseaux de Petri, et concernent uniquement des classes pour la modélisation de systèmes d'allocation de ressources. Pour ne pas avoir recours à l'exploration des états, ces méthodes identifient les situations de blocages grâce à la résolution d'un problème d'optimisation mixte en nombres entiers. Ces méthodes sont par la suite utilisées pour le routage des trains dans un nœud ferroviaire. Une méthode systématique de modélisation de la circulation des trains dans un nœud ferroviaire est proposée. Le modèle ainsi obtenu est un Réseau de Petri pour la modélisation des systèmes d'allocation de ressources. Pour éviter les collisions et les blocages au seins de ce modèle, une méthode de synthèse reposant sur les apports théoriques de cette thèse est appliquée.

Mots clés : Systèmes à événements discrets, Réseaux de Petri, Supervision, Systèmes d'allocation de ressources, Évitement de blocages, Contrôlabilité et observabilité partielle, Systèmes ferroviaires.

Title : Synthesis of Controllers for Petri Net Classes with Partial Controllability and Observability: Application to Automated Train Control

Collisions and deadlocks are unwanted situations in many transport applications such as railway systems. Control synthesis methods can assure that a closed-loop system cannot lead to deadlock situations. Methods issued from the historical approach of Ramadge and Wonham allow dealing with this issue but encounter the problem of combinatorial explosion, that prevents its use on large systems. In the case of deadlock avoidance, the majority of approaches are based on the assumption of total controllability and observability. However, this assumption is not verified in all kinds of systems. This thesis proposes control synthesis methods that do not suffer from the combinatorial explosion. These methods are based on the formalism of Petri nets and uniquely concern classes of resource allocation systems. In a way to not resort to state exploration, these methods identify deadlock situations by resolving a mixed integer programming problem. These methods are then used for routing trains in a railway node. A systematic modeling approach of the trains' circulations in a railway node is developed. The model obtained is a Petri net for resources allocation system. To avoid collisions and deadlocks in the model, a control synthesis method, based on the theoretical background developed, is proposed.

Keywords : Discrete Event Systems, Petri nets, Supervision, Resource allocation system, Deadlock avoidance, Partial controllability and observability, Railway systems.