Modélisation du comportement humain dans les simulations de combat naval

Isabelle Toulgoat

Thèse

présentée à L'UNIVERSITÉ DU SUD TOULON-VAR,

préparée au sein du laboratoire SNC,

par
Isabelle Toulgoat,

pour l'obtention du diplôme de
Docteur de l'Université du Sud Toulon Var

DISCIPLINE : Informatique
SPÉCIALITÉ : Intelligence artificielle

Titre du mémoire :

Modélisation du comportement humain dans les simulations de combat naval

Soutenue le 31 Janvier 2011.

Dr Laurence CHOLVY :	DR à l'ONERA, Toulouse	Rapporteur
Dr Eva CRÜCK :	Ingénieure de recherche, DGA, Paris	Examinatrice
Pr Jean-Paul GAUTHIER :	USTV, Toulon	Examinateur
Pr Yves LACROIX :	USTV, Toulon	Directeur de thèse
Dr Anne MONSALLIER :	DCNS, Toulon	Invitée
Dr Henri PRADE :	Université Paul Sabatier, CNRS, Toulouse	Rapporteur
Pr Pierre SIEGEL :	Université d'Aix Marseille	Co-directeur de thèse

Remerciements

Je tiens à remercier en premier lieu mes directeurs de thèse Yves Lacroix et Pierre Siegel pour leur encadrement tout au long de cette thèse. Je les remercie de la confiance qu'ils m'ont accordée et pour ces nombreux échanges qui ont permis la réalisation de ce projet.

Je remercie l'équipe Simulation de DCNS dans laquelle j'ai effectué cette thèse, et plus particulièrement Anne Monsallier qui a suivi mes travaux tout au long de la thèse. Je remercie également Laurent Pellissero pour sa patience lors de la définition des règles de comportements et pour sa disponibilité au cours de la thèse.

Je remercie Laurence Cholvy et Henri Prade de m'avoir fait l'honneur d'être les rapporteurs de cette thèse et de l'intérêt qu'ils ont porté à mon travail. Merci également aux autres membres du jury, Eva Crück et Jean-Paul Gauthier, qui ont accepté de juger ce travail.

Un grand merci à mes amis pour leur bonne humeur et leur présence : Rémi, Pierre-Guillaume, Jérôme, Emilie et Olivier, Karin et Pierre, Fabrice et Alice, Albane, Mélanie.

Je tiens également à remercier ma famille pour son soutien tout au long de mes études et de mon doctorat. Enfin, je remercie profondément mon mari, Julien Briche, qui m'a supportée et soutenue durant ces trois années et qui a su me réconforter dans les moments difficiles.

Table des matières

Remerciements **3**

Table des matières **5**

Table des figures **11**

Liste des tableaux **15**

Introduction **17**

I Contexte et cas d'application 21

1 Contexte industriel et problématique **23**
1.1 Contexte 23
1.1.1 Présentation de l'entreprise DCNS 23
1.1.2 Le département Simulation 24
1.1.3 Présentation de l'atelier de simulation ATANOR . . . 25
1.2 Problématique 26
1.3 Exigences 27

2 Cas d'application : Scénario sous-marin contre sous-marin **29**
2.1 Contexte particulier du sous-marin 29
2.1.1 Informations fournies par les sonars 29
2.1.2 Informations incomplètes et incertaines 31
2.2 Définitions 33
2.3 Règles de comportements 34
2.4 Illustration de quelques règles de comportements 41

II Représentation des connaissances et modélisation

du comportement **45**

3 Etat de l'art **47**
3.1 L'intelligence artificielle et la représentation des connaissances 47
3.1.1 L'intelligence artificielle 47
3.1.2 La représentation des connaissances 49
3.2 Représentation des connaissances en logique "classique" . . . 49
3.2.1 Le calcul propositionnel 50
3.2.2 La logique du premier ordre 55
3.2.3 Les limites de la logique classique 60
3.3 Représentation des connaissances en logique non-monotone . 61
3.3.1 La logique des défauts [90] 62
3.3.2 Answer Set Programming 67
3.4 Représentation des connaissances avec les systèmes multi-agents 71
3.4.1 Définitions 71
3.4.2 Les approches comportementales 73
3.4.3 Conclusion 79
3.5 Raisonnement temporel en intelligence artificielle 80
3.5.1 Logique du premier ordre 81
3.5.2 Logique modale temporelle 81
3.5.3 Logique temporelle réifiée 82
3.5.4 Interprétation temporelle de la logique des défauts . . . 83
3.5.5 Conclusion 85

4 Modélisation du comportement avec la théorie des schémas **87**
4.1 Définition d'un schéma et de la consigne de déplacement . . . 87
4.2 Formalisation des règles sous forme de schémas 88
4.2.1 Décomposition du scénario en phases 88
4.2.2 Définition des conditions d'activation 89
4.2.3 Définition des schémas 91
4.3 Exemple de calcul pour la consigne de déplacement de l'officier 97
4.4 Validation de cette modélisation 99
4.5 Conclusion 100
4.5.1 Ajout d'un nouveau schéma 100
4.5.2 Limites de la modélisation du comportement avec la théorie des schémas 102

5 Approche proposée : Représentation des connaissances avec la logique des défauts **105**
5.1 Formalisation des règles avec la logique des défauts et une prise en compte du temps 106

5.1.1 L'ensemble des faits W 106
5.1.2 Les défauts D 109
5.1.3 Consignes renvoyées au sous-marin 110
5.1.4 Ecriture des règles 112
5.2 Exemple de calcul 120
5.2.1 Les faits W et les défauts D 120
5.2.2 Calcul des extensions 121
5.3 Implémentation 123
5.3.1 Le langage Prolog 123
5.3.2 Implémentation en Prolog 124
5.3.3 Algorithme de calcul d'extension 127
5.3.4 Interface du programme avec ATANOR 129
5.3.5 Complexité algorithmique 131
5.4 Conclusion 132

III Sélection de l'action 133

6 Etat de l'art 135
6.1 Les préférences en logique des défauts 135
6.1.1 La théorie des défauts ordonnés de Brewka, 1994 . . . 137
6.1.2 L'approche de Rintanen 140
6.1.3 L'approche Delgrande-Schaub 141
6.1.4 Système non monotone de règles annotées 143
6.1.5 La théorie des défauts ordonnés de Brewka et Eiter, 2000 145
6.1.6 Conclusion 148
6.2 La sélection de l'action dans les approches comportementales . 148
6.2.1 L'arbitrage 149
6.2.2 La fusion de commande 152
6.3 Les méthodes d'aide multicritère à la décision 154
6.3.1 Définition 154
6.3.2 Trois grandes familles de méthodes 156
6.3.3 Exemples de méthodes en théorie de l'utilité multi-attribut 158

7 Sélection de l'action avec des préférences et une méthode d'aide multicritère à la décision 165
7.1 Les principes de sélection d'une extension 166
7.1.1 Principe 1 : choix des extensions les plus intéressantes . 166
7.1.2 Principe 2 : choix des extensions obligatoires 166

7.1.3 Principe 3 : gestion du choix entre plusieurs extensions concurrentes . . . 167
7.1.4 Principe 4 : respect du changement minimal . . . 167
7.1.5 Principe 5 : effet de surprise . . . 167
7.2 La fonction de poids pour les extensions . . . 168
7.2.1 Les coefficients de préférences sur les défauts . . . 168
7.2.2 Fonction d'utilité . . . 169
7.2.3 Le caractère de l'officier . . . 169
7.2.4 Fonction de poids . . . 171
7.3 Choix aléatoire d'une extension . . . 171
7.3.1 Choix aléatoire . . . 173
7.3.2 Correction de la fonction de poids des extensions . . . 174
7.3.3 Filtrage des extensions avec des fonctions de poids trop faibles . . . 174
7.4 Prise en compte du changement minimal . . . 175
7.5 Exemple de calcul pour la consigne de déplacement de l'officier 175
7.5.1 Coefficients de préférences sur les défauts en fonction des critères . . . 176
7.5.2 Fonction d'utilité . . . 177
7.5.3 Coefficient de caractère de l'officier . . . 177
7.5.4 Fonction de poids des extensions . . . 178
7.5.5 Correction des fonctions de poids des extensions . . . 178
7.6 Conclusion . . . 179

IV Résultats 181

8 Comparaison des résultats obtenus avec les deux modélisations du comportement 183
8.1 Simulation de différents caractères pour l'officier de quart . . . 184
8.1.1 Paramètres . . . 184
8.1.2 Définition des caractères . . . 184
8.1.3 Mise en pratique de la paramétrisation . . . 185
8.2 Résultats étudiés . . . 188
8.3 Influence de la modélisation du comportement de l'officier . . . 189
8.3.1 Résultats . . . 189
8.3.2 Analyse des résultats . . . 195
8.4 Influence du caractère de l'officier sur la réussite de la mission 196
8.4.1 Résultats . . . 196
8.4.2 Analyse des résultats . . . 198
8.5 Conclusion . . . 198

Conclusions et perspectives **201**

Bibliographie **205**

Table des figures

1.1 Réseau de Pétri 26

2.1 Baffle du sous-marin. 30
2.2 Antennes sonar sur sous-marin [19] 31
2.3 Profil bathycélérimétrique en été en Méditerrannée 32
2.4 Champ sonore émis par une source placée à 5 mètres d'immersion 34
2.5 Définition : cap C, azimut AZ et gisement G 34
2.6 Trajectoire aléatoire d'un sous-marin dans son carré 36
2.7 Manœuvre de mesure de distance 37
2.8 Affinement de la trajectoire estimée grâce à la manœuvre de mesure de distance 38
2.9 Ralliement du poste de pistage 39
2.10 Manœuvre de pistage 39
2.11 Illustration de quelques règles de comportements 42

3.1 Arbre de recherche des solutions pour le calcul d'extensions. . 66
3.2 De gauche à droite : Evitement de la collision ; Alignement des vélocités ; Centrage par rapport au groupe ; 74
3.3 Une décomposition du contrôle d'un robot mobile en comportements [25] 75
3.4 Architecture de subsomption [25] 76
3.5 Mécanisme de suppressions et d'inhibition dans un module [25] 76
3.6 Deux schémas : aller-vers-cible (gauche) et éviter-obstacle (droite) [8] 77
3.7 Formations (de gauche à droite) : ligne , colonne, diamant, en V [10]. 78
3.8 Point de référence déterminé avec les techniques : centre, leader, voisin (de gauche à droite) [10] 78

4.1 Automate à états finis "Phase" représentant l'enchaînement des décisions de l'officier. 89
4.2 Automate à états finis "Baffle" représentant la position du sous-marin A par rapport au baffle du sous-marin adverse B. 90
4.3 Baffle du sous-marin. 90
4.4 Fonction de direction pour le schéma anti-collision. 92
4.5 Fonction de direction pour le premier tronçon de la mesure de distance (cas où le gisement est compris entre 180 ° et 360 °). 93
4.6 Fonction de direction du schéma ralliement du poste de pistage 94
4.7 Fonction d'intensité du schéma ralliement du poste de pistage 95
4.8 Fonction d'intensité du schéma positionnement centré sur la cible 96
4.9 Fonction de direction du schéma pistage 97
4.10 Exemple de calcul de consigne pour le sous-marin A, ralliement de la position de pistage du sous-marin B 99
4.11 Exécution de la modélisation du comportement avec la théorie des schémas, interfacé avec l'atelier de simulation ATANOR . 100
4.12 Automate à états finis Batterie représentant l'état de charge du sous-marin 101
4.13 Exécution de la modélisation du comportement avec la théorie des schémas, interfacé avec l'atelier de simulation ATANOR : Comportement obtenu d'un sous-marin qui doit pister son ennemi et monter au schnorchel. 102
4.14 Résultante de deux schémas : éviter une torpille et éviter un obstacle 103

5.1 Manœuvre de pistage, positionnement de l'ennemi dans des gisements successifs de -30 ° et de 30 ° 116
5.2 Arbre de recherche des solutions pour un calcul d'extensions. 129
5.3 Interface entre NoMROD et ATANOR 129
5.4 Exécution du programme NoMROD, interfacé avec l'atelier de simulation ATANOR 130

6.1 Comportement d'un robot avec une machine à états finis [13] . 150
6.2 Comportement d'un robot avec des réseaux d'activation [66] . 152

8.1 Fonction d'intensité du schéma positionnement centré sur la cible 186
8.2 Vitesse en ralliement du poste de pistage 187
8.3 Vitesse en dérobement 187
8.4 Officier au caractère standard 189

8.5 Modélisation d'un officier au caractère standard avec la théorie des schémas 190
8.6 Modélisation d'un officier au caractère standard avec la logique des défauts 190
8.7 Officier au caractère prudent 191
8.8 Modélisation d'un officier au caractère prudent avec la théorie des schémas 192
8.9 Modélisation d'un officier au caractère prudent avec la logique des défauts 192
8.10 Officier au caractère imprudent 193
8.11 Modélisation d'un officier au caractère imprudent avec la théorie des schémas 194
8.12 Modélisation d'un officier au caractère imprudent avec la logique des défauts 194
8.13 Influence du caractère de l'officier sur la réussite du pistage . 196
8.14 Influence du caractère de l'officier sur le délai avant pistage . 197
8.15 Influence du caractère de l'officier sur la distance en début de pistage 197

Liste des tableaux

2.1 Définitions de termes spécifiques à la navigation 35

3.1 Les 5 connecteurs logiques 50
3.2 Table de vérité du connecteur non 52
3.3 Table de vérité des connecteurs logiques 52

4.1 Définition des schémas et de leurs conditions d'activation pour chaque phase 91

6.1 Espace d'action pour le déplacement 153
6.2 Moyenne pondérée pour chaque étudiant 159
6.3 Limite de la somme pondérée : pas de prise en compte des conflits entre les critères 159
6.4 Moyenne pondérée et intégrale de Choquet pour chaque étudiant 163

7.1 Valeurs des coefficients pour chaque défaut. 168
7.2 Scores des extensions. 170
7.3 Fonctions d'utilité. 170
7.4 Définition des fonctions d'utilité $g_k(E_i)$ et des coefficients de caractère ω_k 172
7.5 Coefficients de préférences pour les défauts 176
7.6 Scores et fonctions d'utilité pour les extensions 177
7.7 Coefficients de caractère de l'officier 178
7.8 Fonctions de poids pour les extensions, en fonction des différents caractères de l'officier 178
7.9 Fonctions de poids avec correction pour l'officier prudent . . . 179
7.10 Fonctions de poids avec correction pour l'officier téméraire . . 179

Introduction

Ce travail de thèse s'est effectué dans le cadre d'une bourse CIFRE (Convention Industrielle de Formation par la Recherche) au sein de l'entreprise DCNS dans l'équipe Simulation, en collaboration avec le laboratoire SNC (Systèmes Navals Complexes). Les simulations consistent à modéliser les navires et les équipements du système de combat à l'aide d'un atelier de simulation, puis à dérouler le scénario en temps simulé. Le but des simulations est d'évaluer les performances opérationnelles de navires militaires, dans un scénario donné. Ces simulations fournissent des réactions qui sont prédéfinies : elles ne prennent pas en compte l'analyse et la décision d'un opérateur, qui peuvent conduire à des réactions inattendues. L'objectif de la thèse est de modéliser le comportement d'un opérateur pour les simulations de combat naval.

Le début de la thèse a été consacré à la définition d'un cas d'application à partir duquel nous avons défini des règles de comportement. Nous avons travaillé sur un scénario faisant intervenir deux sous-marins adverses, et plus particulièrement sur les réactions de l'officier de quart en cas de détection de son adversaire. Les sous-marins évoluent dans un contexte très particulier : un sous-marin est aveugle en immersion ; la plupart du temps, les seules informations dont il dispose proviennent de ses sonars passifs. Les sonars permettent de détecter et de situer des objets sous l'eau, à partir des propriétés de propagation du son dans l'eau. Les sonars passifs permettent d'écouter et de localiser les bruits. Les sonars en mode actif émettent des impulsions sonores et permettent de détecter les échos revenant des cibles. Un sous-marin n'utilise pratiquement jamais ses sonars en mode actif car il est alors très indiscret. Les informations recueillies par les sonars passifs sont incomplètes et incertaines. Le sous-marin ne possède pas toutes les informations sur son environnement extérieur. Il peut, à tout moment, perdre la détection d'un sous-marin adverse. De plus, le sous-marin ne connaît que la position estimée de l'ennemi et non sa position exacte. Les règles de comportements, que nous utilisons, ont été définies avec des opérationnels et elles sont réellement utilisées à bord des sous-marins.

Ensuite, nous avons étudié une méthode de modélisation du comportement déjà utilisée dans l'entreprise DCNS. Dans cette modélisation, la représentation des connaissances est faite avec la théorie des schémas . Cette méthode de modélisation donne de bons résultats mais elle n'était pas adaptée aux exigences fixées pour la thèse.

Nous avons finalement choisi de représenter les connaissances avec la logique non-monotone la plus utilisée : la logique des défauts [90]. Cette logique permet de prendre en compte les informations incomplètes, incertaines et révisables que l'opérateur possède sur l'environnement. Pour le cas d'application traité, les clauses de Horn et les défauts normaux suffisent pour représenter les connaissances. Cependant, ce travail pourrait être généralisé en utilisant d'autres clauses (ou même d'autres formules) ainsi que les défauts généraux. Nous avons ajouté une prise en compte du temps à la logique des défauts. Le temps est défini de façon discrète : à partir des informations connues à l'instant t, nous en déduisons les actions à exécuter à l'instant suivant $t + 1$. La logique des défauts permet de calculer des extensions. Chaque extension correspond à une action possible pour le sous-marin. Pour des raisons pratiques, l'implémentation est réalisée en langage Prolog. Ce langage permet de définir facilement les clauses de Horn, qui représentent l'information, ainsi que les défauts normaux et l'algorithme de calcul d'extension.

Le problème de sélection de l'action a ensuite été abordé. Cette partie consiste à simuler la décision de l'officier. A chaque instant, l'officier doit choisir une action en fonction de sa perception de la situation tactique. Nous définissons une méthode pour choisir entre les extensions. Cette méthode permet de simuler différents comportements, qui dépendent du caractère de l'officier. Pour cela, une fonction de poids est définie pour les extensions. Cette fonction de poids prend en compte des préférences définies sur les défauts, en fonction de différents critères, ainsi que des coefficients de caractère pour l'officier, qui définissent l'importance que l'officier accorde à chaque critère. Cette fonction de poids a été définie pour notre cas d'application mais elle peut être facilement généralisable. Nous avons défini une méthode pour sélectionner une extension, en ajoutant une part d'aléatoire dans le choix de l'extension.

Notre modélisation du comportement en logique des défauts a été implémentée en Prolog, interfacée avec l'atelier de simulation ATANOR, testée et validée par des sous-mariniers. Elle est maintenant utilisée au sein de l'équipe Simulation pour des études. Une étude a permis de comparer la modélisation du comportement avec la théorie des schémas et la modélisation du comportement obtenue avec la logique des défauts, en faisant des simulations avec l'atelier de simulation ATANOR. En jouant sur certains paramètres, nous avons réussi à définir différents caractères pour l'officier. Cette étude met en avant les avantages et inconvénients de ces différents caractères par

rapport à la mission à accomplir.

Plan de la thèse

Ce document s'organise en quatre grandes parties.

La première partie est composée des chapitres 1 et 2. Le chapitre 1 présente le contexte industriel dans lequel la thèse s'est déroulée. Il détaille ensuite la problématique traitée et les exigences auxquelles nous avons dû répondre. Le chapitre 2 décrit le cas d'application sur lequel nous avons travaillé. Il porte sur un scénario faisant intervenir deux sous-marins adverses. Les règles de comportement décrivent les réactions de l'officier de quart à bord du sous-marin.

La seconde partie regroupe les chapitres 3 à 5. Le chapitre 3 présente un état de l'art sur l'intelligence artificielle et la représentation des connaissances. Pour la représentation des connaissances, nous présentons la logique classique, les logiques non-monotones et les systèmes multi-agents avec les approches comportementales. Le raisonnement temporel en intelligence artificielle est ensuite introduit. Le chapitre 4 présente une première méthode de modélisation du comportement, déjà utilisée dans l'entreprise DCNS. Cette méthode utilise la théorie des schémas. Cette modélisation donne de bons résultats. Cependant, elle ne semble pas être réutilisable dans le cadre de la thèse. Dans le chapitre 5, nous exposons notre approche pour modéliser le comportement. Nous utilisons la logique des défauts avec une prise en compte du temps. L'implémentation en langage Prolog est ensuite décrite.

La troisième partie est composée des chapitres 6 et 7. Le chapitre 6 présente un état de l'art sur la sélection de l'action. Différentes approches sont exposées : les préférences en logique des défauts, la sélection de l'action dans les approches comportementales et les méthodes d'aide multicritère à la décision. Dans le chapitre 7, nous exposons notre méthode de sélection de l'action, avec la définition de principes de sélection d'une extension, d'une fonction de poids pour les extensions et finalement d'un choix aléatoire.

La quatrième et dernière partie, avec le chapitre 8, présente une étude qui compare les deux types de modélisation du comportement. Afin de comparer ces modélisations, nous avons fait des simulations sur l'atelier de simulation ATANOR.

Première partie

Contexte et cas d'application

Chapitre 1

Contexte industriel et problématique

1.1 Contexte

1.1.1 Présentation de l'entreprise DCNS

Ma thèse s'est déroulée dans le cadre d'une bourse CIFRE au sein de l'entreprise DCNS. Anciennement DCN (Direction des constructions navales), l'entreprise est devenue une société de droit privé en 2003. Puis elle a acquis la société TNF (Thalès Naval France), ainsi que la filiale Armaris anciennement commune à DCN et Thalès.

DCNS est chargée de la conception, de la réalisation et de la maintenance des bâtiments de la marine nationale et de la plupart des systèmes qui y sont intégrés.

Le client principal de DCNS est la marine nationale française, mais la société développe de plus en plus son activité à l'étranger.

L'organisation de DCNS repose sur :

- Trois divisions :
  - la division Sous-marins avec les centres de Cherbourg, Nantes-Indret et Ruelle. Le rôle de cette division est de réaliser, livrer et commercialiser des sous-marins et leurs systèmes associés.
  - la division Systèmes Navals de Surface avec les centres de Bagneux, Lorient et du Mourillon. Son rôle est de concevoir, commercialiser et livrer à ses clients des systèmes navals de surface répondant à leur besoin (bâtiments de surface, systèmes d'informations et de surveillance).
  - la division Service avec les centres de Brest et Toulon. Son rôle est le maintien en condition opérationnelle. Elle comprend l'ingénierie,

la maintenance et la chaîne d'approvisionnement des navires.
- Trois Business Units :
  - la BU Armes Sous-marines à Saint Tropez,
  - la BU Nucléaire Civil à Paris,
  - la BU Simulateurs et Formation à Ruelle.

Actuellement, DCNS réalise le programme de sous-marins nucléaires d'attaque de type Barracuda ainsi que le développement et la finalisation du programme sous-marin nucléaire lanceur d'engins de nouvelle génération dont le Terrible, pour le compte de la marine française. Le Scorpène est un sous-marin conventionnel (c'est-à-dire diesel), vendu à la Malaisie, deux au Chili, six à l'Inde et plus récemment au Brésil.

En ce qui concerne les bâtiments de surface, une première frégate anti-aérienne de classe Forbin du programme Horizon a été livrée fin 2008 à la marine nationale. Actuellement, DCNS réalise le programme européen des frégates multi-missions de dernière génération FREMM. Pour ce navire, la marine royale marocaine a été le premier client export.

Ma thèse s'est déroulée dans la division Systèmes Navals de Surface au sein du département Simulation.

1.1.2 Le département Simulation

Les simulations sont très utilisées dans l'industrie, elles permettent d'obtenir des résultats sans avoir recours à des moyens trop importants (essais coûteux ou risqués). L'objectif des simulations est d'évaluer les performances technico-opérationnelles dynamiques d'un navire armé ou d'un système naval.

Voici quelques exemples d'applications :
- un scénario d'auto-défense d'une frégate vis-à-vis d'un missile anti-navire (quelle est la probabilité d'échapper à la menace ?) ;
- capacité à détecter un sous-marin adverse avant d'être détecté ;
- scénario de mise en oeuvre de drones en guerre des mines.

Plusieurs types d'études sont réalisés dans le département simulation :
- Etudes de performances : le but est d'estimer sommairement et rapidement des performances comme des portées de détection, ou des risques par rapport à certaines menaces, sur des scénarios simples.
- Simulations de performances : déroulement d'un scénario un grand nombre de fois en temps simulé à l'aide d'outils de simulations. Le but est d'évaluer l'efficacité d'un système dans un environnement opérationnel simulé.
- Démonstrations : visualisation en 2D et 3D d'un scénario dynamique.

1.1.3 Présentation de l'atelier de simulation ATANOR

Les simulations de performances peuvent être exécutées avec l'atelier de simulation ATANOR (Atelier de simulation numérique orienté réutilisation). ATANOR est un atelier logiciel, développé par DCNS, pour produire et exploiter des simulateurs numériques ([112], [19]). Cet atelier permet de simuler le comportement d'un navire ou d'une force navale dans un scénario particulier. Il peut servir à produire et exploiter tout simulateur numérique pour mener une étude sur un système d'armes. ATANOR est orienté réutilisation : les modèles développés pour un simulateur peuvent être réutilisés dans d'autres simulateurs.

Cet outil comporte :

- des modèles d'environnements simulés ;
- des modèles génériques de plateforme (navires, sous-marins, aéronefs, ...) et de menaces (missiles, torpilles, ...) ;
- des modèles génériques d'équipements de système de combat naval (senseurs, systèmes d'armes), comme les sonars ;
- un outil de création et d'animation de scénario ;
- des outils d'enregistrement et d'analyse des résultats ;
- un outil de paramétrisation et de définition des jeux de calcul.

Un scénario de simulation est défini par un ensemble d'acteurs à faire évoluer au cours d'une simulation dynamique et qui communiquent par échanges de messages de données. Les acteurs modélisent les différents systèmes intervenants dans l'étude à mener en simulation (les torpilles, les sous-marins ...) mais aussi les milieux dans lesquels ils évoluent (le milieu acoustique sous-marin, ...).

Un acteur d'une simulation est défini par un ensemble d'équipements qui modélisent des sous-ensembles de cet acteur et qui communiquent par échanges de messages de données. Les équipements de chaque acteur sont de deux types :

- non-terminaux : ils échangent entre eux par des messages externes ;
- terminaux : ils sont définis par une structure de données, des méthodes, qui simulent le fonctionnement du système et un réseau de Pétri.

Le principe d'un réseau de Pétri utilisé dans ATANOR est représenté schématiquement sur la figure 1.1. Le réseau de Pétri [82] permet de modéliser le comportement du système et comporte différentes places (cf. sur la figure 1.1 : Etat initial, Etat 1 , Etat 2 , Fin) qui représentent les états du système. Les transitions entre ces places sont activées par des évènements internes et externes. Une marque détermine la place courante du réseau de Pétri à un instant donné. Une seule place peut être active à la fois, ce qui interdit les fonctionnements parallèles au sein de l'équipement.

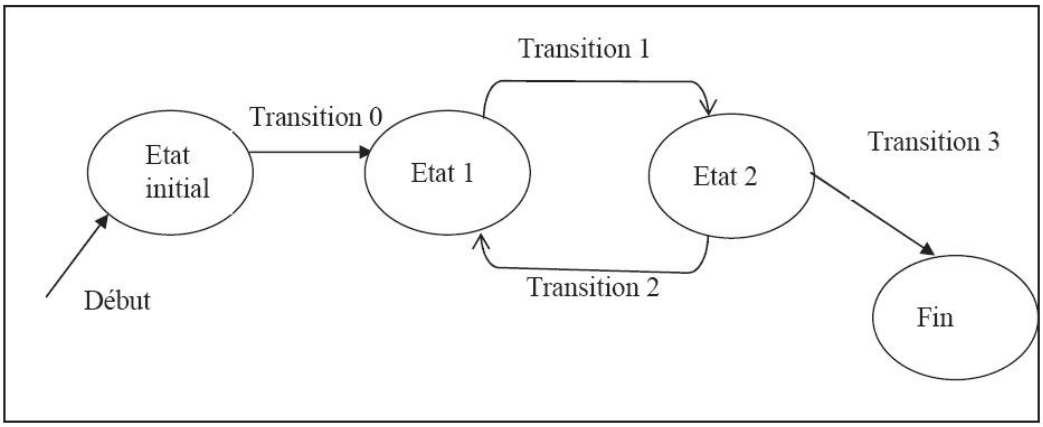

FIG. 1.1 – Réseau de Pétri

Pour chaque scénario, des analyses statistiques type Monte-Carlo sont réalisées, en faisant varier aléatoirement quelques paramètres incertains (comme le bruit sur les données d'entrée ou des données soumises à des variations) pour un même jeu de données d'entrée. L'objectif est de quantifier statistiquement, sur un grand nombre d'exécutions élémentaires de la simulation, le succès de la mission sous forme de résultats globaux. Le nombre de tirages de Monte-Carlo pour les simulations doit être suffisamment élevé pour assurer une bonne convergence des résultats. Bien entendu, les simulations ne sont pas réalisées en temps réel mais en temps simulé accéléré.

1.2 Problématique

Le but des simulations de combat naval est d'évaluer les performances opérationnelles de navires militaires ou de forces navales dans un scénario et un théâtre d'opérations donné. L'atelier de simulation de DCNS, ATANOR, permet de modéliser des navires et des équipements du système de combat et de dérouler un scénario en temps simulé en faisant interagir les différents modèles [112].

Comme nous l'avons vu au paragraphe précédent (cf. paragraphe 1.1.3), dans cet atelier de simulation, le comportement est modélisé avec des réseaux de Pétri. La modélisation des règles de comportement avec un réseau de Pétri donne des réactions prédéfinies et automatiques. En effet, dans un réseau de Pétri, l'enchaînement des actions est défini à l'avance dans une machine à états. Avant même que le scénario soit lancé, les réactions qui vont être enchaînées sont connues.

De plus, dans une situation tactique complexe, l'analyse et la décision d'un opérateur peuvent conduire à une réaction inattendue. En effet, les réactions de l'opérateur vont dépendre de la façon dont il perçoit la situation tactique, ou encore de son caractère, de son état d'esprit ... La définition du comportement avec un réseau de Pétri ne prend pas en compte cet aspect. Cette approche est trop déterministe pour modéliser le comportement d'un opérateur.

Un inconvénient de la modélisation par réseau de Pétri est de devoir implémenter une nouvelle machine à états lorsqu'une nouvelle action doit être ajoutée [40]. Nous ne pouvons pas ajouter de façon simple de nouveaux comportements et nous devons nous soucier des comportements déjà définis.

Le but de ce travail a été de développer un système qui permet de modéliser les lois de comportement d'un opérateur. Cette modélisation doit intégrer des comportements réalistes, l'opérateur doit adapter son comportement en fonction des situations qu'il rencontre. Nous avons travaillé sur un cas d'application faisant intervenir deux sous-marins adverses, et plus spécialement sur les réactions de l'officier de quart en cas de détection de son adversaire. Un des objectifs est de pouvoir interfacer ce système de modélisation du comportement avec l'atelier de simulation ATANOR : après avoir récupéré les données d'ATANOR sur le sous-marin et son environnement, le système doit pouvoir renvoyer les instructions de manœuvre pour le sous-marin à ATANOR.

La modélisation du comportement, interfacée avec ATANOR, servira à effectuer des études. Cette modélisation peut permettre d'étudier l'influence du comportement de l'officier sur la réussite de la mission. Pour cela, nous pouvons définir différents comportements pour l'officier et étudier quel comportement permet d'accomplir le plus souvent la mission. Cette modélisation peut également permettre de définir le meilleur comportement à adopter dans une certaine situation ou encore d'étudier quel sous-marin est le mieux adapté pour un certain scénario.

1.3 Exigences

Nous devons donc réaliser un système qui permet de modéliser les lois de comportement de l'opérateur. Afin de respecter les contraintes industrielles, ce système doit répondre à plusieurs exigences :

1. permettre de raisonner sur des informations incomplètes, révisables et incertaines. En effet, un opérateur a seulement une vue partielle de son environnement. Dans le cas d'un sous-marin, aveugle en immersion, les seules informations proviennent des sonars passifs la plupart du

temps. Ces informations sont incomplètes et incertaines : le sous-marin peut facilement perdre la détection, l'officier a seulement accès à une estimation de la trajectoire de son adversaire ...Les incertitudes sont causées par le milieu marin, le sonar ainsi que par l'ennemi lui-même (cf. paragraphe 2.1.2). Les décisions de l'officier doivent pouvoir être révisées avec l'arrivée de nouvelles informations ([28] [107] [59]).

2. choisir entre plusieurs propositions lorsqu'il propose plusieurs actions possibles pour une même situation.
3. permettre l'addition de nouvelles règles de comportement, sans avoir à modifier la représentation des connaissances et sans avoir à remettre en question les règles précédemment établies (à l'inverse des réseaux de Pétri dans lesquels les modifications sont compliquées). Cette exigence permettra à l'utilisateur non développeur d'ajouter facilement de nouvelles règles de comportements.
4. être capable de raisonner sur des règles générales, sans avoir à compiler de manière très précise toutes les informations. Il n'est pas nécessaire pour l'utilisateur de décrire tous les cas de figures.

Ce travail est initié par la société DCNS pour des applications militaires : nous avons besoin d'un programme simple d'utilisation et robuste. De plus, ce travail devant être interfacé avec ATANOR, l'algorithme doit être rapide. Nous utilisons pour cela la logique non-monotone la plus connue : la logique des défauts [90]. Nous y ajoutons une prise en compte du temps : nous avons les données sur le sous-marin à l'instant t, et nous en déduisons les instructions au temps suivant.

Afin de choisir entre les différentes actions possibles, nous avons utilisé des préférences sur les défauts et nous avons défini une méthode de choix en utilisant une part d'aléatoire.

Ce travail a été implémenté en Prolog et interfacé grâce au langage C avec l'atelier de simulation ATANOR.

Chapitre 2

Cas d'application : Scénario sous-marin contre sous-marin

Pour modéliser l'intervention d'un opérateur, nous avons défini des règles de comportement avec des opérationnels. Le cas d'application que nous avons traité est celui de la modélisation de la décision d'un officier de quart dans un sous-marin en fonction des évènements perçus sur la situation tactique. Nous nous plaçons dans un scénario faisant intervenir deux sous-marins.

Pour définir les règles de comportement spécifiques à ce cas d'application, des sous-mariniers ont décrit un scénario, à partir duquel nous avons défini les règles. Ces règles de comportements représentent la réalité et sont vraiment utilisées à bord des sous-marins. Un contact permanent avec un sous-marinier m'a permis de compléter ces règles tout au long de la thèse.

Avant de définir les règles de comportement, nous décrivons, dans un premier temps, le contexte particulier lié aux sous-marins.

2.1 Contexte particulier du sous-marin

Le sous-marin doit absolument faire preuve de discrétion acoustique lorsqu'il est en mission, afin de ne pas se faire détecter. Pour cela, il reçoit les informations sur son environnement avec des sonars (Sound Navigation and Ranging) passifs.

2.1.1 Informations fournies par les sonars

Un sous-marin est aveugle en immersion. Pour se diriger, il utilise des cartes marines. La plupart du temps, les seules informations qu'il reçoit proviennent des sonars passifs [86]. Les sonars permettent, à partir des pro-

priétés de la propagation du son dans l'eau, de détecter et situer des objets (un sous-marin ennemi) sous l'eau. Le sous-marin utilise principalement les sonars passifs : ils permettent d'écouter et localiser les bruits. Les bruits sont reçus par les antennes de réception sonar.

Les analystes sonars, ou oreilles d'or, sont ensuite chargés d'analyser les bruits, de les classifier et de les identifier. Ils doivent distinguer le son des moteurs de leur propre sous-marin, de ceux des autres bâtiments, le son des baleines et des dauphins .... Parallèlement, à partir des données d'azimut et de fréquence, sont déterminées les informations sur la cinématique du contact (cap, vitesse, distance). Ces informations permettent de comprendre comment le contact se comporte. Chaque sous-marin possédant sa propre signature acoustique, ces informations sont également utilisées pour classifier le contact.

L'utilisation du sonar en mode actif permet d'émettre des impulsions sonores et de détecter les échos revenant des cibles. Le sonar actif fournit des informations plus précises. Un sous-marin n'utilise pratiquement jamais de sonar en mode actif car il commet alors une indiscrétion majeure et il trahit sa position. Il peut néanmoins utiliser le sonar actif dans des situations particulières.

Les sous-marins modernes disposent généralement de deux types d'antennes sonar de détection passive, disposées sur la coque : des antennes d'étrave et des antennes de flanc (cf. figure 2.2). Le sous-marin est sourd dans son baffle (cf. figure 2.1). En effet, à l'arrière du sous-marin, la réception des sonars est amoindrie pour diverses raisons : bruits produits par la propulsion du bâtiment, turbines, hélice et autres équipements mécaniques.

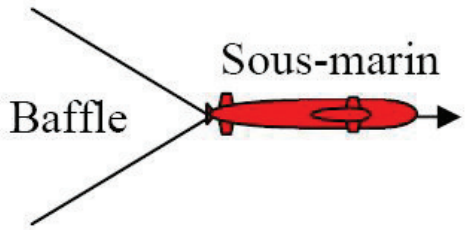

FIG. 2.1 – Baffle du sous-marin.

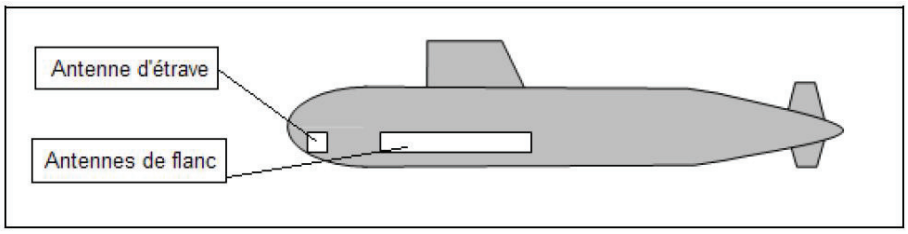

FIG. 2.2 – Antennes sonar sur sous-marin [19]

2.1.2 Informations incomplètes et incertaines

Contrairement à l'aéronautique où le pilote possède généralement des informations précises sur son environnement grâce aux radars embarqués et au sol, les informations obtenues dans un sous-marin avec les sonars passifs sont incomplètes et incertaines. Les données fournies par les sonars contiennent des incertitudes, générées par l'environnement et le sous-marin lui-même ([86], [19], [34]). De plus, le sous-marin risque à chaque instant de perdre la détection du sous-marin ennemi.

Incertitudes liées au milieu marin

L'océan est caractérisé par une célérité non constante (surtout en immersion, et dans une moindre mesure en distance et en temps). La célérité du son dépend de plusieurs facteurs : la pression, la température et la salinité. Les variations de ces facteurs modifient la vitesse et le chemin du son dans l'eau. La propagation du son est plus facilement perturbée dans l'eau car les caractéristiques du fluide peuvent changer très rapidement dans l'océan. La température décroît en général rapidement depuis la surface puis reste quasi-constante aux grandes immersions ($\geq$ 1000 mètres) (cf. figure 2.3). La salinité peut changer rapidement à cause des courants et des facteurs environnementaux. Ces changements ne sont pas facilement prévisibles. La pression, quant à elle, est plus prévisible car elle varie seulement en fonction de la profondeur. Le profil bathycélérimétrique représente la variation de la célérité du son dans l'eau en fonction de l'immersion (cf. figure 2.3). Sur ce profil, la célérité diminue avec la profondeur, lorsque la température diminue (de 0 à 50 mètres de profondeur). Ensuite, la température est constante. La célérité augmente linéairement du fait de la presssion hydrostatique.

Le bruit ambiant (trafic maritime, milieu naturel, bruits biologiques comme le chant des baleines et conditions météorologiques) parasite le son réceptionné par l'antenne.

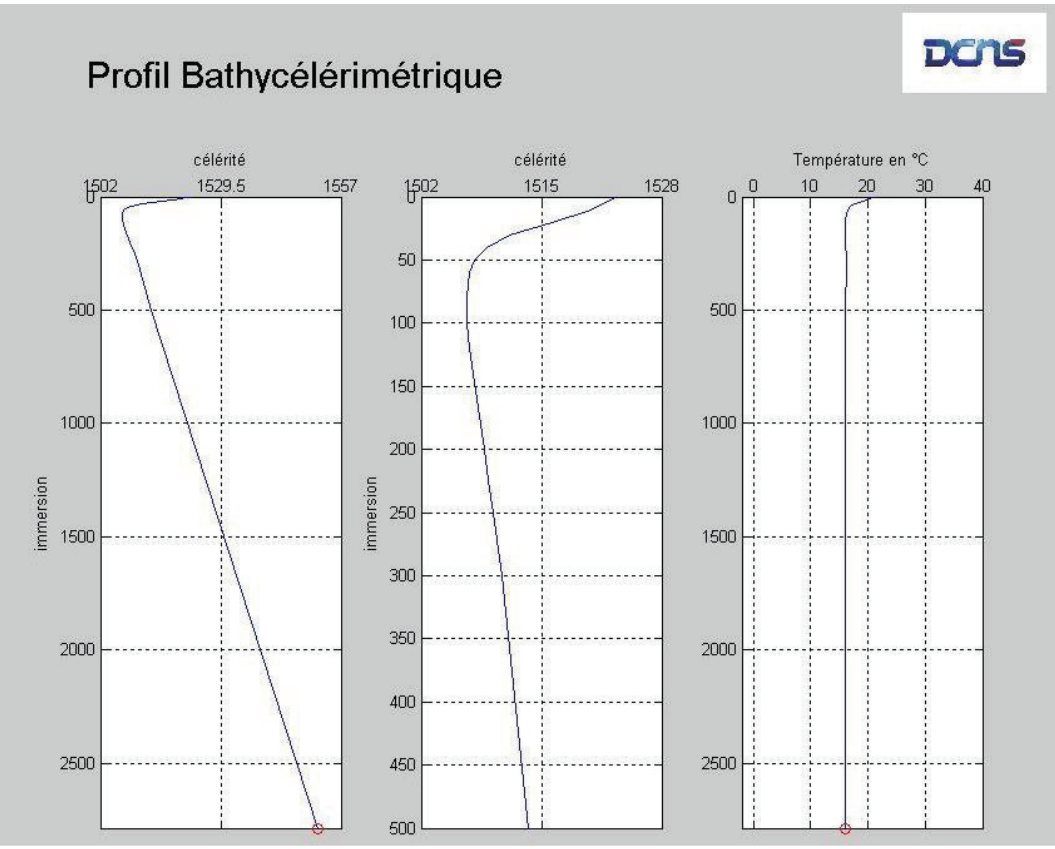

FIG. 2.3 – Profil bathycélérimétrique en été en Méditerrannée

Incertitudes liées au sonar

Les capteurs utilisés dans les sonars passifs ne sont pas parfaits. Lorsqu'un signal a été détecté par le sonar, il est ensuite traité pour être présenté à un opérateur. Une dégradation est générée entre le signal reçu par les capteurs et le signal présenté à l'opérateur (atténuation, ...). Les bruits propres du sous-marin (bruit perturbateur de la plateforme sur les antennes sonar, qui vient s'ajouter au bruit ambiant) viennent également perturber le son. De même, chaque opérateur a sa propre façon et ses propres compétences pour analyser le signal reçu.

De plus, l'officier a seulement accès à une estimation de la trajectoire de son adversaire : cette estimation est obtenue à partir des données d'azimut reçues par le sonar. Il ne possède pas la position et la cinématique exacte de l'ennemi. Son information au sujet de l'ennemi est donc incomplète et incertaine.

Incertitudes causées par l'ennemi

La détection d'un sous-marin ennemi dépend de sa discrétion acoustique c'est-à-dire son niveau de bruit rayonné en champ lointain. Le bruit rayonné en champ lointain est généré par le propulseur, les phénomènes hydrodynamiques et les sources internes. Lorsque le sous-marin a une détection, il peut facilement la perdre. Cette détection dépend du bruit émis par l'ennemi et des aléas de propagation. Si celui-ci arrête de faire du bruit, il ne sera plus détecté. Le champ sonore émis par une source placée à 5 mètres d'immersion en Méditerrannée et en été est représenté sur la figure 2.4. On observe des zones d'ombre, non insonifiées, dans lesquelles le sous-marin ne pourra pas détecter la source. La propagation des rayons sonores explique le fait que le sous-marin peut, à tout moment, perdre la détection de l'ennemi.

Le sous-marin peut également avoir recours à des dispositifs de brouillage, lorsque, par exemple, il s'est fait détecter. Il peut alors décider de lancer un leurre, qui fonctionne de manière acoustique. Un leurre peut, entre autres, imiter le bruit d'un sous-marin et il lance ainsi l'adversaire sur une fausse piste.

Les informations dont l'officier de quart dispose dans son sous-marin, grâce aux sonars passifs, sont donc incomplètes et incertaines. Avec toutes ces incertitudes sur le signal reçu, l'officier n'a pas de précision sur la qualité de l'information dont il dispose. Il doit donc prendre ses décisions dans un contexte difficile.

En logique propositionnelle, l'incertain est représenté par un "ou" ($\vee$) entre les propositions. Dans notre cas d'application, l'incertain est lié aux informations dont le sous-marin dispose. Les informations dont il dispose ne sont pas précises. Par exemple, il ne connaît qu'une position estimée de l'ennemi. Il a, par exemple, l'information suivante : l'ennemi se trouve entre 2500 et 3000 mètres. Cette information peut se traduire de la façon suivante : $2500 \vee 2501 \vee \ldots \vee 3000$. Lorsque l'on veut gérer l'incertain, on veut prendre en compte tous les cas possibles. Une autre façon de gérer l'incertain peut être de traduire cette information par un intervalle : [2500, 3000].

2.2 Définitions

Nous définissons certains termes spécifiques à la navigation dans le tableau 2.1, et sur la figure 2.5. Tout comme les sous-mariniers, nous appelons "but" le sous-marin ennemi.

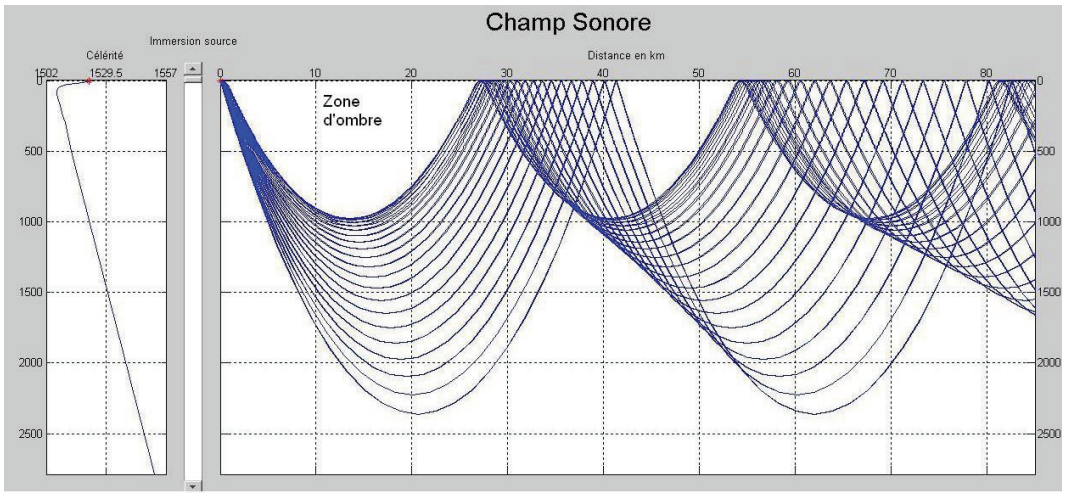

FIG. 2.4 – Champ sonore émis par une source placée à 5 mètres d'immersion

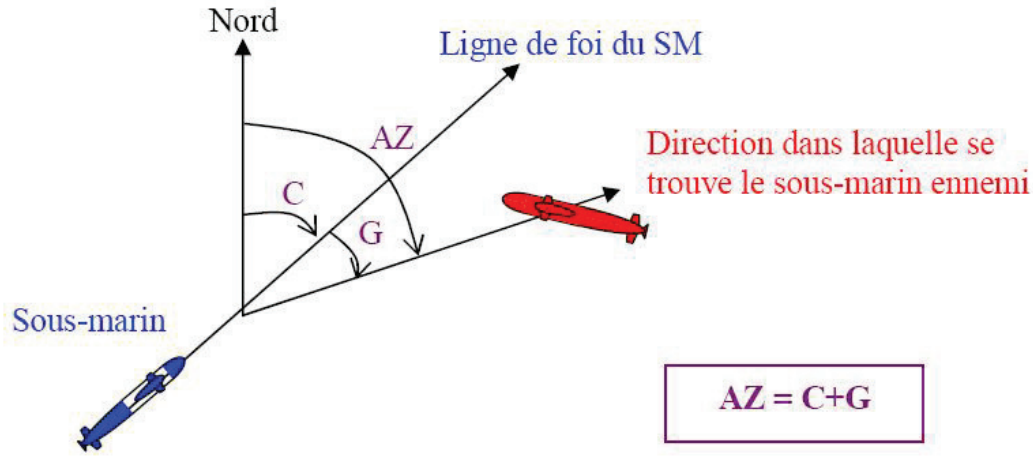

FIG. 2.5 – Définition : cap C, azimut AZ et gisement G

2.3 Règles de comportements

Voici quelques exemples de règles de comportement, définies avec des sous-mariniers, dans le cadre d'un scénario sous-marin contre sous-marin.

Termes	Notation	Définition
Cap	C	Angle entre la direction de progression du sous-marin et le Nord, il est compté entre 0 ° et 359 ° .
Azimut but	AZ	Angle entre la direction de détection du but (c'est-à-dire le sous-marin adverse) et le Nord.
Gisement but	G	Angle entre la direction de détection du but et la ligne de foi du sous-marin qui détecte. Il est compté entre 0 ° et 359 ° .
Défilement but instantané	D	Dérivée par rapport au temps de la valeur de l'azimut. Elle se mesure en degré par minute. Le défilement est positif si le but défile droite et négatif si il défile gauche. Si l'azimut 090 est relevé et dix minutes plus tard, nous avons l'azimut 080, le défilement moyen est alors de - 1 ° .

TAB. 2.1 – Définitions de termes spécifiques à la navigation

Règle 1 : les abattées

Tant qu'un sous-marin n'a pas de détection, il poursuit une trajectoire aléatoire de recherche dans sa zone de patrouille. Une zone de patrouille peut être définie par un carré dans lequel le sous-marin doit surveiller et détecter un éventuel intrus. Ce carré peut faire 50 milles nautiques sur 50 milles nautiques.

Une trajectoire aléatoire de recherche est définie de la façon suivante : le sous-marin avance tout droit et effectue des abattées d'écoute (des changements de cap) dans des intervalles de temps irréguliers. Le sous-marin étant sourd dans son baffle (cf. paragraphe 2.1), les abattées d'écoute lui permettent de vérifier qu'il n'est pas pisté. Le but est de quadriller le plus largement possible la zone, afin d'avoir le plus de chance de détecter un intrus.

La manœuvre pour effectuer une abattée d'écoute est la suivante (nous donnons des exemples de valeurs pour le temps et les angles) :
dans des intervalles de temps irréguliers (allant de 40 minutes à 1h30), le sous-marin doit faire une abattée d'écoute d'amplitude comprise entre 60 ° et 120 ° ou -60 ° et -120 ° , avec une vitesse faible et une immersion éventuellement constante. Le sous-marin continue ensuite éventuellement sur le cap de l'abattée d'écoute (cf. figure 2.6).

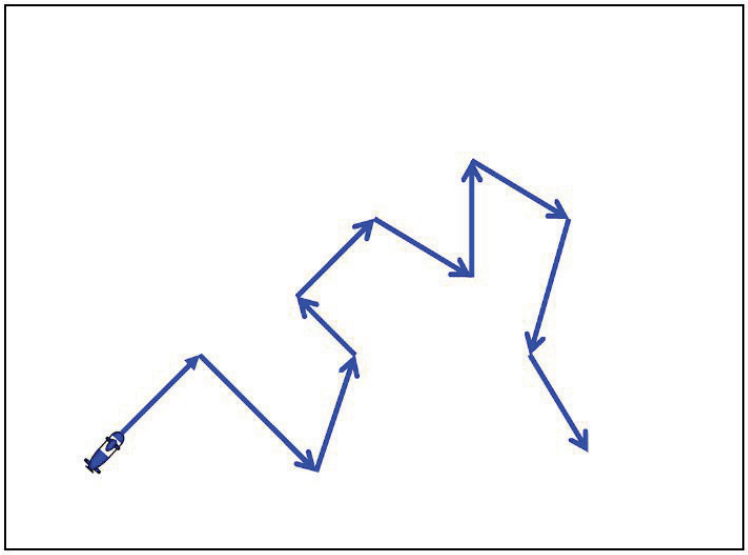

FIG. 2.6 – Trajectoire aléatoire d'un sous-marin dans son carré

Règle 2 : la "chasse"

Si un sous-marin détecte un autre sous-marin, il enclenche les actions suivantes :

Règle 2, phase 1 : l'anti-collision
Suivant les valeurs du défilement et du gisement dans lequel se trouve le sous-marin ennemi, le sous-marin met sa barre à droite ou à gauche, afin d'empêcher que la distance entre les deux sous-marins diminue. Un grand défilement indique que le contact est proche et un petit défilement indique que le contact est éloigné. Durant cette manœuvre, le sous-marin garde la même vitesse (vitesse faible, vitesse de patrouille).

Règle 2, phase 2 : manœuvre de mesure de distance
Cette manœuvre consiste à effectuer des tronçons en route et vitesse stables afin d'obtenir des défilements suffisants pour confirmer les informations de distance, cap et vitesse sur le but (c'est-à-dire le sous-marin adverse) et pour les affiner (cf figure 2.7). La durée des tronçons est d'environ 10 minutes et le sous-marin en effectue deux ou trois.
Durant cette phase, le sous-marin obtient des informations sur la position et

la vitesse du sous-marin ennemi, il sait donc si le sous-marin ennemi manœuvre ou non. Sur la figure 2.8, le sous-marin pisteur, au début du scénario, effectue une trajectoire aléatoire de recherche. Une fois que le sous-marin a détecté l'ennemi, la manœuvre de mesure de distance lui permet d'avoir une meilleure estimation de la position de l'ennemi.

Manœuvrer revient à changer de cap ou de vitesse. Si l'ennemi manœuvre, il y a un risque pour que notre sous-marin soit détecté. Dans ce cas là, il faut alors effectuer un dérobement(règle 3) en temps de crise, ou bien un lancement d'urgence (règle 4) en temps de guerre.

Si le sous-marin ennemi ne manœuvre pas, notre sous-marin peut alors rallier son poste de pistage (phase 3) puis pister (phase 4).

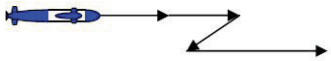

FIG. 2.7 – Manœuvre de mesure de distance

Règle 2, phase 3 : ralliement de la position de pistage
Durant cette phase, le sous-marin doit rallier le baffle du but sans être détecté et se placer à faible distance derrière lui (cf figure 2.9). En restant à cette distance, le pisteur ne sera pas détecté et le but n'aura pas le temps de réagir si le pisteur lui lance une torpille. Le sous-marin doit adopter une vitesse de chasse plus élevée que celle du futur pisté. Cependant, cette vitesse doit permettre de conserver l'écoute du but mais aussi de conserver la discrétion acoustique. De cette façon, le sous-marin ne se fera pas détecter par l'ennemi et pourra conserver sa détection.

Si durant cette phase, le sous-marin but manœuvre et tire une torpille, le sous-marin pisteur effectue un lancement d'urgence (règle 4). Si le sous-marin but manœuvre sans tirer, le sous-marin recommence les actions précédentes :

- phase 1 : manœuvre d'anti-collision,
- phase 3 : ralliement de la position de pistage.

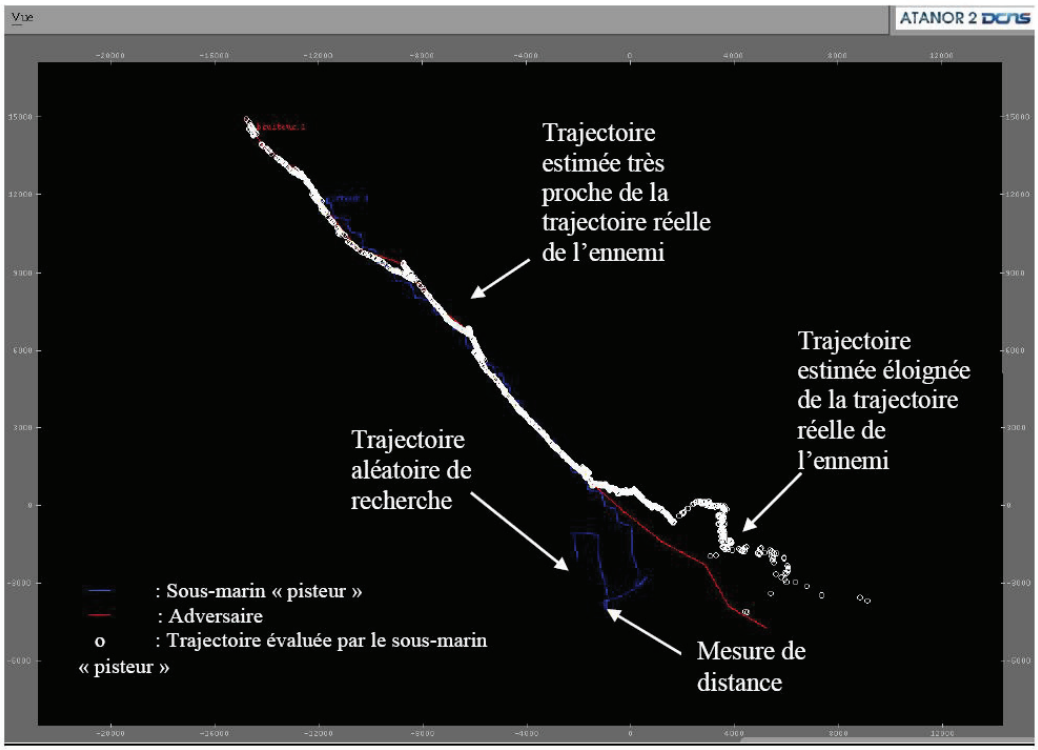

FIG. 2.8 – Affinement de la trajectoire estimée grâce à la manœuvre de mesure de distance
Lorsque le sous-marin pisteur détecte l'adversaire, il a une estimation de la trajectoire de l'adversaire éloignée de la trajectoire réelle. Une fois la manœuvre de mesure de distance effectuée, cette estimation est très proche de la trajectoire réelle de l'adversaire.

Règle 2, phase 4 : pistage du sous-marin ennemi
Le but de cette manœuvre est de contrôler la distance avec le but de façon à ce qu'elle reste constante : ne pas trop s'en rapprocher mais ne pas trop s'en éloigner pour ne pas perdre le contact. Pour contrôler la distance par l'intermédiaire de la mesure du défilement, le sous-marin pisteur effectue des tronçons azimétriques dans le baffle du sous-marin pisté (cf figure 2.10).
Si durant cette phase, le sous-marin but manœuvre mais ne se dérobe pas, il ne nous a, a priori, pas détecté. Le sous-marin doit alors recommencer la manœuvre d'anti-collision (phase 1) et le ralliement du nouveau poste de pistage (phase 3). Par contre, si le sous-marin but manœuvre et s'apprête à tirer une torpille, le sous-marin fait un lancement d'urgence (cf. phase 6).

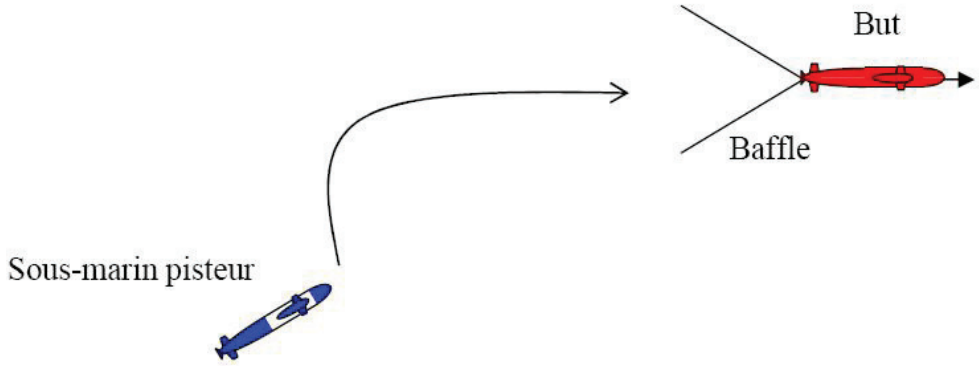

FIG. 2.9 – Ralliement du poste de pistage

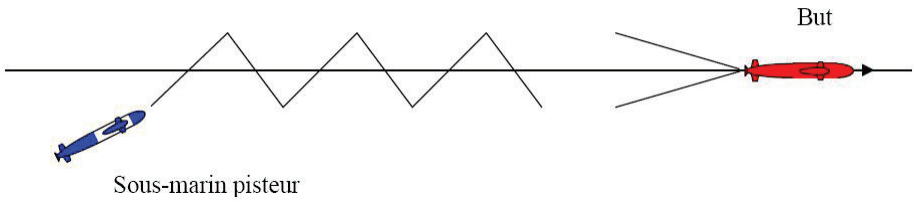

FIG. 2.10 – Manœuvre de pistage

Règle 3 : dérobement

Dans certains cas, le sous-marin doit dérober, c'est-à-dire s'éloigner de l'ennemi. Par exemple, dans les cas suivants :

- pendant la manœuvre de mesure de distance, le sous-marin ennemi manœuvre et nous sommes en temps de crise,
- les deux sous-marins sont trop proches.

La manœuvre de dérobement permet au sous-marin de placer l'ennemi dans un gisement arrière en augmentant sa vitesse. Il s'éloigne ainsi pendant plusieurs heures tout en contrôlant la détection du but.

Règle 4 : lancement d'urgence

Dans certains cas, le sous-marin doit effectuer un lancement d'urgence, c'est-à-dire lancer une torpille sur le but :

- pendant la manœuvre de mesure de distance, le sous-marin ennemi

manœuvre et nous sommes en temps de guerre,
- l'ennemi s'apprête à tirer une torpille.

Le sous-marin doit alors lancer une torpille. Pour cela, il doit avoir une torpille au tube et la porte avant ouverte. Il tire la torpille dans l'azimut de détection.

Une fois la torpille lancée dans l'azimut de détection, le sous-marin manœuvre de façon à dérober, en cassant la figure de lancement présumée du but, et il monte en allure à une vitesse maximale.

Règle 5 : perte de la détection

Si le sous-marin a perdu la détection depuis un certain temps, il doit chercher le but en ralliant sa dernière position connue. S'il ne l'a pas retrouvé au bout d'un certain temps, il le considère complètement perdu et reprend ses abattées (Règle 1).

Règle 6 : schnorchel

Si le sous-marin est un sous-marin diesel et que 4 heures se sont écoulées depuis la dernière recharge de batterie, alors il se met au schnorchel.

Pour se mettre au schnorchel, le sous-marin remonte à l'immersion périscopique (entre 13 et 15 mètres) puis sort son périscope. S'il ne voit rien, il sort le détecteur d'impulsions radar, appelé ARUR. S'il ne détecte rien non plus à l'ARUR, il sort son tube à air, et il recharge ses batteries. Le schnorchel permet au sous-marin d'alimenter ses moteurs diesels en air, sans avoir à faire surface.
S'il voit quelque chose (un bateau de guerre, un hélicoptère ...), il donne l'alerte, rentre tout (périscope, ARUR), et descend à l'immersion de sécurité en adoptant une vitesse d'au moins 10 nœuds.
En fin de charge, il donne l'alerte, rentre tout (ARUR, périscope), et descend à l'immersion opérationnelle à sa vitesse de patrouille (de 4 à 8 nœuds).

Règle 7 : évitement d'obstacle

A l'aide du sonar à évitement d'obstacle MOAS (Mine and Obstacle Avoidance System), le sous-marin peut détecter des mines, des gros rochers, ou encore des falaises. Le MOAS a une portée comprise entre 1500 mètres et 3000 mètres. Si le sous-marin détecte un obstacle, il change de cap de manière à l'éviter.

2.4 Illustration de quelques règles de comportements

La figure 2.11 permet d'illustrer quelques règles de comportements définies dans le paragraphe précédent. Cette figure est le résultat de notre modèle de comportement, interfacé avec ATANOR.
On note le sous-marin ennemi SM2 et le sous-marin pisteur SM1.
L'objectif du sous-marin SM2 est de traverser la zone, sans se faire détecter.
Le comportement du sous-marin SM1 se déroule de la façon suivante :

- Partie 1 : le sous-marin SM1 adopte une trajectoire aléatoire de recherche : avec ses changements de cap, il vérifie si il n'y a pas de sous-marin dans la zone.
- Partie 2 : une fois le sous-marin SM2 détecté, il effectue la manœuvre d'anti-collision.
- Partie 3 : le sous-marin SM1 effectue la manœuvre de mesure de distance, en faisant trois tronçons. Avant cette manœuvre, la trajectoire estimée du sous-marin SM2 par le sous-marin SM1 est éloignée de la trajectoire réelle. Une fois cette manœuvre effectuée, le sous-marin SM1 a une meilleure estimation de la trajectoire.
- Partie 4 : maintenant que le sous-marin SM1 connaît la position et la cinématique du sous-marin SM2, il rallie son poste de pistage en se rapprochant du baffle du sous-marin SM2.
- Partie 5 : le sous-marin SM1 peut finalement pister le sous-marin SM2 en effectuant des tronçons dans son baffle.

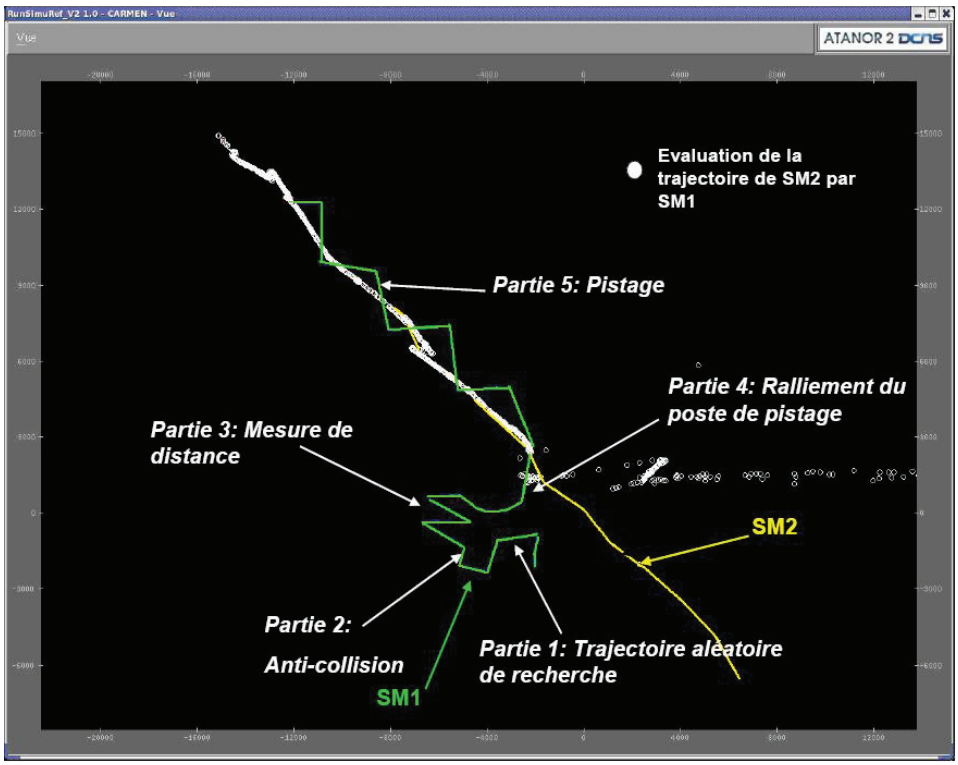

FIG. 2.11 – Illustration de quelques règles de comportements

Deuxième partie

Représentation des connaissances et modélisation du comportement

Chapitre 3

Etat de l'art

Cet état de l'art couvre plusieurs domaines. Les quatre premiers paragraphes portent sur la représentation des connaissances. Afin de décrire le comportement de l'opérateur, plusieurs pistes ont été envisagées. Tout d'abord, nous présentons de manière générale l'intelligence artificielle et la représentation des connaissances. Ensuite, nous présentons la logique classique, puis les logiques non monotones, et plus particulièrement la logique des défauts. Enfin, nous présentons les approches comportementales des systèmes multi-agents.

Enfin, nous nous sommes intéressés à la représentation du temps, que nous devons prendre en compte dans notre modélisation : en effet, l'opérateur possède des données sur l'environnement à un instant et il doit en déduire l'action à exécuter au temps suivant.

3.1 L'intelligence artificielle et la représentation des connaissances

3.1.1 L'intelligence artificielle

Le terme d'intelligence artificielle (IA) a vu le jour au cours d'un congrès organisé à Dartmouth en 1956, avec John McCarthy et Marvin Minsky. De nombreuses définitions ont été données pour décrire la notion d'IA.

- Marvin Minsky [75] définit l'IA comme la science de programmer les ordinateurs pour qu'ils réalisent des tâches qui nécessitent de l'intelligence lorsqu'elles sont réalisées par des êtres humains.
- Pour E. Feigenbaum, l'IA est la partie de l'informatique consacrée à la conception de systèmes informatiques intelligents.

- Selon Jean-Marc Alliot [2], l'IA a pour but de faire exécuter par l'ordinateur des tâches pour lesquelles l'homme, dans un contexte donné, est aujourd'hui meilleur que la machine.

En se limitant à l'informatique, l'IA peut être définie comme un ensemble de disciplines informatiques tendant à faire que des machines imitent des comportements humains.

Le test de Turing marque également le début de l'IA en 1950 [117]. Ce test est basé sur le jeu de l'imitation. Il consiste à savoir si une machine peut se faire passer pour un homme. Un individu communique à l'aide d'un terminal d'ordinateur avec un interlocuteur invisible. L'individu doit décider si l'interlocuteur est un être humain ou un système d'IA imitant un être humain. Turing donnait le test réussi s'il y avait 30% de succès.

Les domaines de l'IA sont nombreux. En voici quelques exemples :

- **reconnaissance d'images, analyse d'images et traitement de signaux.** Reconnaissance de formes, classification d'images, reconnaissance de textes ;
- **robotique.** Systèmes réactifs, robotique autonome, contrôle et planification de trajectoires ;
- **traitement du langage naturel.** Traduction automatique, reconnaissance de la parole, analyse automatique de textes ;
- **raisonnement automatique et acquisition de connaissances : systèmes experts, systèmes intelligents, apprentissage automatique, représentation des connaissances.** Système d'aide au diagnostic (médical, industriel), système d'aide à la décision, systèmes à bases de connaissances ;

  Un des premiers systèmes experts a été développé en 1976 par Shortliffe [105] : MYCIN, qui fait un diagnostic et propose une thérapeutique en médecine. Dans un système expert, il y a séparation entre :
  - les connaissances, c'est-à-dire la base de connaissance, sous forme de faits et de règles ;
  - le programme qui permet de les utiliser : le moteur d'inférence, mécanisme qui permet de résoudre le problème posé en se basant sur les faits de départ et en utilisant les règles d'inférences.
- **IA distribuée et systèmes multi-agents :** définition de comportements intelligents qui sont le produit de l'activité coopérative de plusieurs agents ;
- **logique formelle :** logique pour l'IA, langage d'IA (Prolog, ...), démonstration automatique de théorèmes.

Nous nous intéressons maintenant au domaine de la représentation des connaissances.

3.1.2 La représentation des connaissances

La représentation des connaissances est un des secteurs les plus importants de la recherche en intelligence artificielle. Afin de pouvoir manipuler les connaissances, il faut savoir les représenter symboliquement. On peut représenter les connaissances sur un sujet particulier de différentes façons. Pour cela, plusieurs formalismes peuvent être utilisés, comme le calcul propositionnel, la logique du premier ordre (cf. paragraphe 3.2) ou encore les logiques non-monotones (cf. paragraphe 3.3). La connaissance peut aussi être représentée par d'autres moyens :

- les frames, qui sont des assemblages d'objets qui définissent un contexte. Les frames ont été proposés par Marvin Minsky [76] pour représenter initialement des concepts visuels ;
- les scripts, qui décrivent une séquence d'évènements ;
- les réseaux sémantiques, qui sont des réseaux de concepts liés par des associations, proposés par Quillian [87].

Le but est de représenter les connaissances de façon à pouvoir les utiliser le plus efficacement par la suite. Une fois les connaissances représentées, elles peuvent être manipulées et exploitées par des outils informatiques.

Avant de représenter les connaissances, le chercheur doit commencer par acquérir ces connaissances. Les connaissances nécessaires pour décrire un domaine spécifique peuvent être réunies de différentes façons. On peut interroger des experts et en déduire ces connaissances, ou bien l'expert peut expliquer directement ces connaissances. On peut également extraire les connaissances à partir de textes ou de manuels. La psychologie peut aussi servir à étudier et recueillir les modes des raisonnements des individus.

3.2 Représentation des connaissances en logique "classique"

Dans cette thèse, nous appelons logique classique le calcul propositionnel et la logique du premier ordre, qui définissent les bases de toutes les logiques. Une logique classique est un système formel qui se compose d'un langage, qui va permettre de décrire l'information, d'une grammaire et des techniques de déductions (pour déduire de l'information).

Il existe deux aspects pour étudier un système et déduire de l'information :

- l'aspect syntaxique : il revient à définir le système formel et à déduire l'ensemble des formules qui sont des théorèmes. Pour cela, des axiomes et des règles d'inférence sont utilisés ;

– l'aspect sémantique : il considère l'interprétation des formules et déduit les formules qui sont sémantiquement vraies.

En logique classique, ces deux aspects sont équivalents : ce qui est déductible de manière sémantique l'est de manière syntaxique.

3.2.1 Le calcul propositionnel

Le calcul propositionnel est le plus simple des langages logiques. Il permet de représenter des connaissances et de déduire des informations à partir de ces connaissances (existence de solutions, ou non, pour le problème représenté, réponse à une question se rapportant au problème ...). Le lecteur pourra se référer aux ouvrages suivants : [110], [29].

Vocabulaire

Le vocabulaire du calcul propositionnel est formé :
– d'un ensemble de variables propositionnelles (X,Y,...), appelées propositions par la suite,
– de symboles de ponctuations (les parenthèses ouvrantes et fermantes),
– de 5 connecteurs logiques (cf tableau 3.1).

NOM DU CONNECTEUR	SYMBOLE USUEL
négation	$\neg$
conjonction	$\wedge$
disjonction	$\vee$
implication	$\rightarrow$
équivalence	$\leftrightarrow$

TAB. 3.1 – Les 5 connecteurs logiques

Syntaxe

Une formule est une suite d'éléments du vocabulaire, répondant à des règles syntaxiques. Cette syntaxe est définie de manière inductive : à partir des formules de base (les propositions), d'autres formules sont construites par assemblages de formules déjà construites à l'aide de connecteurs.

DÉFINITION 1 : Formule
L'ensemble des formules est l'ensemble tel que :
– toute variable propositionnelle est une formule ;

- si X et Y sont des formules, alors ¬X, (X ∧ Y), (X ∨ Y), (X → Y), (X ↔ Y) sont toutes des formules (règle d'induction) ;
- une formule s'obtient uniquement avec les règles de base et d'induction.

Exemple 1 : **Formules**

$$(p \wedge (q \vee r)), ((p \vee r) \wedge (p \rightarrow \neg(r \leftrightarrow \neg t))), p, \neg\neg q$$

Définition 2 : **Littéral**
Un littéral est une variable propositionnelle ou sa négation. Si X est une proposition, on peut donc parler du littéral positif X et du littéral négatif ¬X.

La syntaxe du calcul propositionnel revient à définir un système formel dans lequel les déductions qui peuvent être faites conduisent à des théorèmes. Pour cela, nous utilisons des axiomes, c'est-à-dire des propriétés évidentes admises sans démonstration. Différents ensembles d'axiomes ont été proposés, dont le système de Lukasiewicz, basé sur trois axiomes suivants :

- $p \rightarrow (q \rightarrow p)$;
- $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$;
- $(\neg p \rightarrow \neg q) \rightarrow (q \rightarrow p)$;

avec p, q et r des formules.

La règle d'inférence, appelée *Modus Ponens*, est définie de la façon suivante :

$$\frac{\vdash p, \vdash p \rightarrow q}{\vdash q}$$

Sémantique

La sémantique va permettre de donner une valeur de vérité VRAI ou FAUX à toute formule dans un certain contexte. Elle permet de faire de la déduction : à partir d'un ensemble d'information, on implique d'autres informations.
Elle dira également si une formule est toujours vraie (une tautologie), ou encore si une formule est toujours fausse (inconsistante).
Pour cela, la base est l'interprétation : c'est une application qui donne à toute formule une valeur de vérité VRAI ou FAUX. On notera la valeur de vérité FAUX par 0 et VRAI par 1.

Définition 3 : **Interprétation**
Soit F l'ensemble des formules propositionnelles. Une interprétation est une

application $I : F \rightarrow \{0,1\}$, définie de la façon suivante, avec X et Y des formules :

1. si X est une proposition, $I[X]$ appartient à $\{0,1\}$;
2. $I[\neg X] = 1 - I[X]$
   (inversion de la valeur de vérité) ;
3. $I[X \vee Y] = max\{I(X), I(Y)\}$
   ($X \vee Y$ est vraie si X est vrai ou Y est vrai, fausse sinon) ;
4. $I[X \wedge Y] = min\{I[X], I[Y]\}$
   (vrai si et seulement si X est vrai et Y vrai) ;
5. $I[X \rightarrow Y] = max\{1 - I[X], I[Y]\}$
   (faux si et seulement si X est vrai et Y est faux, vrai dans les trois autres cas) ;
6. $I[X \leftrightarrow Y] = 1$ si et seulement si $I[X] = I[Y]$;
7. si nous introduisons les formules Vrai et Faux dans le langage, nous avons, pour tout I, $I[Vrai] = 1$ et $I[Faux] = 0$.

Pour représenter les règles 2 à 6, les tables de vérité des connecteurs (cf tableaux 3.2 et 3.3) peuvent être utilisées.

X	¬ X
0	1
1	0

TAB. 3.2 – Table de vérité du connecteur non

X	Y	X ∧ Y	X V Y	X → Y	X↔Y
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

TAB. 3.3 – Table de vérité des connecteurs logiques

Remarque : la formule (X → Y) a la même sémantique que la formule (¬X ∨ Y). Pour la logique, ces formules sont équivalentes.

DÉFINITION 4 : Modèles, consistance
Soit X une formule :

1. un modèle de X est une interprétation I telle que $I[X] = 1$. On dit aussi que I satisfait X ;
2. si X n'a aucun modèle, X est inconsistante ($I[X] = 0$ pour toute interprétation I). On dit aussi que X est incohérente, ou contradictoire, ou insatisfaisable) ;
3. si X a au moins un modèle, X est consistante (il existe une interprétation I telle que $I[X] = 1$). On dit aussi cohérente, ou non-contradictoire ou satisfaisable) ;
4. si toutes les interprétations du langage sont des modèles de X, alors X est une tautologie (une formule universellement valide) ;
5. une formule qui n'est ni inconsistante, ni une tautologie, est dite contingente (X est vrai dans certaines interprétations, faux dans d'autres).

Déduction

Le problème de déduction consiste à déterminer si une formule est une conséquence logique d'une autre formule. Soit E un ensemble de formules représentant un problème donné, si F est une formule logique, poser la question "Est-ce que F ?", c'est demander exactement : "dans le cadre du problème exprimé par E, est-ce que F est vraie ?". Cette question signifie : est-ce que $E \models F$?

Théorèmes de base :
Si A et B sont des formules, alors : $A \models B$ si et seulement si A $\wedge \neg B$ est inconsistant.
Si A est un ensemble fini ($A = \{F1, F2, F3, \ldots, Fi\}$) de formules, B, une formule, alors $A \models B$ si et seulement si $\{F1, F2, F3, \ldots, Fi, \neg B\}$ est inconsistant.
Cette règle est généralisable au cas où A est un ensemble infini de formules.

Les clauses

Parmi les formules, des formules très utiles sont utilisées : les clauses.
Une clause représente intuitivement une règle du type

(f1 et f2 et ... fk) implique (g1 ou g2 ou ... ou gk),

où les fi et gi sont des informations élémentaires, c'est-à-dire soit des variables propositionnelles, soit leurs négations.

Définition 5 : Clause

Une clause est la disjonction d'un nombre fini de littéraux, c'est-à-dire une formule du type $(I1 \vee I2 \vee I3 \vee \ldots \vee In)$, où chaque Ii est un littéral.

Une clause positive n'a que des littéraux positifs et une clause négative n'a que des littéraux négatifs. Une clause unaire est une clause avec un littéral, une clause binaire est une clause avec deux littéraux ...

La définition formelle d'une clause se traduit en logique par la formule :

$$(f1 \wedge f2 \wedge f3 \wedge \ldots \wedge fk) \rightarrow (g1 \vee g2 \vee g3 \vee \ldots \vee gk)$$

Comme la formule (X $\rightarrow$ Y) a la même sémantique que $(\neg X \vee Y)$, la formule de la clause s'écrit également :

$$\neg(f1 \wedge f2 \wedge f3 \wedge \ldots \wedge fk) \vee (g1 \vee g2 \vee g3 \vee \ldots \vee gk)$$

ou encore :

$$\neg f1 \vee \neg f2 \vee \neg f3 \vee \ldots \vee \neg fk \vee g1 \vee g2 \vee g3 \vee \ldots \vee gk$$

Définition 6 : Clause de Horn

- Une clause de Horn est une clause avec au plus un littéral positif (une clause de Horn a zéro ou un littéral positif). Les clauses négatives sont des clauses de Horn.
- Une clause de Horn avec un seul littéral positif s'écrit :

  $$g \vee \neg f1 \vee \neg f2 \vee \neg f3 \vee \ldots \vee \neg fk.$$

  ou encore :

  $$(f1 \wedge f2 \wedge f3 \wedge \ldots \wedge fk) \rightarrow g$$

  où les fi et g sont des littéraux positifs.
- Une clause de Horn ne contenant aucun littéral positif (clause négative) s'écrit :

  $$\neg f1 \vee \neg f2 \vee \neg f3 \vee \ldots \vee \neg fk$$

  ou encore

  $$(f1 \wedge f2 \wedge f3 \wedge \ldots \wedge fk) \rightarrow FAUX$$

  ou encore

  $$\neg(f1 \wedge f2 \wedge f3 \wedge \ldots \wedge fk).$$

Les clauses de Horn en calcul propositionnel ont l'avantage d'avoir une faible complexité, quasi-linéaire. Lorsque l'on utilise d'autres clauses que les clauses de Horn, la complexité augmente de façon très importante (jusqu'à une complexité exponentielle). De plus, les clauses de Horn sont intuitives et faciles à comprendre. Enfin, elles ont l'avantage de donner des algorithmes qui fonctionnent très bien, par exemple, implémentés en Prolog.

3.2.2 La logique du premier ordre

Dès que l'on veut manipuler des propriétés générales un peu compliquées, ou des relations entre des objets, on est conduit à utiliser des énoncés dont la valeur de vérité dépend de variables, comme : "x est un diviseur de y". De tels énoncés se représentent par des prédicats et leur théorie, qui généralise le calcul propositionnel, s'appelle logique du premier ordre ou encore calcul des prédicats (du premier ordre).

En pratique, le calcul des prédicats permet de décrire (à l'aide de formules logiques) des relations portant sur les objets d'un certain domaine, appelé domaine d'individu. Ce domaine d'individu est soit un ensemble fini d'objets (par exemple un ensemble fini d'entiers), soit un ensemble infini dénombrable (en général, l'ensemble des entiers naturels $\mathbb{N}$ est considéré).

Vocabulaire

Le vocabulaire de la logique du premier ordre est composé de :
- connecteurs logiques du calcul propositionnel ($\neg, \vee, \wedge, \rightarrow, \leftrightarrow$),
- symboles de ponctuation : (,)
- quantificateurs $\forall$ (pour tout), et $\exists$ (il existe),
- variables, généralement représentées par les lettres minuscules de la fin de l'alphabet : x , y , z , ..., éventuellement indicées. On appelle V l'ensemble des variables.
- constantes, généralement représentées par les lettres minuscules du début de l'alphabet : a, b , c , ..., éventuellement indicées. On appelle C l'ensemble des constantes.
- symboles prédicatifs, généralement représentés par les lettres minuscules de l'alphabet, comme p, q, r , ..., éventuellement indicées. On appelle P_j l'ensemble des constantes prédicatives à j arguments.
- symboles fonctionnels, généralement représentés par les lettres minuscules du milieu de l'alphabet : f, g, h , ..., éventuellement indicées. On appelle F_j l'ensemble des constantes fonctionnelles à j arguments.

Définition 7 : Prédicat

Les prédicats sont constitués d'un symbole prédicatif, suivi d'un certain nombre d'arguments (entre parenthèses, et séparés par des virgules). Par exemple : inf(x,y).

L'arité est le nombre d'arguments d'un prédicat. Un prédicat d'arité 0 correspond en fait à une proposition : le calcul des prédicats contient tout le calcul propositionnel.

Définition 8 : Fonction
Les fonctions (ou termes fonctionnels) sont constitués d'un symbole fonctionnel, suivi d'un certain nombre d'arguments (entre parenthèses, et séparés par des virgules).
Il y a les fonctions mathématiques, comme les fonctions PLUS et MOINS : PLUS(MOINS(3,2),5) signifie 3-2+5.

Définition 9 : Terme
Les arguments d'un prédicat ou d'une fonction sont des termes. Les termes peuvent être soit des variables, soit des constantes, soit des fonctions. Un terme est censé représenter un objet d'un ensemble particulier, appelé domaine d'individu. En fait, une variable représente n'importe quel objet du domaine, une constante représente un objet du domaine, et une fonction représente un objet du domaine pour chaque valeur possible des variables apparaissant dans la fonction. Ainsi, une constante individuelle est analogue à une fonction sans argument.

Syntaxe : termes, atomes, variables libres et liées

Définition 10 : Terme
L'ensemble des termes, noté Terme, est défini de la façon suivante :
- toute variable est un terme,
- toute constante est un terme,
- toute fonction est un terme : $\forall t_1, t_2, \ldots t_j \in$ Terme, et $\forall$ f $\in F_j$, alors : $f(t_1, t_2, \ldots t_j) \in$ Terme.

Exemple 2 : Termes
$x, a, f(a), g(a, x, f(y, a)), f(f(f(x_1, x_2), x_3), x_4)$
Si Toto est une constante, soient PERE et FRERE des symboles fonctionnels, alors PERE(Toto) est un terme et FRERE(PERE(Toto)) est également un terme.

Définition 11 : Prédicat
Un prédicat d'arité n est défini par un symbole prédicatif R et de n termes : $t_1, t_2, \ldots t_n$, noté R($t_1, t_2, \ldots t_n$).

Remarque sur la différence entre un terme et un prédicat :
Si $f(x_1, x_2, x_3, x_4)$ est un terme, alors $f(f(x_1, x_2, x_3, x_4))$ est un terme.
Alors que si $R(x_1, x_2, x_3, x_4)$ est un prédicat, nous ne pouvons pas déduire que $R(R(x_1, x_2, x_3, x_4))$ est un prédicat.

Définition 12 : Atome

L'ensemble des atomes, noté Atome, est le plus petit ensemble tel que :

- $\forall t_1, t_2, \ldots t_j \in$ Terme, $\forall$ p $\in P_j$, alors : $p(t_1, t_2, \ldots t_j) \in$ Atome (un atome est un prédicat ou une égalité).

Une atome est aussi parfois appelé une formule atomique.

Exemple 3 : Atomes

$$p; q(x, y); r(f(x), g(a, a), c); x = a; f(a) = g(x, b)$$

Définition 13 : Formules bien formées

L'ensemble des formules bien formées du calcul des prédicats du premier ordre, noté F_{p_r}, est le plus petit ensemble tel que :

- $\forall p \in Atome, p \in F_{p_r}$ (tout atome est une formule),
- si $A \in F_{p_r}$ et $B \in F_{p_r}$ et $x \in V$, alors
  $(A) \in F_{p_r}$,
  $\neg A \in F_{p_r}$,
  $A \vee B \in F_{p_r}$,
  $A \wedge B \in F_{p_r}$,
  $A \rightarrow B \in F_{p_r}$,
  $A \leftrightarrow B \in F_{p_r}$,
  $\forall x(A) \in F_{p_r}$,
  $\exists x(A) \in F_{p_r}$.

Exemple 4 : Formules

$$\forall x(p(x, y) \rightarrow (q \wedge r(f(a))))$$

$$\neg p(z) \vee \exists y r(y)$$

Définition 14 : Ensemble des variables d'une formule A

L'ensemble des variables d'une formule A est l'ensemble des éléments de V qui apparaissent dans A. On note var(A) cet ensemble.

- L'ensemble des variables liées d'une formule A, noté varliée(A), est défini par induction de la façon suivante :
  - si A $\in$ Atome, varliée$(A) = \{\}$,
  - si A est de la forme : $\neg B$, (B), alors varliée(A) = varliée(B),
  - si A est de la forme : $B \vee C, B \wedge C, B \rightarrow C$ ou $B \leftrightarrow C$, alors : varliée(A) = varliée (B) $\cup$ varliée (C),

  - si A est de la forme : $\forall x(B), \exists x(B)$, alors :
    varliée(A)=varliée(B) $\cup$ x.
- L'ensemble des variables libres d'une formule A, noté varlib(A), est défini par induction de la façon suivante :
  - si A $\in$ Atome, varlib(A) = var(A),
  - si A est de la forme : $\neg B$, (B), alors varlib(A) = varlib(B),
  - si A est de la forme : $B \vee C, B \wedge C, B \rightarrow C, B \leftrightarrow C$, alors :
    varlib(A) = varlib (B) $\cup$ varlib (C),
  - si A est de la forme : $\forall x(B), \exists x(B)$, alors :
    varlib(A)=varlib(B) - {x}.

  Une formule A sans variable libre (tel que varlib(A) = {}) est appelée une formule close, ou une formule fermée.

Définition 15 : La portée

La portée d'une quantification est la formule à laquelle cette quantification s'applique. L'occurrence de la variable x apparaissant dans la quantification est dite quantifiée. Ainsi, toute occurrence de la variable x dans la portée de cette quantification est liée. Une occurrence de la variable x est libre si elle n'est ni quantifiée, ni liée. On peut aussi remarquer qu'une variable ayant une occurrence liée dans une formule A a aussi forcément une occurrence quantifiée dans cette formule A.

Exemple 5 : La portée

A = p(f(x,y)) $\vee \forall$ z r(a,z)
var(A) = $\{x, y, z\}$
varliée(A)= $\{z\}$
varlib(A)= $\{x, y\}$
B= $\forall x$ p(x,y,z) $\vee \forall z$ (q(z) $\rightarrow$ r(z))
var(B) = $\{x, y, z\}$
varliée(B)= $\{x, z\}$
varlib(B)= $\{y, z\}$

Sémantique

Les formules closes du calcul des prédicats, comme les formules propositionnelles, sont susceptibles d'être interprétées, c'est-à-dire de recevoir une valeur de vérité. Cependant, les composants d'une formule du calcul des prédicats ne sont pas seulement des sous-formules, mais aussi des termes. Il faut donc également construire une interprétation pour les termes. Chaque terme sera interprété par un objet, parmi un ensemble d'objets donné au

départ : le domaine.

Définition 16 : Domaine

On définit un ensemble non vide d'objets D, appelé domaine, ou domaine d'individu, ou domaine d'interprétation, ou base de l'interprétation.

Définition 17 : Interprétation

Une interprétation I est une application qui associe :

- à toute constante prédicative p à n places ($\forall p \in P_n$) une fonction $I(p) : D^n \rightarrow 0, 1$.
  Cas particulier : si p est une proposition, c'est-à-dire si $p \in P_0$, I(p) = 0 ou I(p) = 1.

- à toute constante fonctionnelle f à m places ($\forall f \in F_m$) une fonction $I(f) : D^m \rightarrow D$.
  Cas particulier : si f est une constante individuelle, c'est-à-dire si $f \in F_0$, $I(f) = a$, $a \in D$.

On généralise, par induction, une interprétation aux termes, aux atomes, et aux formules de la façon suivante :

- Généralisation aux termes :
  si f est une constante fonctionnelle à m places ($f \in F_m$), et si $t_1, t_2, \ldots t_m$ sont des termes, alors $I(f(t_1, t_2, \ldots, t_m)) = I(f)(I(t_1), I(t_2), \ldots, I(t_m))$.
- Généralisation aux atomes :
  si p est une constante prédicative à n places ($p \in P_n$), et si $t_1, t_2, \ldots t_n$ sont des termes, alors $I(p(t_1, t_2, \ldots, t_n)) = I(p)(I(t_1), I(t_2), \ldots, I(t_n))$.
  Si s et t sont des termes, alors I(s = t) = 1 si I(s) = I(t), et I(s = t) = 0 sinon.
- Généralisation aux formules :
  Si A et B sont des formules, alors $\neg A, A \vee B, A \wedge B, A \rightarrow B, A \leftrightarrow B$, s'interprètent comme dans le calcul des propositions.
  Si A est une formule, et x une variable, alors $I(\forall x(A)) = 1$ si et seulement si $\forall d \in D, I_{x/d}(A) = 1$.
  Si A est une formule, et x une variable, alors $I(\exists x(A)) = 1$ si et seulement si $\exists d \in D, I_{x/d}(A) = 1$ (où $I_{x/d}$ est l'interprétation égale partout à I, sauf en x, et telle que I(x) = d).

Ces généralisations permettent d'associer à toute formule close A une valeur de vérité I(A) = 0 ou 1, et à tout terme t un élément $I(t) = d \in D$. On peut remarquer aussi que l'interprétation d'une sous-formule contenant i variables libres est égale à une fonction de $D^i \rightarrow \{0, 1\}$.

Définition : Comme dans le calcul propositionnel, les formules closes du calcul des prédicats se répartissant en trois classes :

- les formules valides (ou tautologies) qui sont vraies pour toute interprétation,
- les formules inconsistantes qui sont fausses pour toute interprétation,
- les formules contingentes qui sont vraies pour certaines interprétations, mais pas pour d'autres.

De même, la notion de conséquence logique est la même qu'en calcul propositionnel : $A \models B$ si et seulement si tout modèle de A est un modèle de B.

3.2.3 Les limites de la logique classique

Les logiques classiques, comme la logique mathématique ou encore le calcul propositionnel, sont des logiques monotones. En logique monotone, tout ce qui est déductible à partir d'un ensemble d'information sera toujours vrai après un ajout de nouvelles informations. Cette logique est appelée monotone, dans le sens où le nombre d'énoncés que nous savons vrais est strictement croissant : un savoir ajouté ne peut jamais être remis en cause avec l'arrivée de nouvelles informations.

La logique classique ne permet pas de raisonner lorsque les informations sont incomplètes.

Prenons la règle : "Généralement, les oiseaux volent". Cette règle exprime une information incomplète, car elle ne précise pas à quel type d'oiseaux elle s'applique. C'est une règle générale tolérant des exceptions. A première vue, ce type d'information peut être exprimée en logique du premier ordre. "Généralement, les oiseaux volent" se traduira alors par "Tous les oiseaux volent" : $\forall x, Oiseau(x) \rightarrow Vole(x)$.
Cette formulation est cohérente si la seule information connue est "Titi est un oiseau". Cette règle possède des exceptions, certains oiseaux ne volent pas, comme les manchots. Il faut alors modifier la règle :
$\forall x, Oiseau(x) \wedge \neg Manchot(x) \rightarrow Vole(x)$.
Cette formulation possède des inconvénients. Il est difficile de vouloir gérer des règles générales contenant un nombre important d'exceptions. De plus, il est impossible de déduire avec cette règle qu'un oiseau vole si l'on n'établit pas que ce n'est pas un manchot.

La logique classique ne permet pas la révision des informations.

Elle ne prévoit pas de remettre en cause des déductions établies auparavant. Prenons l'énoncé suivant :

- "Généralement, les oiseaux volent."
- "Les manchots ne volent pas."
- "Titi est un oiseau."

On peut exprimer cet énoncé en logique du premier ordre :

- $\forall x, Oiseau(x) \rightarrow Vole(x)$
- $\forall x, Manchot(x) \rightarrow \neg Vole(x)$
- $Oiseau(Titi)$

Sachant que les oiseaux volent et que Titi est un oiseau, on peut déduire que "Titi vole".
Or, si l'on sait que Titi est un manchot, on conclut "Titi ne vole pas".
On obtient alors deux conclusions inconsistantes. La logique classique ne permet donc pas le raisonnement révisable qui peut remettre en cause des conclusions. Ce type de raisonnement est courant en intelligence artificielle, ainsi que dans la vie de tous les jours.
Prenons un autre exemple, avec la règle générale "Généralement, un officier doit assurer la sauvegarde de son sous-marin". Si nous ajoutons l'information que l'officier est kamikaze, nous ne pourrons pas déduire qu'il assurera cette sauvegarde.

La logique classique ne nous permet donc pas de raisonner sur des informations incomplètes et révisables. Or, nous voulons modéliser le comportement d'un officier de quart dans un sous-marin. Les données disponibles dans le sous-marin proviennent des sonars passifs, elles sont incomplètes, incertaines et révisables(cf. exigence 1, paragraphe 1.3, page 27). La logique classique ne nous convient donc pas pour modéliser le comportement de l'officier de quart.

3.3 Représentation des connaissances en logique non-monotone

La logique non monotone permet de prendre en compte des informations incomplètes, révisables et incertaines. Elle permet de réviser les informations et les conclusions précédemment établies avec l'arrivée de nouvelles connaissances. Elle présente une similitude naturelle avec le raisonnement humain : par manque d'information ou manque de temps, une personne peut raisonner avec des connaissances partielles et réviser les conclusions au besoin

lorsqu'elle a plus d'informations. Cette logique est appelée non monotone car l'ensemble des conclusions que l'on fait à partir d'une base de connaissance ne va pas croître si la base de connaissance croît, contrairement à la logique classique.

Les premiers travaux en logique non-monotone datent de la fin des années 70 avec les travaux de McCarthy, Mc Dermott & Doyle et Reiter. En 1978, Reiter [89] formalise un premier mode de raisonnement non-monotone avec l'hypothèse du monde clos. McCarthy [72] étend ensuite l'hypothèse du monde clos avec la circonscription en 1980. En 1980, Mc Dermott et Doyle [74] ont proposé une logique modale non-monotone. La même année, Reiter propose la logique des défauts [90]. En 1985, Moore introduit la logique autoépistémique [77].

Citons d'autres logiques non monotones : les logiques de l'incertain comme la logique possibiliste, proposée par Dubois et Prade [35] en 1987, ou encore la logique floue, proposée par Zadeh en 1965 [122].

Pour la formalisation des règles de comportements, nous avons choisi d'utiliser la logique des défauts, une des premières logiques non monotones. Introduite par Ray Reiter en 1980 [90] comme un formalisme de raisonnement en l'absence d'informations, elle est aujourd'hui la logique non monotone la plus utilisée.

3.3.1 La logique des défauts [90]

La logique des défauts formalise le raisonnement par défaut. Elle permet de traiter les règles admettant des exceptions sans être obligé de remettre en cause les règles précédemment établies à chaque fois qu'une nouvelle exception apparaît.

Une théorie des défauts est composée d'un ensemble de faits W, qui sont des formules issues du calcul propositionnel ou bien de la logique du premier ordre et d'un ensemble de défauts D, qui sont des règles d'inférence à contenu spécifique. Les défauts permettent de gérer les informations incomplètes.

Définition 18 : **Défaut**

De sa forme la plus simple, un défaut est une expression de la forme

$$\frac{A(X) : B(X)}{C(X)}$$

où A(X), B(X) et C(X) sont des formules bien formées du premier ordre qui contiennent X comme variable libre ou $X = (x_1, x_2, x_3, \ldots, x_n)$ comme vecteur de variables libres. A(X) est le prérequis, B(X) est la justification et C(X) est le conséquent.

Le défaut $\frac{A(X) : B(X)}{C(X)}$ signifie informellement : si A(X) est vérifié, si il est possible que B(X) soit vrai (B(X) est consistant), alors on infère C(X).

De manière générale, les défauts peuvent être définis de la façon suivante :

$$\frac{A(X) : B_1(X), \ldots, B_n(X)}{C(X)},$$

où A(X), $B_1(X), \ldots, B_n(X)$ et C(X) sont des formules bien formées du premier ordre qui contiennent X comme variable libre ou $X = (x_1, x_2, x_3, \ldots, x_n)$ comme vecteur de variables libres.

Pour définir la règle "Tous les oiseaux volent, sauf les manchots", nous pouvons définir le défaut

$$\frac{oiseau(x) : \neg manchot(x)}{vole(x)}$$

qui signifie "Si x est un oiseau et que rien ne prouve que x est un manchot, alors x vole".

Si B(X)=C(X), le défaut est dit normal (la justification et le conséquent sont identiques). Les défauts normaux peuvent signifier "Les A sont des B, sauf exceptions", "Normalement, les A sont B".
Le défaut

$$\frac{oiseau(x) : vole(x)}{vole(x)}$$

signifie "Sauf exception, si x est un oiseau, si il n'est pas contradictoire que x vole, alors x vole.". La variable x peut être remplacée par toutes les valeurs possibles, constante ou termes.

Il existe également d'autres types de défauts, comme les défauts semi-normaux de la forme :

$$\frac{A(X) : B(X) \wedge C(X)}{C(X)}.$$

La justification implique le conséquent.

Les défauts sans prérequis sont de la forme :

$$\frac{: B(X)}{C(X)}.$$

Les défauts normaux sans prérequis, de la forme

$$\frac{: C(X)}{C(X)}$$

sont des défauts supernormaux.

Définition 19 : Extension
L'utilisation des défauts augmente le nombre de formules déduites de la base de connaissance W : nous obtenons des extensions qui sont des ensembles de théorèmes dérivables de façon monotone.

Une extension de la théorie des défauts $\Delta = (D, W)$ est un ensemble E de formules, clos pour la déduction, contenant W et vérifiant la propriété suivante : si d est un défaut de D dont le prérequis est dans E, sans que la négation de sa justification soit dans E, alors le conséquent de d est dans E. Formellement, les extensions sont définies de la façon suivante :

E est une extension de Δ si et seulement si $E = \cup_{i=0,\infty} E_i$, avec

$$E_0 = W \text{ et pour } i \geq O,$$
$$E_{i+1} = Th(E_i) \cup \{C/(\frac{A : B}{C}) \in D, A \in E_i, \neg B \notin E\}$$

où $Th(E_i)$ désigne l'ensemble des théorèmes obtenus de façon monotone à partir de E_i : $Th(E_i) = \{w/E_i \vdash w\}$.

Il est important de noter que E apparaît dans la définition de E_{i+1}. Il n'est donc pas possible de construire E, puisqu'il faut déjà connaître E pour obtenir les E_i.

Dans le cas des défauts normaux, l'extension est définie de la façon suivante :

E est une extension de Δ si et seulement si $E = \cup_{i=0,\infty} E_i$, avec

$$E_0 = W \text{ et pour } i \geq O,$$
$$E_{i+1} = Th(E_i) \cup \{C/(\frac{A:C}{C}) \in D, A \in E_i, \neg C \notin E_i\}$$

où $Th(E_i)$ désigne l'ensemble des théorèmes obtenus de façon monotone à partir de E_i : $Th(E_i) = \{w/E_i \vdash w\}$.

Cette fois, il faut vérifier que la négation de la justification n'appartient pas à E_i. Un algorithme incrémental peut donc être utilisé pour calculer E. Selon Reiter [90], lorsque tous les défauts sont normaux, l'existence d'au moins une extension est assurée.

Exemple 6 : Calcul d'extension
Prenons maintenant un exemple. Soit la théorie des défauts Δ=(D ,W) avec :
$W = \{\forall x, Manchot(x) \rightarrow Oiseau(x), Manchot(Titi)\}$
$D = \left\{d_1 : \frac{Oiseau(x) : Vole(x)}{Vole(x)}, d_2 : \frac{Manchot(x) : \neg Vole(x)}{\neg Vole(x)}\right\}$

La base des faits W signifie que les manchots sont des oiseaux et que Titi est un manchot.
On peut traduire les deux défauts par :
d_1 : "Si x est un oiseau et qu'il est consistant de supposer que x peut voler, alors on conclue que x peut voler." ou encore "Sauf exception, les oiseaux volent."
d_2 : "Sauf exception, les manchots ne volent pas."

On calcule les extensions :
- Cas 1 : on applique le défaut d_1 à W, on a alors :
  $E_0 = \{Manchot(Titi) \rightarrow Oiseau(Titi), Manchot(Titi), Oiseau(Titi)\}$

  $E_1 = Th(E_0) \cup \{Vole(Titi) \ tel \ que \ \frac{Oiseau(Titi) : Vole(Titi)}{Vole(Titi)} \in D,$
  $Oiseau(Titi) \in E_0, \neg Vole(Titi) \notin E_0\}$

  Le défaut d_1 peut donc bien être appliqué et Vole(Titi) peut donc être ajouté dans l'extension. Une fois ce premier défaut appliqué, il n'est pas

possible d'appliquer le défaut d_2 puisque le justification de ce défaut ne serait plus vraie.
On a donc :
$E = \{Manchot(Titi) \rightarrow Oiseau(Titi), Manchot(Titi), Oiseau(Titi), Vole(Titi)\}$

– Cas 2 : on applique le défaut d_2 à W, alors on a :

$E_0 = \{Manchot(Titi) \rightarrow Oiseau(Titi), Manchot(Titi), Oiseau(Titi)\}$

$E_1 = Th(E_0) \cup \{\neg Vole(Titi) \ tel \ que \ \frac{Manchot(Titi) : \neg Vole(Titi)}{\neg Vole(Titi)} \in D, Manchot(Titi) \in E_0, Vole(Titi) \notin E_0\}$

Le défaut d_2 peut donc bien être appliqué et $\neg Vole(Titi)$ peut donc être ajouté dans l'extension. Une fois ce défaut appliqué, il n'est plus possible d'appliquer le défaut d_1 puisque la justification de ce défaut ne serait plus vrai.
On obtient donc l'extension suivante :
$E = \{Manchot(Titi) \rightarrow Oiseau(Titi), Manchot(Titi), Oiseau(Titi), \neg Vole(Titi)\}$

On peut suivre cette démarche sur l'arbre de la figure 3.1.

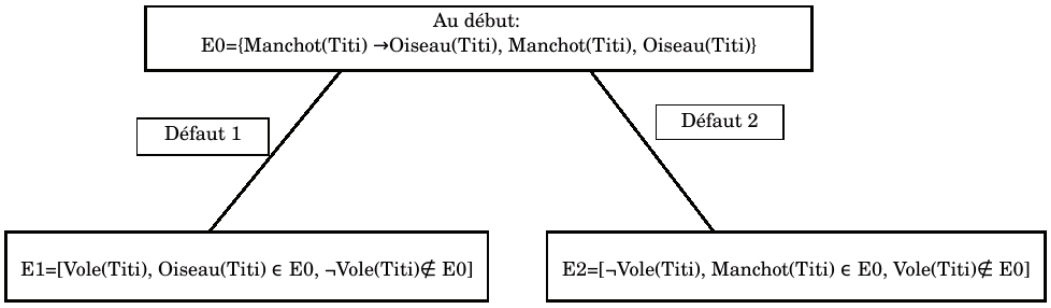

FIG. 3.1 – Arbre de recherche des solutions pour le calcul d'extensions.

On obtient donc deux extensions qui sont contradictoires. Si nous cherchons à répondre à la question : "Est-ce que Titi vole ?", il faut pouvoir choisir entre ces deux extensions. On peut préférer les défauts les plus particuliers, ou encore établir des préférences entre les défauts.

Comme nous venons de voir dans cet exemple, les théories des défauts peu-

vent avoir plusieurs extensions. Il y a également des cas où elles ne présentent aucune extension.

Par exemple, prenons la théorie des défauts $\Delta = (D, W)$, avec :
$W = \emptyset$ et $D = \{d_1 = \frac{: A(X)}{\neg A(X)}\}$.
Le défaut d_1 peut être lu : "Si la négation de $A(X)$ ne peut être prouvée, alors on conclut la négation de $A(X)$".

Le calcul d'extension se fait de la façon suivante : on a $E_0 = W = \emptyset$.
Si l'extension E ne contient pas $\neg A(X)$, alors on utilise le défaut d_1 et $\neg A(X) \in E$, cette extension n'est donc pas possible. Il n'y a donc aucune extension possible à cette théorie des défauts : $E = \emptyset$.

Dans certains cas, les défauts classiques peuvent donc poser des problèmes lorsqu'il n'y a pas d'extension. Selon Reiter, l'existence d'au moins une extension est assurée dans le cas où tous les défauts sont normaux.

Conclusion

La logique des défauts semble satisfaire en de nombreux points les exigences fixées pour la thèse (cf. paragraphe 1.3) :

- les défauts et les formules du calcul propositionnel permettent de définir des règles générales. Cette logique permet donc de définir facilement les règles de comportements de l'opérateur ;
- la logique des défauts est un formalisme de raisonnement en l'absence d'informations. Elle peut donc nous permettre de raisonner sur les informations incomplètes, incertaines et révisables, ce qui est le cas de celles dont l'officier dispose dans son sous-marin ;
- ce formalisme permet de traiter les règles admettant des exceptions sans être obligé de remettre en cause les règles précédemment établies.

La logique des défauts nous paraît donc être une formalisme adapté pour représenter les connaissances de l'officier en charge du sous-marin. Il semble difficile de présenter la logique des défauts, sans présenter la programmation avec sémantique de modèles stables, les Answer Set Programming (ASP) en anglais.

3.3.2 Answer Set Programming

La notion d'ASP a été définie par Lifschitz [48] en 1988. ASP signifie programmation par ensemble de réponse. Il s'agit d'un formalisme pour représenter différents problèmes de l'intelligence artificielle, dans lesquels l'information disponible est incomplète. Les applications des ASP sont le raisonnement

non monotone, la planification, le diagnostic, la bioinformatique...

Dans les ASP, l'information est codée par des règles logiques et les solutions correspondent à des modèles. Ces solutions sont parfois contradictoires, ce qui permet un raisonnement non-monotone. Chaque modèle correspond à un ensemble minimal d'atomes représentant des informations sûres (des faits) et des déductions obtenues à partir de règles par défaut. Les conclusions sont basées sur de l'information présente et absente, elles sont plausibles, plus ou moins certaines. On retrouve le même principe que dans le raisonnement par défaut avec la logique des défauts.

Il existe différentes variantes de l'ASP, comme les programmes logiques normaux, interprétés par la sémantique des modèles stables. Il existe également les programmes logiques définis qui sont plus simples mais ils ne permettent pas la gestion des exceptions.

Définition 20 : Programme logique normal
Un programme logique normal est un ensemble de règles de la forme :

$$c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m$$

où c, a_i $(1 \leq i \leq n)$, b_j $(1 \leq j \leq m)$ sont des atomes de la logique propositionnelle, *not* est appelé négation par défaut. Cette règle signifie "Si tous les a_i appartiennent à un ensemble réponse et si aucun des b_j n'y appartient, alors c doit appartenir à cet ensemble réponse.
Soit r une règle $r : c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m$. On appelle :
- la tête de règle c, notée $tete(r)$,
- le corps de la règle $\{a_1, \ldots, a_n, b_1, \ldots, b_m\}$, noté $corps(r)$,
- le prérequis positif de r est noté $corps^+(r) = \{a_1, \ldots, a_n\}$,
- le prérequis négatif de r est noté $corps^-(r) = \{b_1, \ldots, b_m\}$,
- la projection positive de r , notée r^+, la règle $tete(r) \leftarrow corps^+(r)$.

Définition 21 : Programme logique défini
Si un programme P ne contient pas de négation par défaut, c'est-à-dire $corps^-(P) = \emptyset$, alors P est un programme logique défini et il possède un seul modèle de Herbrand minimal, noté $Cn(P)$.

Définition 22 : Le réduit
Le réduit P^X d'un programme logique normal P par rapport à un ensemble d'atomes X est le programme logique défini par : $P^X = \{r^+ | r \in P, corps^-(r) \cap X = \emptyset\}$

Définition 23 : Modèle stable
Soit P un programme logique normal et S un ensemble d'atomes. S est un

modèle stable de P si et seulement si $S = Cn(P^S)$. Un programme peut avoir aucun, un ou plusieurs modèles stables. S'il n'en a aucun, il est inconsistant, sinon il est consistant.

Exemple 7 : Soit P le programme logique suivant :

$$P = \left\{ \begin{array}{r} a \leftarrow not\ b. \\ c. \\ b \leftarrow d,\ not\ c. \\ d \leftarrow c,\ not\ a. \end{array} \right\}$$

Soit X un ensemble d'atomes, $X = \{a,\ c\}$, alors, le réduit P^X est le programme logique suivant :

$$P^X = \left\{ \begin{array}{r} a. \\ c. \end{array} \right\}$$

On a alors $Cn(P^X) = \{a, c\} = X$ et X est un modèle stable.

Soit X' un ensemble d'atomes, $X' = \{c,\ d\}$ alors le réduit $P^{X'}$ est le programme logique suivant :

$$P^{X'} = \left\{ \begin{array}{r} a. \\ c. \\ d \leftarrow c \end{array} \right\}$$

On a alors $Cn(P^{X'}) = \{a, c, d\} \neq X'$ et X' n'est pas un modèle stable.

Un formalisme a été proposé par Nicolas et al. ([80], [84]) pour gérer à la fois un raisonnement non-monotone et incertain. Pour cela, les concepts de la théorie des possibilités [123] sont introduits au sein de la sémantique des modèles stables. Un degré de certitude est affecté aux règles du programme. L'exemple suivant y est proposé :

Exemple 8 : Un patient souffre de deux maladies. Chaque maladie peut être soignée par un médicament mais les deux médicaments sont incompatibles. Le programme suivant :

$$P = \left\{ \begin{array}{r} med1 \leftarrow mal1, not\ med2. \\ med2 \leftarrow mal2, not\ med1. \\ g1 \leftarrow med1,\ mal1. \\ g2 \leftarrow med2,\ mal2. \\ mal1 \leftarrow . \\ mal2 \leftarrow . \end{array} \right\}$$

signifie que le médicament $med1$ (resp. $med2$) est donné à un patient s'il est atteint de la maladie $mal1$ (resp. $mal2$) sauf si il prend le médicament $med2$ (resp. $med1$). Si un patient atteint de la maladie $mal1$ (resp. $mal2$) prend le médicament $med1$ (resp. $med2$) alors il est guéri $g1$ (resp. $g2$) de cette maladie.

Le patient est atteint des deux maladies $mal1$ et $mal2$. On a deux modèles stables : $\{mal1, mal2, med1, g1\}$ et $\{mal1, mal2, med2, g2\}$. Le patient peut être guéri soit de la maladie $mal1$, soit de l'autre maladie $mal2$.

Le médecin doit alors choisir entre ces deux solutions. L'introduction de niveaux de certitudes sur les règles va pouvoir le guider dans son choix. On obtient le programme logique normal possibiliste suivant :

$$P = \left\{ \begin{array}{r} (med1 \leftarrow mal1, not\ med2.\ 1) \\ (med2 \leftarrow mal2, not\ med1.\ 1) \\ (g1 \leftarrow med1,\ mal1.\ 0.7) \\ (g2 \leftarrow med2,\ mal2.\ 0.3) \\ (mal1 \leftarrow .\ 0.9) \\ (mal2 \leftarrow .\ 0.7) \end{array} \right\}$$

Les deux premières règles sont totalement certaines. La troisième règle exprime le fait que le médicament $med1$ a une efficacité assez certaine de guérir la maladie $mal1$, et la quatrième règle signifie que le médicament $med2$ a une efficacité peu certaine de guérir la maladie $mal2$.

Les deux modèles possibilistes de P sont :

– $\{\ (mal1, 0.9),\ (mal2, 0.3),\ (med1, 0.9),\ (g1, 0.7)\}$,

– $\{\ (mal1, 0.9),\ (mal2, 0.3),\ (med2, 0.7),\ (g2, 0.3)\}$.

Le médecin doit alors choisir entre donner le médicament $med1$ et il sera assez certain que le patient guérisse de la maladie $mal1$. S'il donne le médicament $med2$, il est peu certain que le patient guérisse de la maladie $mal2$. Le médecin a alors une information supplémentaire qui peut l'aider à faire son choix. Cependant, cette information est à prendre avec précaution. Par exemple, si la maladie $mal2$ est beaucoup plus grave que la maladie $mal1$, le médecin préférera sûrement donner le médicament $med2$.

Conclusion

La programmation logique avec sémantique des modèles stables est un outil efficace pour la représentation des connaissances et le raisonnement non-monotone. De nombreux solveurs ASP ont été développés : Smodels [106] (premier solveur ASP), Nomore [4], Assat [65], Clasp [47].

Cependant, en ce qui concerne notre cas d'application, la logique des défauts semble plus simple à utiliser. Elle nous permet de définir entièrement

notre propre programme, sans avoir recourt à un solveur ASP. De cette façon, nous avons une parfaite maîtrise de ce programme.

3.4 Représentation des connaissances avec les systèmes multi-agents

Née au début des années 70, l' intelligence artificielle distribuée s'intéresse à des comportements intelligents qui sont le produit de l'activité coopérative de plusieurs agents ([53], [55]). L'évolution des cette discipline a conduit aux systèmes multi-agents.

Un système multi-agents est un système composé d'agents autonomes qui poursuivent des objectifs individuels tout en interagissant dans un environnement commun afin de résoudre une tâche commune.

Les domaines d'application sont nombreux : résolution de problèmes, robotique distribuée, simulation ...

Avant de donner quelques exemples de systèmes multi-agents, commençons par donner quelques définitions.

3.4.1 Définitions

Il existe de nombreuses définitions de l'agent. Celle donnée ici est une des premières définitions, tirée du livre de Ferber [40].

Définition 24 : **Agent** On appelle agent une entité physique ou virtuelle :

- qui est capable d'agir dans un environnement,
- qui peut communiquer directement avec d'autres agents,
- qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
- qui possède des ressources propres,
- qui est capable de percevoir (mais de manière limitée) son environnement,
- qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
- qui possède des compétences et offre des services,
- qui peut éventuellement se reproduire,
- dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Un agent est une entité autonome au sens où il sait prendre ses décisions seuls, en fonction de sa structure interne et de ses perceptions.

Un agent est une entité intelligente, c'est-à-dire capable de raisonner. On distingue différents types d'agents : les agents cognitifs, les agents réactifs et les agents hybrides. Dans un rapport de recherche, Labidi et al. [62] comparent ces deux types d'agents. Les agents réactifs sont les plus sommaires. Ils ont un comportement du type "stimulus/réponse", ils ne peuvent pas prendre en compte leurs actions passées et ils ne possèdent pas une représentation complète de leur environnement. Les agents cognitifs sont plus proches du comportement d'un être humain. Ils ont une représentation explicite de leur environnement et l'utilisent pour planifier leurs actions. Les agents hybrides ont les capacités à la fois cognitives et réactives. Ils possèdent la rapidité des agents réactifs ainsi que les capacités de raisonnement des agents cognitifs.

Le comportement d'un agent peut être vu comme un cycle perception, décision, action. L'agent perçoit son environnement, et décide en fonction de cette perception quelle action il va effectuer.

Un système multi-agents est un système composé de plusieurs agents, organisés et immergés dans un environnement commun dans le but de résoudre une tâche commune.

Définition 25 : **Système multi-agents [40]** On appelle système multi-agents, un système composé des éléments suivants :

- un environnement E, c'est-à-dire un espace disposant généralement d'une métrique,
- un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents,
- un ensemble d'agents A, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système,
- un ensemble de relations R qui unissent des objets (et donc des agents) entre eux,
- un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O,
- des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification.

Le comportement d'un agent pouvant être vu comme un cycle perception, décision, action, il faut définir le modèle décisionnel de l'agent. Une approche possible pour définir ce modèle décisionnel est d'utiliser des approches dites comportementales, qui décomposent le modèle décisionnel entre différentes

entités : les comportements.

3.4.2 Les approches comportementales

Les approches comportementales, connues également sous le nom de "Behaviour based Systems", ont été introduites en robotique à la fin des années 80 ([25], [66], [83], [8]). Elles ont permis la réalisation des travaux sur des robots en interaction avec le monde réel.
Ces approches sont basées sur l'idée qu'un modèle décisionnel peut être réparti entre différentes entités : les comportements. Les entrées des comportements sont des senseurs et d'autres comportements et les sorties sont des effecteurs et d'autres comportements.

Chaque comportement a un rôle particulier. Dans [10], les robots possèdent les comportements suivants : Aller vers le but, Eviter un obstacle, Eviter un robot, Rester grouper ...

Les *boids* de Reynolds

En 1987, C.W. Reynolds [91] a présenté un comportement émergeant appelé *flocking* qui simule le vol d'oiseaux. Cette approche est basée sur le fait que le *flocking* est simplement le résultat de l'interaction entre les comportements d'oiseaux individuels. Pour qu'un oiseau participe à un *flocking*, il doit avoir les comportements qui lui permettent de coordonner ses mouvements avec ceux des autres oiseaux. Pour cela, il a besoin de deux comportements opposés : rester proche de la volée et éviter la collision avec les autres oiseaux.

Dans le modèle, chaque oiseau est nommé *boid*. Les *boids* sont caractérisés par une position, une orientation et une vélocité. Chaque *boid* a accès à la position, à l'orientation et à la vélocité de tous les autres objets du système (c'est-à-dire les autres *boids* et les obstacles). Dans le cas du *flocking*, chaque *boid* a seulement besoin de connaître ces informations pour les proches voisins.

Trois règles de comportements suffisent à animer le groupe de *boids* :
- l'évitement de la collision avec les *boids* voisins ;
- l'alignement de la vélocité : il faut conserver une vélocité proche de celle des voisins ;
- le centrage par rapport au groupe : il faut rester le plus près possible de ses voisins ;

Ces trois règles sont traduites en vecteur vitesse. Elles sont représentées de façon schématique sur la figure 3.2.

Chaque règle de comportement est associée à un poids, compris entre 0 et 1, qui permet d'atténuer l'influence de la proposition. A chaque instant, le vecteur vitesse qui détermine le mouvement du *boid* est la somme pondérée des trois vecteurs vitesses qui traduisent les trois règles de comportements. Ces règles permettent au *boid* d'évoluer dans la direction du groupe tout en restant à une distance raisonnable de ses voisins.

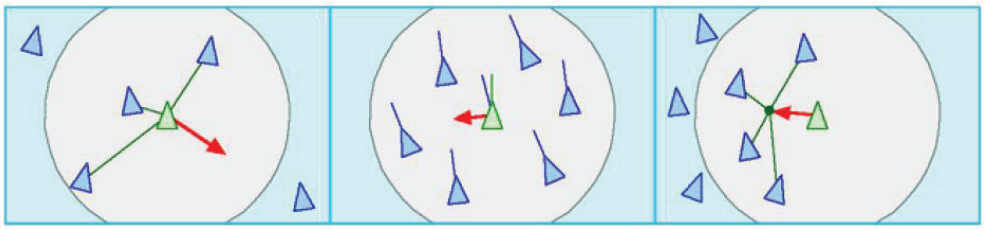

FIG. 3.2 – De gauche à droite : Evitement de la collision ; Alignement des vélocités ; Centrage par rapport au groupe ;

Le *flocking* est un comportement émergeant : des comportements complexes globaux peuvent résulter de l'interaction de simples règles locales.

Ces règles s'avèrent suffisantes pour générer un comportement global des *boids*, semblable à un vol d'oiseaux migrateurs. Depuis 1987, cette méthode a été utilisée pour d'autres animations comportementales. En 1992, Tim Burton a été le premier à l'utiliser pour le film *Batman Returns*, pour simuler une nuée de chauves-souris et un vol de pingouins.

Cette technique a été développée pour une application précise de vol en formation et elle a fait ses preuves. Elle est cependant limitée à trois règles de comportement et elle est trop spécifique pour être réutilisée pour notre cas d'application qui consiste à modéliser le comportement d'un sous-marin.

L'architecture de subsomption de Brooks

En 1985, Brooks [25] a défini une architecture pour contrôler un robot mobile. Pour cela, il a décomposé le comportement du robot en différents niveaux de compétences : éviter des objets, errer, explorer ... (Figure 3.3).

L'architecture de subsomption est basée sur l'idée clé qu'il est possible d'ajouter des niveaux supérieurs, sans avoir à modifier l'architecture déjà existante.

Un niveau supérieur implique un comportement plus spécifique. Il est capable d'examiner son niveau inférieur et de lui supprimer ses données d'entrée. De cette façon, un niveau supérieur peut bloquer, subsumer le com-

portement du niveau inférieur (Figure 3.4). Dans l'exemple de Brooks, le niveau le plus inférieur est l'évitement d'obstacle.

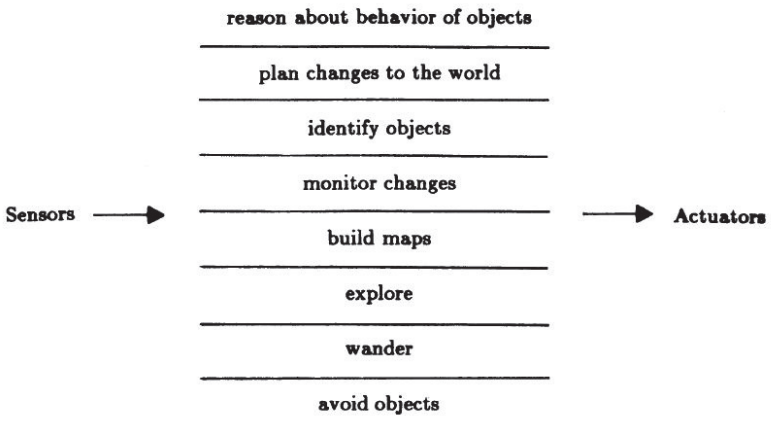

FIG. 3.3 – Une décomposition du contrôle d'un robot mobile en comportements [25]

Chaque niveau de compétence est composé d'un ensemble de petits modules qui s'envoient des messages les uns aux autres. Chaque module est une machine à états finis. La subsomption permet de supprimer les messages d'entrée des modules et de bloquer les messages de sortie. La figure 3.5 explique comment les modules communiquent entre eux : ils ont des lignes d'entrée et des lignes de sortie. Dans cet exemple, la ligne de sortie peut être inhibée pendant un certain temps (3 unités de temps) après l'arrivée d'un message et la ligne d'entrée peut être supprimée pendant 10 unités de temps.

Cette approche comportementale est intéressante. Elle a ensuite été reprise dans d'autres travaux comme ceux de Mataric [70]. Cependant, elle semble lourde à mettre en place. En effet, dans son article, Brooks précise qu'il faut construire l'architecture des comportements de manière incrémentale, en commençant par construire le niveau de compétence 0. Une fois que ce niveau fonctionne bien, on peut alors construire le niveau 1, et ainsi de suite. Cette méthode semble compliquée à être réutilisée pour notre cas d'application et elle ne permet pas d'ajouter simplement de nouvelles règles, sans avoir à modifier la représentation des connaissances (exigence 3, cf. paragraphe 1.3, page 27). En effet, l'ajout d'une nouvelle règle nécessite la prise en compte des règles déjà existantes et de leur hiérarchie.

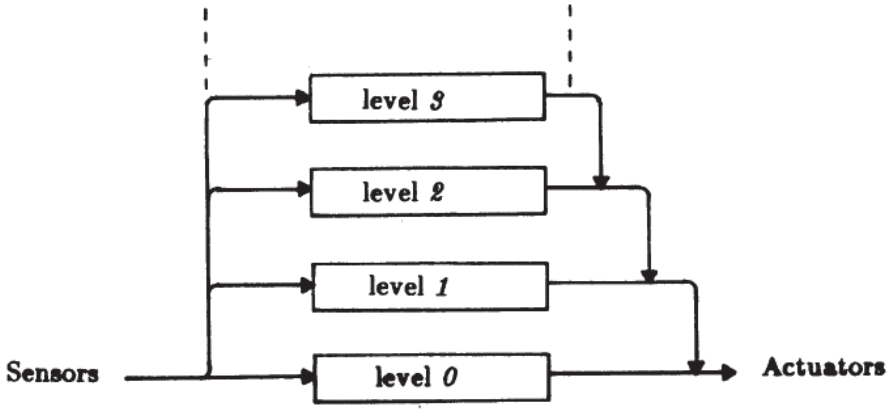

FIG. 3.4 – Architecture de subsomption [25]

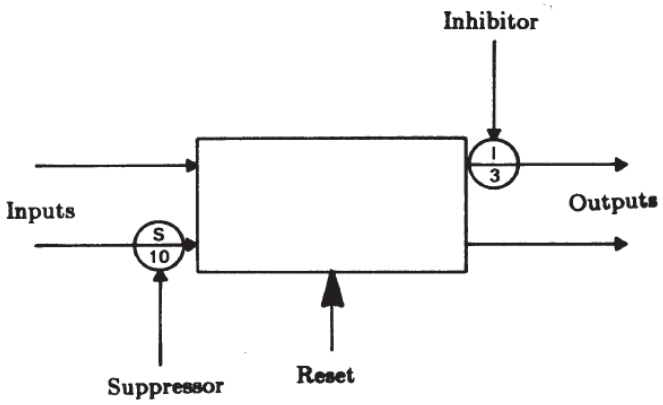

FIG. 3.5 – Mécanisme de suppressions et d'inhibition dans un module [25]

La théorie des schémas d'Arkin

La théorie des schémas vient à l'origine des travaux de Arbib ([5], [6]). En particulier, Arbib et House [7] ont développé, en 1987, un modèle de comportements pour les grenouilles : les grenouilles doivent acquérir des vers de terre dans un environnement contenant des obstacles. Ils développent deux modèles pour le comportement, le second est le plus facilement réutilisable dans le domaine des robots mobiles. Ils utilisent des champs de vecteurs :

- des champs attractifs pour les proies,
- des champs répulsifs pour les obstacles,

– un champ pour la grenouille elle même.

Ces champs, une fois combinés, donnent un modèle de comportement qui est consistant avec les observations expérimentales des grenouilles.

La théorie des schémas a ensuite été proposée par Arkin en 1989 [8], appliquée à la navigation de robots autonomes. Elle décompose le comportement en entités indépendantes, les schémas. Ces schémas s'expriment sous forme de forces attractives ou répulsives. Chaque schéma correspond à un comportement de base : suivre un chemin, éviter les obstacles, aller vers le but, changer de cap, changer de vitesse...
Des conditions d'activation sont définies pour chaque schéma. La sortie d'un schéma est un vecteur vélocité qui donne une consigne de déplacement. Ce vecteur est généré selon des méthodes à base de champs de potentiel : les cibles induisent des champs de potentiel attractifs et les obstacles des champs de potentiel répulsifs (cf. sur la figure 3.6).
Les vecteurs de chaque schéma activé sont ensuite additionnés pour donner une consigne finale de déplacement. Chaque schéma est pondéré d'un poids, qui traduit l'importance du comportement élémentaire.

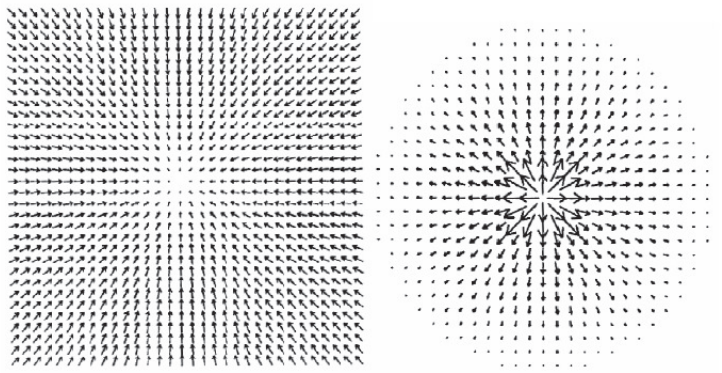

FIG. 3.6 – Deux schémas : aller-vers-cible (gauche) et éviter-obstacle (droite) [8] .

Combinée à un système de planification, cette méthode a été testée sur un système d'architecture de robots autonomes AuRA (Autonomous Robot Architecture). Elle a été appliquée à trois robots chargés de collecter des déchets et de les mettre à la poubelle [9].

Elle a également prouvé son applicabilité à des problèmes de coordination spatiale robotique : une équipe de robots devait naviguer vers un but, éviter

les obstacles tout en restant en formation [10]. Différentes formations ont été définies : ligne, colonne, diamant, en V (Figure 3.7). Chaque robot calcule sa position par rapport à un point de référence. Trois techniques pour déterminer ce point de référence ont été testées : centre, leader et voisin (Figure 3.8).

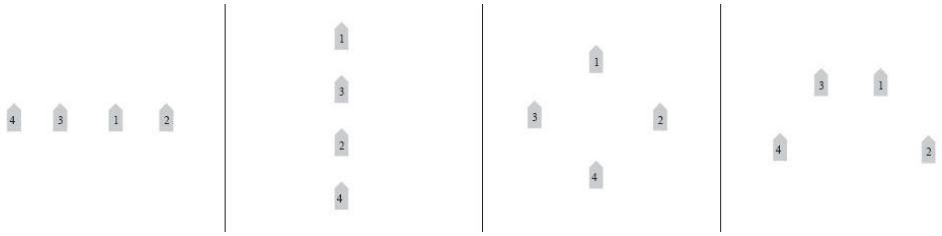

FIG. 3.7 – Formations (de gauche à droite) : ligne , colonne, diamant, en V [10].

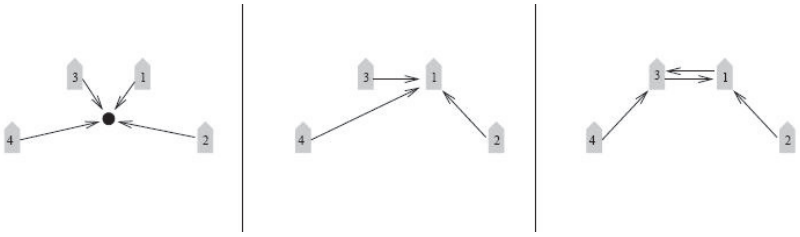

FIG. 3.8 – Point de référence déterminé avec les techniques : centre, leader, voisin (de gauche à droite) [10] .

Des simulations ont montré que les formations utilisées ont des performances qui ne sont pas équivalentes. Dans un scénario où les robots doivent avancer tout droit pendant 250 mètres, puis tourner d'un angle de 90 ° , puis de nouveau avancer tout droit, la meilleure performance revient à la formation en diamant avec pour point de référence le centre ou encore aux formations ligne et en V si la technique leader est utilisée. Par exemple, pour la formation en diamant avec pour point de référence le centre, l'erreur de position (qui mesure la moyenne des déplacements par rapport à la position correcte dans la formation) et le temps passé hors de la formation sont minimisés. De plus, il n'y a pas de robot à plus de 50 mètres du centre alors qu'avec les formations ligne, colonne et V, les robots vont jusqu'à 75 mètres du centre.

Classiquement, les poids des schémas sont donnés par l'animateur. Dans certains cas, il est possible d'utiliser des algorithmes évolutionnistes pour

déterminer ces poids. Ram [88] a utilisé un algorithme génétique : le robot mobile est contrôlé par des schémas, qui utilisent différents paramètres. Ces paramètres ont été optimisés par un algorithme génétique afin de réduire le travail du concepteur.

De même, Flacher [43] s'est intéressé au contrôle de coordination spatiale d'un ensemble d'agents mobiles chargés d'exhiber un comportement collectif. Il propose une méthodologie GACS (Génération Ascendante de Coordination Spatiale) qui consiste à :

- générer un ensemble de contrôleurs, solutions potentielles,
- exploiter un mécanisme de sélection automatique pour isoler un contrôleur conforme aux spécifications du problème.

Pour cela, il a utilisé un modèle de synthèse de contrôleur avec la théorie des schémas, combiné à un algorithme évolutionniste de sélection automatique, qui permet de fixer des paramètres des schémas, comme le poids. L'avantage important de cette méthode est qu'elle demande peu d'effort au concepteur.

Cette méthodologie a été testée sur :

- le contrôle de troupeau de moutons simulés par une équipe de robots bergers,
- le maintien en formation par un groupe d'aéronefs dans un espace aérien encombré de zones dangereuses.

Cette technique est intéressante car les schémas sont des comportements indépendants. On n'a donc pas à se soucier des comportements déjà définis lorsque l'on veut en ajouter des nouveaux. Le choix entre les comportements est géré par l'addition des consignes de tous les schémas activés, pondérées par les poids.

3.4.3 Conclusion

Les systèmes multi-agents, et plus particulièrement les approches comportementales, semblent peu correspondre à notre problème. En effet, ils portent sur des systèmes qui comportent un grand nombre d'agents, alors que dans notre application, nous avons besoin d'entités peu nombreuses mais intelligentes. Les exemples donnés en approche comportementale semblent difficilement réutilisables pour notre cas d'application : les *boids* de Reynolds ont été conçus pour une application bien précise, l'architecture de subsomption semble lourde à mettre en place et elle ne permet pas d'ajouter simplement de nouveaux comportements. Cependant, la théorie des schémas répond à certaines de nos exigences : un nouveau comportement, sous forme de schéma, peut être ajouté facilement et le choix entre les comportements est géré par l'addition des consignes de chaque schéma actif.

Nous avons travaillé sur une modélisation du comportement avec la théorie des schémas (cf. paragraphe 4, page 87), déjà utilisé dans l'entreprise DCNS. Ayant trouvé des limites à cette approche, nous nous sommes tournés vers une approche orientée logique, en utilisant la logique des défauts.

3.5 Raisonnement temporel en intelligence artificielle

La prise en compte du temps en intelligence artificielle est utilisée dans différents domaines tels que :

- les diagnostics médicaux : afin de déterminer correctement la cause d'une maladie et le traitement à prescrire, il faut prendre en compte quand et dans quel ordre les symptômes de la maladie sont apparus, ainsi que leur évolution au cours du temps ;
- la planification et l'ordonnancement des tâches : pour mettre en place une planification, il faut considérer la durée des actions et des tâches à accomplir. Il faut également déterminer l'ordre le plus approprié dans lequel les tâches vont devoir être accomplies en prenant en compte leurs interactions au cours du temps ;
- la représentation de l'évolution et du changement d'état dans l'environnement,
- les prédictions, ...

Une définition du raisonnement temporel a été proposée par Vila [119] et reprise par Pani et al. [81]. Le raisonnement temporel consiste à formaliser la notion de temps et à fournir des moyens de représenter l'aspect temporel des connaissances et de raisonner à son sujet.

Pour définir un raisonnement temporel, on doit avoir :

- une extension du langage pour représenter l'aspect temporel des connaissances : cela revient à ajouter une dimension supplémentaire pour définir la vérité d'une information ;
- un système de raisonnement temporel : une méthode pour raisonner sur le langage qui prend en compte ce nouvel aspect temporel.

D'après Vila [119] et Pani et al. [81], il y a trois façons principales d'introduire le temps en logique :

- avec la logique du premier ordre,
- avec la logique modale temporelle,
- avec la logique temporelle réifiée.

Après avoir présenté ces logiques temporelles, une interprétation temporelle de la logique des défauts sera présentée succinctement.

3.5.1 Logique du premier ordre

Une façon simple et naturelle de prendre en compte le temps est de le considérer comme un argument d'un prédicat du premier ordre. Cet argument temporel permet de changer l'interprétation du prédicat dans lequel il apparaît. Il n'y a pas de traitement spécial accordé au temps. Prenons un exemple. On peut traduire : "Il fait chaud à Toulon à 12h" par *chaud(Toulon, 12h)*. Ou encore, "Le magasin est ouvert de 10h à 19h" peut être traduit par *ouvert(magasin, 10h-19h)*.

Cette méthode est utilisée pour raisonner sur les processus physiques et dans beaucoup d'applications mathématiques, lorsque le temps peut être représenté par une variable. Cette approche permet de résoudre des problèmes de faible complexité. Dans certains cas, le fait que le temps n'a pas de statut particulier peut poser problème. En effet, rien n'empêche de considérer la formule suivante : *chaud(Toulon, Nantes)*, alors que *Nantes* n'est pas un argument temporel. Une solution peut être d'utiliser une logique multi-sorte, où les objets temporels sont séparés des autres objets.

Cette représentation simple, qui a l'avantage d'avoir une faible complexité algorithmique, convient à notre cas d'application. Nous choisissons de représenter le temps de manière discrète. Le temps est incrémenté d'un pas de temps à chaque itération : nous passons d'un instant t à un instant $t+1$. Le temps est considéré comme un argument dans les formules.

Dans certains cas, cette représentation du temps peut être insuffisante. On peut avoir besoin d'exprimer des notions temporelles comme : maintenant, depuis, tant que, jusqu'à ce que ... Ces concepts ne peuvent pas être représentés en définissant le temps par un nombre.

3.5.2 Logique modale temporelle

Une autre façon d'inclure le concept de temps est de complexifier l'interprétation des formules. En 1955, Prior introduisit la première logique temporelle [85]. Les logiques modales temporelles sont construites à base de logiques modales où l'interprétation d'une formule prend en compte le contexte temporel. Pour cela, des opérateurs faisant référence au passé et au futur sont introduits.

L'article de Prior [85] définit les opérateurs temporels suivants :

- Fp : "p sera vrai dans le futur",
- Gp : "p sera toujours vrai dans le futur",
- Pp : "p a été vrai dans le passé",
- Hp : "p a toujours été vrai dans le passé".

Les formules sont qualifiées temporellement en prenant en compte le présent ou les autres évènements.

La sémantique des mondes possibles [61] est réinterprétée dans le contexte temporel : chaque élément du temps représente un monde possible. L'interprétation d'une formule p, si elle est vraie ou fausse, doit se faire par rapport à un modèle du monde M et un élément du temps t, noté : $M, t \models p$.

La logique modale temporelle est un langage riche et efficace. Le système de notation permet d'utiliser facilement la logique modale temporelle dans des applications sur le langage naturel [45]. Mais le domaine dans lequel cette logique a été la plus reconnue est celui de la programmation. L'idée est de définir un programme en utilisant cette logique et en appliquant des méthodes de déduction. Pour cela, de nouveaux opérateurs modaux ont été définis : ensuite (pour l'instant d'après), jusqu'à, ...

3.5.3 Logique temporelle réifiée

La *réification* ([1], [73]) consiste à transformer une proposition temporelle en un *terme propositionnel*, qui devient un objet ordinaire du langage ayant le même statut que les autres termes. L'avantage de la réification est qu'elle permet de décrire la vérité d'une proposition tout en réutilisant la logique du premier ordre.

On peut alors raisonner sur la valeur de vérité des expressions grâce à des prédicats comme le prédicat *TRUE*. Le prédicat *TRUE* a pour arguments le terme et le temps :

$$TRUE(expression\ atemporelle, qualification\ temporelle).$$

Ce prédicat signifie que le premier argument est vrai pendant le temps indiqué dans le second argument. Par exemple :

$$TRUE(voyager(Isabelle), [9h00, 18h00]).$$

La réification a été utilisée dans plusieurs approches : Allen [1], Mc Dermott [73], Kowalski et al [60] et Shoham [104]. Elle a plusieurs avantages : elle accorde un statut spécial au temps et elle permet d'utiliser des prédicats pour déterminer la valeur de vérité d'un terme propositionnel dans un intervalle de temps. Elle permet donc de discuter des relations entre les termes propositionnels et les aspects temporels avec une capacité importante de généralisation.

Cependant, la réification a été critiquée entre autres par Bacchus et al [12] qui proposent une logique nommée BTK ou encore Galton [46] qui propose une technique pour éviter l'utilisation de la réification en utilisant des *tokens*.

Après avoir introduit le raisonnement temporel en intelligence artificielle, nous détaillons ici une approche proposée par Engelfriet [36], dans laquelle il a défini une interprétation temporelle de la logique des défauts. Pour cela, il utilise la logique temporelle modale épistémique S5.

3.5.4 Interprétation temporelle de la logique des défauts

Durant les années 90, Engelfriet et Treur ont travaillé sur la logique des défauts temporelle ([37],[36],[38]). La prise en compte du temps dans la logique des défauts semble naturelle. Lorsque nous utilisons un défaut, il faut vérifier si il n'y a pas de contradiction à la justification. Cette propriété doit être vérifiée au moment où l'on applique le défaut. Il paraît également important de vérifier si cette propriété sera bien vérifiée dans le futur. Cela suggère qu'un défaut peut être utilisé en tant que règle temporelle avec une de ses conditions référant au futur du processus de raisonnement.

Engelfriet et Treur ont défini deux interprétations temporelles de la logique des défauts. Ils ont utilisé une représentation linéaire du temps et une représentation avec ramification. La représentation du temps avec ramification a été introduite dans le but de prendre en compte les incertitudes à propos du futur. Elle est basée sur l'idée que pour chaque moment à venir, il existe plusieurs alternatives possibles.

Interprétation de la logique des défauts en logique temporelle : cas linéaire

L'interprétation temporelle de la logique des défauts avec la représentation linéaire utilise la logique temporelle épistémique, basée sur la logique modale épistémique S5 avec une prise en compte du temps. On retrouve les opérateurs temporels : P, H, F et G, définis au paragraphe 3.5.2. D'autres opérateurs temporels sont utilisés :

- Yp : "p était vrai au temps précédent",
- Xp : "p est vrai au prochain temps",
- □p : "p est toujours vrai".

Lorsque nous utilisons un défaut, il faut vérifier qu'il n'y a pas de contradiction à sa justification. Pour vérifier cela, nous nous réfèrons aux informations déduites jusqu'à ce moment. Une fois ce défaut utilisé, le raisonnement à venir permettra de vérifier si cette condition était justifiée ou non. Pour prendre en compte le temps dans les défauts, l'idée est d'interpréter le défaut en tant que règle temporelle avec une de ses conditions qui réfère au futur.

Soit $\Delta = (D, W)$ une théorie des défauts. L'état initial d'un modèle temporel de raisonnement par défaut M inclut les connaissances W, par conséquent M devrait être un modèle de $\{K\alpha | \alpha \in W\}$.

On a un défaut

$$d = \frac{\alpha : \beta_1, \ldots \beta_n}{\gamma},$$

avec $\alpha, \beta_1, \ldots \beta_n, \gamma$ des formules propositionnelles. α est vraie dans M au temps t :

$$(M, t) \models K\alpha.$$

Si le défaut est applicable alors son conséquent doit être vrai à l'état suivant : $XK\gamma$.

Il reste à exprimer si l'utilisation d'un défaut est justifiée. Pour cela, il faut que les β_i soient consistants dans le contexte du processus de raisonnement, en prenant en compte la partie du contexte qui sera générée dans les prochaines étapes. Si la raisonnement est conservatif, cela signifie qu'il n'y a pas de futur dans lequel $\neg\beta_i$ est généré, pour tout i. En logique temporelle, cela s'exprime de la façon suivante :

$$(M, t) \models \neg FK(\neg\beta_i).$$

Pour résumer, pour définir un défaut en logique temporelle, il faut définir : si α est vrai au temps t et il n'y a pas de i tel que $\neg\beta_i$ soit vrai à tout instant après t, alors γ est vrai au temps $t + 1$.

En logique temporelle épistémique, cela se traduit :

$$K\alpha \wedge \neg FK(\neg\beta_1) \wedge \ldots \wedge \neg FK(\neg\beta_n) \rightarrow XK\gamma$$

Définition 26 : Interprétation temporelle de la logique des défauts

Soit $\Delta = (D, W)$ une théorie des défauts.

1. La fonction τ, qui associe à chaque défaut $d = \frac{\alpha : \beta_1, \ldots \beta_n}{\gamma}$ son interprétation temporelle est définie par :
   $\tau : \frac{\alpha : \beta_1, \ldots \beta_n}{\gamma} \rightarrow K\alpha \wedge \neg FK(\neg\beta_1) \wedge \ldots \wedge \neg FK(\neg\beta_n) \rightarrow XK\gamma$
   L'ensemble $\tau(D)$ est appelé interprétation temporelle de l'ensemble de défauts D.
2. L'interprétation temporelle de W est définie par $\tau(W) = \{K\alpha | \alpha \in W\}$.
3. L'interprétation temporelle de la logique des défauts Δ est définie par : $\tau(\Delta) = \tau(D) \cup \tau(W)$.

Conclusion Engelfriet a donc défini une interprétation temporelle de la logique des défauts. Dans sa thèse, il compare ses travaux avec d'autres approches : Gabbay en 1982 [44], Etherington en 1987 [39],Besnard et Schaub en 1994 [18], Marek, Schwarz et Truszczynski en 1993 [69], Amati, Aiello et Pirri en 1997 [3], Lin et Shoham en 1992 [64]. Cette interprétation en logique des défauts temporelle utilise une logique temporelle linéaire. Le comportement dynamique d'un agent est formalisé par un ensemble de modèles temporels linéaires et l'agent choisit un de ces modèles. Cette représentation linéaire ne permet pas de traiter le déroulement en parallèle de deux modèles distincts. Une différente façon de formaliser les différents modèles possibles est d'utiliser un modèle temporel avec ramifications, où chaque branche est un de ces modèles. Cette seconde interprétation est proposée dans [36]. L'inconvénient de cette représentation est qu'elle peut amener à résoudre un problème très complexe.

3.5.5 Conclusion

La prise en compte du temps a été introduite de différentes façons en intelligence artificielle. Nous avons préféré représenter le temps comme argument dans les formules. Nous incrémentons le temps, de manière discrète, d'un pas de temps à chaque itération : nous passons de l'instant t à l'instant suivant $t+1$. Cette représentation convient à notre cas d'application, elle est simple et a l'avantage d'avoir une faible complexité algorithmique.

Chapitre 4

Modélisation du comportement avec la théorie des schémas

La théorie des schémas a été la première technique utilisée à DCNS pour modéliser le comportement humain. Le début de ma thèse a porté sur l'étude de cette modélisation. Un des objectifs était de savoir si ce module était réutilisable comme point de départ pour la thèse, ou bien si il était préférable de partir sur une autre solution. Dans ce chapitre, je vais commencer par définir ce qu'est un schéma et par décrire la formalisation des règles sous forme de schémas. J'illustrerai cette approche avec un exemple de calcul. Enfin, je conclurai sur cette modélisation en expliquant pourquoi nous n'avons pas voulu la réutiliser comme support de travail pour la thèse.

La théorie des schémas a été proposée par Arkin en 1989 [8] (cf paragraphe 3.4.2). Cette théorie consiste à décomposer le comportement en entités indépendantes : les schémas. Dans le cas du sous-marin, un ensemble de schémas a donc été défini. Suivant le contexte dans lequel se trouve l'officier de quart, un ensemble de schémas sera activé. La consigne de déplacement donnée par l'officier est ensuite une combinaison des schémas activés.

Commençons par définir plus précisément ce qu'est un schéma.

4.1 Définition d'un schéma et de la consigne de déplacement

On note l'officier (et son sous-marin) A. La consigne de déplacement $\overrightarrow{C_A}$ de l'officier est modélisée comme la résultante de δ schémas activés $\overrightarrow{F_k}$. Chacun des schémas $\overrightarrow{F_k}$ est construit à partir :

– d'une fonction de direction, qui calcule le point visé par l'officier, noté

P_k : c'est le point que l'officier va chercher à atteindre. Ce point est calculé en fonction des λ éléments de l'environnement : $E_1, E_2, \ldots, E_\lambda$ perçus par l'officier. Ces éléments peuvent être des objets physiques de l'environnement (obstacle, cibles, autres agents...) directement perceptibles, ou bien des variables internes ;

- d'une fonction d'intensité, qui calcule l'intensité I_k : elle permet de moduler la norme du schéma. Elle est calculée en fonction de la distance entre l'officier A et le point P_k ;
- d'une fonction de gain, qui calcule le gain G_k : il précise le niveau de priorité du schéma par rapport aux autres schémas ;
- de conditions d'activation : elles déterminent dans quelles situations le schéma doit être activé. Elles sont définies à partir de variables d'états. Une telle variable permet de stocker les différents états que peut posséder le sous-marin. Ces variables sont implémentées par un automate à états finis.

Seuls les schémas dont les conditions d'activation sont remplies sont pris en compte dans le calcul de déplacement futur de l'agent.
La consigne de déplacement $\overrightarrow{C_A}$ à un instant t est la somme des δ schémas $\overrightarrow{F_k}$ activés à cet instant t :

$$\overrightarrow{C_A} = \sum_{k=1}^{\delta} \overrightarrow{F_k} = \sum_{k=1}^{\delta} G_k * I_k * \frac{\overrightarrow{AP_k}}{||AP_k||}$$

Cette consigne est finalement normalisée :

$$\overrightarrow{C_A} = \frac{\sum_{k=1}^{\delta} \overrightarrow{F_k}}{\sum_{k=1}^{\delta} G_k} = \frac{\sum_{k=1}^{\delta} G_k * I_k * \frac{\overrightarrow{AP_k}}{||AP_k||}}{\sum_{k=1}^{\delta} G_k}$$

4.2 Formalisation des règles sous forme de schémas

4.2.1 Décomposition du scénario en phases

La théorie des schémas a été appliquée aux règles qui décrivent les actions de l'officier de quart en cas de détection d'un sous-marin ennemi (cf. règle 2 au paragraphe 2.3).
Ces règles ont été décomposées en quatre phases :

1. anti-collision ;
2. mesure de distance ;
3. ralliement de la position de pistage ;
4. pistage du sous-marin adverse.

4.2.2 Définition des conditions d'activation

Pour chaque phase, des schémas ont été définis. Pour spécifier les conditions d'activation de ces schémas, deux automates à états finis ont été définis (cf. figure 4.1 et figure 4.2).

Sur le premier automate à états finis (figure 4.1), les états reprennent les actions effectuées par l'officier. Les deux premiers états "petit défilement" et "moyen défilement" indiquent les défilements du sous-marin ennemi : l'officier doit effectuer la manœuvre d'anti-collision tant que le sous-marin est en petit et moyen défilement. Une fois qu'il a atteint un grand défilement, il n'y a plus de risque de collision.

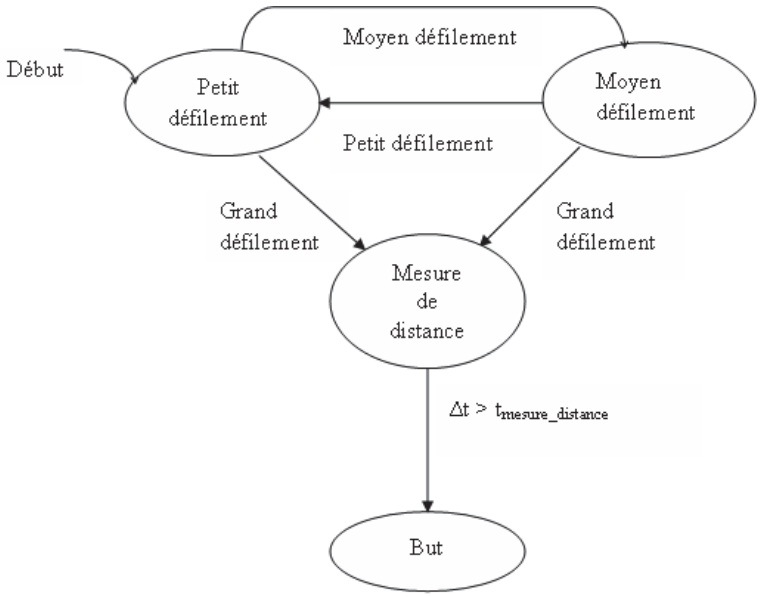

FIG. 4.1 – Automate à états finis "Phase" représentant l'enchaînement des décisions de l'officier.

Dans l'état "mesure de distance", le sous-marin effectue la mesure de distance qui lui permet d'obtenir une meilleure information sur la cinématique de l'ennemi. Le sous-marin doit y rester un certain temps ($t_{mesure\_distance}$)

afin d'acquérir une bonne information. Enfin, dans l'état "but", l'officier va pouvoir rallier le poste de pistage et finalement pister l'ennemi.

Le second automate à états finis (figure 4.2) définit les conditions dans lesquelles le sous-marin se trouve dans le baffle et hors du baffle du sous-marin ennemi. On note B la position du sous-marin ennemi (le but) et $\overrightarrow{V_B}$, la vitesse du sous-marin de l'officier B. L'angle du baffle est noté α. Le baffle se trouve à l'arrière du sous-marin (figure 4.3). Le sous-marin de l'officier A se trouve dans le baffle du sous-marin ennemi B si l'angle $(\overrightarrow{V_B}, \overrightarrow{BA})$ est compris entre $\pi - \frac{\alpha}{2}$ et $\pi + \frac{\alpha}{2}$. Sinon, il se trouve hors du baffle.

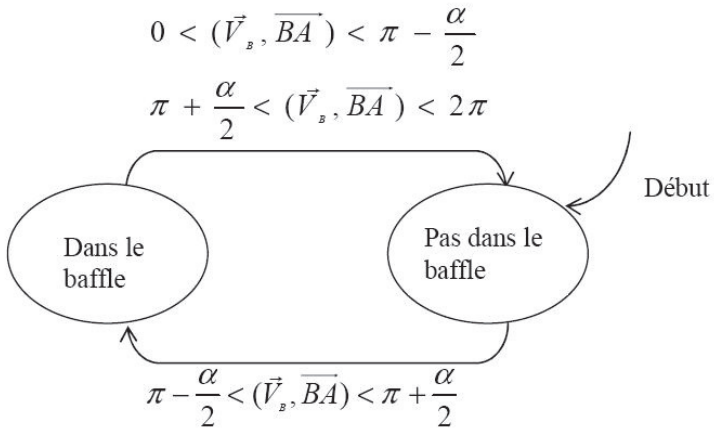

FIG. 4.2 – Automate à états finis "Baffle" représentant la position du sous-marin A par rapport au baffle du sous-marin adverse B.

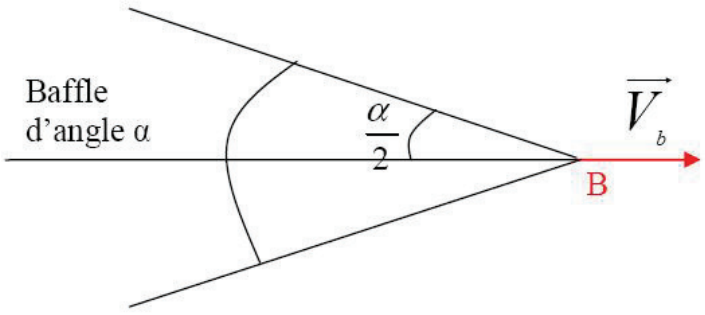

FIG. 4.3 – Baffle du sous-marin.

PHASES DU SCENARIO	SCHEMAS	CONDITIONS D'ACTIVATION
Anti-collision	Anti-collision	Phase(Petit défilement) ou Phase(Moyen défilement)
Mesure de distance	Mesure de distance	Phase(Mesure de distance)
Ralliement de la position de pistage	Ralliement de la position de pistage Positionnement centré sur la cible	Phase(But) et Baffle (Hors du baffle)
Pistage	Pistage de la cible Positionnement centré sur la cible	Phase(But) et Baffle (Dans le baffle)

TAB. 4.1 – Définition des schémas et de leurs conditions d'activation pour chaque phase

4.2.3 Définition des schémas

Les schémas suivants ont été définis :
- anti-collision,
- mesure de distance,
- ralliement de la position de pistage,
- positionnement centré sur la cible,
- pistage de la cible.

Dans le tableau 4.1, les conditions d'activation pour chaque schéma ainsi que les phases du scénario correspondantes sont décrites.

Description du schéma anti-collision

Le schéma d'anti-collision est activé dès que le sous-marin détecte un sous-marin ennemi.

Conditions d'activation Pour être activé, les deux états suivants doivent être actifs :

- Phase(Petit défilement),
- Phase(Moyen défilement).

Le sous-marin reste en manœuvre d'anti-collision tant que le sous-marin ennemi se trouve en petit ou en moyen défilement.

Fonction de direction La fonction de direction permet au sous-marin de placer l'ennemi dans un gisement proche de 150 ° (si l'ennemi se trouve dans un gisement compris entre 0 et 180 °) ou de 210 ° (si l'ennemi se trouve dans un gisement compris entre 180 et 360 °) (cf. figure 4.4). Pour cela, le sous-marin vise le point P_k, qui lui permettra de mettre l'ennemi dans le gisement voulu.

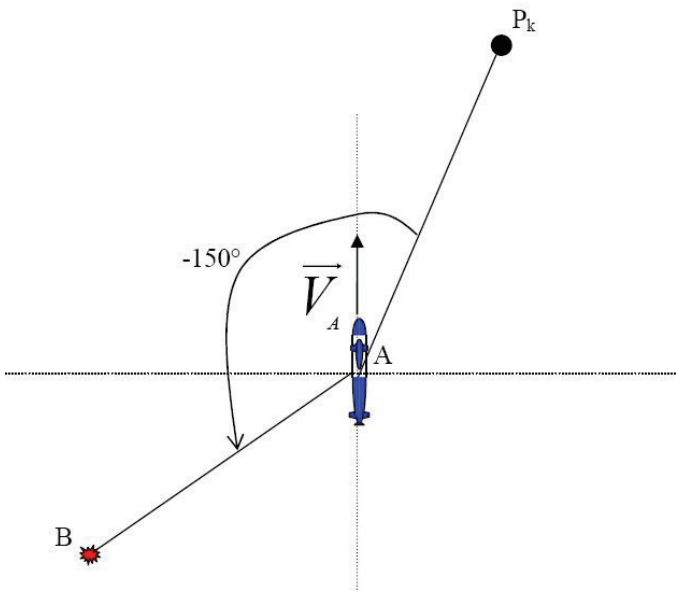

FIG. 4.4 – Fonction de direction pour le schéma anti-collision.

Fonction d'intensité La fonction d'intensité est constante égale à 1. La norme du schéma n'est pas modulée.

Description du schéma mesure de distance

Le schéma mesure de distance permet au sous-marin d'affiner son information sur la cinématique de l'ennemi.

Conditions d'activation Ce schéma est activé lorsque l'état Phase(Mesure de distance) est actif.

Fonction de direction La fonction de direction permet au sous-marin d'effectuer deux tronçons. Lors du premier tronçon, il place le sous-marin dans un gisement 90 ° si l'ennemi est dans un gisement compris entre 0 et 180 ° . Sinon, il place l'ennemi dans un gisement de -90 ° si l'ennemi est dans un gisement compris entre 180 et 360 ° .

Une fois qu'il a effectué ce premier tronçon, il place l'ennemi dans un gisement de 150 ° si l'ennemi est dans un gisement compris entre 0 et 180 ° . Sinon, il place l'ennemi dans un gisement de 2100 ° si l'ennemi est dans un gisement compris entre 180 et 360 ° .

La manœuvre de mesure de distance doit durer $t_{mesure\_distance}$, la première moitié de ce temps est consacré au premier tronçon et la seconde moitié au deuxième.

La fonction de direction du premier tronçon est illustrée sur la figure 4.5.

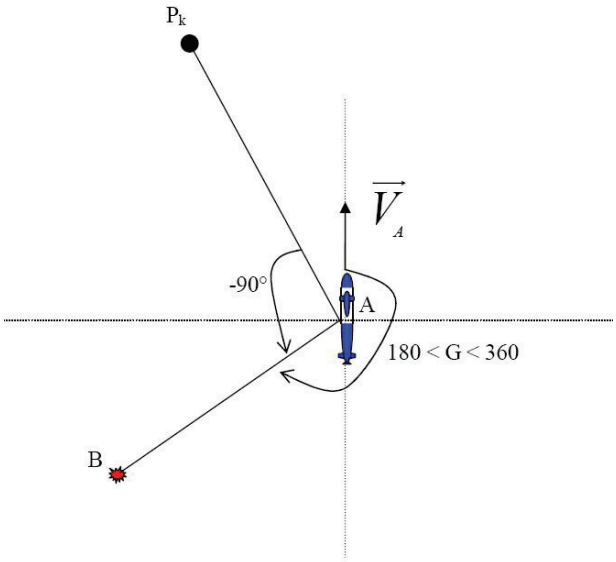

FIG. 4.5 – Fonction de direction pour le premier tronçon de la mesure de distance (cas où le gisement est compris entre 180 ° et 360 °).

Fonction d'intensité La fonction d'intensité est constante égale à 1.

Description du schéma ralliement de la position de pistage

Ce schéma permet de placer le sous-marin sur l'arrière du sous-marin ennemi, afin qu'il se retrouve dans son baffle, position dans laquelle il ne sera pas détecté.

Conditions d'activation Ce schéma possède deux conditions d'activation :

- Phase(But) : l'officier a déjà effectué la mesure de distance, il est alors dans l'état "But" ;
- Baffle(Hors du baffle) : le sous-marin ne se trouve pas encore dans le baffle du sous-marin ennemi.

Fonction de direction La fonction de direction de ce schéma permet de viser un point P_k, qui se trouve à l'arrière du sous-marin ennemi et à une distance D. Ce point est défini de la façon suivante (cf figure 4.6) :

- $(\overrightarrow{BP_k}, \overrightarrow{V_b}) = 180\,^\circ$, avec $\overrightarrow{V_b}$ la vitesse du but,
- $\left\|\overrightarrow{BP_k}\right\| = D$.

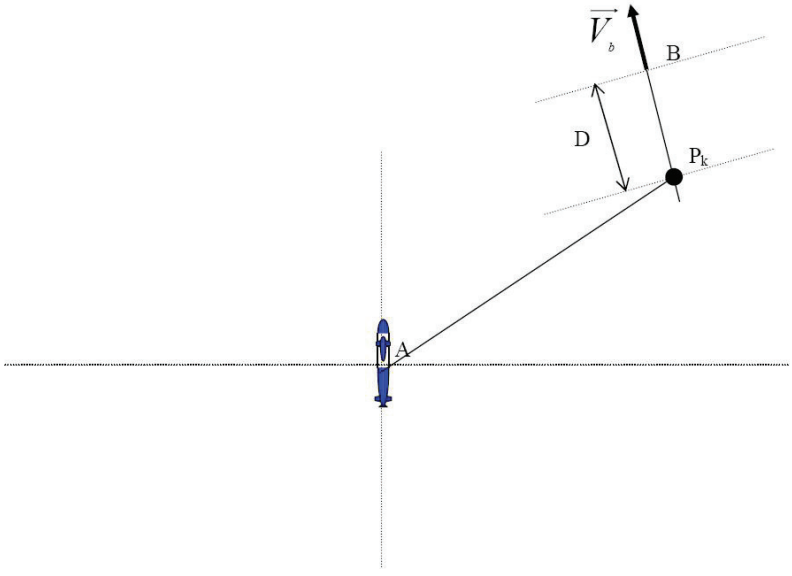

FIG. 4.6 – Fonction de direction du schéma ralliement du poste de pistage

Fonction d'intensité La fonction d'intensité est une fonction linéaire par morceaux qui dépend de la distance entre le sous-marin de l'officier A et le point visé (cf figure 4.7). En dessous d'une distance à l'ennemi de d_{min}, le sous-marin A sera moins attiré vers le sous-marin B. Cette fonction permet au sous-marin A de ne pas approcher le sous-marin B de trop près.

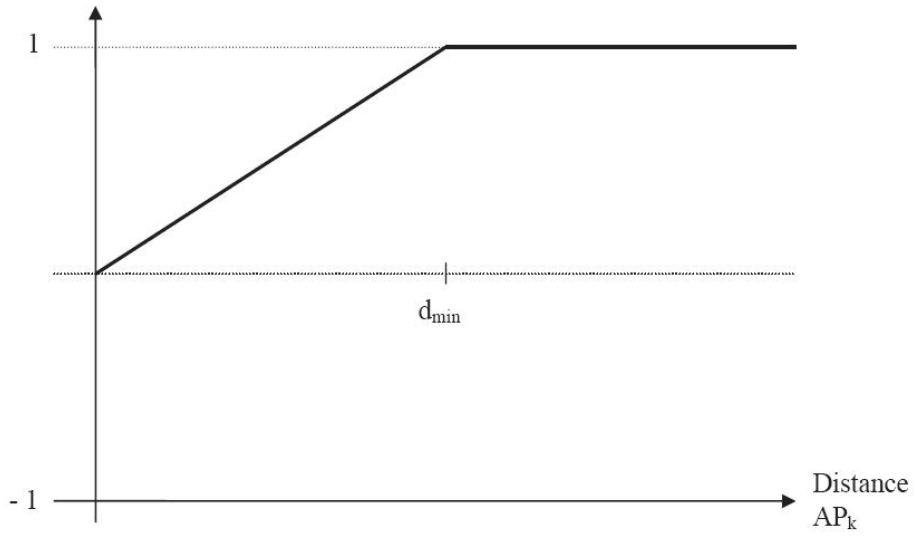

FIG. 4.7 – Fonction d'intensité du schéma ralliement du poste de pistage

Description du schéma positionnement centré sur la cible

Ce schéma permet de maintenir le sous-marin à une certaine distance de l'ennemi : il ne doit pas se trouver trop proche de l'ennemi pour ne pas être contre-détecté mais il ne doit pas en être trop éloigné afin de ne pas perdre le contact.

Conditions d'activation Ce schéma possède les conditions d'activation suivantes :

- Phase(But) ;
- Baffle(Hors du baffle) ou Baffle(Dans le baffle).

Fonction de direction Le point visé P_k par ce schéma est directement la position du but.

Fonction d'intensité La fonction d'intensité est une fonction linéaire par morceaux qui dépend de la distance entre le sous-marin de l'officier A et le point visé (cf. figure 4.8).

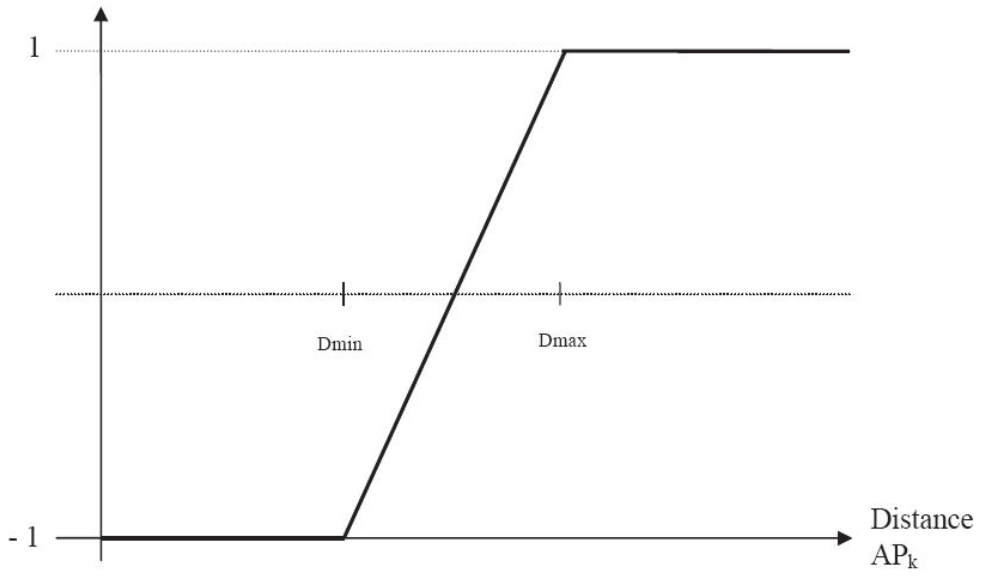

FIG. 4.8 – Fonction d'intensité du schéma positionnement centré sur la cible

Elle maintient l'officier A à une distance de l'ennemi comprise entre :
- la zone limite de détection de l'ennemi d_{min} ;
- une distance maximale d_{max} au delà de laquelle l'officier pourrait perdre le contact avec le sous-marin but.

Description du schéma pistage

Conditions d'activation Les conditions d'activation sont Phase(But) et Baffle(Dans le baffle).

Fonction de direction Le schéma pistage permet au sous-marin de faire des tronçons dans le baffle du sous-marin ennemi. Pour cela, il place le sous-marin ennemi successivement dans un gisement 30 ° puis dans un gisement -30 ° (cf. figure 4.9).

Fonction d'intensité La fonction d'intensité est similaire à celle du schéma ralliement du poste de pistage (cf. figure 4.7).

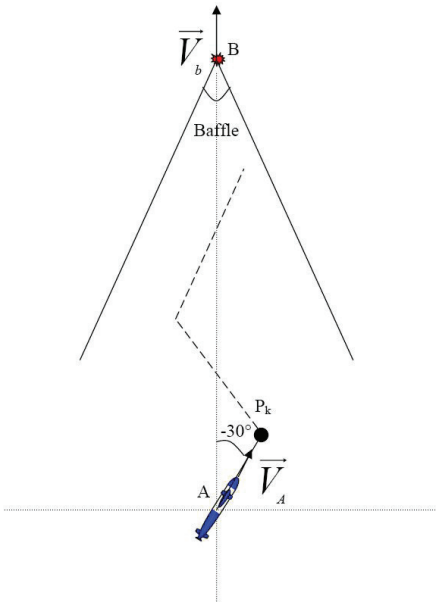

FIG. 4.9 – Fonction de direction du schéma pistage

4.3 Exemple de calcul pour la consigne de déplacement de l'officier

On suppose que les états dans lesquels se trouve le sous-marin sont :

- Phase(But) : l'officier a déjà effectué la mesure de distance, il est alors dans l'état "But" ;
- Baffle (Hors du baffle) : le sous-marin ne se trouve pas encore dans le baffle du sous-marin ennemi.

A ce moment, l'officier connaît la position de l'ennemi. Il veut alors rallier son baffle pour pouvoir ensuite le pister. Sur la figure 4.10, le sous-marin de l'officier est noté A et le sous-marin ennemi est noté B.

Deux schémas sont alors activés : le schéma ralliement de la position de pistage et le schéma positionnement centré sur la cible.

Le schéma ralliement de la position de pistage vise le point P_1, qui se trouve à l'arrière du sous-marin ennemi et à une distance D (cf figure 4.10). Dans cet exemple, la distance entre les deux sous-marins est supérieure à d_{min}, l'intensité du schéma est donc de $I_1 = 1$. Le gain est également égal à $G_1 = 1$ dans cet exemple. On a donc le premier schéma qui renvoie la

consigne :

$$\overrightarrow{F_1} = G_1 * I_1 * \frac{\overrightarrow{AP_1}}{||AP_1||}$$

On note $\overrightarrow{V_1}$ le vecteur $\frac{\overrightarrow{AP_1}}{||AP_1||}$.
On obtient donc :

$$\overrightarrow{F_1} = \overrightarrow{V_1}$$

Le schéma positionnement centré sur la cible vise le point B (c'est-à-dire le but). Dans cet exemple, la distance est comprise entre d_{min} et d_{max} et l'intensité correspondante est de $I_2 = -0.5$. Le gain est égal à $G_2 = 1$.
Le second schéma renvoie la consigne suivante :

$$\overrightarrow{F_2} = G_2 * I_2 * \frac{\overrightarrow{AB}}{||AB||}$$

On note $\overrightarrow{V_2}$ le vecteur $\frac{\overrightarrow{AB}}{||AB||}$ et $\overrightarrow{V_3}$ le vecteur $-0.5 * \overrightarrow{V_2}$.

$$\overrightarrow{F_2} = -0.5 * \overrightarrow{V_2} = \overrightarrow{V_3} \ (cf\ \ figure\ 4.10).$$

Finalement, la consigne finale renvoyée au sous-marin est la somme des deux schémas activés :

$$\overrightarrow{C_A} = \frac{\overrightarrow{F_1} + \overrightarrow{F_2}}{G_1 + G_2} = \frac{\overrightarrow{V_1} + \overrightarrow{V_3}}{2}$$

La combinaison de ces deux schémas permet donc de rallier le baffle du sous-marin ennemi, tout en restant à une certaine distance de sécurité.

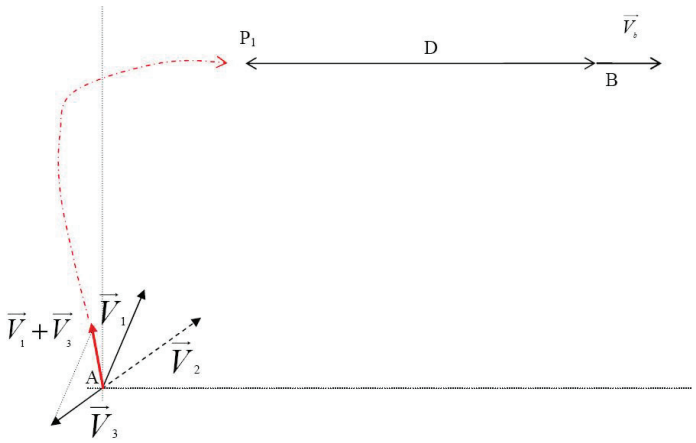

FIG. 4.10 – Exemple de calcul de consigne pour le sous-marin A, ralliement de la position de pistage du sous-marin B

4.4 Validation de cette modélisation

L'intégration du module de comportement avec la théorie des schémas dans ATANOR donne des résultats de simulations conformes à ceux attendus pour le scénario demandé. En effet, ce module de comportement permet bien au sous-marin d'effectuer la séquence d'actions attendue en cas de détection d'un sous-marin ennemi. Sur la figure 4.11, nous avons une exécution de la modélisation du comportement avec la théorie des schémas, interfacée avec l'atelier de simulation ATANOR.

On appelle SM1 le sous-marin auquel nous appliquons les règles de comportement. définies avec la théorie des schémas. On appelle SM2 le sous-marin ennemi, dont le comportement est défini par ATANOR. Dès le début du scénario, le sous-marin SM1 détecte le sous-marin ennemi. Il enclenche alors les actions suivantes :

- Anti-collision,
- Mesure de distance,
- Ralliement de la position de pistage,
- Pistage.

Au début, la trajectoire estimée du sous-marin SM2 par le sous-marin SM1 est éloignée de la trajectoire réelle. La manœuvre de mesure de distance permet d'obtenir une meilleure estimation de la trajectoire de l'ennemi.

Ce module a été validé par des sous-mariniers qui reconnaissent le comportement adopté dans une telle situation. Il nous reste maintenant à savoir s'il est intéressant de réutiliser ce module dans la thèse.

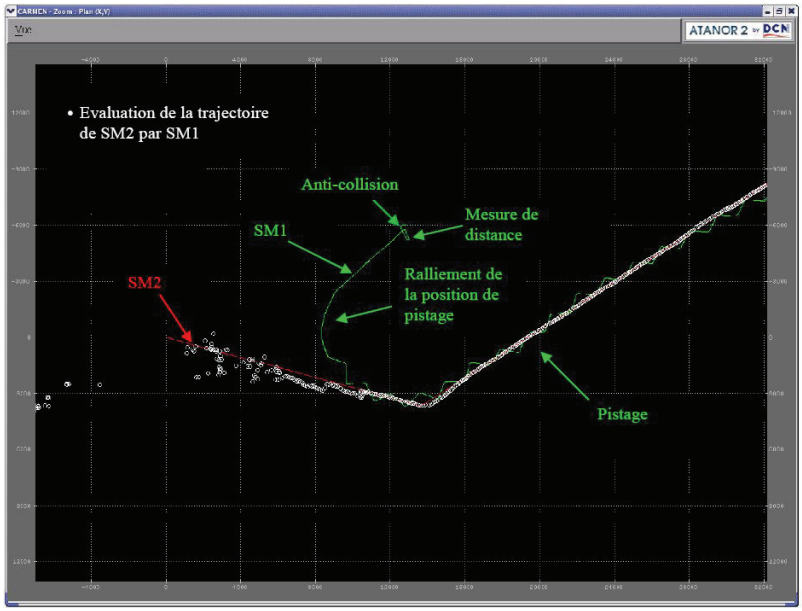

FIG. 4.11 – Exécution de la modélisation du comportement avec la théorie des schémas, interfacé avec l'atelier de simulation ATANOR

4.5 Conclusion

4.5.1 Ajout d'un nouveau schéma

Afin de pouvoir conclure sur la réutilisation de cette modélisation pour la thèse, j'ai défini un nouveau schéma. Pour cela, il a fallu étudier l'architecture logicielle. Cette étude m'a permis de définir les éléments essentiels à la définition de ce nouveau schéma au niveau programmation (définition des variables d'états, des fonctions de direction ...). Par contre, elle a aussi mis en avant le fait que le code est difficilement réutilisable : il a été conçu de façon complexe et il n'a pas été imaginé dans l'objectif d'être réutilisé par un utilisateur non développeur.

Le nouveau schéma ajouté est le schéma recharge batterie, qui permet au sous-marin d'aller au schnorchel (cf. la règle de comportement 6 au paragraphe 2.3).

Pour ajouter ce schéma, il a fallu créer :

- un automate à état fini (cf figure 4.12), dans lequel les états "Batterie chargée" et "Batterie à charger" ont été définis. Au bout d'un temps, noté t_{charge}, le sous-marin doit monter au schnorchel.
- la fonction de direction, qui permet de rallier un point, dont les co-

ordonnées sont données par l'utilisateur. Le sous-marin montera au schnorchel une fois ces coordonnées atteintes.

Avec ce schéma, on va se retrouver dans une situation où le sous-marin aura commencé les actions de pistage (anti-collision, mesure de distance, ...) et il va devoir exécuter sa remontée au schnorchel. Un comportement possible est alors de stopper les actions de pistage afin d'aller au schnorchel, et de reprendre ensuite les actions de pistage.

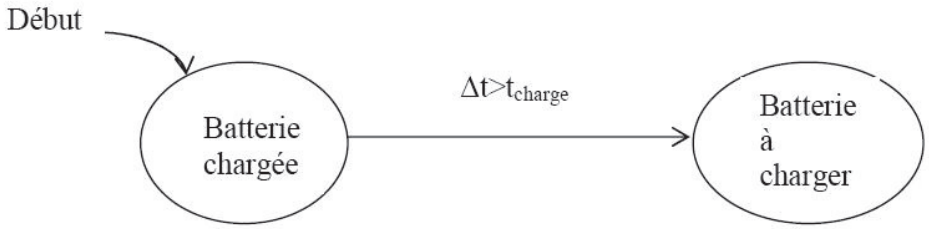

FIG. 4.12 – Automate à états finis Batterie représentant l'état de charge du sous-marin

Il nous faut maintenant fixer le gain du schéma recharge batterie, afin qu'il prenne le dessus par rapport aux comportements de pistage. Pour cela, nous fixons les gains des schémas du pistage à 1 et le gain du schéma Recharge batterie à 10.

On aura alors le comportement suivant :
Supposons que deux schémas soient activés : pistage, qui calcule la consigne $\overrightarrow{F_1}$, et recharge batterie, qui calcule la consigne $\overrightarrow{F_2}$. La consigne renvoyée au sous-marin sera alors :

$$\overrightarrow{C_A} = \frac{\overrightarrow{F_1} + \overrightarrow{F_2}}{G_1 + G_2} = \frac{G_1 * I_1 * \dfrac{\overrightarrow{AP_1}}{||AP_1||} + G_2 * I_2 * \dfrac{\overrightarrow{AP_2}}{||AP_2||}}{G_1 + G_2}$$

$$\overrightarrow{C_A} = \frac{1 * I_1 * \dfrac{\overrightarrow{AP_1}}{||AP_1||} + 10 * I_2 * \dfrac{\overrightarrow{AP_2}}{||AP_2||}}{11}$$

Avec de tels coefficients, le schéma recharge batterie devient prépondérant : le sous-marin rallie alors le point auquel il devra monter au schnorchel (cf figure 4.13).

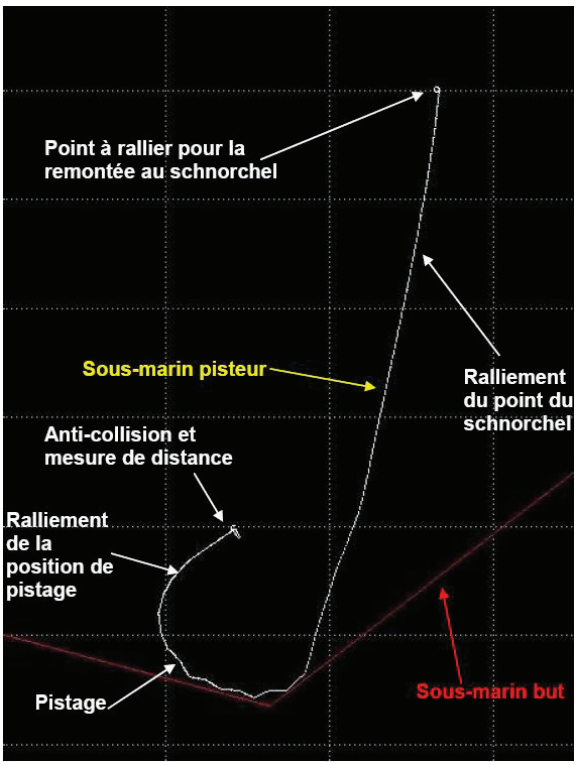

FIG. 4.13 – Exécution de la modélisation du comportement avec la théorie des schémas, interfacé avec l'atelier de simulation ATANOR : Comportement obtenu d'un sous-marin qui doit pister son ennemi et monter au schnorchel.

4.5.2 Limites de la modélisation du comportement avec la théorie des schémas

Ce nouveau schéma met en avant certaines limites de cette modélisation.

La manière de fixer les gains des schémas deviendra vite compliquée lorsque le nombre de comportements va augmenter. Il faudra en effet faire attention aux comportements qui peuvent être activés en même temps.

De plus, à chaque nouveau comportement, il faudra ajouter un nouvel automate à états finis ou encore revoir l'implémentation d'automates déjà définis. Cette contrainte ne permet pas de répondre à une de nos exigences : on veut pouvoir ajouter de nouveaux comportements sans avoir à modifier la représentation des connaissances (cf. exigence 3, paragraphe 1.3, page 27).

Les schémas ont pour objectif de donner un comportement cinématique. Ils ne peuvent pas servir à d'autres types de comportements, comme le tir de torpille.

Il faut également se demander si une somme de tous les schémas activés a bien un sens. En effet, une fois les schémas activés, la consigne finale doit permettre de remplir les objectifs de chaque schéma.

Prenons l'exemple suivant : A un même moment, le sous-marin doit :

- éviter une torpille,
- éviter un obstacle.

Deux schémas sont alors activés :

- le schéma éviter une torpille, qui calcule le vecteur $\overrightarrow{F_1}$,
- le schéma éviter un obstacle, qui calcule le vecteur $\overrightarrow{F_2}$.

Dans la configuration de la figure 4.14, le somme des deux schémas activés ne permet de remplir aucun objectif $\overrightarrow{C_A}$. A priori, le sous-marin va continuer d'avancer sur l'obstacle, sans éviter la torpille. En jouant sur les gains des schémas, nous pourrions peut être résoudre ce problème.

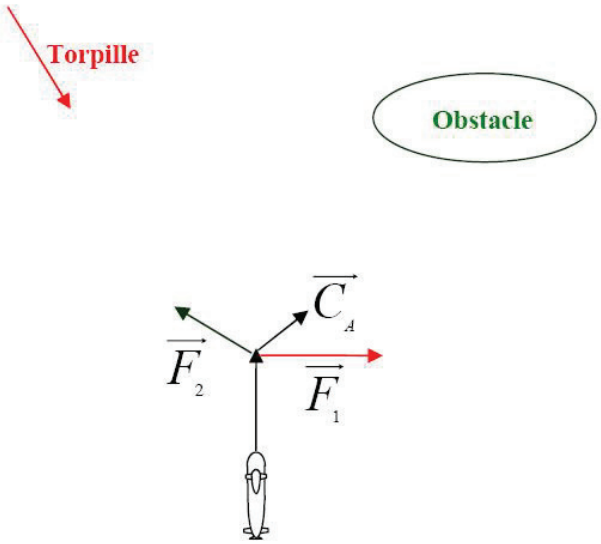

FIG. 4.14 – Résultante de deux schémas : éviter une torpille et éviter un obstacle

Finalement, la modélisation du comportement avec la théorie des schémas est efficace lorsque nous avons besoin d'un petit nombre de schémas. Par contre, cette méthode semble devenir compliquée lorsque l'on veut ajouter de nouveaux comportements.

Chapitre 5

Approche proposée : Représentation des connaissances avec la logique des défauts

Cette première étude sur la modélisation du comportement avec la théorie des schémas nous aura finalement permis de bien comprendre les règles de comportements. De plus, l'étude de ses limites nous confirme qu'il est important de pouvoir ajouter facilement de nouveaux comportements. De cette façon, la modélisation du comportement peut être reprise et enrichie par un utilisateur non développeur. Nous pouvons en déduire que, tout comme les réseaux de Pétri, l'utilisation d'automates à états finis n'est pas adaptée.

Nous utilisons une logique non-monotone pour représenter les connaissances, afin de prendre en compte les informations incomplètes, incertaines et révisables que l'opérateur possède sur son environnement. Nous avons choisi la logique non-monotone la plus utilisée : la logique des défauts. En effet, elle permet d'inférer certaines connaissances, tant qu'il n'y a pas de contradiction. Les défauts sont des règles générales, elles peuvent admettre des exceptions. L'apparition d'une nouvelle exception ne remet pas en cause les règles précédemment établies.

La logique des défauts est composée d'un ensemble de faits W et d'un ensemble de défauts D. Pour définir les faits, nous avons choisi d'utiliser les formules et les clauses de Horn. L'ensemble des défauts est défini avec des défauts normaux ([116], [111]). Notre cas d'application nous amène à prendre en compte le temps dans notre raisonnement. En effet, nous disposons des informations sur le sous-marin à un instant t et nous voulons en déduire quelle action le sous-marin va pouvoir exécuter à l'instant suivant $t+1$. On utilise donc des défauts normaux qui sont des défauts temporels.

Notre programme a été implémenté en langage Prolog. Pour le cas d'ap-

plication traité, les clauses de Horn et les défauts normaux suffisent pour représenter les connaissances. Cette représentation, implémentée en langage Prolog, nous permet d'obtenir un algorithme qui est efficace et rapide, ce qui répond aux contraintes industrielles. Ce travail pourrait être généralisé en utilisant les défauts classiques et d'autres clauses.

5.1 Formalisation des règles avec la logique des défauts et une prise en compte du temps

Nous utilisons la logique des défauts, présentée au paragraphe 3.3.1 (page 62), pour représenter les connaissances et formaliser les règles de comportement, à laquelle nous ajoutons une représentation du temps. A partir des informations dont le sous-marin dispose à l'instant t, nous en déduisons la consigne à lui renvoyer au temps suivant. Nous utilisons une gestion discrète du temps (il est incrémenté d'un pas de temps à chaque itération).

5.1.1 L'ensemble des faits W

Nous avons vu, dans le paragraphe 3.3.1, que l'ensemble des faits est défini avec le calcul propositionnel ou bien la logique du premier ordre. Les faits sont les formules toujours vraies.

Pour définir l'ensemble des faits, nous avons choisi d'utiliser :
- des formules, des littéraux terminaux (sans variables),
- des clauses de Horn.

En se limitant aux formules et aux clauses de Horn, nous obtenons un programme simple, qui nous permet de définir toutes les règles dont nous avons besoin pour décrire le comportement. Cependant, nous pourrions généraliser ce programme en utilisant d'autres clauses que les clauses de Horn.

Les formules, les littéraux terminaux (sans variables)

Les formules nous permettent de définir les informations sur notre sous-marin à un certain temps t. Notons X_t le sous-marin au temps t. Nous définissons :

1. la cinématique du sous-marin : son cap ($cap(X_t, c_t)$, avec c_t le cap à l'instant t), sa vitesse ($vitesse(X_t, v_t)$ avec v_t la vitesse à l'instant t), son immersion, sa position ;
2. les informations que le sous-marin possède sur l'ennemi : est-il détecté ? ($détection(X_t, d_t)$ avec $d_t = 0$ si le sous-marin n'a pas de détection et

$d_t = 1$ si le sous-marin a une détection), quelle est sa position estimée, son cap, sa vitesse, son gisement, son défilement ? Le sous-marin est-il dans le baffle de l'ennemi ?

3. les informations sur l'environnement : y-a-t-il un obstacle ? quelle est sa position ? ...
4. les informations temporelles : à quand remonte la dernière charge ? la dernière abattée ? quel est le temps passé en mesure de distance ? ...
5. l'état du sous-marin : dans quel état se trouve le sous-marin ?
   Les différents états possibles du sous-marin sont :
   - abattée,
   - schnorchel,
   - chasse,
   - anti-collision,
   - mesure de distance,
   - rallier le pistage,
   - pistage,
   - évitement d'obstacle,
   - lancement d'urgence,
   - dérobement.
6. la consigne que nous lui renvoyons : quelle action effectuer ? ($abattée(X_t)$, $anti\_collision(X_t)$, ...), quel cap ($tourner(X_t, angle_t)$), quelle vitesse ($changer\_vitesse(X_t, v_t)$), quelle immersion($changer\_immersion(X_t, i_t)$) doit-il prendre ? le sous-marin doit-il tirer une torpille ?

Les clauses de Horn

Nous avons défini les clauses de Horn dans le paragraphe 3.2.1 (page 53, définition 6). Ce sont des clauses avec, au plus, un littéral positif. Elles nous permettent principalement de définir les règles de navigation : quel est le cap à prendre, quelles actions ne sont pas compatibles ...

Utilisation des clauses de Horn avec un littéral positif Ce sont les règles de la forme :

$$(f1 \wedge f2 \wedge f3 \wedge \ldots \wedge fk) \rightarrow g$$

où les fi et g sont des littéraux positifs. On ajoute la représentation du temps :

$$(f1(t) \wedge f2(t) \wedge f3(t) \wedge \ldots \wedge fk(t)) \rightarrow g(t)$$

où les fi(t) et g(t) sont des littéraux positifs correspondant à l'état du sous-marin au temps t.

Ce type de règles nous permet de définir des règles toujours vraies, que nous appellerons règles dures. Ce sont les règles classiques de système expert. Nous définissons la règle suivante :

$$r1 : abattée(X_t) \rightarrow tourner(X_t, (\alpha, \beta)),$$

qui signifie : "Si le sous-marin est en abattée (au temps t), il tournera d'un angle compris entre α et β.".

Les clauses de Horn sont des règles facilement compréhensibles pour les utilisateurs.

Utilisation des clauses de Horn avec aucun littéral positif (clauses négatives) Ce sont les règles de la forme :

$$\neg f1 \vee \neg f2 \vee \neg f3 \vee \ldots \vee \neg fk$$

ou encore

$$(f1 \wedge f2 \wedge f3 \wedge \ldots \wedge fk) \rightarrow FAUX,$$

ou encore

$$\neg(f1 \wedge f2 \wedge f3 \wedge \ldots \wedge fk).$$

Avec la représentation du temps, nous définissons des règles de la forme :

$$\neg(f1(t+1) \wedge f2(t+1) \wedge f3(t+1) \wedge \ldots \wedge fk(t+1)).$$

Ce type de clause nous permet de définir les exclusions mutuelles, c'est-à-dire les prédicats qui ne peuvent être exécutés en même temps.
Par exemple, nous définissons la règle suivante :

$$w1 = \neg(abattée(X_{t+1}) \wedge schnorchel(X_{t+1})).$$

Au même instant $t+1$, un sous-marin ne peut pas faire à la fois une abattée et remonter au schnorchel.

On peut également définir des règles de la forme :

$$\neg(f1(t) \wedge f2(t) \wedge f3(t) \wedge \ldots \wedge fk(t+1)).$$

Cela signifie que si les faits f1 et f2 et f3 ...sont vérifiés au temps t, il n'est pas possible d'exécuter l'action fk au temps suivant (t+1).
Prenons un autre exemple, la règle : "Si le sous-marin ennemi manœuvre, le sous-marin ne peut pas rallier le poste de pistage" peut être écrite de la façon suivante :

$$w2 = \neg(ennemi\_manoeuvre(X_t) \wedge rallier\_pistage(X_{t+1})).$$

5.1.2 Les défauts D

Les défauts sont des règles d'inférence à contenu spécifique qui permettent de gérer l'incertitude (cf. paragraphe 3.3.1). Ils nous permettent d'exprimer le fait que, si il n'y a pas de contradiction à exécuter une action, le sous-marin peut le faire.

Un défaut est défini de la façon suivante :

$$\frac{A(X) : B(X)}{C(X)},$$

où A(X), B(X) et C(X) sont des formules bien formées qui contiennent X comme variable libre ou $X = (x_1, x_2, x_3, \ldots, x_n)$ comme vecteur de variable libre.

Nous utilisons seulement les défauts normaux :

$$\frac{A(X) : C(X)}{C(X)}.$$

D'après Reiter [90], l'utilisation des défauts normaux nous permet d'assurer l'existence d'au moins une extension. De plus, l'algorithme de calcul d'extensions sera plus simple. Même si nous nous sommes limités aux défauts normaux, ce travail pourrait être généralisé en utilisant des défauts généraux.

Pour utiliser un défaut normal, on vérifie que A(X) est vrai et qu'il est possible que C(X) soit vrai (c'est-à-dire C(X) est consistant). On peut alors inférer C(X). Dans notre cas d'application, nous prenons en compte le temps : à partir des données dont le sous-marin dispose à l'instant t, quelles actions va-t-il pouvoir exécuter à l'instant suivant ? Pour utiliser un défaut à l'instant t, on vérifie que A(X) est vrai à cet instant t. Par contre, on vérifie la consistance de C(X) avec la base de connaissance à l'instant suivant $t + 1$. Et on infère C(X) à l'instant $t+1$. Pour utiliser les défauts, on procède donc de la façon suivante :

- on vérifie le prérequis $A(X)$ à un instant t : $A(X_t)$, la variable X est alors instanciée à l'instant t : $X_t = (x_{1t}, x_{2t}, x_{3t}, \ldots, x_{nt})$;
- on vérifie qu'il n'y a pas de contradiction à la justification $C(X)$ à l'instant $t + 1 : C(X_{t+1})$;
- on ajoute le conséquent $C(X)$ au temps $t + 1 : C(X_{t+1})$.

On définit finalement les défauts de la façon suivante :

$$\frac{A(X_t) : C(X_{t+1})}{C(X_{t+1})}$$

Ce défaut signifie : "Si le prérequis A(X) est vérifié au temps t, et si il est possible que C(X) soit vrai au temps t+1, alors C(X) est vrai au temps t+1."

On pourrait également vérifier la consistance de C(X) avec la base de connaissance à l'instant suivant $t+2$, $t+3$, ..., $t+n$, cependant nous aurions alors des problèmes de complexité.

Pour définir la règle " Si le sous-marin n'a pas de détection, alors il fait une abattée. ", nous définissons le défaut suivant :

$$\frac{détection(X_t, 0) : abattée(X_{t+1})}{abattée(X_{t+1}, 1)}.$$

Finalement, ce défaut signifie : " Si le sous-marin n'a pas de détection à l'instant t, et si il lui est possible de faire une abattée à l'instant t+1, alors il fera une abattée à l'instant t+1.".

La révision

Nous possédons une base de connaissance dynamique : la base de connaissance peut changer. Par exemple, le sous-marin n'a pas de détection et, à un moment, il détecte un ennemi. Ou encore, la distance entre les deux sous-marin change à chaque instant, en fonction de leurs trajectoires ...

Des nouvelles informations peuvent contredire la base de connaissance. Il faut donc la réviser afin d'éviter les inconsistances et de garder une base de connaissance cohérente.

A chaque pas de temps, les formules de la base de connaissance sont mises à jour :

- les informations sur l'ennemi (détection, position, cap, vitesse, gisement, défilement, ...)
- les informations sur l'environnement (position d'un éventuel obstacle ...)
- les informations temporelles (temps depuis la dernière charge, le dernière abattée, ...)
- les informations sur le sous-marin : en fonction de la consigne calculée (cap, vitesse, immersion), quelle est la nouvelle position du sous-marin et son nouvel état ? A-t-il tiré une torpille ?

5.1.3 Consignes renvoyées au sous-marin

Dans le paragraphe 5.1.1, nous avons défini les consignes données au sous-marin : nous devons lui renvoyer un cap, une vitesse, une immersion, et l'ordre éventuel de tirer une torpille.

Nous devons prendre en compte le fait qu'une même action peut avoir plusieurs consignes possibles. En effet, pour certaines actions, nous avons une marge de manœuvre. Le sous-marin ne doit pas forcément atteindre un cap

précis mais ce cap peut être compris dans un certain intervalle. On raisonne de la même façon avec la vitesse et l'immersion.

Prenons la règle de l'abattée (cf paragraphe 2.3) simplifiée :

- si le sous-marin n'a pas de détection, il doit se mettre en abattée ;
- un sous-marin en abattée doit tourner d'un angle compris entre 60 ° et 120 °.

On peut traduire ces règles de la façon suivante :

- avec un défaut :

$$\frac{détection(X_t, 0) : abattée(X_{t+1})}{abattée(X_{t+1})},$$

- et une règle :

$$abattée(X_{t+1}) \rightarrow tourner(X_{t+1}, 60) \ \vee \ tourner(X_{t+1}, 61) \ \vee \ \ldots \\ \ldots \vee \ tourner(X_{t+1}, 120).$$

Le "ou" entre chaque prédicat *tourner* est un "ou" exclusif : le sous-marin ne peut prendre qu'un seul cap.

Le calcul obtenu avec l'algorithme d'extension donnera plusieurs extensions :

$$E1 = [abattée(X_{t+1}), tourner(X_{t+1}, 60)],$$

$$E2 = [abattée(X_{t+1}), tourner(X_{t+1}, 61)],$$

$$\ldots$$

$$E61 = [abattée(X_{t+1}), tourner(X_{t+1}, 120)].$$

Ce calcul est long alors que nous obtenons des extensions pratiquement identiques, avec pour seule différence la consigne de cap. Ce calcul est donc répétitif, sans pour autant apporter de l'intérêt à la résolution.

Nous avons donc décidé de regrouper toutes ces extensions en une seule, en introduisant la consigne qui doit permettre au sous-marin de tourner d'un angle compris dans une certaine plage de cap. Cette consigne est donnée avec le prédicat $tourner\_plage(\alpha, \beta)$: le sous-marin doit tourner d'un angle compris entre α ° et β °. Pour simplifier l'écriture, on note ce prédicat $tourner(\alpha, \beta)$. Nous reprenons l'idée des symétries [17] qui permet de simplifier considérablement la résolution des problèmes qui présentent une structure très symétrique.

L'exemple précédent est alors simplifié : nous retrouvons le même défaut

$$\frac{détection(X_t, 0) : abattée(X_{t+1})}{abattée(X_{t+1})}.$$

La règle pour donner la consigne de cap est maintenant simplifiée. Nous avons la clause de Horn suivante :

$$abattée(X_{t+1}) \rightarrow tourner(X_{t+1}, (60, 120)).$$

Et le calcul d'extension ne nous donne qu'une seule extension, ce qui simplifie la complexité du calcul :

$$E1 = [abattée(X_{t+1}), tourner(X_{t+1}, (60, 120))].$$

On peut raisonner de même avec les consignes de vitesse et d'immersion, en définissant les prédicats suivants :

- changer_vitesse$(X_{t+1}, (V_1, V_2))$: ce prédicat permet au sous-marin de changer de vitesse, il doit prendre une vitesse comprise entre V_1 et V_2 ;
- changer_immersion$(X_{t+1}, (I_1, I_2))$: ce prédicat permet au sous-marin de changer d'immersion, il doit prendre une immersion comprise entre I_1 et I_2.

5.1.4 Ecriture des règles

Dans ce paragraphe, nous donnons la formalisation des règles décrites au paragraphe 2.3 en logique des défauts.

Pour chaque état possible, les règles dures concernant la consigne d'immersion et de vitesse ne sont pas détaillées. Elles sont de la même forme que les règles suivantes, données pour l'exemple de l'abattée :

$r1 : abattée(X_t) \rightarrow changer\_vitesse(X_t, (Va, Va))$, avec Va, la vitesse à adopter en abattée.

$r2 : abattée(X_t) \rightarrow changer\_immersion(X_t, (Ia, Ia))$, avec Ia, la différence d'immersion que le sous-marin doit adopter par rapport à l'immersion actuelle.

Les défauts et les clauses de Horn avec un littéral positif

Règle 1 : Les abattées Nous avons déjà utilisé comme exemple le défaut pour la règle de l'abattée :

$$D1 = \frac{détection(X_t, 0) : abattée(X_{t+1})}{abattée(X_{t+1})}$$

Nous définissons ensuite les règles dures. On note Tc, le temps courant et Ta le temps auquel l'officier a décidé de faire la prochaine abattée (c'est-à-dire le prochain changement de cap). Si ce temps est atteint, il peut alors changer de cap, sinon il continue tout droit.
On a alors les deux règles suivantes :

$$r3 : abattée(X_t) \wedge (Tc < Ta(X_t)) \rightarrow tourner(X_t, (0, 0))$$

$$r4 : abattée(X_t) \wedge (Tc \geq Ta(X_t)) \rightarrow tourner(X_t, (60, 120)) \vee tourner(X_t, (-120, -60))$$

Le "ou" $\vee$ n'a pas été défini. On le remplace par deux règles :
$r4a : abattée(X_t) \wedge (Tc >= Ta(X_t)) \rightarrow tourner(X_t, (60, 120))$

$r4b : abattée(X_t) \wedge (Tc >= Ta(X_t)) \rightarrow tourner(X_t, (-120, -60))$

Un choix aléatoire permet de choisir une des deux règles. Ce choix aléatoire reflète le comportement de l'officier, qui doit effectuer une trajectoire aléatoire de recherche.

Règle 2 : La "chasse" : A partir du moment où le sous-marin a une détection (on note $détection(X_t, 1)$), il se trouve en état "chasse" :

$$D2 = \frac{détection(X_t, 1) : chasse(X_{t+1})}{chasse(X_{t+1})}$$

L'anti-collision : Lorsque l'officier détecte un sous-marin, il doit immédiatement effectuer la manœuvre d'anti-collision. On a défini le défaut suivant :

$$D3 = \frac{chasse(X_t) \wedge \neg var\_anti\_collision(X_t) : anti\_collision(X_{t+1})}{anti\_collision(X_{t+1})}$$

La formule $var\_anti\_collision(X_t)$ est à 0 si la manœuvre d'anti-collision n'a pas été faite et passe à 1 lorsqu'elle est terminée.

On a ensuite les règles dures qui permettent d'indiquer le changement de cap, en fonction du défilement De et du gisement G :

$r6$: $anti\_collision(X_t) \wedge (\gamma < |De| < \omega) \wedge (\mu <= |G| <= \rho) \rightarrow tourner(X_t, (10, 20))$ (le sous-marin doit aller à droite).

$r7$: $anti\_collision(X_t) \wedge (\gamma' < |De| < \omega') \wedge (\mu' <= |G| <= \rho') \rightarrow tourner(X_t, (-20, -10))$ (le sous-marin doit aller à gauche).

Manœuvre de mesure de distance : On note Tmd le temps passé en mesure de distance. Sachant que cette manœuvre est composée de trois tronçons de 10 minutes chacun, on y reste tant que le temps Tmd n'a pas dépassé 30 minutes. On définit le défaut suivant :

$$D4 = \frac{chasse(X_t) \wedge Tmd(X_t) \leq 30 : mesure\_distance(X_{t+1})}{mesure\_distance(X_{t+1})}$$

Afin de ne pas avoir de conflit entre les défauts D3 et D4, nous avons défini l'exclusion mutuelle suivante :

$$w1 = \neg(\neg var\_anti\_collision(X_t) \wedge mesure\_distance(X_{t+1}))$$

Elle empêche d'exécuter la mesure de distance, tant que l'anti-collision n'a pas été effectuée.

Les changements de cap pour les tronçons sont effectués toutes les 10 minutes. On définit les règles suivantes :

$$r8 : mesure\_distance(X_t) \wedge Tmd(X_t) \neq 10 \wedge Tmd(X_t) \neq 20 \rightarrow tourner(X_t, (0, 0))$$

$$r9 : mesure\_distance(X_t) \wedge Tmd(X_t) = 10 \rightarrow tourner(X_t, (150, 160))$$

$$r10 : mesure\_distance(X_t) \wedge Tmd(X_t) = 20 \rightarrow tourner(X_t, (-160, -150))$$

Ralliement de la position de pistage Une fois la mesure de distance effectuée, le sous-marin peut effectuer le ralliement de la position de pistage. Il cherche à se mettre dans le baffle de l'ennemi.

$$D5 = \frac{chasse(X_t) \wedge Tmd(X_t) > 30 : rallier\_pistage(X_{t+1})}{rallier\_pistage(X_{t+1})}$$

Si l'ennemi manœuvre, le sous-marin ne pourra plus rallier le pistage mais il devra effectuer un lancement d'urgence. On a donc défini l'exclusion mutuelle suivante, avec la formule *ennemi_manoeuvre* :

$$w2 = \neg(ennemi\_manoeuvre(X_t) \wedge rallier\_pistage(X_{t+1}))$$

Une fois positionné dans le baffle de l'ennemi, le sous-marin aura fini son ralliement de la position de pistage. On définit cette règle avec l'exclusion mutuelle :

$$w3 = \neg(dans\_baffle(X_t) \wedge rallier\_pistage(X_t))$$

Pour rallier la position de pistage, le sous-marin doit prendre le cap qui vise la position de l'ennemi. On a donc la règle suivante :

$$r11 : rallier\_pistage(X_t) \rightarrow tourner(X_t, ((Ce - C), (Ce - C))),$$

avec Ce le cap qui vise l'ennemi et C le cap actuel du sous-marin.
La consigne $tourner(X_t, ((Ce - C), (Ce - C)))$ revient à donner au sous-marin la consigne de tourner d'un angle $(Ce - C)$. Le sous-marin doit alors prendre un cap de $C + (Ce - C) = Ce$. Il prend donc bien le cap qui vise l'ennemi.

Pistage du sous-marin ennemi Le défaut suivant permet au sous-marin de se mettre en pistage :

$$D6 = \frac{chasse(X_t) \wedge Tmd(X_t) > 30 : pistage(X_{t+1})}{pistage(X_{t+1})}$$

Le pistage est effectué en tronçons de 10 minutes. Toutes les dix minutes, le sous-marin doit changer de cap. Ce cap doit permettre au sous-marin de placer l'ennemi successivement dans un gisement de 330 ° (ou -30 °)et de 30 ° (cf figure 5.1). On note Tpp le temps auquel le sous-marin devra changer de cap. La formule *basc* change de valeur à chaque fin de tronçon et permet de se placer successivement d'un côté et de l'autre de l'ennemi.

$$r12 : pistage(X_t) \wedge (Tc > Tpp(X_t)) \wedge basc(1) \rightarrow tourner(X_t, ((30 + Ce - C), (30 + Ce - C)))$$

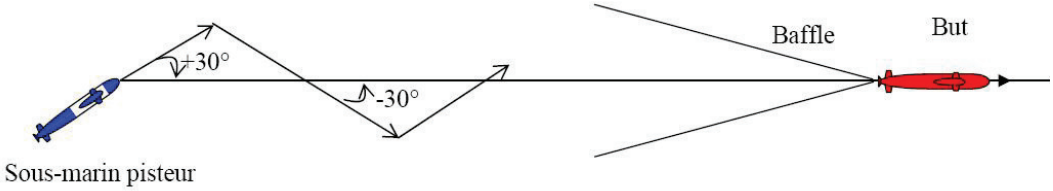

FIG. 5.1 – Manœuvre de pistage, positionnement de l'ennemi dans des gisements successifs de -30 ° et de 30 °

$$r13 : pistage(X_t) \wedge (Tc > Tpp(X_t)) \wedge basc(0) \rightarrow tourner(X_t, ((-30 + Ce - C), (-30 + Ce - C)))$$

$$r14 : pistage(X_t) \wedge (Tc \leq Tpp(X_t)) \rightarrow tourner(X_t, (0, 0))$$

Si l'ennemi manœuvre, le pistage doit être stoppé :

$$w4 = \neg(ennemi\_manoeuvre(X_t) \wedge pistage(X_t))$$

Règle 3 : Dérobement Le sous-marin doit se dérober dans les cas suivants :

- pendant la manœuvre de mesure de distance, le sous-marin ennemi manœuvre et nous sommes en temps de crise (noté TC),
- les deux sous-marins sont trop proches (on note la distance entre les deux sous-marins D_ennemi). On considère ici que les deux sous-marins sont trop proches si la distance est inférieure à 2000 mètres.

On a donc deux défauts :

$$D7 = \frac{détection(X_t,1) \wedge Tmd(X_t) <= 30 \wedge ennemi\_manoeuvre(X_t) \wedge TC : dérobement(X_{t+1})}{dérobement(X_{t+1})}$$

$$D8 = \frac{détection(X_t, 1)) \wedge D\_ennemi <= 2000 : dérobement(X_{t+1})}{dérobement(X_{t+1})}$$

L'objectif est de placer l'ennemi dans un gisement arrière. Le sous-marin peut, par exemple, placer l'ennemi dans un gisement de 150 °. La consigne

de cap est donc définie par :

$$r15 : \textit{dérobement}(X_t) \rightarrow \textit{tourner}(X_t, ((150 + Ce - C), (150 + Ce - C)))$$

Règle 4 : Lancement d'urgence Le sous-marin doit effectuer un lancement d'urgence dans les cas suivants :

- pendant la manœuvre de mesure de distance, le sous-marin ennemi manœuvre et nous sommes en temps de guerre (noté TG),
- l'ennemi s'apprête à tirer une torpille (défini par la formule tir_ennemi).

On a les défauts :

$$D9 = \frac{\textit{détection}(X_t,1) \wedge Tmd(X_t) <= 30 \wedge ennemi\_manoeuvre(X_t) \wedge TG : lancement\_urgence(X_{t+1})}{lancement\_urgence(X_{t+1})}$$

$$D10 = \frac{\textit{détection}(X_t, 1) \wedge tir\_ennemi(X_t) : lancement\_urgence(X_{t+1})}{lancement\_urgence(X_{t+1})}$$

Dans un premier temps, le sous-marin doit tirer une torpille :

$$r16 : lancement\_urgence(X_{t+1}) \rightarrow tir(X_{t+1}, 1)$$

Une fois la torpille lancée dans l'azimut de détection, le sous-marin manœuvre de façon à dérober. Il peut placer l'ennemi dans un gisement arrière de 140 °.

$$r17 : lancement\_urgence(X_t) \rightarrow \textit{tourner}(X_t, ((135 + Ce - C), (145 + Ce - C)))$$

Règle 5 : Perte de la détection Lorsque le sous-marin a perdu la détection depuis un certain temps, on note la détection $\textit{détection}(X_t, 2)$.

Dans ce cas là, le sous-marin doit alors rechercher la piste :

$$D11 = \frac{\textit{détection}(X_t, 2) : recherche\_piste(X_{t+1})}{recherche\_piste(X_{t+1})}$$

Il rallie la dernière position connue du but :

$$\text{r18} : recherche\_piste(X_t) \rightarrow \textit{tourner}(X_t, ((Ce - C), (Ce - C)))$$

Règle 6 : Schnorchel Un sous-marin diesel doit recharger ses batteries toutes les 4 heures. On définit le défaut suivant :

$$D12 = \frac{Tdch > 4h : schnorchel(X_{t+1})}{schnorchel(X_{t+1})}$$

Une fois au schnorchel, le sous-marin doit remonter à la surface. On note I l'immersion du sous-marin et Tdch le temps passé en surface. On suppose que le sous-marin doit passer 20 minutes en surface.

$$r19 : schnorchel(X_t) \wedge (I < 0) \wedge (Tch < 20min) \rightarrow changer\_immersion(X_t, (20, 30))$$

$$r20 : schnorchel(X_t) \wedge (I = 0) \wedge (Tch < 20min) \rightarrow tourner(X_t, (0, 0))$$

$$r21 : schnorchel(X_t) \wedge (Tch = 20min) \rightarrow changer\_immersion(X_t, (-30, -20))$$

Règle 7 : Evitement d'obstacle A l'aide du sonar à évitement d'obstacle MOAS, le sous-marin peut détecter des mines, des gros rochers, ou encore des falaises. Le MOAS a une portée comprise entre 1500 mètres et 3000 mètres.

Si le sous-marin détecte un obstacle, il change de cap de manière à l'éviter. Par exemple, il peut le placer dans un gisement proche de 150 °.

Lorsqu'un obstacle est trop proche du sous-marin, il doit l'éviter. On note $D\_obstacle$, la distance entre l'obstacle et le sous-marin :

$$D13 = \frac{D\_obstacle < 2000 : eviter\_obstacle(X_{t+1})}{eviter\_obstacle(X_{t+1})}$$

Le sous-marin cherche à placer l'obstacle dans un gisement proche de 150 °. On note :

$$r22 : éviter\_obstacle(X_t) \rightarrow tourner((X_t), (145 + Cob - C), (155 + Cob - C))$$

Les exclusions mutuelles (ou clauses négatives)

Nous pouvons facilement définir les exclusions mutuelles. Nous en avons déjà énuméré quelques exemples dans le paragraphe précédent (w1,w2,w3,w4).

Afin d'avoir la consigne la plus efficace possible pour le sous-marin, il est intéressant de pouvoir lui faire exécuter plusieurs actions à la fois. Pour cela, nous spécifions les cas dans lesquels le sous-marin va pouvoir accomplir plusieurs actions à la fois.

Prenons un cas dans lequel le sous-marin pourrait, à un même instant, faire deux actions à la fois. Notons les "action 1" et "action 2". Chacune de ces actions va proposer une consigne de cap, vitesse, immersion et tir.

Si l'action 1 propose $tourner(X_t, (60, 100))$ et l'action 2 propose $tourner(X_t, (80, 110))$, la meilleure consigne que l'on puisse renvoyer au sous-marin est : $tourner(X_t, (80, 100))$ de façon à ce qu'il accomplisse les deux actions à la fois. Le sous-marin aura alors le choix de tourner d'un angle compris entre 80 ° et 100 ° . Ce choix est possible à condition que les actions 1 et 2 ne soient pas exclues mutuellement.

Nous pouvons également combiner les consignes sur la vitesse et l'immersion de la même façon.

Nous définissons les règles d'exclusion mutuelle entre les consignes de cap (c'est-à-dire les cas où il n'existe pas de plage de cap commune), les consignes de vitesse et les consignes d'immersion.

$$w5 = \neg(tourner(X_t, (\alpha, \beta)) \wedge tourner(X_t, (\delta, \lambda)) \wedge (\beta < \delta))$$

$$w6 = \neg(tourner(X_t, (\alpha, \beta)) \wedge tourner(X_t, (\delta, \lambda)) \wedge (\alpha > \lambda))$$

$$w7 = \neg(changer\_vitesse(X_t, (\alpha, \beta)) \wedge changer\_vitesse(X_t, (\delta, \lambda)) \wedge (\beta < \delta))$$

$$w8 = \neg(changer\_vitesse(X_t, (\alpha, \beta)) \wedge changer\_vitesse(X_t, (\delta, \lambda)) \wedge (\alpha > \lambda))$$

$$w9 = \neg(changer\_immersion(X_t, (\alpha, \beta)) \wedge changer\_immersion(X_t, (\delta, \lambda)) \wedge (\beta < \delta))$$

$$w10 = \neg(changer\_immersion(X_t, (\alpha, \beta)) \wedge changer\_immersion(X_t, (\delta, \lambda)) \wedge (\alpha > \lambda))$$

5.2 Exemple de calcul

Pour illustrer l'utilisation de la logique des défauts, nous donnons dans ce paragraphe un exemple de calcul.

5.2.1 Les faits W et les défauts D

Les informations sur le sous-marin sont les suivantes :
- $détection(X_t, 1)$: le sous-marin a détecté l'ennemi ;
- $Tmd(X_t) > 30$: le sous-marin a déjà effectué le manœuvre de mesure de distance ;
- $dans\_baffle(X_t, 1)$: le sous-marin est dans le baffle de l'ennemi ;
- $D\_ennemi < 2000m$: la distance entre les deux sous-marins est inférieure à 2000 mètres ;
- $Tc \leq Tpp(X_t)$: le sous-marin n'a pas encore atteint le temps auquel il devra changer de cap dans son pistage (c'est-à-dire faire un nouveau tronçons).

On reprend quelques règles définies au paragraphe 5.1.4 :

$$w3 = \neg(dans\_baffle(X_t) \wedge rallier\_pistage(X_t))$$

$$r12 : pistage(X_t) \wedge (Tc > Tpp(X_t)) \wedge basc(1) \rightarrow tourner(X_t, ((30 + Ce - C), (30 + Ce - C)))$$

$$r13 : pistage(X_t) \wedge (Tc > Tpp(X_t)) \wedge basc(0) \rightarrow tourner(X_t, ((-30 + Ce - C), (-30 + Ce - C)))$$

$$r14 : pistage(X_t) \wedge (Tc \leq Tpp(X_t)) \rightarrow tourner(X_t, (0, 0))$$

$$r15 : dérobement(X_t) \rightarrow tourner(X_t, ((150 + Ce - C), (150 + Ce - C)))$$

$$D2 = \frac{détection(X_t, 1) : chasse(X_{t+1})}{chasse(X_{t+1})}$$

$$D5 = \frac{chasse(X_t) \wedge Tmd(X_t) > 30 : rallier\_pistage(X_{t+1})}{rallier\_pistage(X_{t+1})}$$

$$D6 = \frac{chasse(X_t) \wedge Tmd(X_t) > 30 : pistage(X_{t+1})}{pistage(X_{t+1})}$$

$$D8 = \frac{détection(X_t, 1) \wedge D\_ennemi <= 2000 : dérobement(X_{t+1})}{dérobement(X_{t+1})}$$

On rajoute les règles suivantes :

$$w11 = \neg(dérobement(X_t) \wedge pistage(X_t))$$

(exclusion mutuelle entre le dérobement et le pistage)

$$r23 : dérobement(X_t) \rightarrow changer\_vitesse(X_t, (13, 15))$$

$$r24 : pistage(X_t) \rightarrow changer\_vitesse(X_t, (1.2 * Ve, 1.2 * Ve)),$$

avec Ve la vitesse de l'ennemi.

$$r25 : dérobement(X_t) \rightarrow changer\_immersion(X_t, (0, 0))$$

$$r26 : pistage(X_t) \rightarrow changer\_immersion(X_t, (0, 0))$$

5.2.2 Calcul des extensions

Pour le calcul des extensions, nous avons :

$$E_0 = \{détection(X_t, 1), Tmd(X_t) > 30, dans\_baffle(X_t, 1), \\ D\_ennemi < 2000m, Tc \leq Tpp(X_t)\}.$$

Le défaut D5 ne pourra en aucun cas être appliqué : son prérequis est bien rempli mais on a l'exclusion mutuelle :

$$w3 = \neg(dans\_baffle(X_t) \wedge rallier\_pistage(X_t)),$$

qui nous empêche de l'utiliser. Le sous-marin se trouvant dans le baffle du sous-marin ennemi, il a déjà rallié son poste de pistage.

On calcule la première extension :

$$\begin{aligned} E_1 = Th(E_0) \cup \{ & chasse(X_{t+1}) / \frac{détection(X_t, 1) : chasse(X_{t+1})}{chasse(X_{t+1})} \in D, \\ & détection(X_t, 1) \in E_0, \neg chasse(X_{t+1}) \notin E_0, \\ & pistage(X_{t+1}) / \frac{chasse(X_t) \wedge Tmd(X_t) > 30 : pistage(X_{t+1})}{pistage(X_{t+1})} \in D, \\ & chasse(X_t) \in E_1, \neg pistage(X_{t+1}) \notin E_0\} \end{aligned}$$

Les défauts D2 et D6 peuvent donc être utilisés et on ajoute $chasse(X_{t+1})$ et $pistage(X_{t+1})$ dans l'extension. On ne peut alors pas appliquer le défaut D8 à cause de l'exclusion mutuelle w11.

On obtient alors une première extension :

$$\begin{aligned} E_1 = \{ & détection(X_t, 1), Tmd(X_t) > 30, dans\_baffle(X_t, 1), \\ & D\_ennemi < 2000m, Tc \leq Tpp(X_t), chasse(X_{t+1}), pistage(X_t), \\ & tourner(X_t, (0, 0)), changer\_vitesse(X_t, (1.2 * Ve, 1.2 * Ve)), \\ & changer\_immersion(X_t, (0, 0))\} \end{aligned}$$

De la même façon, nous calculons la deuxième extension en utilisant les défauts D2 et D8 :

$$\begin{aligned} E_2 = Th(E_0) \cup \{ & chasse(X_{t+1}) / \frac{détection(X_t, 1) : chasse(X_{t+1})}{chasse(X_{t+1})} \in D, \\ & détection(X_t, 1) \in E_0, \neg chasse(X_{t+1}) \notin E_0, dérobement(X_{t+1}) / \\ & \frac{détection(X_t, 1) \wedge D\_ennemi \leq 2000 : dérobement(X_{t+1})}{dérobement(X_{t+1})} \in D, \\ & D\_ennemi \leq 2000 \in E_1, détection(X_t, 1) \in E_1, \neg dérobement(X_{t+1}) \notin E_0\} \end{aligned}$$

On ajoute donc $dérobement(X_{t+1})$ et $chasse(X_{t+1})$ dans l'extension. On ne peut alors pas appliquer le défaut D6 à cause de l'exclusion mutuelle w11.

On obtient finalement la deuxième extension :

$$E_2 = \{détection(X_t, 1), Tmd(X_t) > 30, dans\_baffle(X_t, 1),$$
$$D\_ennemi < 2000m, Tc \leq Tpp(X_t), chasse(X_{t+1}), dérobement(X_{t+1}),$$
$$tourner(X_t, ((150+Ce-C), (150+Ce-C))), changer\_vitesse(X_t, (13, 15)),$$
$$changer\_immersion(X_t, (0, 0))\}$$

Finalement, nous obtenons deux extensions : une qui propose le pistage et la seconde qui propose au sous-marin de se dérober. La logique des défauts nous permet bien d'obtenir l'ensemble des manœuvres possibles pour le sous-marin. Il reste maintenant à gérer les choix entre les extensions. Le choix de l'extension est présenté par la suite dans le chapitre 7.

5.3 Implémentation

Nous avons choisi d'implémenter notre programme en langage Prolog. Nous l'appelons NoMROD, pour Non-Monotonic Reasoning for Operator Decision.

5.3.1 Le langage Prolog

Prolog, abréviation de PROgrammation LOGique, est le premier langage de programmation logique, créé par Alain Colmerauer [27], Philippe Roussel, Jean Trudel, Robert Kowalski et Robert Pasero en 1972. Ce langage a été développé afin de représenter les connaissances en logique et non pas de décrire la suite d'instructions que les machines doivent exécuter.

C'est un langage initialement basé sur la logique du premier ordre, restreinte aux clauses de Horn. Il a ensuite été enrichi avec, entre autres, la négation. Ce langage continue d'être enrichi encore aujourd'hui. Prolog est adapté aux problèmes de planification, raisonnement automatique (systèmes experts, diagnostics), traitement de la langue naturelle ...

En Prolog, toutes les fonctionnalités sont programmées de manière récursive. Pour appliquer un prédicat sur une liste, Prolog applique le prédicat sur le premier élément de la liste. Ensuite, il applique récursivement le prédicat sur le reste de la liste. Prolog n'est pas un langage impératif comme les langages C ou C++.

La description de connaissances se fait en définissant :

- des faits. Par exemple, "Le sous-marin sm1 a un cap de 5 ° " se définit par "cap([sm1,5])", avec cap un prédicat.

- des règles. Une règle de la forme "$P \wedge Q \rightarrow R$" s'écrit
  "R(X) :- P(X),Q(X)".
  La règle suivante : "Si le sous-marin a une détection, il se met en chasse" peut s'écrire : $chasse(X)$:- $detection(X)$.

A partir de ces connaissances, le moteur d'inférence déduit de nouvelles connaissances. Une des particularités de Prolog est que l'on peut construire une base de connaissance dans un ordre indéterminé. Prolog peut ensuite résoudre des séries de problèmes logiques relatifs à une telle base de connaissance.

5.3.2 Implémentation en Prolog

Finalement, le langage Prolog nous permet très simplement de définir l'ensemble des faits et les défauts.

L'ensemble des faits

Les formules Les formules sont définies comme des prédicats dynamiques, afin de pouvoir ajouter de nouvelles clauses de manière simple. Pour cela, en début de programme, il faut déclarer les prédicats dynamiques. Pour déclarer le prédicat dynamique *cap* d'arité 1, on définit :
:- $dynamic(cap/1)$.

Lorsque le sous-marin change de cap, on utilise le prédicat *retract* (*exemple : retract(cap([sm1,60])), avec sm1 le nom du sous-marin concerné et 60 ° son cap actuel*) pour retirer la clause puis le prédicat *asserta* (*exemple : asserta(cap([sm1,90]))*) pour ajouter la nouvelle clause.

Les clauses de Horn

Les clauses de Horn avec un littéral positif Ces clauses, appelées règles dures, sont définies avec le prédicat "regle_dure".
On reprend une règle dure définie pour les abattées, au paragraphe 5.1.4 :

$$r1 : abattée(X_t) \rightarrow changer\_vitesse(X_t, (Va, Va)),$$

avec Va, la vitesse à adopter en abattée. Cette règle permet de fixer la vitesse du sous-marin lorsqu'il est en abattée.

En prolog, cette règle est définie de la façon suivante :

$$regle\_dure(abattee([Nom]), changer\_vitesse([Nom], (Va, Va))) :- v\_abattee([Nom, Va]).$$

Avec :

- $abattee([Nom])$: prédicat qui définit l'état abattée du sous-marin, de nom "Nom".
- $changer\_vitesse([Nom], (Va, Va))$: prédicat qui définit la consigne de vitesse du sous-marin de nom "Nom". La vitesse du sous-marin doit être comprise entre Va et Va, le sous-marin doit donc adopter la vitesse Va.
- $v\_abattee([Nom, Va])$: ce prédicat définit la vitesse Va que le sous-marin doit adopter en abattée.

Les clauses de Horn sans littéral positif Pour définir les exclusions mutuelles, nous avons défini un prédicat *non*. Prenons une exclusion mutuelle :

$$w1 = \neg(abattee(X_{t+1}) \wedge schnorchel(X_{t+1})),$$

qui signifie que, au même instant $t + 1$, un sous-marin ne peut pas faire à la fois une abattée et remonter au schnorchel. Cette règle est définie de la façon suivante :

$$non(abattee([Nom]), schnorchel([Nom])) :- !.$$

Le coupe-choix !, ou Cut, permet, lors de l'évaluation d'une clause, d'indiquer que l'on ne souhaite pas évaluer les autres clauses.

Si, au cours d'un calcul, on trouve une exclusion mutuelle entre deux prédicats, ce calcul n'est pas possible. On utilise alors la négation par échec, qui permet de ne pas aller plus loin dans ce calcul. Pour cela, nous utilisons le prédicat prédéfini *fail*, qui échoue toujours. On définit un prédicat

$verification(X, Y)$, qui vérifie que X et Y ne sont pas impliqués dans une exclusion mutuelle :
$verification(X, Y)$:- $non(X, Y), !, fail.$
$verification(X, Y).$
Si X et Y sont impliquées dans une exclusion mutuelle, le prédicat $verification(X, Y)$ échoue. $verification(X, Y)$ est vraie si $non(X, Y)$ échoue.

Les défauts

Les défauts sont définis avec le prédicat $defaut$. Prenons le défaut suivant :

$$D_1 = \frac{détection(X_t, 0) : abattée(X_{t+1})}{abattée(X_{t+1})},$$

qui signifie que si le sous-marin n'a pas de détection et qu'il lui est possible de faire une abattée, alors il fait une abattée. Il est défini dans le programme Prolog par :

$$defaut(detection([Nom, 0]), abattee([Nom]), abattee([Nom])).$$

Le premier argument du prédicat $defaut$ contient le prérequis du défaut, le second argument contient la justification et le dernier le conséquent.

Le programme

La programme est défini de la façon suivante :

- Déclaration des prédicats dynamiques.
- Lancement de la boucle qui permet de définir le comportement de tous les sous-marins : pour chaque sous-marin, à chaque pas de temps
  - mise à jour des données suivantes : le sous-marin a-t-il une détection ? Quelle est la position de l'ennemi, quelle est sa cinématique ? Y-a-t-il un obstacle à proximité ?
  - calcul des extensions avec l'algorithme de calcul d'extension (cf. paragraphe 5.3.3) ;
  - choix d'une extension (cf. chapitre 7) ;
  - incrémentation du temps ;
  - exécution des consignes de cap, vitesse, immersion, tir pour le sous-marin ;
  - calcul de la nouvelle position du sous-marin, en fonction de sa cinématique.

5.3.3 Algorithme de calcul d'extension

Le calcul des extensions permet d'étudier les défauts un à un et de retenir ceux qui répondent au problème posé et qui sont compatibles entre eux. Chaque extension correspond donc à une solution possible du problème : en fonction de l'état du sous-marin à l'instant t, une extension donne une solution possible d'action du sous-marin au temps suivant.

L'algorithme défini ici permet de traiter le calcul d'extension pour des défauts normaux. Les défauts normaux assurent l'existence d'au moins une extension. En général, on peut même en compter plusieurs pour une même base de connaissance.

Rappelons la définition d'une extension dans le cas des défauts normaux :

E est une extension de Δ si et seulement si $E = \cup_{i=0,\infty} E_i$, avec

$$E_0 = W \text{ et pour } i \geq O,$$
$$E_{i+1} = Th(E_i) \cup \{C/(\frac{A:C}{C}) \in D, A \in E_i, \neg C \notin E_i\}$$

où $Th(E_i)$ désigne l'ensemble des théorèmes obtenus de façon monotone à partir de E_i : $Th(E_i) = \{w/E_i \vdash w\}$.

Pour calculer une extension, on doit vérifier que la négation de la justification n'appartient pas à E_i. On peut donc utiliser un algorithme incrémental pour calculer les extensions.

Pour une théorie des défauts normaux $\Delta = (D, W)$, avec D l'ensemble des défauts normaux et W la base de connaissance, le calcul d'extension se fait suivant l'algorithme suivant :

Entrée : $E = \emptyset$ (ensemble d'extension E vide).
Sortie : $E = \cup_{i=0,N} E_i$.
calcul_extension(E) :
{ **tantque** il y a un défaut $D_i = \frac{A_i(X) : C_i(X)}{C_i(X)}$ qui n'a pas encore été inspecté **faire**

- Sélectionner ce défaut D_i,
- Vérifier que le prérequis $A_i(X)$ est vrai (vérification avec Prolog),
- Vérifier que la justification $C_i(X)$ est consistante avec W (on utilise la négation par échec en cas d'inconsistance),
- Ajouter $C_i(X)$ à W (prédicat *asserta* de Prolog).

fin tantque
Fin du calcul pour une extension.
Backtracking (Suppression du dernier $C_i(X)$ ajouté à W, prédicat *retract* de Prolog).
calcul_extension(E).
}

Algorithme : Calcul d'extension

La vérification de la consistance de la justification $C_i(X)$ avec la base de connaissance W est faite en deux temps. L'algorithme doit vérifier que la justification $C_i(X)$ ne sera pas impliquée dans une exclusion mutuelle au temps suivant. Il doit également vérifier que les conséquents de règles dures entraînés par le défaut $D_i(X)$ ne seront pas impliqués dans une exclusion mutuelle au temps suivant.

En fin de calcul, nous obtenons alors l'ensemble des extensions. On note N le nombre d'extensions trouvées, on a donc l'ensemble des extensions : $E = \cup_{i=0,N} E_i$. Chaque extension correspond à une manœuvre possible pour le sous-marin.

Cet algorithme peut être présenté par un arbre de recherche des solutions (cf. figure 5.2). Dans cet exemple, nous notons les défauts $D_1, D_2, D_3 \ldots D_9$, et respectivement, les conséquents des défauts $C_1, C_2, C_3, \ldots C_9$. Nous obtenons le résultat suivant :

$$E = [[C_1, C_3, C_5], [C_1, C_3, C_8, C_9], [C_1, C_3, C_8], [C_1, C_3], [C_1], [C_2, C_4], [C_2], []].$$

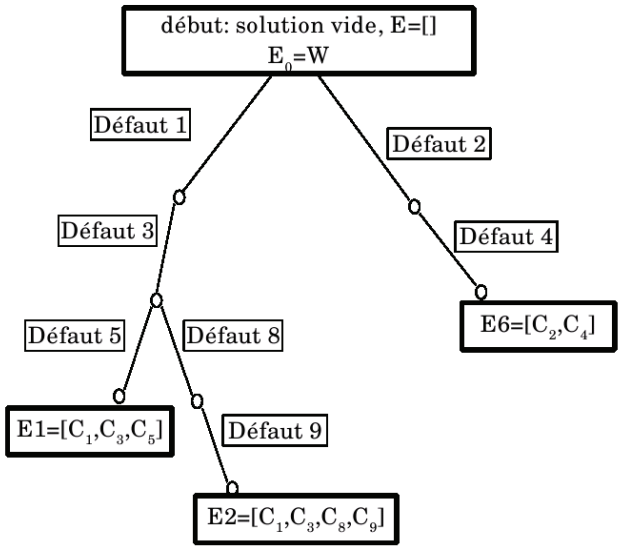

FIG. 5.2 – Arbre de recherche des solutions pour un calcul d'extensions.

5.3.4 Interface du programme avec ATANOR

Une interface en langage C a été réalisée entre NoMROD et l'atelier de simulation ATANOR (cf. figure 5.3). ATANOR étant programmé en langage C, nous avons réalisé cette interface dans le même langage.

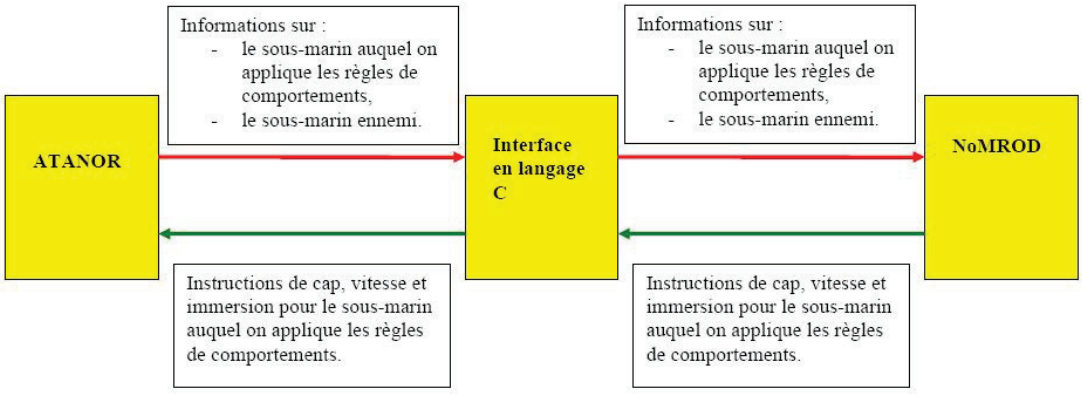

FIG. 5.3 – Interface entre NoMROD et ATANOR

ATANOR envoie à NoMROD les informations sur :

- le sous-marin auquel nous appliquons les règles de comportement (cap, vitesse, immersion, position,...)
- le sous-marin ennemi (détection, position estimée, vitesse estimée, ...)

NoMROD compile les règles, sélectionne une extension et renvoie les instructions de cap, vitesse et immersion à ATANOR.

Sur la figure 5.4 (précédemment présentée au paragraphe 2.4), nous avons une exécution du programme NoMROD, interfacé avec l'atelier de simulation ATANOR. Cette exécution simule un scénario d'un peu moins de trois heures en temps réel.

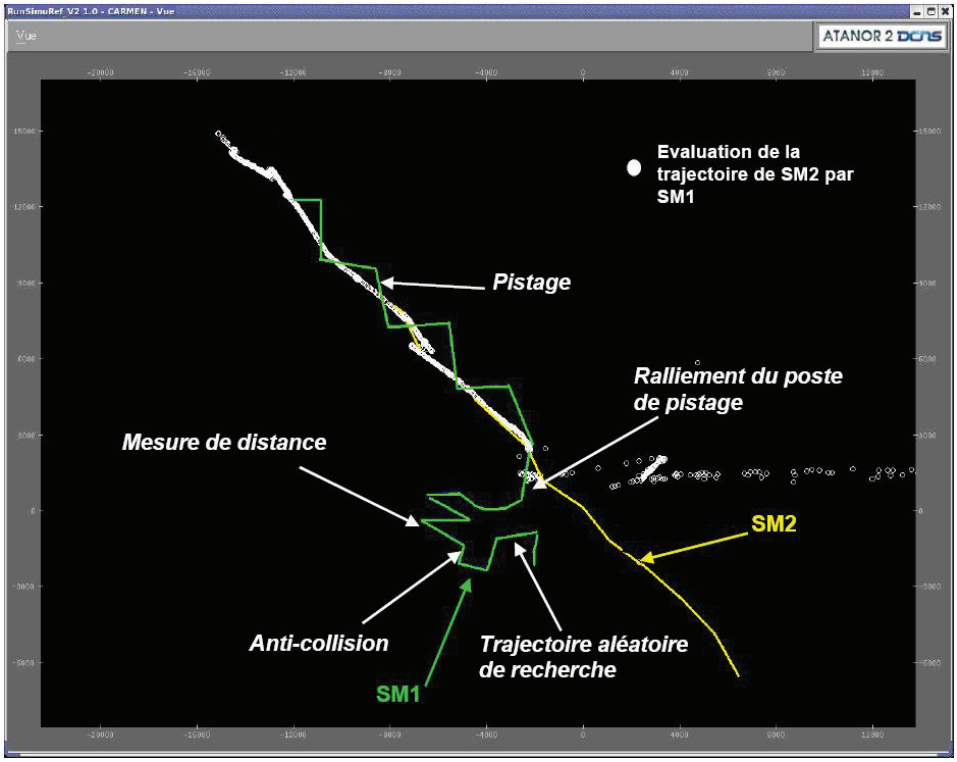

FIG. 5.4 – Exécution du programme NoMROD, interfacé avec l'atelier de simulation ATANOR

On appelle SM1, le sous-marin auquel nous appliquons les règles de comportement de NoMROD. Et on note SM2 le sous-marin ennemi, dont le comportement est défini par ATANOR. L'objectif du sous-marin ennemi est de traverser la zone de patrouille sans être détecté.

Dans ce scénario, le sous-marin SM1 fait une trajectoire aléatoire de recherche, car il n'a pas de détection. Lorsqu'il détecte l'ennemi, l'officier

enclenche les actions suivantes : anti-collision, mesure de distance, ralliement du poste de pistage, pistage. Au début, la trajectoire évaluée du sous-marin SM2 par le sous-marin SM1 est éloignée de la trajectoire réelle. La manœuvre de mesure de distance permet d'obtenir une meilleure estimation de la trajectoire du sous-marin ennemi.

Ce scénario a été validé par des sous-mariniers, qui retrouvent bien les actions qu'ils engagent en cas de détection d'un sous-marin ennemi. De plus, nous obtenons une modélisation robuste du comportement : dans les simulations, les informations fournies au sous-marin pisteur au sujet de la cinématique du sous-marin ennemi sont éloignées de la cinématique réelle. Cependant, la modélisation du comportement permet de mener à bien la mission. Cette modélisation a servi à une étude sur l'influence du comportement d'un officier sur les performances opérationnelles d'un sous-marin, avec des analyses statistiques de type Monte-Carlo. Cette étude est présentée au chapitre 8.

5.3.5 Complexité algorithmique

L'utilisation des clauses de Horn et des défauts normaux, ainsi que la programmation en langage Prolog nous permettent d'avoir une bonne complexité. Ce point était important pour la thèse. En effet, un des objectifs était de pouvoir interfacer le module de comportement avec ATANOR, afin de pouvoir faire des études statistiques de type Monte-Carlo. Nous avions donc besoin d'un programme efficace.

La complexité des clauses de Horn en calcul propositionnel est quasi-linéaire.

En ce qui concerne les défauts, en 1992, Gottlob [50] a démontré que la complexité est élevée avec le raisonnement par défaut. Cependant, en pratique, pour notre cas d'application, nous obtenons de très bons temps de calcul.

Nous obtenons un programme efficace : pour simuler un scénario d'environ 2h40, le temps de calcul utilisé par ATANOR et NoMROD est de 6 secondes et le progamme NoMROD ne prend que 20 % de ce temps de calcul.

Nous aurions pu utiliser la programmation par ensemble réponse (ou ASP, cf. paragraphe 3.3.2, [80], [84]) pour calculer les extensions. Afin de garder un programme simple, nous avons préféré implémenter le calcul d'extension avec le langage Prolog.

5.4 Conclusion

La logique des défauts nous a permis de répondre à une grande partie des exigences fixées au paragraphe 1.3.

La logique des défauts nous permet de représenter les connaissances de manière efficace et de raisonner sur des informations incomplètes, incertaines et révisables. Nous avions besoin d'un programme simple et robuste pour qu'il puisse être utilisé pour des applications militaires. L'utilisation des clauses de Horn et des défauts normaux nous permet d'obtenir un tel programme. Et nous avons l'avantage de pouvoir généraliser ce programme en utilisant d'autres clauses et d'autres défauts.

De plus, il était important de pouvoir interfacer ce programme avec la plateforme de simulation ATANOR, afin de pouvoir faire des études statistiques. La bonne complexité du programme nous permet également d'atteindre cet objectif, nous obtenons un bon temps de calcul, qui nous permettra de faire ces études.

Avec la logique des défauts, nous nous sommes affranchis de la représentation sous forme de réseau de Pétri ou encore de machine à état. L'utilisateur pourra facilement ajouter des nouvelles règles et de nouveaux défauts, sans avoir à modifier les règles précédentes. Il aura juste à gérer les exceptions entre les règles s'il y a besoin. Les défauts nous donnent l'avantage de pouvoir travailler avec des règles générales.

Cette modélisation a été validée par des sous-mariniers. Ils retrouvent bien les actions engagées à la mer. Cette modélisation est suffisamment robuste pour que le sous-marin mène à bien sa mission, malgré les informations incomplètes et incertaines qu'il possède sur l'ennemi.

L'algorithme de calcul d'extensions calcule toutes les extensions possibles. Chaque extension correspond à une solution possible d'action pour le sous-marin. Une fois toutes les extensions calculées, il reste à en choisir une, qui correspondra au choix final de l'officier. Nous nous intéressons, dans le chapitre suivant, à ce problème de choix et de sélection de l'action. Ce problème a été étudié dans différents domaines : on retrouve les approches comportementales et les préférences pour la logique des défauts. Les méthodes d'aide à la décision multicritère proposent aussi des solutions.

Troisième partie

Sélection de l'action

Chapitre 6

Etat de l'art

Pour une même situation, un agent peut avoir plusieurs possibilités d'actions. Par exemple, en ce qui concerne l'officier de quart, il peut avoir à choisir entre deux actions :

- doit-il continuer à pister l'ennemi ?
- doit-il remonter au schnorchel ?

Pour trouver une réponse à un tel problème, différentes méthodes existent. Ce problème a aussi bien été étudié dans les approches comportementales, que dans la logique des défauts avec les préférences, ou bien encore en définissant des méthodes d'aide multicritères à la décision. Dans ce paragraphe, nous donnons un aperçu des méthodes proposées dans ces trois domaines.

6.1 Les préférences en logique des défauts

La notion de préférence est très utilisée dans les raisonnements de sens commun, dans la vie de tous les jours. En effet, elle permet de résoudre de façon naturelle et efficace des situations indéterminées. Elles permettent de trouver des compromis intéressants pour résoudre des problèmes de décision. De même, lorsque des lois entrent en conflit, ce conflit peut être résolu en privilégiant les règles les plus récentes ou bien les règles de plus grande autorité.

En intelligence artificielle, une approche standard pour prendre en compte les préférences est d'utiliser un système de raisonnement non-monotone et d'y ajouter des préférences. Des préférences ont été ajoutées à la logique des défauts ([22], [30]).

Reprenons l'exemple avec lequel nous avons illustré le calcul d'extension (exemple 6, paragraphe 3.3.1). On a la théorie des défauts Δ=(D ,W) avec :

- $W = \{\forall x, Manchot(x) \rightarrow Oiseau(x), Manchot(Titi)\}$
- $D = \left\{ d_1 : \frac{Oiseau(x) : Vole(x)}{Vole(x)}, d_2 : \frac{Manchot(x) : \neg Vole(x)}{\neg Vole(x)} \right\}$

Le calcul des extensions donne deux extensions possibles :

$$E1 = \{Manchot(Titi) \rightarrow Oiseau(Titi), Manchot(Titi), Oiseau(Titi), Vole(Titi)\}$$

$$E2 = \{Manchot(Titi) \rightarrow Oiseau(Titi), Manchot(Titi), Oiseau(Titi), \neg Vole(Titi)\}$$

On obtient donc deux extensions qui sont contradictoires. Si l'on cherche à répondre à la question : "Est-ce que Titi vole ?", il faut pouvoir choisir entre ces deux extensions. Pour cela, on peut établir des préférences entre les défauts.
Une solution pour résoudre ce choix est de prendre en compte la spécificité des défauts. Le défaut d_2 contient une information plus précise que le défaut d_1, il est plus spécifique. Le défaut d_2 devrait donc être prioritaire par rapport au défaut d_1 et on préférerait normalement l'extension E2.
On note la relation de préférence avec la relation d'ordre suivante : $d2 < d1$. Cette relation signifie que le défaut d2 est prioritaire par rapport au défaut d1[1].

Nous donnons un autre exemple, tiré de [30].
Exemple 9 : Des préférences sont établies entre trois défauts :

- généralement, les Canadiens parlent anglais ;
- généralement, les Québécois parlent français ;
- généralement, les habitants du Nord du Québec parlent Cri.

Une relation d'ordre peut être exprimée de la façon suivante :

$$\frac{NQué(x) : cri(x)}{cri(x)} < \frac{Qué(x) : francais(x)}{francais(x)} < \frac{Can(x) : anglais(x)}{anglais(x)}$$

Cette relation signifie : si un habitant du Nord du Québec ne parle pas cri, il est raisonnable de penser que cette personne parle français, et si cette personne ne parle pas français, alors elle parle anglais. Nous avons une relation de spécificité (ou de subsomption) entre les défauts.

[1] C'est la convention utilisée par Brewka [22] et c'est celle que nous utilisons dans la thèse. Dans d'autres articles, on trouve que le relation $d2 < d1$ signifie que $d1$ est préféré à $d2$.

Changeons maintenant la relation d'ordre pour définir le fait que, dans le nord du Québec, la première langue est le français, ensuite l'anglais et ensuite le cri. La relation de préférence est la suivante :

$$\frac{Qué(x) : francais(x)}{francais(x)} < \frac{Can(x) : anglais(x)}{anglais(x)} < \frac{NQué(x) : cri(x)}{cri(x)}$$

Cette relation d'ordre ne prend pas en compte la spécificité, car le défaut le plus prioritaire n'est pas le défaut le plus spécifique, le plus précis.

Nous présentons maintenant quelques approches sur les préférences en logique des défauts.

La première formalisation des priorités en logique des défauts a été proposée par Brewka [21] en 1991. Cette approche ne s'appliquait qu'aux défauts normaux sans prérequis. D'autres approches ont ensuite été proposées pour les défauts normaux : Brewka [22] [23] en 1994, Baader et Hoolunder [11] en 1993, Rintanen [92],[93] en 1995, Delgrande et Schaub [33] en 1997. Dans ces approches, les préférences sont définies par des relations d'ordre entre les défauts.

Une approche différente a été proposée par Marek, Nerode et Remmel ([67], [68]) dans les années 90, il s'agit des systèmes non monotones de règles annotées : un poids est attribué à chaque information. Ce poids peut représenter un indice de confiance, une probabilité ...

En 2000, Brewka et Eiter [24] présentent une nouvelle approche de la théorie des défauts ordonnés. En 2009, Delgrande et al [32] reprennent leurs précédents travaux [33] et ils réutilisent la théorie développée par Brewka et Eiter [24].

6.1.1 La théorie des défauts ordonnés de Brewka, 1994

En 1994, Brewka [22] [23] propose une formalisation des priorités pour les défauts normaux. Afin de gérer la notion de préférence entre les extensions mutuellement contradictoires, Brewka a étendu la logique des défauts à la théorie des défauts ordonnés (Prioritized default logic), en introduisant un ordre partiel strict.

Définition 27 : La théorie des défauts normaux ordonnés

Une théorie des défauts normaux ordonnés est le triplet (D,W,<) où W est un ensemble de formules, D est un ensemble de défauts normaux et < est un ordre partiel strict sur D.

La relation d'ordre $D1 < D2$, avec D1 et D2 des défauts, signifie que le défaut D1 est prioritaire par rapport au défaut D2, ou que D1 est le défaut $<$-minimal.

Afin de définir les extensions prioritaires, Brewka définit la notion d'*activité* et la notion d'ordre total $\ll$, issue de l'ordre partiel strict $<$.

Définition 28 : Activité [23]

Soient E un ensemble de formules et $d = \frac{a : c}{c}$ un défaut normal. On dit que d est actif dans E si et seulement si :

1. $a \in E$,
2. $c \notin E$,
3. $\neg c \notin E$.

Un défaut est donc actif si son prérequis est satisfait, s'il n'a pas déjà été déclenché et s'il n'y a pas de contradiction à son conséquent.

Un ordre total $\ll$, extension de l'ordre partiel $<$, permet de calculer les extensions prioritaires d'une théorie des défauts ordonnés :

Définition 29 : Extension prioritaire

Soit (D,W,$<$) une théorie des défauts ordonnés et soit $\ll$ un ordre total issu de l'ordre partiel $<$. E est une extension prioritaire générée par $\ll$ si et seulement si $E = \cup_{i=0,\infty} E_i$ où $E_0 = W$ et pour $i \geq O$,

$$E_{i+1} = \begin{cases} E_i & si\ il\ n'\ y\ a\ pas\ de\ défaut\ actif\ dans\ E_i,\ ou\ bien \\ Th(E_i \cup \{c\})\ où\ c\ est\ le\ conséquent\ du\ défaut\ \ll -minimal \\ & actif\ dans\ E_i. \end{cases}$$

Pour construire l'extension E_{i+1}, on ajoute à E_i le conséquent du défaut $\ll$-minimal actif, c'est-à-dire le défaut actif le plus prioritaire dans E_i.

Reprenons l'exemple 6 en le modifiant :

Exemple 10 :

- $W = \{Manchot(Titi)\}$
- $D = \{d1, d2, d3\}$

avec

$$d1 = \frac{Oiseau(x) : Vole(x)}{Vole(x)},$$

$$d2 = \frac{Manchot(x) : \neg Vole(x)}{\neg Vole(x)},$$
$$d3 = \frac{Manchot(x) : Oiseau(x)}{Oiseau(x)}.$$

Pour prendre en compte la spécificité entre les défauts, on définit la relation de préférence suivante : d2 $<$ d1. Le défaut $d2$ est plus spécifique.
A partir de cet ordre partiel, il y a trois ordres totaux :

- $d3 \ll d2 \ll d1$,
- $d2 \ll d1 \ll d3$,
- $d2 \ll d3 \ll d1$.

Pour chacun de ces trois ordres totaux, on obtient la même extension prioritaire :

$$E = \{Manchot(Titi), Oiseau(Titi), \neg Vole(Titi)\}.$$

Prenons le premier ordre total. Le calcul est fait de la façon suivante :

- $E_0 = \{Manchot(Titi)\}$
  On a alors les défauts $d2$ et $d3$ qui sont actifs. et le défaut d1 qui est inactif ;
- $E_1 = \{Manchot(Titi), Oiseau(Titi)\}$
  car on utilise le défaut $d3$ qui est le défaut $\ll$-minimal actif. On a toujours le défaut $d2$ qui est actif. et le défaut $d1$ est alors actif ;
- $E_2 = \{Manchot(Titi), Oiseau(Titi), \neg Vole(Titi)\}$.
  car on utilise le défaut $d2$ qui est le défaut $\ll$-minimal actif. Le défaut $d1$ est alors inactif ;
- $E_3 = E_2$ car il n'y a pas de défaut actif.
- ...

D'après la classification des approches qui traitent des préférences dans les raisonnements non-monotones de Delgrande et al [31], l'approche développée par Brewka est une approche :

- où les préférences sont définies au méta-niveau, c'est-à-dire qu'elles sont ajoutées aux règles ;
- prescriptive : la relation d'ordre spécifie l'ordre dans lequel les défauts doivent être appliqués.

Cette approche semble être efficace. Cependant, un inconvénient est le calcul de tous les ordres totaux. Certains ordres totaux peuvent donner le même ordre d'application des défauts, la même extension est alors calculée plusieurs fois.

Baader et Hoolunder [11] ont également introduit une approche prescriptive en logique des défauts. La définition d'extension classique de Reiter [90]

a été modifiée en ajoutant une condition sur l'applicabilité des défauts. Cette méthode a été utilisée dans un cas d'application portant sur les réseaux sociaux sur le web par Katz et Golbeck [57]. Dans les réseaux sociaux, les utilisateurs révèlent des informations sur leurs relations avec d'autres gens. En particulier, ils donnent des valeurs qui représentent leur degré de confiance avec ces gens. Ces valeurs peuvent générer des recommandations sur la confiance que l'on peut accorder à une personne que l'on ne connaît pas. L'article montre comment ces valeurs de confiance peuvent être utilisés pour ordonner des défauts.

6.1.2 L'approche de Rintanen

En 1995, Rintanen [92],[93] s'est intéressé aux préférences descriptives pour les théories de défauts normaux. Avec des préférences descriptives, la situation préférée est celle qui utilise le plus de défauts préférés. L'approche de Rintanen est basée sur l'applicabilité d'un défaut dans un ensemble de formules.

Définition 30 : Un défaut normal $d = \frac{a : b}{b}$ est applicable dans E un ensemble de formules si a et b $\in$ E.

Les extensions prioritaires sont définies de la façon suivante :

Définition 31 : Soit (D,W,<) une théorie des défauts ordonnés. Soit E une extension de la théorie (D,W). E est une extension prioritaire par rapport à E' si et seulement si il y a un défaut $\delta \in D$ appliqué dans E, mais pas dans E', de telle façon à ce que si δ' est préféré à δ et que δ' est appliqué dans E', alors δ' est également appliqué dans E.

Pour illustrer cette approche, reprenons l'exemple 10, avec l'ordre total suivant :

$$\frac{Manchot(x) : Oiseau(x)}{Oiseau(x)} \ll \frac{Manchot(x) : \neg Vole(x)}{\neg Vole(x)} \ll \frac{Oiseau(x) : Vole(x)}{Vole(x)}$$

On doit choisir entre deux extensions :

$$E1 = \{Manchot(Titi), Oiseau(Titi), Vole(Titi)\}$$

$$E2 = \{Manchot(Titi), Oiseau(Titi), \neg Vole(Titi)\}$$

L'extension E2 est prioritaire par rapport à E1 pour cet ordre lexicographique : le défaut

$$\frac{Manchot(x) : \neg Vole(x)}{\neg Vole(x)}$$

est appliqué dans E2 mais pas dans E1. Le défaut

$$\frac{Manchot(x) : Oiseau(x)}{Oiseau(x)}$$

est le seul défaut préféré à

$$\frac{Manchot(x) : \neg Vole(x)}{\neg Vole(x)}$$

et il est appliqué à la fois dans E2 et E1. Dans cet exemple, l'approche de Rintanen donne un résultat bien conforme à l'intuition.

6.1.3 L'approche Delgrande-Schaub

En 1997, Delgrande et Schaub [33] définissent, à partir d'une théorie des défauts ordonnés (D,W,<), une théorie des défauts (D',W') dans laquelle les relations d'ordre partiel sont compilées dans D' et W'. Dans cette approche, les préférences sont définies au niveau objet : elles permettent de définir les préférences à l'intérieur de la théorie.

Un nom est associé à chaque défaut. Pour cela, le langage original est étendu avec un ensemble de constantes N, en définissant une fonction bijective $n : D \rightarrow N$. On note n_δ pour $n(\delta)$ avec δ le défaut $\delta = \frac{\alpha : \beta}{\gamma}$.
Pour une relation d'ordre partielle $\delta < \delta'$, il faut s'assurer que δ est appliqué avant δ'.

Il y a deux cas pour lesquels un défaut $\delta = \frac{\alpha : \beta}{\gamma}$ ne peut pas être appliqué :

- soit le prérequis α n'est pas vrai ($\alpha \notin E$),
- soit la justification β n'est pas consistante ($\neg\beta \in E$).

Pour détecter ces cas, trois prédicats ont été introduits :

- $ok(n_\delta)$, pour contrôler l'applicabilité d'un défaut δ,
- $bl(n_\delta)$, pour contrôler le blocage d'un défaut δ,
- $ap(n_\delta)$, pour détecter le cas où le défaut a été appliqué.

Chaque défaut $\delta = \frac{\alpha : \beta}{\gamma}$ est traduit de la façon suivante :

$$\frac{\alpha \wedge ok(n_\delta) : \beta}{\gamma \wedge ap(n_\delta)}, \frac{ok(n_\delta) : \neg\alpha}{bl(n_\delta)}, \frac{\neg\beta \wedge ok(n_\delta) :}{bl(n_\delta)}.$$

Aucune de ces trois règles ne peut être appliquée sans que $ok(n_\delta)$ soit vrai. Les deux derniers défauts donnent les conditions dans lesquelles un défaut ne peut pas être appliqué.

La relation d'ordre $\delta < \delta'$ signifie : si δ a été appliqué ($ap(n_\delta)$ est vrai), ou si δ n'est pas applicable ($bl(n_\delta)$), alors δ' est applicable. Cette propriété est traduite de la façon suivante :

$$\forall y \in N, [\forall x \in N, (x \prec y) \rightarrow bl(x) \vee ap(x)] \rightarrow ok(y)$$

Définition 32 : Soit (D,W,<) une théorie des défauts ordonnés et $N = \{n_\delta | \delta \in D\}$. ($D'$,$W'$) est défini de la façon suivante :

$$D' = \left\{ \frac{\alpha \wedge ok(n_\delta) : \beta}{\gamma \wedge ap(n_\delta)}, \frac{ok(n_\delta) : \neg\alpha}{bl(n_\delta)}, \frac{\neg\beta \wedge ok(n_\delta)) :}{bl(n_\delta)} | n : \frac{\alpha : \beta}{\gamma} \in D \right\} \cup D_\prec$$

$$W' = W \cup W_\prec \cup \{DCA_N, UNA_N\}$$

où

$$D_\prec = \left\{ \frac{: \neg(x \prec y)}{\neg(x \prec y)} \right\}$$

$$W_\prec = \{n_\delta \prec n_{\delta'} | (\delta, \delta') \in <\} \cup \{ok(n_T)\} \cup \{\forall y \in N, [\forall x \in N, (x \prec y) \rightarrow bl(x) \vee ap(x)] \rightarrow ok(y)\}$$

avec UNA_N (Unique Name Assumtion) et DCA_N (Domain Closure Assumption) définis de la façon suivante :

$$UNA_N : (n_i \neq n_j) \textit{ pour tout } n_i, n_j \in N \textit{ avec } i \neq j$$

$$DCA_N : \forall x, name(x) \equiv (x = n_1 \vee \ldots \vee x = n_m)$$

Le premier sous-ensemble de $W_\prec$ représente les connaissances relatives à $\prec$ issues de l'ordre $<$ et $ok(n_T)$ signifie que le défaut $\frac{T : T}{T}$ est toujours applicable. La dernière partie contrôle l'application des défauts : pour tout $n_i \prec n_j$, $ok(n_j)$ est prouvé si on a $ap(n_i)$ ou $bl(n_i)$.

Les informations contenues dans $W_\prec$ donnent des conditions nécessaires mais pas suffisantes pour rendre δ_i applicable. Si $((\delta_i, \delta_j) \notin <)$, alors $(n_i \prec n_j) \notin W_\prec$. Pour travailler proprement, nous devons pouvoir conclure que $\neg(n_i \prec n_j)$. C'est ce qui est fait dans $D_\prec$.

Pour illustrer cette méthode, nous présentons un exemple tiré de [16] :

Exemple 11 : Soit la théorie des défauts normaux ordonnés suivante :

$$D = \left\{ \frac{T : T}{T}, \frac{a : \neg c}{\neg c}, \frac{b : c}{c} \right\}$$
$$W = \{a, b\}$$

On a la relation d'ordre :

$$\frac{a : \neg c}{\neg c} < \frac{b : c}{c}.$$

La théorie correspondante est :

$$D' = \left\{ \begin{array}{ccc} \frac{a \wedge ok(n1) : \neg c}{\neg c \wedge ap(n1)}, & \frac{ok(n1) : \neg a}{bl(n1)}, & \frac{c \wedge ok(n1) :}{bl(n1)}, \\ \frac{b \wedge ok(n2) : c}{c \wedge ap(n2)}, & \frac{ok(n2) : \neg b}{bl(n2)}, & \frac{\neg c \wedge ok(n2) :}{bl(n2)}, \\ \frac{T \wedge ok(nT) : T}{T \wedge ap(nT)}, & \frac{ok(nT) : \neg T}{bl(nT)}, & \frac{\neg T \wedge ok(nT) :}{bl(nT)} \end{array} \right\}$$

$$W' = \left\{ \begin{array}{r} a, b, n1 \prec n2, nT \prec n1, nT \prec n2, ok(nT), \\ bl(nT) \vee ap(nT) \rightarrow ok(n1), \\ ((nT \prec n2) \wedge (n1 \prec n2)) \rightarrow \\ ((bl(nT) \vee ap(nT)) \wedge (bl(n1) \vee ap(n1)))) \rightarrow ok(n2). \end{array} \right\}$$

avec

$$D_{\prec} = \left\{ \frac{: \neg(n2 \prec n1)}{\neg(n2 \prec n1)}, \frac{: \neg(n1 \prec nT)}{\neg(n1 \prec nT)}, \frac{: \neg(n2 \prec nT)}{\neg(n2 \prec nT)} \right\}$$

Cette théorie a une extension :

$$E = Th(\left\{ \begin{array}{r} a, b, \neg c, (n1 \prec n2), (nT \prec n1), (nT \prec n2), \\ \neg(n2 \prec n1), \neg(n2 \prec nT), \neg(n1 \prec nT), \\ ok(nT), ok(n1), ok(n2), \\ ap(nT), ap(n1), bl(n2) \end{array} \right\})$$

6.1.4 Système non monotone de règles annotées

Inspiré de Reiter [90], Marek, Nerode et Remmel ([67], [68]) ont introduit la notion de système non monotone de règles dans les années 90. Cette approche est différente des précédentes : des poids sont attribués à chaque information. Nous n'en donnons ici qu'un bref aperçu.

Une règle non monotone est un triplet (P,G,ϕ), où $P = \{\alpha_1, \ldots, \alpha_n\}$, $G = \{\beta_1, \ldots, \beta_m\}$ sont des listes finies d'objet dans un ensemble U et $\phi \in U$. Une telle règle s'écrit

$$r = \frac{(\alpha_1 \ldots, \alpha_n) : (\beta_1, \ldots, \beta_m)}{\phi},$$

où les $\{\alpha_1, \ldots, \alpha_n\}$ sont les prémisses de la règle r, $\{\beta_1, \ldots, \beta_m\}$ sont les contraintes de la règles r et ϕ est la conclusion de r.
Cette règle signifie : si $(\alpha_1 \ldots, \alpha_n)$ ont été établis et $(\beta_1, \ldots, \beta_m)$ n'ont pas encore été établis, alors on conclut ϕ.

Définition 33 : Système non monotone de règles
Un système non monotone de règles S est une paire (U,N), où U est un ensemble non vide et N est une ensemble de règles non-monotones sur U.

Les systèmes non-monotones de règles généralisent les formalismes de raisonnement non monotone comme la logique des défauts, ou encore les logiques modales non monotones. Des parallélismes sont établis entre les systèmes non monotones et les formalismes de raisonnements non-monotones.

La notion de préférence a ensuite été introduite par Nerode et al [79] en 1997, en ajoutant un poids attribué à chaque information, en réutilisant les logiques annotées de Subrahmanian [108]. Ce poids peut représenter un indice de confiance, une probabilité, la qualité des données ...

On note P=(P, $\leq_P$) un ensemble d'ordre partiel ou un ensemble de préordre et on note U un univers.
Cette approche utilise des déclarations de la forme : (ϕ,p) où ϕ est un élément de U et $p \in P$.
Voici quelques exemples, avec P=([0,1], $\leq$) :

- (ϕ,1) signifie que l'information est vraie à coup sûr ;
- (ϕ,0.9) signifie que ϕ est vrai à au moins 90% ou que ϕ a une probabilité d'au moins 90% d'être vrai ;
- (ϕ,0) signifie que que nous ne savons rien sur la vérité de ϕ.

Une règle non-monotone et annotée est définie de la façon suivante :

$$r = \frac{(\alpha_1, a_1), \ldots, (\alpha_n, a_n) : (\beta_1, b_1), \ldots, (\beta_m, b_m)}{(\phi, c)}.$$

$(\alpha_1, a_1), \ldots, (\alpha_n, a_n)$ sont les prémisses de la règle, $(\beta_1, b_1), \ldots, (\beta_m, b_m)$ sont les contraintes de la règle r et (ϕ, c) est la conclusion de la règle r.

Cette règle signifie que si α_i a été établie avec une confiance $\geq a_i$ pour $i = 1, \ldots, n$ et qu'aucun β_i ne peut être établie avec une confiance $\geq b_j$ pour $j = 1, \ldots, m$, alors on peut conclure que ϕ est vrai avec une confiance d'au moins c.

Définition 34 : Système non-monotone de règles annotées
Un système non-monotone de règles annotées S est un triplet (U,P,N) où U est un ensemble appelé univers de S, P=(P, $\leq_P$) est un ensemble de préordre et N est un ensemble de règles de la forme

$$r = \frac{(\alpha_1, a_1), \ldots, (\alpha_n, a_n) : (\beta_1, b_1), \ldots, (\beta_m, b_m)}{(\phi, c)},$$

où les α_i, β_j et ϕ sont dans U pour tout i et j et a_i, b_j et c sont dans P pour tout i et j.

Les systèmes non-monotones de règles annotées donnent une théorie et des algorithmes qui peuvent être appliqués à la logique des défauts, aux modèles stables ...

6.1.5 La théorie des défauts ordonnés de Brewka et Eiter, 2000

En 2000, Brewka et Eiter [24] ont présenté une nouvelle approche de la théorie des défauts ordonnés. Pour introduire le concept d'extension préférée, ils ont commencé par travailler sur les défauts normaux sans prérequis, aussi appelés les défauts supernormaux. Ils ont ensuite étendu cette notion aux défauts sans prérequis et finalement aux défauts généraux.

Prendre en compte les préférences dans une théorie des défauts supernormaux ordonnés est le cas le plus simple. Dans cette définition, l'opérateur C est introduit :

Définition 35 : Soit $\Delta = (D, W, <)$ une théorie des défauts sans prérequis ordonnés $E = \cup_{i=0,\infty} E_i$. L'opérateur C est défini de la façon suivante : $C(\Delta) = \cup_{i \geq 0} E_i$, où $E_0 = W$ et pour tout $i \leq 0$:

$$E_i \quad = \quad \begin{cases} \cup_{j \leq i} E_j \quad \textit{si il n'y a pas de défaut actif dans } \cup_{j \leq i} E_j, \\ \textit{ou bien} \\ Th(\cup_{j \leq i} E_j \cup \{c\}) \textit{ où } c \textit{ est le conséquent du défaut} \\ \ll -\textit{minimal actif dans } \cup_{j \leq i} E_j. \end{cases}$$

Dans le cas de la théorie des défauts supernormaux, les extensions obtenues sont toujours des extensions au sens de Reiter. Les extensions prioritaires sont définies directement de la façon suivante :

Définition 36 : Soit $\Delta = (D, W, <)$ une théorie des défauts supernormaux ordonnés. E est une extension prioritaire de Δ si et seulement si $E = C(\Delta)$.

On utilise la notation suivante : on note *prérequis*, le prérequis du défaut d, $justification(d)$ la justification du défaut d et $conséquent(d)$ le conséquent du défaut d.

Cette définition est ensuite étendue aux défauts sans prérequis :

Définition 37 : Soit $\Delta = (D, W, <)$ une théorie des défauts sans prérequis ordonnés. Un ensemble de formules E est une extension prioritaire de Δ si et seulement si $E = C(\Delta^E)$ où $\Delta^E = (D^E, W, <)$ et D^E est obtenu en enlevant les défauts dont le conséquent est dans E ($conséquent(d) \in E$) et qui ne peuvent pas être utilisés dans E ($\neg justification(d) \in E$) (les défauts normaux ne sont pas concernés).

Les extensions préférées sont différentes des extensions classiques de Reiter.

Enfin, cette définition est étendue à la théorie des défauts généraux :

Définition 38 : Soit $\Delta = (D, W, <)$ une théorie des défauts ordonnés et E un ensemble de formules. La théorie des défauts $\Delta^E = (D_E, W, <_E)$ est obtenue de Δ de la façon suivante :
D_E résulte de D en éliminant tous les défauts $d \in D$ tels que $prérequis(d) \notin E$, et en remplaçant le prérequis du défaut d par T dans tous les défauts restants. T est une formule vraie. On note d^T la version sans prérequis du défaut d.

La relation d'ordre $<_E$ est obtenue à partir de la relation d'ordre $<$ de la façon suivante : pour tout défaut d et d' dans D_E, on a $d <_E d'$ si et seulement si $d1 < d1'$ avec $d1^T = d$ et $d1'^T = d'$.

La théorie des défauts $\Delta^E = (D_E, W, <_E)$ ne contient alors que des défauts sans prérequis.

Les extensions préférées sont définies de la façon suivante :

Définition 39 : Soit $\Delta = (D, W, <)$ une théorie des défauts ordonnés. E est une extension prioritaire de Δ si E est une extension classique de Δ, et si E est une extension prioritaire de Δ_E.

Illustrons cette approche avec l'exemple suivant :

Exemple 12 : On considère la théorie des défauts suivante Δ :

– $d1 = \dfrac{a : b}{b}$,

- $d2 = \dfrac{T : \neg b}{\neg b}$,
- $d3 = \dfrac{T : a}{a}$,

avec la relation de préférence suivante : $d1 < d2 < d3$. La théorie des défauts de Reiter donne deux extensions classiques : $E1 = Th(\{a, b\})$ et $E2 = Th(\{a, \neg b\})$.

Les règles de Δ_{E1} sont les suivantes :

- $d1 = \dfrac{T : b}{b}$,
- $d2 = \dfrac{T : \neg b}{\neg b}$,
- $d3 = \dfrac{T : a}{a}$.

On a $C(\Delta_{E1}) = E1$. E1 est donc une extension prioritaire.

Les règles de Δ_{E2} sont les mêmes que les règles de Δ_{E1}. On obtient $C(\Delta_{E2}) = E1$. L'extension E2 n'est donc pas prioritaire.

Exemple 13 : On considère la théorie des défauts suivante Δ :

- $d1 = \dfrac{a : b}{b}$,
- $d2 = \dfrac{T : \neg a}{\neg a}$,
- $d3 = \dfrac{T : a}{a}$,

avec la relation de préférence suivante : $d1 < d2 < d3$. La théorie des défauts de Reiter donne deux extensions classiques : $E1 = Th(\{\neg a\})$ et $E2 = Th(\{a, b\})$.

Les règles de Δ_{E1} sont les suivantes :

- $d2 = \dfrac{T : \neg a}{\neg a}$,
- $d3 = \dfrac{T : a}{a}$.

On a $C(\Delta_{E1}) = E1$. E1 est donc une extension prioritaire.

Les règles de Δ_{E2} sont les suivantes :

- $d1 = \dfrac{T : b}{b}$,
- $d2 = \dfrac{T : \neg a}{\neg a}$,
- $d3 = \dfrac{T : a}{a}$.

On obtient $C(\Delta_{E2}) = Th(\{b, \neg a\})$. Cette extension est différente de E2, l'extension E2 n'est donc pas prioritaire.

Cette approche a été réutilisée en 2009 par Delgrande et al [32]. Ils ont exprimé la théorie développée par Brewka et Eiter dans leurs précédents

travaux [33].

6.1.6 Conclusion

Les préférences en logique des défauts ont fait l'objet de nombreux travaux. Pour notre problème, nous ne voulons pas modifier la définition de la logique des défauts, comme dans l'approche Delgrande-Schaub, avec la logique des défauts (D',W'). De même, nous ne voulons pas modifier la façon de calculer les extensions, comme avec dans l'approche de Brewka et Eiter. Nous voulons que la notion de préférence reste simple et intuitive, afin qu'elle puisse être facilement maîtrisée par un utilisateur. Nous avons choisi de définir les préférences avec des poids sur les défauts. Pour définir l'extension prioritaire, nous utilisons une méthode d'aide multicritère à la décision.

6.2 La sélection de l'action dans les approches comportementales

Dans le paragraphe 3.4.2, trois approches basées sur les comportements ont été présentées. Dans ces approches, le modèle décisionnel est réparti entre plusieurs entités indépendantes : les comportements. Les comportements peuvent être les schémas, les champs de potentiels, les règles de comportements, les niveaux de compétences .... Chaque comportement gère un aspect particulier du problème.

Une fois les comportements définis, il faut mettre en place une méthode de sélection de l'action : quel(s) comportement(s) doit-on sélectionner en fonction de la situation ?
Maes [66] définit le problème de sélection de l'action de la façon suivante :
On a un agent autonome qui possède un certain nombre de buts et qui fait face à une situation particulière à un moment spécifique. Comment cet agent peut-il sélectionner une action de telle façon qu'il ait un comportement sensé ?
Voici les caractéristiques d'un comportement sensé :

- orienté par des buts : il favorise des actions qui contribuent à un ou plusieurs buts ;
- situé : il favorise les actions qui sont pertinentes avec la situation présente ;
- persistant : il favorise les actions qui contribuent au but en cours, il persiste dans ses choix ;
- planning : il doit se projeter afin d'éviter les situations dangereuses ;
- réactivité : il fournit des réponses rapides ;

- robuste : un dysfonctionnement local ne doit pas remettre en cause tout le système.

Tyrrell [118] ajoute les caractéristiques suivantes :

- Un bon compromis : il doit satisfaire un maximum des buts.
- Opportuniste : il doit être capable d'interrompre une action en cours pour réaliser un autre objectif.

Plusieurs mécanismes de sélection de l'action ont été mis en place. Pirjanian a rédigé un état de l'art sur les mécanismes de coordination de comportements [83]. Il distingue deux politiques : l'arbitrage et la fusion d'actions. Cet état de l'art a été repris dans la thèse de Hanon [52].
L'arbitrage sélectionne un ou plusieurs comportements pertinents qui vont prendre le contrôle de l'agent jusqu'au prochain pas de calcul. La fusion de commande consiste à prendre en compte les différentes propositions des comportements afin d'obtenir un compromis.

Il existe plusieurs mécanismes d'arbitrage et de fusion de commandes. Nous donnons ici quelques exemples.

6.2.1 L'arbitrage

Les mécanismes d'arbitrage peuvent être divisés en trois types de technique : les priorités, les machines à états et le gagnant emporte la mise.

Par priorités

La sélection par priorité est la plus ancienne. Un comportement est actif s'il propose une réponse. Le principe des priorités consiste à sélectionner le comportement actif le plus prioritaire.

Cette technique est utilisée dans l'architecture de subsomption de Brooks, décrite au paragraphe 3.4.2. Le comportement est décomposé en niveaux de compétences qui sont ordonnés hiérarchiquement en fonction de leur degré de priorité. Cette architecture prend également en compte des mécanismes d'inhibition et de suppression entre les niveaux de compétence.

Le principe des priorités est simple. Cependant, le concepteur doit connaître la liste exhaustive de tous les comportements possibles et doit être capable de fixer les priorités, ce qui peut vite devenir compliqué si le nombre de comportements augmente.

Par machines à états finis

Les machines à états finis permettent de modéliser les relations de successions entre les comportements. Une machine à états finis est composée d'états, qui correspondent à des comportements, et de transitions entre les états, qui définissent dans quels cas les états sont actifs. Balch et al [13] ont réalisé des robots chargés de collecter des ordures et de les mettre à la poubelle. Pour définir leur comportement, ils ont utilisé une machine à états finis. Pour accomplir leur tâche, les robots doivent accomplir une séquence d'actions : trouver un déchet, chercher une poubelle, aller à la poubelle, déposer le déchet ...La machine à état est représentée sur la figure 6.1. Les états sont représentés par des cercles et les transitions sont représentées par les flèches.

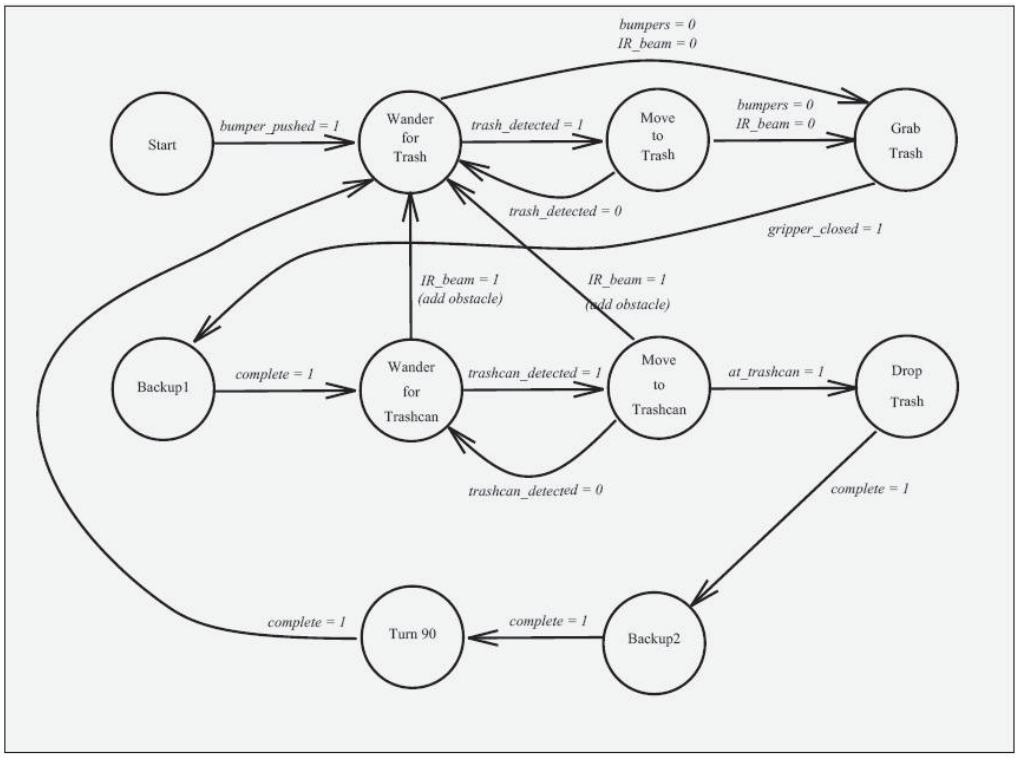

FIG. 6.1 – Comportement d'un robot avec une machine à états finis [13] .

Dans sa thèse, Moreau [78] utilise également des machines à états finis dans son modèle HPTS (Hierarchical Parallel Transition System), appliqué à la construction d'un conducteur de voiture virtuel.

Les machines à états finis sont une solution simple pour décrire un com-

portement. Cependant, on retrouve le même inconvénient que les réseaux de Pétri : on ne peut pas ajouter de nouveaux comportements, sans avoir à reprendre la machine à états préexistante.

Le gagnant emporte la mise (winner takes all)

Dans la technique du gagnant emporte la mise, la sélection de l'action résulte de l'interaction entre un ensemble de comportements qui sont en compétition, jusqu'à ce qu'un comportement gagne la compétition et prenne le contrôle.

Les réseaux d'activation de Maes [66] sont un exemple de méthode du gagnant emporte la mise. Chaque comportement i est défini par un tuple : (c_i, a_i,d_i, α_i), où :

- c_i sont les conditions d'activation du comportement,
- a_i sont les effets attendus de l'action en terme de liste d'ajout,
- d_i sont les effets attendus de l'action en terme de liste d'annulation d_i,
- α_i est le niveau d'activation du comportement.

Le comportement sélectionné est celui qui a le niveau d'activation le plus élevé. Le niveau d'activation est calculé en fonction de facteurs externes : l'activation est augmentée si les préconditions du comportement sont remplies et si le comportement réalise un des buts de l'agent. Par contre, elle diminue si le comportement peut annuler un des buts. Le niveau d'activation est également calculé en fonction de facteurs internes. L'activation est augmentée si le comportement permet d'activer d'autres comportements et de bloquer des comportements rivaux.

Avec ces échanges d'énergie d'activation, les comportements rivalisent et coopèrent pour sélectionner une action qui est la plus appropriée à l'environnement et aux buts à atteindre. La façon dont l'énergie d'activation est échangée entre les comportements favorise la sélection d'une séquence de comportements, qui transforme l'état courant de l'environnement en un état dans lequel les buts sont atteints. Cependant, cette séquence de comportements n'est pas déterminée une fois pour toutes et peut s'adapter aux situations. Une séquence peut être naturellement interrompue pour une autre. On a donc un système réactif et opportuniste. La figure 6.2 illustre le comportement d'un robot qui doit poncer une planche.

Conclusion

Ces méthodes semblent assez simples à utiliser mais peuvent vite devenir compliquées si le nombre de comportements augmente. Dans le cas de l'architecture de subsomption et avec les machines à états, l'ajout de nou-

veaux comportements nécessite de revoir le modèle déjà existant. De plus, les méthodes d'arbitrage permettent de choisir un seul comportement, en ignorant totalement les autres propositions. Ces méthodes sont qualifiées de compétitives.

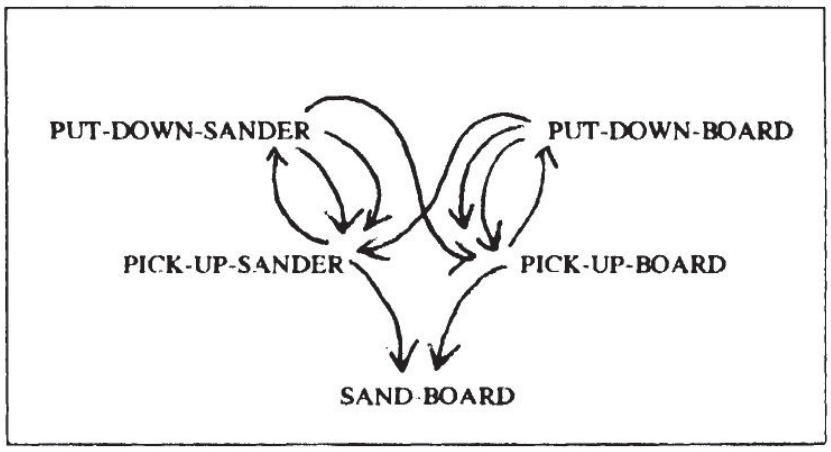

FIG. 6.2 – Comportement d'un robot avec des réseaux d'activation [66] .

6.2.2 La fusion de commande

La fusion de commande permet de combiner les recommandations des différents comportements afin d'obtenir un compromis. Les comportements peuvent simultanément contribuer au contrôle du système de façon coopérative plutôt que compétitive.

La fusion de commande est exécutée en trois étapes :

1. les recommandations des actions : un module est implémenté pour générer les actions recommandées en fonction de plusieurs critères comportementaux ;
2. l'agrégation des comportements : les actions recommandées par les comportements sont combinées en fonction de certaines règles ;
3. la sélection de l'action : une action appropriée est sélectionnée sur les recommandations combinées.

Plusieurs techniques ont été définies :

– le vote : les techniques de vote interprètent la sortie de chaque comportement comme un vote, qui est combiné en prenant en compte les votes et en sélectionnant l'action qui a reçu le maximum de votes ;

- la superposition : elle combine les recommandations des comportements en utilisant des combinaisons linéaires ;
- le flou : cette technique est très proche du vote, avec en plus des techniques d'inférence floue ;
- les objectifs multiples : c'est un mécanisme qui permet de sélectionner une action en prenant en compte plusieurs objectifs.

Nous donnerons ici un exemple pour le vote et pour la superposition. Pour le flou et les objectifs multiples, le lecteur pourra se reporter à l'état de l'art de Pirjanian [83].

Le vote

Hostetler et al. [54] ont utilisé le vote de préférence distribué pour simuler des groupes de piétons en environnement urbain. Le système de vote trouve des compromis qui permettent aux membres d'un groupe de marcher côte à côte, d'éviter des obstacles et de passer dans des rues resserrées. Chaque piéton est contrôlé par deux paramètres : la vitesse et la direction. L'espace d'action est défini dans le tableau 6.1.

accélérer tourner à gauche	accélérer tout droit	accélérer tourner à droite
vitesse identique tourner à gauche	vitesse identique tout droit	vitesse identique tourner à droite
décélérer tourner à gauche	décélérer tout droit	décélérer tourner à droite

TAB. 6.1 – Espace d'action pour le déplacement

Chaque comportement attribue une note à chaque option de l'espace d'action. Les notes sont comprises entre -1 et 1. Un vote positif indique une inclination favorable à l'option, tandis qu'une note négative indique une opposition. La note à zéro indique de l'indifférence.

Plusieurs comportements sont définis : poursuivre sa route, rester en formation, éviter les autres, éviter les obstacles, maintenir sa vitesse ...

Chaque comportement possède un poids, qui traduit son importance. Ce poids est obtenu expérimentalement ou par apprentissage. Les notes sont multipliées par ces poids avant d'être additionnées afin d'obtenir une note globale pour chaque option. L'option qui obtient la meilleure note est finalement retenue.

D'autres travaux ont utilisé cette technique du vote, comme Rosenblatt [94] avec DAMN, ou encore Sukthankar [109], qui a ajouté la notion de veto

(si un comportement veut interdire une option, il pose un veto et cette option ne pourra pas être retenue, même si elle obtient la meilleure note).

La superposition

Cette technique est utilisée dans la théorie des schémas, décrite au paragraphe 3.4.2. La consigne de déplacement est la somme pondérée des consignes générées par chaque schéma activé. Cette méthode permet donc de prendre en compte chaque comportement actif.

On peut cependant se demander si cette méthode est facilement réutilisable. En effet, il faut s'assurer que la somme de tous les schémas activés ait bien un sens et si cette somme permet bien de répondre aux objectifs de chaque comportement. Si deux schémas sont activés : éviter un obstacle et aller vers un but, il faut s'assurer que la consigne obtenue par la somme pondérée va bien permettre d'éviter l'obstacle tout en allant vers le but.

Conclusion

Les méthodes de fusion de commande permettent de sélectionner une action qui sera un compromis acceptable pour la majorité des comportements. Elles semblent donc intéressantes. Il faut cependant faire attention à ce que ce compromis soit cohérent avec les objectifs à atteindre et qu'il soit sensé.

6.3 Les méthodes d'aide multicritère à la décision

6.3.1 Définition

Un problème de décision consiste à devoir faire un choix entre plusieurs solutions possibles (ou *alternatives*). Avant l'apparition de l'analyse multicritère, les problèmes de décision se ramenaient le plus souvent à l'optimisation d'une fonction économique [71]. Cette méthode permet de poser les problèmes de façon simple, mais pas forcément de façon représentative de la réalité :

- la comparaison des solutions se fait rarement suivant un seul critère ;
- les préférences sur un critère sont, dans bien des cas, difficilement modélisables par une fonction ;
- lorsqu'il y a plusieurs objectifs, il est impossible de les atteindre tous à la fois.

L'analyse multicritère a été développée dans le cadre des sciences économiques et du génie industriel. Elle connaît un développement très important depuis 1970 ([41], [98], [97]). L'aide multicritère à la décision fournit à un décideur des modèles pour résoudre des problèmes décisionnels [100].

Il existe plusieurs façons de résoudre un problème d'aide multicritère à la décision :

- on cherche la meilleure alternative ou un ensemble des meilleures alternatives (problématique du choix) ;
- on range les alternatives de la meilleure à la moins bonne (problématique du tri) ;
- on classe les alternatives dans des catégories (problématique du rangement).

Dans ces problèmes, plusieurs points de vue sont à prendre en considération. Le décideur possède des préférences qu'il exprime sur chaque point de vue par un *critère*. Les *critères* sont donc les aspects suivant lesquels les alternatives sont analysées.

Les différents points de vue sont généralement contradictoires : un problème ne possède pas une solution qui soit la meilleure pour tous les critères. La meilleure solution est celle qui fournit le meilleur compromis pour tous les critères.

La résolution d'un problème d'aide multicritère à la décision se fait en plusieurs étapes ([63], [14]) :

1. définition de la liste des alternatives potentielles à étudier : on doit avoir la liste, la plus complète possible, des solutions au problème. On note A l'ensemble des alternatives : A= $\{a, b, \ldots, j\}$;
2. définition de la liste des critères à prendre en considération. Vincke [120] définit le critère de la façon suivante : un critère est "une fonction g, définie sur A, qui prend ses valeurs dans un ensemble totalement ordonné, et qui représente les préférences du décideur selon un point de vue". On note les critères $X = \{g_1, g_2, \ldots, g_n\}$;
3. pondération des critères : on définit l'importance des critères les uns par rapport aux autres avec des *poids* ;
4. établir le tableau de performances : il contient l'évaluation des alternatives selon chaque critère. Il est constitué, en ligne, des n alternatives a_i et en colonne, des n critères g_j. Les valeurs qui remplissent le tableau sont les $g_j(a_i)$, qui correspondent à l'évaluation de l'action a_i suivant le critère g_j. Les g_j sont des fonctions d'utilité. Ce tableau peut également contenir d'autres informations, comme les poids ;
5. agréger les performances : il s'agit d'agréger les performances afin de définir quelle solution possède globalement les meilleures évaluations.

Les quatre premières étapes sont communes à toutes les méthodes d'aide multicritère à la décision. La dernière étape est différente suivant la méthode utilisée.

6.3.2 Trois grandes familles de méthodes

Les méthodes se différencient par la façon d'effectuer la synthèse de l'information contenue dans chaque critère (étape 5 au paragraphe précédent). Ces méthodes ont été divisées en trois grandes familles [120] :

- la théorie de l'utilité multi-attribut,
- les méthodes de surclassement,
- les méthodes interactives.

Roy[99] les appelle respectivement :

- approche du critère unique de synthèse évacuant toute incomparabilité,
- approche du surclassement de synthèse, acceptant l'incomparabilité,
- approche du jugement local interactif avec itérations essai-erreur.

Et Scharlig [103] les appelle respectivement méthodes d'agrégation :

- complètes,
- partielles,
- locales.

La théorie de l'utilité multi-attribut

Cette méthode, d'inspiration anglo-saxonne et très employée aux Etats-Unis, est très utilisée pour les problèmes d'aide à la décision, mais également pour les problèmes d'économie et de finances. Elle consiste à inclure toutes les performances dans une fonction d'utilité ou d'agrégation. Elle repose sur l'axiome suivant [120] :

"Tout décideur essaye inconsciemment de maximiser une fonction $u = f(g_1, g_2, \ldots, g_n)$, qui agrège tous les points de vue à prendre en compte."

Cette fonction permet alors de comparer les actions entre elles et de les ranger de la meilleure à la moins bonne.
La relation de préférence du décideur sur les alternatives est notée $\succsim$, $a \succsim b$ signifie que le décideur préfère l'alternative a à l'alternative b.

La fonction $u : A \rightarrow \mathbb{R}$ est la fonction d'utilité globale qui représente une relation de préférences globale $\succsim$ et que tout décideur tente de maximiser :

$$a \succsim b \Leftrightarrow u(a) \geq u(b)$$

L'utilité globale dépend seulement des critères : $u(a) = f(g_1(a), g_2(a), \ldots g_n(a))$. La fonction f est la fonction d'agrégation.

Voici quelques exemples de méthodes :

- la somme ou moyenne pondérée,
- MAUT (Multiple Attribute Utility Theory) ([41], [42], [58]),
- UTA (Utilité additive) [56],
- AHP (Anaytic Hierarchy Process) [102].

Les méthodes de surclassement

On doit cette méthode, plutôt européenne, à Roy [96]. Cette approche consiste à comparer les alternatives deux à deux et à vérifier si, selon certaines conditions préétablies, l'une des deux actions surclasse l'autre ou pas.

Une relation de surclassement aSb signifie que a est au moins aussi bonne que b.

La relation de surclassement aSb est construite en deux étapes :

- les actions sont comparées par paires : a et b sont comparées indépendamment des autres actions en comparant les $g_i(a)$ et les $g_i(b)$ pour les n indices,
- on agrège les résultats des comparaisons avec une fonction qui donne la préférence globale.

Une alternative a surclasse b si, par exemple, a est meilleure que b sur au moins un critère tout en étant au moins aussi bonne qu'elle sur tous les autres critères.

Contrairement à la méthode de la théorie de l'utilité multi-attribut dans laquelle on agrège puis on compare, dans la méthode de surclassement, on compare d'abord puis on agrège.

Voici quelques exemples de méthodes : Electre ([101], [99]), Prométhée [20], Oreste [95].

Les méthodes interactives

Cette méthode consiste à alterner entre des étapes de calcul et des étapes de dialogue avec le décideur. Elle est utilisée dans les cas où il y a un grand nombre d'alternatives. Après un premier calcul, le résultat est présenté au décideur qui peut alors apporter plus d'informations sur ses préférences. Ces nouvelles informations sont ensuite prises en compte dans le calcul suivant. Ainsi le décideur intervient à deux niveaux : il définit le problème (définition des critères ...) puis il construit la solution en interagissant dans la méthode. Voici quelques exemples de méthodes : Stem [15], la méthode de Geoffrion,

Dyer et Feinberg [49] ...

Nous avons présenté rapidement les trois familles de méthodes de l'aide multicritère à la décision. La théorie de l'utilité multi-attribut semble naturellement adaptée pour modéliser le choix de l'opérateur. Elle permettrait de prendre en compte tous ses critères de choix dans la fonction d'utilité. Afin d'illustrer plus particulièrement la théorie de l'utilité multi-attribut, nous donnons trois exemples de méthodes : la somme pondérée, la moyenne pondérée ordonnée et l'intégrale de Choquet.

6.3.3 Exemples de méthodes en théorie de l'utilité multi-attribut

Exemple de fonction d'agrégation additives : la somme pondérée

Les fonctions d'agrégation additives sont de la forme :

$$u(a) = \sum_{i=1}^{n} f_i(g_i(a))$$

avec f_i monotones croissantes.

Un exemple très utilisé de fonctions d'agrégation additives est la somme pondérée, qui est de la forme :

$$u(a) = \sum_{i=1}^{n} f_i(g_i(a))$$

avec f_i monotones croissantes.

La somme pondérée est de la forme :

$$u(a) = \sum_{i=1}^{n} w_i.g_i(a),$$

avec w_i le poids associé au critère g_i.

En général, on a : $\sum_{i=1}^{n} w_i = 1$.

L'exemple cité ici est tiré de [51]. Il s'agit de l'évaluation d'étudiants dans un lycée pour trois matières : les mathématiques, la physique et la littérature. Chaque matière a un coefficient. On suppose que le proviseur du lycée attribue plus d'importance aux matières scientifiques. Prenons, respectivement, les coefficients 3, 3 et 2. Prenons trois étudiants : A, B et C. Les moyennes sont calculées dans la tableau 6.2.

La moyenne pondérée possède quelques limites qui peuvent être gênantes. Reprenons l'exemple des étudiants. Si l'école veut favoriser les élèves qui n'ont pas de points faibles, la somme pondérée n'est pas satisfaisante. En effet, l'élève qui obtient la meilleure moyenne est l'élève A, alors qu'il possède une note basse en littérature. L'élève C obtient une note un peu inférieure alors que cet élève ne semble pas avoir de point faible. On obtient de tels résultats car des coefficients trop importants ont été donnés aux mathématiques et à la physique, alors que, en général, si un élève est bon en mathématiques, il sera bon en physique, et vice versa. Ces critères sont donc redondants. Cette évaluation attribue donc trop d'importance aux matières scientifiques.

	Poids	A	B	C
Mathématiques	3	18	10	14
Physique	3	16	12	15
Littérature	2	10	18	15
Moyenne		15.25	12.75	14.62

TAB. 6.2 – Moyenne pondérée pour chaque étudiant

De plus, la somme pondérée ne permet pas de prendre en compte les conflits entre les critères. L'exemple du tableau 6.3 est celui de Vincke [120]. Il contient les évaluations de a, b, c et d suivant les critères g_1 et g_2. A chacun de ces critères est attribué le poids de 0.5.

Le calcul de la somme pondérée donne : $u(a) = 100, u(b) = 50, u(c) = 100, u(d) = 50$.

Critères	Poids	a	b	c	d
g_1	0.5	100	50	200	0
g_1	0.5	100	50	0	200

TAB. 6.3 – Limite de la somme pondérée : pas de prise en compte des conflits entre les critères

Les alternatives a et c ainsi que les alternatives c et d obtiennent les mêmes moyennes. On peut en déduire les relations suivantes : $a \succsim b$ et $c \succsim d$. L'alternative a est bien meilleure que b pour les deux critères alors qu'il y a un conflit important entre les deux critères pour la comparaison de c et d.

La somme pondérée est simple et pratique à utiliser. Cependant elle souffre de limites qui peuvent être gênantes :

- elle peut favoriser des alternatives peu équilibrées,
- elle ne prend pas en compte le caractère conflictuel entre les critères.

Les moyennes pondérées ordonnées (Ordered Weighted Averaging, OWA)

La moyenne pondérée ordonnée a été introduite par Yager [121].

C'est une déformation de la somme pondérée, les alternatives sont rangées et les poids portent sur les rangs :

$$u_{owa}(a) = \sum_{i=1}^{n} w_i.g_i(a)^*,$$

avec les poids w_i tels que

$$\sum_{i=1}^{n} w_i = 1.$$

La notation $g_i(a)^*$ signifie que les $g_i(a)$ ont été réordonnées :
$g_1(a)^* \leq g_2(a)^* \leq \ldots \leq g_n(a)^*$.

La moyenne est un cas particulier d'OWA avec les poids w_i égaux à $\frac{1}{n}$ pour tout i.

Exemple de modèle non-additifs : l'intégrale de Choquet

L'intégrale de Choquet [26] permet de pallier aux limites de la somme pondérée : elle prend en compte l'interaction entre les critères (aussi bien les interactions négatives que positives entre critères). Pour cela, on introduit, en plus des poids sur les critères pris individuellement, des poids définis sur des groupes de critères.

L'intégrale de Choquet est une déformation de la somme pondérée et de la moyenne pondérée ordonnée (OWA).
Elle utilise une capacité ou mesure floue :

Définition 40 : Mesure floue Une mesure floue sur l'ensemble X est une fonction $\mu : 2^X \rightarrow [0,1]$, avec $\mu(X) = 1$, $\mu(\emptyset) = 0$, et vérifiant la propriété de monotonie : $S \subset T \Rightarrow \mu(S) \leq \mu(T)$

La définition de monotonie traduit le fait que l'importance d'un groupe de critères ne peut décroître si on ajoute un critère au groupe.

Définition 41 : Intégrale de Choquet Soit μ une mesure floue définie sur un ensemble de critères cardinaux $X = \{g_1, g_2, \ldots, g_n\}$. L'intégrale de Choquet sur a est définie par :

$$u_\mu(a) = \sum_{i=1}^{n} w_i.g_i(a)^*,$$

avec $w_i = \mu(\{g_i^*, \ldots g_n^*\}) - \mu(\{g_{i+1}^*, \ldots g_n^*\})$.

La notation $g_i(a)^*$ signifie que les $g_i(a)$ ont été réordonnées :
$g_1(a)^* \leq g_2(a)^* \leq \ldots \leq g_n(a)^*$.

Cette fois, les poids w_i dépendent :
– du rang (comme dans la moyenne pondérée ordonnée),
– du rang des autres composantes.
La propriété de monotonie permet d'avoir des w_i toujours positifs.

Reprenons l'exemple des étudiants (cf. tableau 6.2). La somme pondérée donne la meilleure moyenne à l'élève A, alors qu'il possède une note basse en littérature, et donne une moins bonne moyenne à l'étudiant C alors que ces notes sont équilibrées dans les trois matières. Nous voulons maintenant favoriser les étudiants qui ont des notes équilibrées, sans points faibles particuliers. Définissons la fonction de mesure floue :

1. dans cet exemple, les matières scientifiques sont plus importantes que les matières littéraires. On fixe les mesures floues suivantes :
   – $\mu(Mathématiques) = \mu(Physique) = 0.45$,
   – $\mu(Littérature) = 0.3$.
   On a gardé le même ratio que pour les poids initiaux des critères (3,3,2) ;
2. les critères Mathématiques et Physique sont redondants, on attribue donc un poids à l'ensemble (Mathématiques, Physique) moins important que la somme des poids des Mathématiques et Physique.
   $\mu(Mathématiques, Physique) = 0.5 < 0.45 + 0.45$;
3. on favorise les étudiants qui sont bons en science et en littérature, le poids attribué à l'ensemble (Mathématiques, Littérature) doit être plus

grand que les sommes des poids des Mathématiques et de la Littérature. $\mu(Mathématiques, Littérature) = 0.9 > 0.45 + 0.3$;

4. on raisonne de la même façon pour le poids attribué à l'ensemble (Physique, Littérature) :
   $\mu(Physique, Littérature) = 0.9 > 0.45 + 0.3$;
5. enfin, on a, par définition, $\mu(\emptyset) = 0$ et
   $\mu(Mathématiques, Physique, Littérature) = 1$.

On calcule l'intégrale de Choquet pour l'étudiant A :
- on classe les évaluations de A, suivant les critères, dans l'ordre : Littérature(A)<Physique(A)<Mathématiques(A) ;
- on définit l'intégrale de Choquet :

$$u_\mu(A) = w_1 * Littérature(A) + w_2 * Physique(A) + w_3 * Mathématiques(A)$$

avec

$$w_1 = \mu(Mathématiques, Physique, Littérature) - \mu(Mathématiques, Physique)$$

$$w_2 = \mu(Mathématiques, Physique) - \mu(Mathématiques)$$

$$w_3 = \mu(Mathématiques) - \mu(\emptyset)$$

- l'application numérique donne :

$$u_\mu(A) = 0.5 * 10 + 0.05 * 16 + 0.45 * 18 = 13.9.$$

De la même façon, on obtient (cf. tableau 6.4) :

$$u_\mu(B) = 13.6 \; et \; u_\mu(C) = 14.9.$$

L'utilisation de l'intégrale de Choquet permet donc bien d'attribuer à l'étudiant C, qui a les notes les plus équilibrées, la meilleure note.

	Poids	A	B	C
Mathématiques	3	18	10	14
Physique	3	16	12	15
Littérature	2	10	18	15
Moyenne		15.25	12.75	14.62
Intégral de Choquet		13.9	13.6	14.9

TAB. 6.4 – Moyenne pondérée et intégrale de Choquet pour chaque étudiant

Chapitre 7

Sélection de l'action avec des préférences et une méthode d'aide multicritère à la décision

L'objectif de cette partie est de simuler la décision de l'officier. Dans une situation tactique, la décision d'un opérateur est un aspect clé. A chaque instant, l'officier doit choisir une action en fonction de sa perception de la situation tactique. Nous définissons une méthode qui permet de choisir entre les différentes extensions ([115],[114], [113]) . Cette méthode permet de simuler différents types de comportements, qui dépendent du caractère de l'officier.

Cette méthode de sélection a été définie en trois étapes :

1. la définition de principes généraux auxquels doit répondre la sélection de l'extension ;
2. la définition d'une fonction de poids pour chaque extension. Cette fonction de poids prend en compte le comportement de l'officier à deux niveaux :
   - Niveau 1 : l'importance de chaque défaut en fonction de différents critères. Pour cela, on définit des coefficients de préférences $C_1, \ldots, C_n$, qui permettent de définir des préférences en fonction des différents critères. Ces coefficients décrivent l'importance de chaque action ;
   - Niveau 2 : le caractère de l'officier avec des coefficients de caractère $w_1, \ldots, w_n$. Ces coefficients définissent l'importance que l'officier accorde à chaque critère. Grâce à ces coefficients de caractère, nous pouvons modéliser différents officiers : prudent, téméraire ...

   Les principes de sélection de l'extension ainsi que la fonction de poids des extensions ont été développés pour notre cas d'application mais

peuvent être facilement généralisés ;

3. la définition d'une méthode pour sélectionner une extension, en ajoutant une part d'aléatoire dans le choix de l'extension.

7.1 Les principes de sélection d'une extension

Premièrement, nous étudions des principes et les exigences auxquels la sélection de l'extension doit répondre.

7.1.1 Principe 1 : choix des extensions les plus intéressantes

Dans certains cas, l'officier aura à choisir entre des comportements qui correspondent à des règles de changement minimal et d'autres comportements. Les règles de changement minimal doivent être appliquées tant que le sous-marin ne possède pas de nouvelles informations. Si d'autres comportements sont possibles, le choix doit donc porter sur ces nouveaux comportements.

Prenons l'exemple suivant :

Exemple 14 : NoMROD propose deux extensions :

- faire une trajectoire aléatoire de recherche,
- aller au schnorchel.

La trajectoire aléatoire de recherche est une règle de changement minimal : elle est appliquée tant que le sous-marin n'a pas de nouvelles informations. Dans cette situation, l'officier choisira donc toujours d'aller au schnorchel.

7.1.2 Principe 2 : choix des extensions obligatoires

Dans certains cas, des extensions seront obligatoires, en particulier pour la survie de l'équipage.

Par exemple, l'officier peut repousser la montée au schnorchel, si il est en train d'exécuter une autre action importante (par exemple le pistage). Lorsque cette règle est vérifiée, l'officier a approximativement trente minutes de batterie. Quand cette réserve sera pratiquement vide, l'officier sera obligé d'aller au schnorchel.

7.1.3 Principe 3 : gestion du choix entre plusieurs extensions concurrentes

NoMROD va devoir gérer les choix entre les extensions en concurrence.

Exemple 15 : NoMROD propose deux extensions :
- éviter la collision avec un sous-marin,
- éviter la collision avec un gros rocher.

Ces deux comportements sont très importants pour la sauvegarde du sous-marin. NoMROD doit être capable de choisir entre les extensions qui semblent aussi importantes les unes que les autres.

7.1.4 Principe 4 : respect du changement minimal

Tant que le sous-marin n'a pas de nouvelles informations, il reste dans le même état, il ne change pas de comportement. Pour respecter cette règle, on peut donner plus d'importance à une action déjà commencée. Avec cette règle, l'officier persistera dans ses choix, il n'oscillera pas entre plusieurs comportements. Cependant, NoMROD doit être capable de stopper une action si une autre devient obligatoire.

Exemple 16 : NoMROD propose deux extensions :
- pister l'ennemi,
- monter au schnorchel.

Supposons que le système choisisse le pistage. Ces deux extensions seront proposées tant que la montée au schnorchel n'aura pas été effectuée. Le meilleur choix pour l'officier est de poursuivre le pistage, et lorsque le schnorchel devient obligatoire (le sous-marin a juste assez d'électricité pour monter à la surface), l'officier doit effectuer cette action.

7.1.5 Principe 5 : effet de surprise

Si le sous-marin ennemi ne peut pas deviner quelle action notre officier va choisir, il sera moins prévisible et aura l'avantage de l'effet de surprise sur son adversaire.

Défauts	$C_{sauvegarde}$	$C_{efficacite}$
D_1	500	600
D_2	10	800
D_3	1000	900
D_4	50	60

TAB. 7.1 – Valeurs des coefficients pour chaque défaut.

7.2 La fonction de poids pour les extensions

Nous définissons une fonction de poids pour donner des poids aux extensions. Ces poids quantifient l'importance des extensions. Nous utilisons une méthode d'aide multicritères à la décision, afin de modéliser les préférences de l'officier. Ces méthodes servent à résoudre des problèmes décisionnels complexes, où plusieurs critères doivent être pris en compte dans le choix (cf. le paragraphe 6.3). Nous utilisons la théorie de l'utilité multi-attribut, qui permet d'agréger les différents points de vue en une fonction unique. Nous utilisons une fonction d'agrégation simple : la somme pondérée.

Chaque extension utilise des défauts. Pour quantifier l'importance des extensions, nous introduisons des coefficients de préférences sur les défauts.

7.2.1 Les coefficients de préférences sur les défauts

Chaque défaut est un comportement général. Pour spécifier l'importance des défauts, nous leurs attribuons des coefficients de préférences.

Nous définissons différents coefficients de préférences $C_1, \ldots, C_n$ pour spécifier l'importance des défauts en fonction de différents critères. Nous pouvons spécifier l'importance des défauts pour la sauvegarde du sous-marin, pour l'efficacité dans la mission du sous-marin, pour l'obéissance aux ordres, pour des critères écologiques ...

Pour chaque défaut D_j, nous attribuons les valeurs de ces coefficients $C_{1j} \ldots C_{kj}, \ldots, C_{nj}$. Le coefficient C_{kj} définit l'importance du défaut j par rapport au critère k. Ces coefficients sont fixés de façons arbitraire par l'utilisateur, entre 0 et 1000.

EXEMPLE 17 : On définit quatre défauts : $\{D_1, D_2, D_3, D_4\}$, et deux coefficients : la sauvegarde du sous-marin $C_{sauvegarde}$ et l'efficacité dans la mission du sous-marin $C_{efficacite}$. Pour chaque défaut, on définit la valeur des coefficients de sauvegarde et d'efficacité (cf tableau 7.1).

7.2.2 Fonction d'utilité

Chaque extension E_i utilise des défauts : $E_i = \{D_1 \ldots, D_j, \ldots, D_m\}$. Dans notre cas, chaque extension utilise au moins un défaut.

Pour chaque extension E_i, nous calculons les scores

$$Score_1(E_i), \ldots, Score_k(E_i), \ldots, Score_n(E_i),$$

qui correspondent respectivement aux critères $1, \ldots, n$.

Un score $Score_k(E_i)$ est la somme des coefficients C_k de chaque défaut utilisé dans l'extension E_i :

$$Score_k(E_i) = \sum_{j=1}^{m} C_{kj}$$

Supposons que NoMROD propose p extensions. Pour chaque $Score_k(E_i)$ pour une extension E_i, nous calculons une fonction d'utilité $g_k(E_i)$, de façon à ce que la somme des fonctions d'utilité g_k soit égale à 1 :

$$\sum_{j=1}^{p} g_k(E_j) = 1$$

Le score $Score_k(E_i)$ est divisé par la somme de tous les $Score_k(E_j)$ des p extensions proposées par le système :

$$g_k(E_i) = \frac{Score_k(E_i)}{\sum_{j=1}^{p} Score_k(E_j)}$$

Reprenons les défauts donnés dans l'exemple 17.

On suppose que l'on doit choisir entre deux extensions : $E_1 = \{D_1, D_4\}$ et $E_2 = \{D_2, D_3, D_4\}$. Les coefficients de préférences sont définis dans le tableau 7.1.

On calcule les scores des extensions (cf tableau 7.2) et les fonctions d'utilité (cf tableau 7.3).

7.2.3 Le caractère de l'officier

Nous voulons modéliser le caractère de l'officier. En fonction de son caractère, l'officier adoptera des tactiques différentes. Un officier prudent aura tendance à donner plus d'importance aux comportements qui assurent la

Scores	$E_1 = \{D_1, D_4\}$	$E_2 = \{D_2, D_3, D_4\}$
$Score_{sauvegarde}$	550	1060
$Score_{efficacite}$	660	1760

TAB. 7.2 – Scores des extensions.

g	$E_1 = \{D_1, D_4\}$	$E_2 = \{D_2, D_3, D_4\}$
$g_{sauvegarde}$	$\frac{550}{1610}$	$\frac{1060}{1610}$
$g_{efficacite}$	$\frac{660}{2420}$	$\frac{1760}{2420}$

TAB. 7.3 – Fonctions d'utilité.

sauvegarde du sous-marin. Un officier téméraire favorisera les comportements importants pour la réussite de la mission du sous-marin.

Pour modéliser ces caractères, nous définissons des coefficients de caractère $w_1, \ldots, w_n$, de façon à ce que la somme de ces coefficients soit égale à 1 :

$$\sum_{k=1}^{n} w_k = 1$$

Le coefficient w_k correspond à l'importance accordée par l'officier au critère k.

Définissons un officier plutôt prudent. En reprenant l'exemple 17, on doit attribuer deux coefficients de caractère : un coefficient pour la sauvegarde $w_{sauvegarde}$ et un coefficient pour l'efficacité $w_{efficacite}$. Un officier prudent va préférer favoriser la sauvegarde de son sous-marin plutôt que l'efficacité dans la réussite de la mission, prenons : $w_{sauvegarde} = 0.6$ et $w_{efficacite} = 0.4$.

7.2.4 Fonction de poids

Finalement, nous obtenons la fonction de poids de chaque extension E_i, en faisant la somme des fonctions d'utilité g_k, pondérées par les coefficients de caractère w_k :

$$P(E_i) = \sum_{k=1}^{n} w_k g_k(E_i)$$

Les définitions des fonctions d'utilité et des coefficients de caractère sont reprises dans le tableau 7.4.

La somme des fonctions de poids des extensions vérifie la propriété suivante :

$$\sum_{i=1}^{p} P(E_i) = 1$$

En reprenant l'exemple précédent, on obtient les fonctions de poids suivantes :

$$P(E_1) = w_{sauvegarde} * g_{sauvegarde}(E_1) + w_{efficacite} * g_{efficacite}(E_1),$$

$$P(E_2) = w_{sauvegarde} * g_{sauvegarde}(E_2) + w_{efficacite} * g_{efficacite}(E_2).$$

Finalement, on a $P(E_1) = 0.31$ et $P(E_2) = 0.69$.

7.3 Choix aléatoire d'une extension

Dans une situation tactique, la décision d'un opérateur est un aspect clé, qui peut conduire à des réactions inattendues. En effet, chaque opérateur possède ses propres réactions, qui dépendent de la façon dont il appréhende la situation. Dans notre cas d'application, le choix ne doit donc pas toujours être déterministe : face à une même situation, des officiers peuvent prendre des décisions différentes. Par exemple, l'officier peut avoir à choisir entre la poursuite du pistage du sous-marin ennemi et le dérobement dans la cas où l'ennemi se trouve à une distance critique. Le caractère de l'officier, ou encore la façon dont il ressent la situation, influenceront son choix final. Pour une même situation, les décisions prises par les officiers ne seront donc peut être pas les mêmes.

De plus, des réactions inattendues permettent à l'officier de jouer sur l'effet de surprise : le sous-marin ennemi pourra difficilement deviner les réactions de l'officier (principe 5, cf. paragraphe 7.1).

Coef. de préférences suivant différents critères	Défauts de l'extension E_i					$Scores(E_i)$	Fonction d'utilité, choix entre p extensions	Poids : coef. de caractère de l'officier
	D_1	..	D_j	..	D_m			
C_1 Exemple : importance du défaut pour la sauvegarde du sous-marin	C_{11}	..	C_{1j}	..	C_{1m}	$Score_1(E_i) = \sum_{j=1}^{m} C_{1j}$	$g_1(E_i) = \frac{Score_1(E_i)}{\sum_{j=1}^{p} Score_1(E_j)}$	ω_1
...	...	..	...	..	...	...	...	...
C_k Exemple : importance du défaut pour la réalisation de la mission	C_{k1}	..	C_{kj}	..	C_{km}	$Score_k(E_i) = \sum_{j=1}^{m} C_{kj}$	$g_k(E_i) = \frac{Score_k(E_i)}{\sum_{j=1}^{p} Score_k(E_j)}$	ω_k
...	...	..	...	..	...	...	...	...
C_n	C_{n1}	..	C_{nj}	..	C_{nm}	$Score_n(E_i) = \sum_{j=1}^{m} C_{nj}$	$g_n(E_i) = \frac{Score_n(E_i)}{\sum_{j=1}^{p} Score_n(E_j)}$	ω_n

TAB. 7.4 – Définition des fonctions d'utilité $g_k(E_i)$ et des coefficients de caractère ω_k

Nous avons donc besoin d'une méthode qui prend en compte ces réactions inattendues. Pour ces raisons, nous ne choisissons pas toujours les extensions avec une fonction de poids maximum. Nous préférons introduire davantage d'incertitude avec un choix aléatoire. Cependant, ce choix aléatoire doit être cohérent avec les principes qui guident la décision de l'officier et avec les principes de sélection d'une extension définis au paragraphe 7.1.

7.3.1 Choix aléatoire

Le choix aléatoire est basé sur un tirage aléatoire : on a p extensions $E_1, \ldots, E_p$, et les fonctions de poids respectives : $P(E_1), \ldots, P(E_p)$, telles que

$$\sum_{i=1}^{p} P(E_i) = 1.$$

Chaque fonction de poids $P(E_i)$ correspond à la probabilité pour l'extension E_i d'être choisie. Avec ce choix aléatoire, nous pouvons donc gérer les choix entre plusieurs extensions (principe 3, cf. paragraphe 7.1). Prenons l'exemple suivant :

Exemple 18 : NoMROD doit choisir entre deux extensions :
- E_1 avec la fonction de poids $P(E_1) = 0.45$.
- E_2 avec la fonction de poids $P(E_2) = 0.55$.

Dans une telle situation, le tirage aléatoire semble très naturel, car les poids sont proches. Par contre, il peut poser problème dans des cas comme le suivant :

Exemple 19 : NoMROD doit choisir entre deux extensions :
- E_1 avec la fonction de poids $P(E_1) = 0.1$.
- E_2 avec la fonction de poids $P(E_2) = 0.9$.

Dans une telle situation, il semble plus naturel de choisir la deuxième extension. Or, le tirage aléatoire ne reflète pas ce choix : l'extension E_1 a une chance sur dix d'être choisie.

Nous avons le même problème dans le cas suivant :

Exemple 20 : Nous devons choisir entre six extensions :
- cinq extensions avec une même fonction de poids à 0.1 ;
- une extension avec une fonction de poids de 0.5.

Le tirage aléatoire donne autant de chance d'être choisies aux cinq extensions avec la fonction de poids à 0.1 : $5 * 0.1 = 0.5$, qu'à l'extension avec la fonction de poids à 0.5.

Afin de résoudre ce problème, nous apportons une correction aux fonctions de poids des extensions.

7.3.2 Correction de la fonction de poids des extensions

Nous devons corriger la fonction de poids, afin de donner un poids plus important aux extensions importantes et un poids moins important aux autres. Pour cela, nous appliquons une correction aux fonctions de poids : la fonction puissance $f(x) = x^k$, avec $k > 1$. La correction est appliquée de la façon suivante :

- nous devons choisir entre p extensions : $E_1, \ldots, E_p$, avec les fonctions de poids respectives : $P(E_1), \ldots, P(E_p)$, telles que

$$\sum_{i=1}^{p} P(E_i) = 1;$$

- nous appliquons la fonction puissance $f(x) = x^k$, avec $k > 1$: $P(E_1)^k, \ldots, P(E_p)^k$. Plus la puissance k est importante, plus les extensions avec des poids faibles seront réduites.
- la somme des fonctions de poids est ensuite ramenée à 1 : nous divisons par la somme des fonctions de poids de toutes les extensions :

$$P_{cor}(E_j) = \frac{P(E_j)^k}{\sum_{i=1}^{p} P(E_i)^k}.$$

Cette correction donne plus d'importance aux extensions avec des fonctions de poids importantes, et moins d'importance aux autres.

Appliquons cette correction à l'exemple 20. Nous prenons la fonction puissance $f(x) = x^2$. La fonction de poids 0.1 devient $0.1^2 = 0.01$ et la fonction de poids à 0.5 devient $0.5^2 = 0.25$. On ramène la somme des fonctions de poids à 1. La somme des fonctions de poids avec correction est $\sum_{i=1}^{6} P(E_i) = 0.3$. On obtient cinq extensions avec la fonction de poids $P_{cor}(E_i) = \frac{0.1^2}{0.3} = 0.04$ avec $i = 1, \ldots, 5$, et une extension avec la fonction de poids $P_{cor}(E_6) = \frac{0.5^2}{0.3} = 0.8$. Avec la correction, on donne plus de chance à l'extension avec un poids important d'être choisi.

7.3.3 Filtrage des extensions avec des fonctions de poids trop faibles

Nous voulons choisir les extensions les plus intéressantes, ou encore les extensions obligatoires, en laissant les autres de côté (principes 1 et 2, para-

graphe 7.1). Pour cela, nous éliminons les extensions avec des fonctions de poids trop faibles. On fixe un seuil : les extensions avec une fonction de poids inférieure à ce seuil sont supprimées. Ce seuil est fixé de manière arbitraire, nous choisissons un pourcentage de la fonction de poids maximum des extensions.

7.4 Prise en compte du changement minimal

Pour respecter le changement minimal (principe 4, cf. paragraphe 7.1), nous définissons une règle générale de changement minimal. Pour cela, on garde en mémoire le comportement du sous-marin en cours (trajectoire aléatoire, l'évitement de la collision, le pistage, ...) au temps t. On appelle ce comportement $Comportement(t)$. Dans notre cas, chaque comportement est le conséquent d'un défaut D_i :

$$D_i = \frac{Prerequis(t) : Comportement(t+1)}{Comportement(t+1)}.$$

Avec la règle du changement minimal, nous voulons donner plus de chance à une action déjà commencée. Pour cela, nous définissons le défaut suivant :

$$D_{ch\_min} = \frac{Comportement(t) \wedge Prerequis(t) : Comportement(t+1)}{Comportement(t+1)}.$$

Cette règle signifie : "Si le prérequis du comportement précédent est toujours vrai au temps t, et si il est possible de rester dans ce comportement au temps $t+1$, le sous-marin peut garder le même comportement au temps $t+1$".

Cette règle donne plus de chance à une action déjà commencée et permet à l'officier de persister dans ses choix.

7.5 Exemple de calcul pour la consigne de déplacement de l'officier

Afin d'illustrer cette partie concernant la sélection de l'extension, nous reprenons l'exemple donné au paragraphe 5.2.

Le calcul des extensions avait donné deux extensions : l'extension E_1 qui propose le pistage et l'extension E_2 qui propose au sous-marin de faire un

dérobement.

L'extension E_1 utilise deux défauts :

$$D2 : \frac{détection(X_t, 1) : chasse(X_{t+1})}{chasse(X_{t+1})},$$

$$D6 : \frac{chasse(X_t) \wedge Tmd(X_t) > 30 : pistage(X_{t+1})}{pistage(X_{t+1})}.$$

Et l'extension E_2 utilise les deux défauts suivants :

$$D2 : \frac{détection(X_t, 1) : chasse(X_{t+1})}{chasse(X_{t+1})},$$

$$D8 : \frac{détection(X_t, 1) \wedge D\_ennemi <= 2000 : dérobement(X_{t+1})}{dérobement(X_{t+1})}.$$

7.5.1 Coefficients de préférences sur les défauts en fonction des critères

On reprend les critères donnés en exemple précédemment :
- $C_{sauvegarde}$, qui représente l'importance du défaut pour la sauvegarde du sous-marin,
- $C_{efficacite}$, qui représente l'importance du défaut pour la réalisation de la mission.

On définit les coefficients de préférence pour les trois défauts dans le tableau 7.5

	Défaut D_2 (Chasse)	Défaut D_6 (Pistage)	Défaut D_8 (Dérobement)
$C_{sauvegarde}$	500	500	1000
$C_{efficacite}$	1000	1000	800

TAB. 7.5 – Coefficients de préférences pour les défauts

Critère $C_{efficacite}$

Le comportement "Chasse" correspond à la mission du sous-marin : il regroupe toutes les actions que le sous-marin doit accomplir en cas de détection.

Le comportement "Pistage" permet d'accomplir la dernière phase de la mission, le pistage de l'ennemi.

Le dérobement consiste à s'éloigner du sous-marin ennemi pour ne pas être contre-détecté. Une fois que le sous-marin est hors de danger, il peut reprendre son action de pistage. Le dérobement contribue donc aussi à accomplir la mission, même si le pistage sera moins rapidement atteint.

Pour ces raisons, nous mettons les coefficients pour les comportements "Chasse" et "Pistage" au maximum pour le critère $C_{efficacite}$, et le coefficient pour le dérobement est un peu moins élevé, à 800.

Critère $C_{sauvegarde}$

Les comportements "Chasse" et "Pistage" font prendre le risque au sous-marin de se faire contre-détecter : on leur met donc le coefficient à 500 pour le critère de sauvegarde du sous-marin.

Le comportement "Dérobement" a pour finalité d'éviter au sous-marin de se faire contre-détecter et il lui permet de s'éloigner de l'ennemi. On met donc le coefficient maximum pour le critère $C_{sauvegarde}$.

7.5.2 Fonction d'utilité

Pour chaque extension, les scores $Score_{sauvegarde}$ et $Score_{efficacite}$ et les fonctions d'utilité $g_{sauvegarde}$ et $g_{efficacite}$ sont calculés dans le tableau 7.6.

	$Score_{sauvegarde}$	$Score_{efficacite}$	$g_{sauvegarde}$	$g_{efficacite}$
$E_1 = \{D_2, D_6\}$	1000	2000	$\frac{1000}{2500}$	$\frac{2000}{3800}$
$E_2 = \{D_2, D_8\}$	1500	1800	$\frac{1500}{2500}$	$\frac{1800}{3800}$

TAB. 7.6 – Scores et fonctions d'utilité pour les extensions

7.5.3 Coefficient de caractère de l'officier

Définissons deux officiers différents :
- un officier prudent, qui privilégiera les actions de sauvegarde,
- un officier téméraire, qui privilégiera les actions d'efficacité.

Les coefficients de caractère sont définis dans le tableau 7.7.

	Officier prudent	Officier téméraire
$C_{sauvegarde}$	0.8	0.1
$C_{efficacite}$	0.2	0.9

TAB. 7.7 – Coefficients de caractère de l'officier

7.5.4 Fonction de poids des extensions

Finalement, les fonctions de poids pour les extensions sont calculées dans le tableau 7.8.

	$g_{sauvegarde}$	$g_{efficacite}$	Fonction de poids pour l'officier prudent	Fonction de poids pour l'officier téméraire
E_1 (Pistage)	$\frac{1000}{2500}$	$\frac{2000}{3800}$	0.42	0.51
E_2 (Dérobement)	$\frac{1500}{2500}$	$\frac{1800}{3800}$	0.58	0.49

TAB. 7.8 – Fonctions de poids pour les extensions, en fonction des différents caractères de l'officier

L'officier prudent accorde un peu plus d'importance au comportement de dérobement qu'au comportement pistage et l'officier téméraire apporte pratiquement autant d'importance aux deux comportements.

7.5.5 Correction des fonctions de poids des extensions

On applique la correction aux fonctions de poids, en utilisant la fonction puissance carrée (cf. tableaux 7.9 et 7.10).

L'utilisation de la fonction de correction permet pour l'officier prudent, d'avoir une fonction de poids plus importante pour le dérobement. Il aura donc plus de chance de choisir le dérobement. Pour l'officier téméraire, les

fonctions de poids restent très proches, la correction ne les modifie pas de façon significative.

	Fonction de poids pour l'officier prudent	Fonction à la puissance carrée	Somme ramenée à 1
E_1 (Pistage)	0.42	0.1764	0.34
E_2 (Dérobement)	0.58	0.3364	0.66

TAB. 7.9 – Fonctions de poids avec correction pour l'officier prudent

	Fonction de poids pour l'officier prudent	Fonction à la puissance carrée	Somme ramenée à 1
E_1 (Pistage) (Pistage)	0.51	0.2601	0.52
E_2 (Dérobement)	0.49	0.2401	0.48

TAB. 7.10 – Fonctions de poids avec correction pour l'officier téméraire

7.6 Conclusion

Nous choisissons une extension grâce à un tirage aléatoire sur les fonctions de poids. Cette technique nous permet de gérer le choix entre plusieurs extensions concurrentes (principe 3, cf. paragraphe 7.1), tout en ayant un effet de surprise (principe 5). La définition de la fonction de poids permet de définir l'importance des comportements, en fonction de différents critères avec les coefficients de préférence et elle permet de prendre en compte le caractère de l'officier avec les coefficients de caractère.

La correction sur la fonction de poids des extensions et le filtrage des extensions avec des fonctions de poids trop faibles permet de choisir les extensions les plus intéressantes ou encore les extensions obligatoires (principes 1 et 2).

Enfin, le défaut du changement minimal permet à l'officier de favoriser une action déjà commencée et de persister dans ses choix (principe 5).

Quatrième partie

Résultats

Chapitre 8

Comparaison des résultats obtenus avec les deux modélisations du comportement

Nous avons effectué une comparaison des résultats obtenus avec la modélisation du comportement avec la théorie des schémas (cf. chapitre 4) et avec la modélisation du comportement obtenue avec la logique des défauts (cf. chapitre 5), en faisant des simulations avec l'atelier de simulation ATANOR (cf. paragraphe 1.1.3).

Afin de rendre ces deux méthodes comparables, nous devons limiter notre modélisation du comportement avec la logique des défauts aux règles qui ont été définies pour la théorie des schémas. Pour effectuer cette étude, nous avons donc utilisé les règles de comportements suivantes (cf. paragraphe 2.3) :

- Règle 1 : les abattées,
- Règle 2 : la chasse (anti-collision, mesure de distance, ralliement de la position de pistage, pistage),
- Règle 3 : le dérobement,
- Règle 5 : la perte de détection.

Pour la même raison, la méthode de sélection de l'action pour la modélisation avec la logique des défauts a été modifiée par rapport à la méthode de sélection de l'action décrite au chapitre 7 :

- nous définissons un seul coefficient de préférence pour chaque défaut : on définit l'importance du défaut suivant un seul critère ;
- la fonction de poids d'une extension est définie par la somme des coefficients de préférences des défauts de l'extension ;
- on choisit l'extension avec la fonction de poids maximale.

Avec ces règles de comportement, les seuls comportements qui peuvent entrer en compétition sont les règles de la chasse et du dérobement. Les poids

sont fixés de façon à ce que le comportement dérobement soit prioritaire par rapport aux comportements de la chasse.

Pour comparer ces deux modélisations, nous avons effectué une étude paramétrique qui permet de définir différents caractères pour l'officier de quart.

8.1 Simulation de différents caractères pour l'officier de quart

8.1.1 Paramètres

Les paramètres avec lesquels nous avons défini ces différents comportements sont :

- le temps écoulé entre la première détection et la manœuvre d'anti-collision,
- le temps passé en manœuvre de mesure de distance,
- la distance de sécurité à laquelle l'officier maintient son sous-marin par rapport à l'ennemi lorsqu'il n'a pas encore atteint le baffle de l'ennemi (c'est-à-dire lorsque le sous-marin rallie sa position de pistage).

8.1.2 Définition des caractères

Trois caractères ont été définis :

- **standard** : les valeurs qui définissent ce comportement viennent de l'expérience générale des sous-mariniers. L'officier maintient son sous-marin à une distance de sécurité par rapport à l'ennemi : il se place entre la distance de détection de son sous-marin (afin de ne pas perdre le contact) et la distance de détection du sous-marin adverse (afin de ne pas se faire contre détecter) ;
- **prudent** : en comparaison avec l'officier standard, l'officier prudent commence plus rapidement la manœuvre d'anti-collision. Ensuite, il prend plus de temps pour effectuer la mesure de distance : il préfère avoir une très bonne information sur la position et la vitesse de l'ennemi. Durant le ralliement du poste de pistage, il reste plus éloigné de l'ennemi ;
- **imprudent** : en comparaison avec l'officier standard, l'officier imprudent prend beaucoup de temps avant de commencer la manœuvre d'anti-collision (ce qui lui permet d'avoir une meilleure estimation de la cinématique de l'ennemi). Ensuite, il effectue rapidement la manœuvre de

mesure de distance. Enfin, durant le ralliement de pistage, il se maintient à une distance très proche de l'ennemi.

8.1.3 Mise en pratique de la paramétrisation

Temps écoulé entre la première détection et la manœuvre d'anti-collision La définition de ce paramètre donne exactement le même comportement pour les deux modélisations : un temps à respecter entre la première détection et la manœuvre d'anti-collision est défini.

Temps passé en manoœuvre de mesure de distance La définition de ce paramètre donne exactement le même comportement pour les deux modélisations : nous définissons le temps à passer en manœuvre de mesure de distance.

Distance de sécurité à laquelle l'officier maintient son sous-marin par rapport à l'ennemi Ce paramètre est défini de façon différente pour les deux modélisations.

Modélisation avec la théorie des schémas Le schéma positionnement centré sur la cible permet de maintenir le sous-marin à une certaine distance de l'ennemi.

La distance de sécurité est définie avec les paramètres D_{min} et D_{max} de la fonction d'intensité du schéma positionnement centré sur la cible (cf. figure 8.1).

Comme il est indiqué dans le tableau 4.1, ce schéma est activé en même temps que le schéma ralliement de la position de pistage, lorsque le sous-marin n'est pas dans le baffle de l'ennemi. Le schéma ralliement de la position de pistage permet de rallier l'arrière de l'ennemi.

Les deux schémas activés en même temps permettent finalement au sous-marin de :

- rallier le baffle de l'ennemi,
- se maintenir à une distance comprise entre D_{min} et D_{max}.

Modélisation avec la logique des défauts Avec la logique des défauts, ces règles sont définies différemment.

Nous avons défini le défaut qui permet au sous-marin de rallier sa position de pistage (cf. paragraphe 5.1.4) :

$$D5 = \frac{chasse(X_t) \wedge Tmd(X_t) > 30 : rallier\_pistage(X_{t+1})}{rallier\_pistage(X_{t+1})}.$$

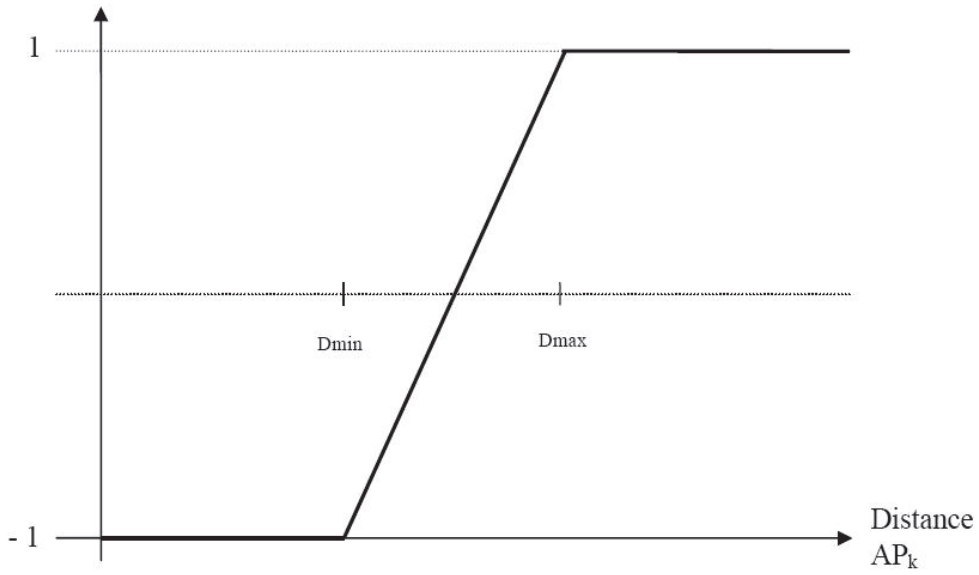

FIG. 8.1 – Fonction d'intensité du schéma positionnement centré sur la cible

Une fois positionné dans le baffle de l'ennemi, le sous-marin aura fini son ralliement de la position de pistage. On définit cette règle avec l'exclusion mutuelle :

$$w3 = \neg(dans\_baffle(X_t) \wedge rallier\_pistage(X_t)).$$

La vitesse qui correspond à l'état $rallier\_pistage(X_{t+1})$ est définie sur la figure 8.2.

Nous avons défini un deuxième défaut qui permet au sous-marin de maintenir une certaine distance avec l'ennemi, grâce à une manœuvre de dérobement :

$$D7 = \frac{détection(X_t, 1)) \wedge D\_ennemi <= D_{min} : dérobement(X_{t+1})}{dérobement(X_{t+1})}$$

La vitesse qui correspond à l'état $dérobement(X_{t+1})$ est définie sur la figure 8.3.

Les défauts $D5$ et $D7$ vont entrer en compétition dans certaines situations. Nous mettons un poids beaucoup plus important au défaut $D7$ afin qu'il soit toujours prépondérant sur le défaut $D5$. Le sous-marin préférera donc se dérober plutôt que de continuer à rallier sa position de pistage. Le sous-marin va alors manœuvrer pour placer l'ennemi dans un gisement de 150 °.

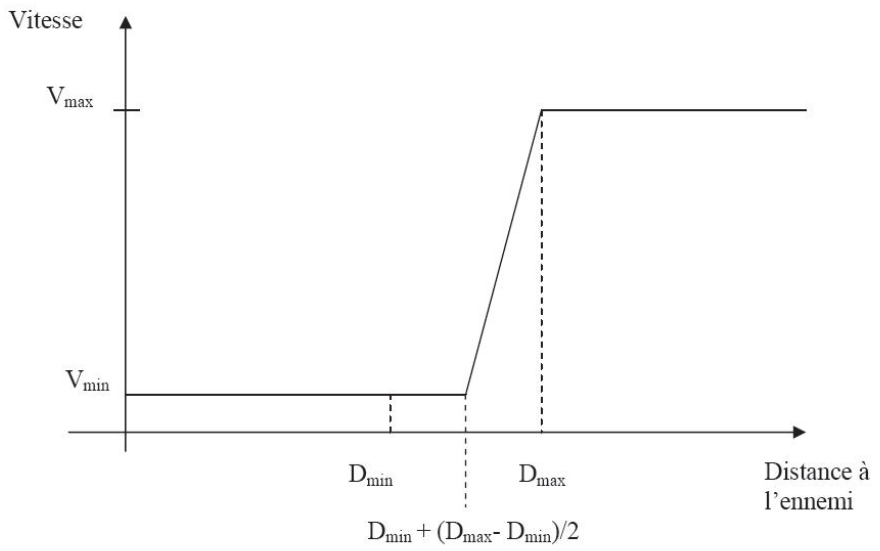

FIG. 8.2 – Vitesse en ralliement du poste de pistage

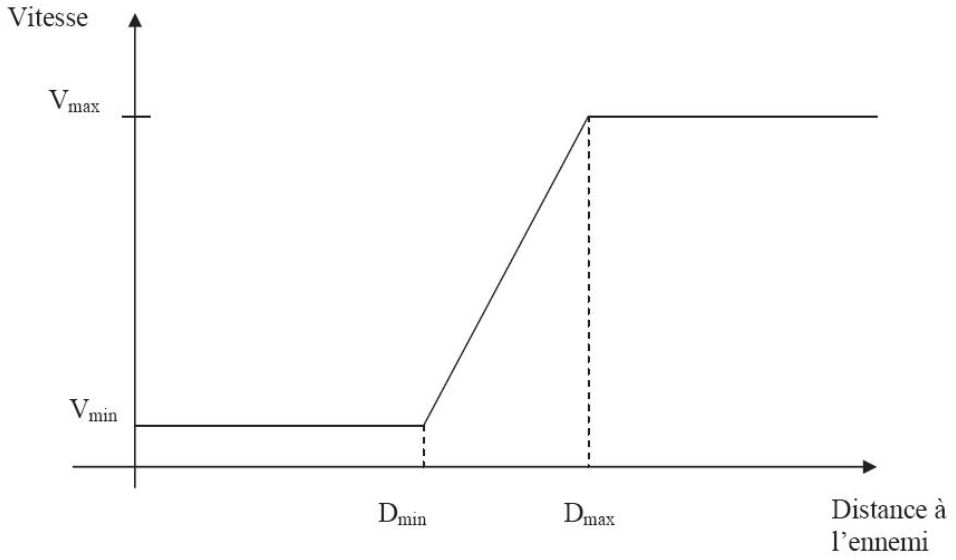

FIG. 8.3 – Vitesse en dérobement

Une fois qu'il se sera suffisamment éloigné, il pourra reprendre le ralliement sans risquer de se faire contre détecter.

8.2 Résultats étudiés

Nous avons fait une étude statistique de type Monte-Carlo, en faisant varier aléatoirement quelques paramètres incertains pour un même jeu de données d'entrée. Afin d'obtenir une bonne convergence des résultats, nous avons lancé les calculs sur 1000 exécutions de simulations.

On appelle pisteur le sous-marin qui possède la modélisation du comportement de l'officier et on appelle bruiteur le sous-marin adverse. Les résultats étudiés pour les simulations sont les suivants :

Taux d'exécution efficace Il représente le pourcentage d'exécutions lors desquelles des détections ont lieu entre les deux sous-marins du scénario pour une simulation Monte-Carlo.

Probabilité de pistage réussi La probabilité de pistage réussi est égale au nombre d'exécutions lors desquelles le pisteur est entré en phase de pistage sans être contre-détecté par le bruiteur, divisé par le nombre d'exécutions efficaces d'une simulation Monte-Carlo.

Distance de première détection Les distances moyennes des premières détections sont calculées pour les sonars du sous-marin sur l'ensemble de simulations Monte-carlo.

Délai avant le pistage Le délai avant le pistage est le temps qui s'écoule entre le première détection effectuée par le pisteur et la date de passage en pistage. La moyenne est calculée sur l'ensemble des simulations de Monte-Carlo.

Distance en début de pistage La distance au début du pistage est la distance qui sépare les deux sous-marins au commencement de la phase de pistage. La moyenne est calculée sur l'ensemble des simulations de Monte-Carlo.

Nous étudions les résultats des simulations à deux niveaux :
- l'influence de la modélisation du comportement (c'est-à-dire théorie des schémas ou bien logique des défauts) de l'officier sur les résultats,
- l'influence du caractère de l'officier (standard, prudent, imprudent) sur la réussite de la mission.

8.3 Influence de la modélisation du comportement de l'officier

8.3.1 Résultats

Dans cette partie, nous comparons les résultats obtenus pour les différents caractères avec la théorie des schémas et la logique des défauts. On fixe à 1 les résultats obtenus avec la logique des défauts.

Sur la figure 8.4, nous comparons les résultats pour les deux modélisations avec un officier de caractère standard.

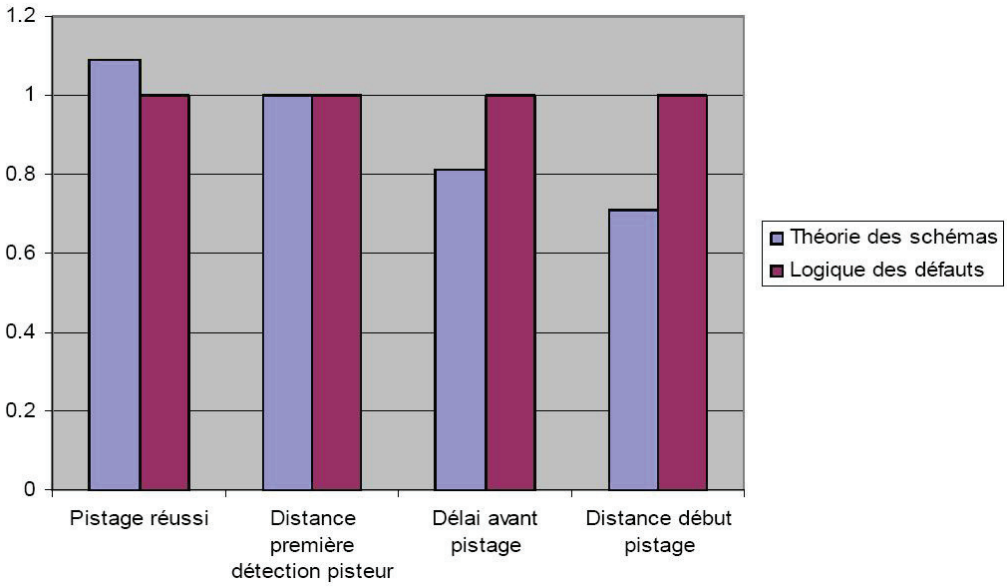

FIG. 8.4 – Officier au caractère standard

Nous avons une exécution d'un scénario avec un officier au caractère standard avec les théories des schémas sur la figure 8.5 et la logique des défauts sur la figure 8.6.

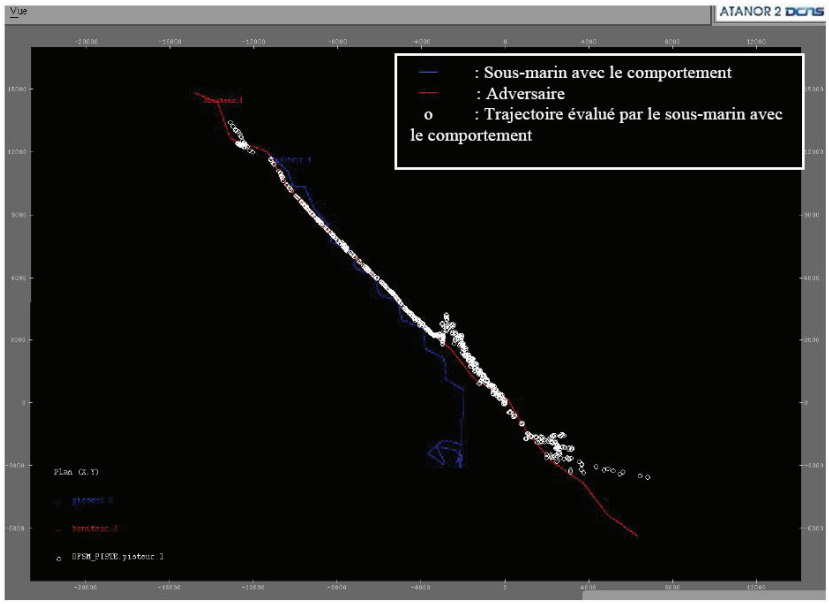

FIG. 8.5 – Modélisation d'un officier au caractère standard avec la théorie des schémas

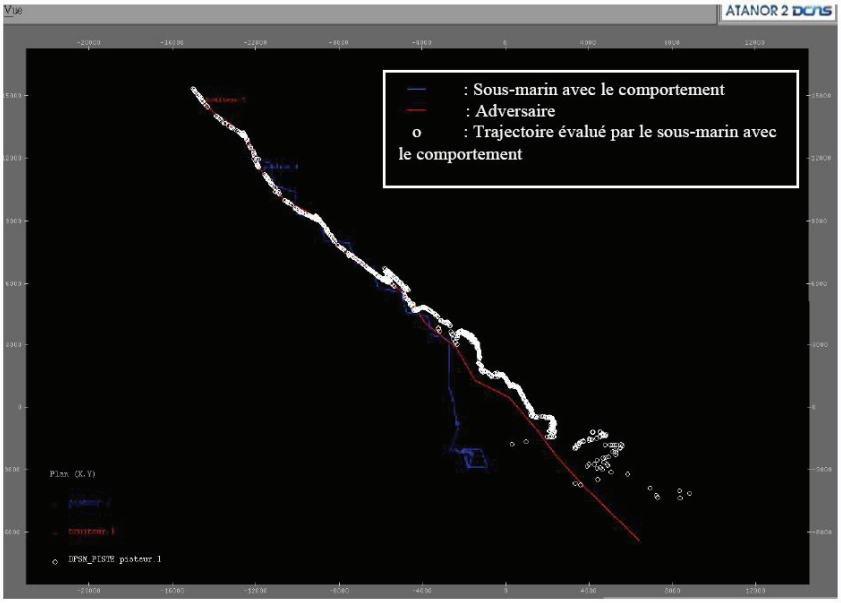

FIG. 8.6 – Modélisation d'un officier au caractère standard avec la logique des défauts

Sur la figure 8.7, nous comparons les résultats pour les deux modélisations avec un officier de caractère prudent.

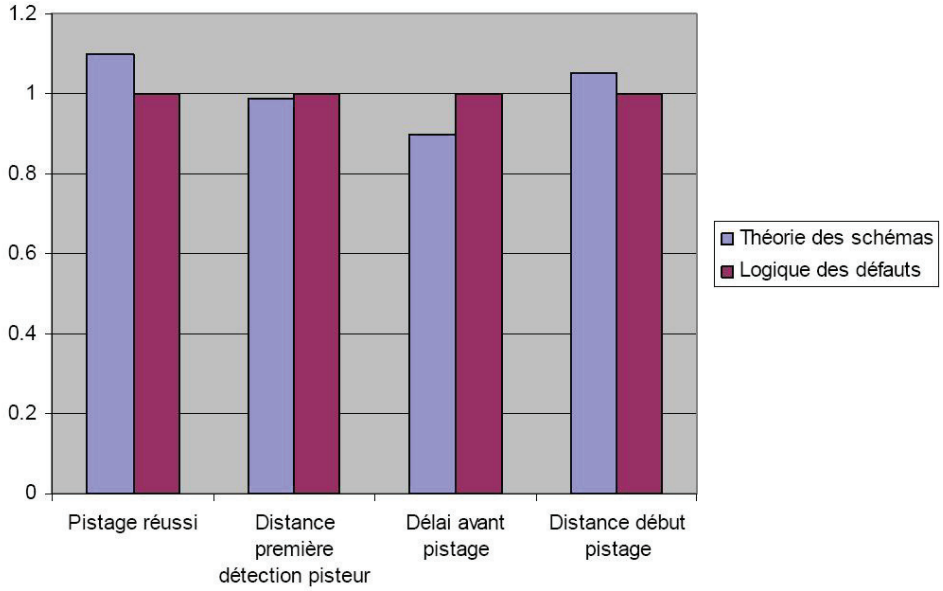

FIG. 8.7 – Officier au caractère prudent

Nous avons une exécution d'un scénario avec un officier au caractère prudent avec les théories des schémas sur la figure 8.8 et la logique des défauts sur la figure 8.9.

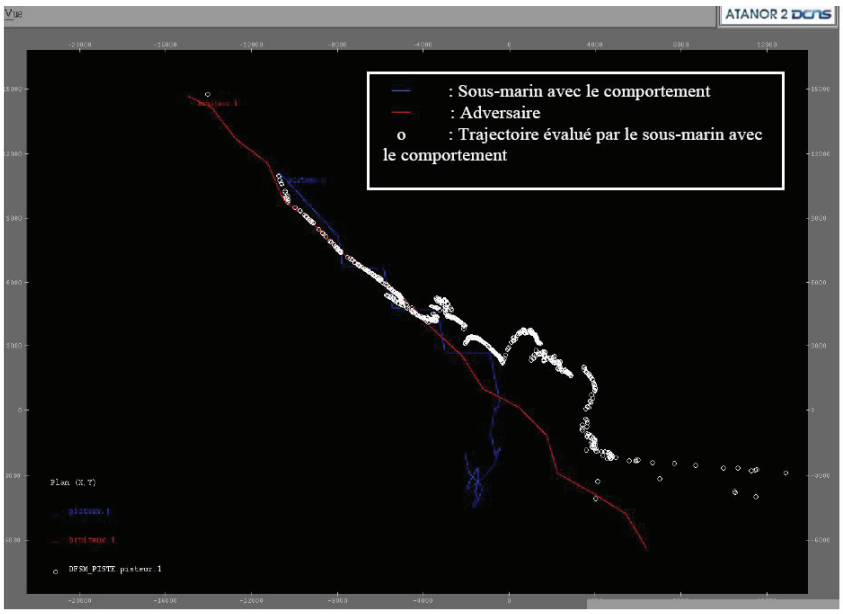

FIG. 8.8 – Modélisation d'un officier au caractère prudent avec la théorie des schémas

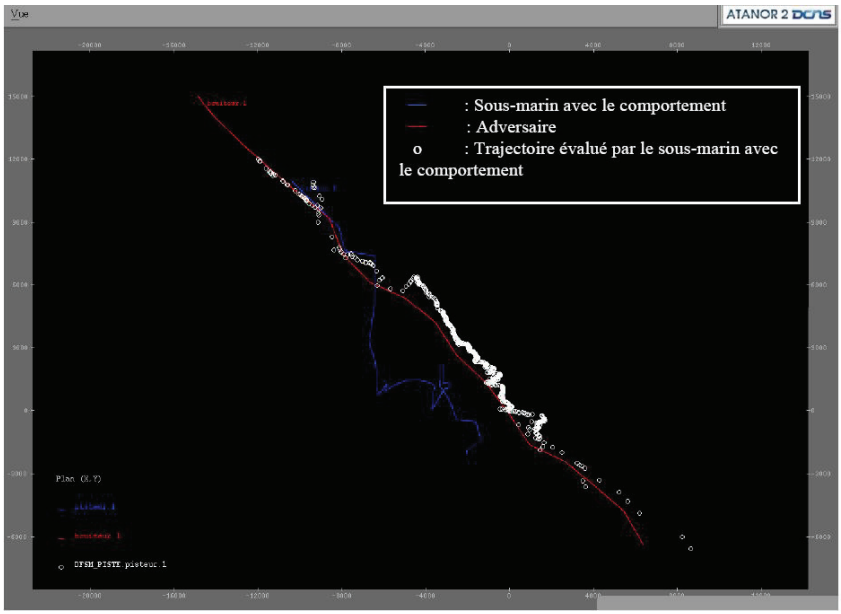

FIG. 8.9 – Modélisation d'un officier au caractère prudent avec la logique des défauts

Sur la figure 8.10, nous comparons les résultats pour les deux modélisations avec un officier de caractère imprudent.

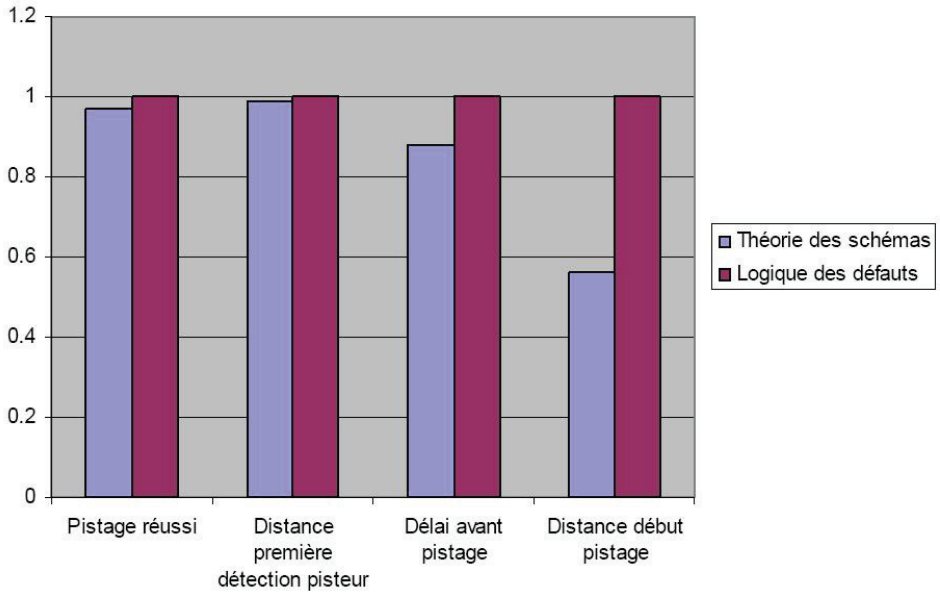

FIG. 8.10 – Officier au caractère imprudent

Nous avons une exécution d'un scénario avec un officier au caractère imprudent avec les théories des schémas sur la figure 8.11 et la logique des défauts avec la figure 8.12.

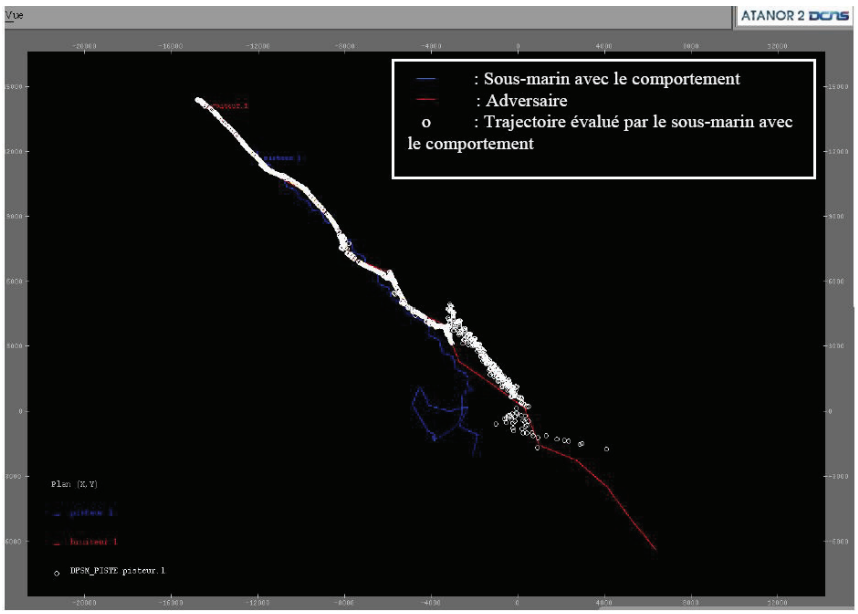

FIG. 8.11 – Modélisation d'un officier au caractère imprudent avec la théorie des schémas

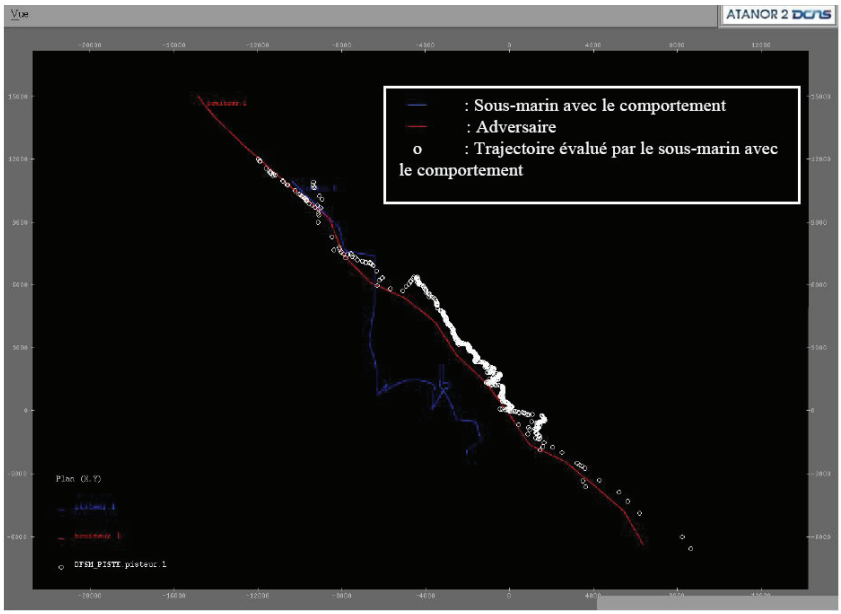

FIG. 8.12 – Modélisation d'un officier au caractère imprudent avec la logique des défauts

8.3.2 Analyse des résultats

Dans tous les cas, nous obtenons la même distance de première détection. Ce résultat est normal : la distance de première détection dépend uniquement des sonars du sous-marin.

On observe des légères différences pour les taux de pistage réussi en fonction de la modélisation utilisée. La modélisation avec la théorie des schémas fournit un taux un peu plus élevé pour les caractères standards et prudents. Ces différences viennent de la manœuvre d'anti-collision qui n'est pas définie exactement de la même façon pour les deux modélisations. Cette légère différence peut amener le sous-marin à se faire contre-détecter juste à la fin de cette manœuvre ou bien à obtenir une moins bonne estimation de la position de l'ennemi.

Pour tous les caractères, la modélisation par la théorie des schémas permet au sous-marin de rentrer un peu plus rapidement en pistage et à une distance un peu plus proche que pour la logique des défauts. Cette différence s'explique par les règles qui permettent de fixer la distance de sécurité du sous-marin par rapport à l'ennemi.
En effet, avec la théorie des schémas, la règle du ralliement du poste de pistage (avec le schéma ralliement de la position de pistage) et la règle qui maintient le sous-marin à une certaine distance de l'ennemi (avec le schéma positionnement centré sur la cible) sont déclenchées en même temps. Le sous-marin rallie sa position de pistage tout en gardant une distance de sécurité. Suivant la distance à l'ennemi, le sous-marin est plus ou moins attiré vers l'ennemi.
Dans le cas de la modélisation avec la logique des défauts, le sous-marin rallie sa position de pistage en modulant sa vitesse en fonction de la distance à l'ennemi. Cependant, cette condition ne suffit pas pour le maintenir suffisamment éloigné de l'ennemi. Lorsqu'il se retrouve trop proche, il doit s'en éloigner en exécutant un dérobement. Une fois suffisamment éloigné, il peut reprendre son ralliement. Ce comportement correspond au comportement réellement employé à la mer : un officier préfère s'assurer de ne pas se faire contre-détecter avant de se mettre à pister son adversaire.

8.4 Influence du caractère de l'officier sur la réussite de la mission

8.4.1 Résultats

Dans cette partie, nous étudions l'influence du caractère de l'officier sur la mission. On fixe les résultats obtenus pour le caractère standard avec la logique des défauts à 1.

Pour les différents caractères, nous comparons les résultats obtenus pour
- la réussite du pistage sur la figure 8.13,
- le délai avant le pistage sur la figure 8.14,
- la distance en début de pistage sur la figure 8.15.

On ne montre pas les résultats sur la distance de première détection. Comme nous l'avons vu précédemment, la distance de première détection ne dépend que des sonars du sous-marins. Le comportement de l'officier n'a donc aucune influence dessus.

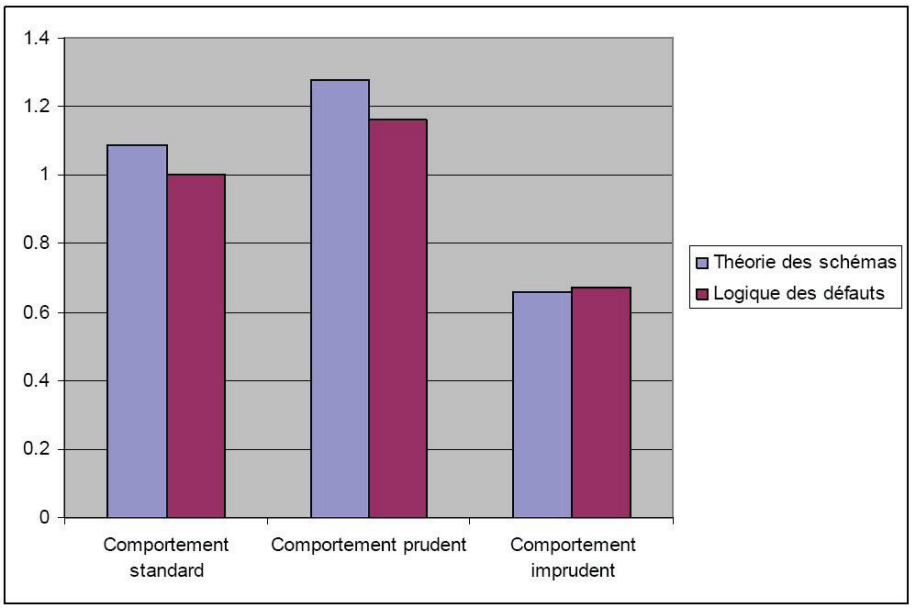

FIG. 8.13 – Influence du caractère de l'officier sur la réussite du pistage

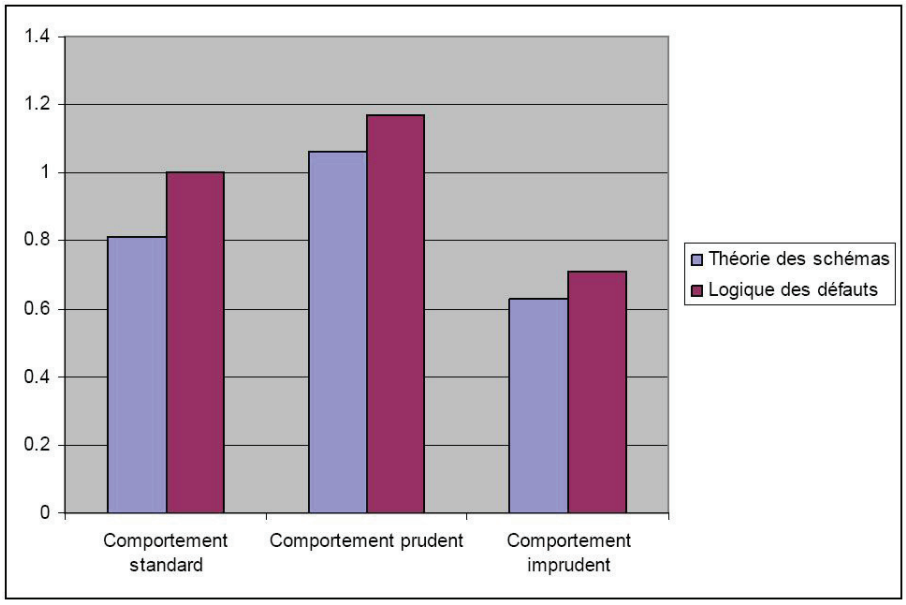

FIG. 8.14 – Influence du caractère de l'officier sur le délai avant pistage

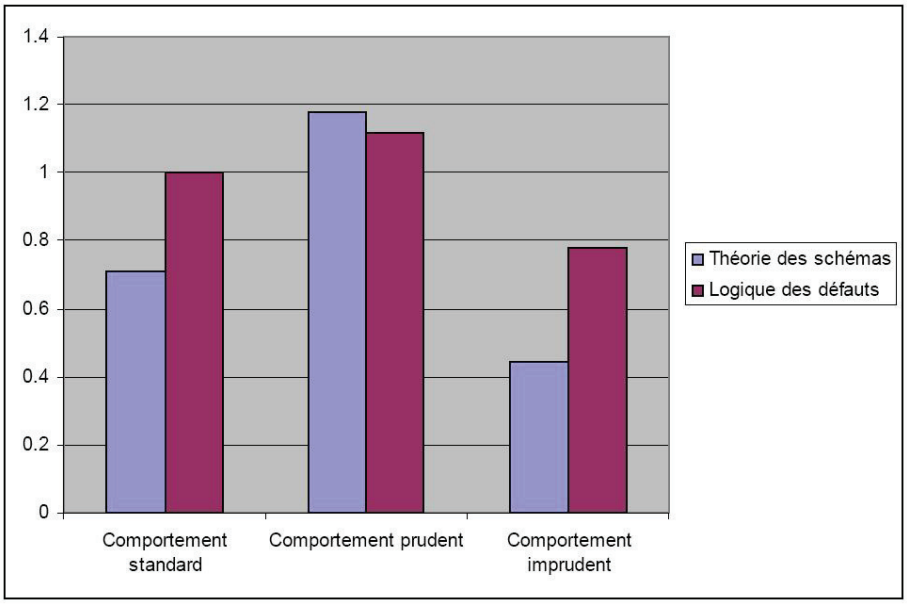

FIG. 8.15 – Influence du caractère de l'officier sur la distance en début de pistage

8.4.2 Analyse des résultats

Ces résultats confirment que le comportement de l'officier a une influence sur la réussite de la mission.

Le taux de pistage réussi est plus important avec l'officier au caractère prudent et moins important avec celui au caractère imprudent, en comparant avec l'officier au caractère standard (cf. figure 8.13). L'officier au caractère prudent passe plus de temps en manœuvre de mesure de distance. Il a donc une meilleure estimation de la cinématique de l'ennemi. De plus, il se maintient davantage éloigné de l'ennemi, ce qui lui permet de se faire moins souvent contre-détecter (cf. figure 8.15). Le risque de ce comportement est de perdre la détection du sous-marin ennemi et de ne pas pouvoir engager une arme. De plus, ce comportement implique un plus grand délai de mise en œuvre du pistage (cf. figure 8.14), ce qui peut avoir un impact sur l'autonomie du sous-marin et la réussite de la mission.

La distance qui sépare les deux sous-marins en début de pistage dépend du caractère de l'officier (cf. figure 8.15). L'officier imprudent va se retrouver plus proche de l'ennemi, ce qui provoque plus de risque de se faire contre-détecter.

8.5 Conclusion

Afin de rendre les deux modélisations comparables, nous avons limité notre modélisation du comportement avec la logique des défauts aux règles qui avaient été définies pour la théorie des schémas. Pour la même raison, nous n'avons pas pris en compte notre méthode de sélection de l'action, nous nous sommes contentés de choisir l'extension avec la fonction de poids la plus importante. Notre modélisation avec la logique des défauts nous a permis de nous affranchir de la représentation des connaissances par automates à états finis. L'utilisateur pourra ainsi ajouter de nouvelles règles facilement. De plus, contrairement à la théorie des schémas qui prend en compte tous les comportements activés, la logique des défauts nous a permis de définir des exclusions mutuelles entre des comportements incompatibles. Enfin, cette représentation avec la logique des défauts nous offre la possibilité de pouvoir choisir entre les extensions calculées et elle nous permet de prendre en compte différents caractères pour les officiers.

L'étude nous montre que les deux modélisations du comportement (logique des défauts et théorie des schémas) nous donnent globalement les mêmes résultats. De légères différences sont dues au fait que certaines règles ne sont pas définies exactement de la même façon.

En jouant sur certains paramètres, nous avons réussi à définir différents caractères pour l'officier. Cette étude met en avant les avantages et inconvénients de ces différents caractères.

Conclusions et perspectives

Cette thèse propose une réponse aux besoins de l'entreprise DCNS en matière de modélisation du comportement humain dans les simulations de combat naval. Suivant la situation tactique, l'analyse et la décision d'un opérateur peuvent conduire à des réactions inattendues. L'objectif de la thèse est de développer une modélisation du comportement de cet opérateur.

Pour développer cette modélisation, nous nous plaçons dans le cas d'un scénario faisant intervenir deux sous-marins adverses. L'opérateur est l'officier de quart en charge du sous-marin. Le sous-marin évolue dans un contexte très difficile : il doit faire preuve de discrétion acoustique afin de ne pas se faire détecter. La plupart des informations dont il dispose viennent des sonars passifs. Ces informations sont incomplètes, incertaines et révisables. Le sous-marin risque, à tout moment, de perdre le contact de son adversaire. De plus, il raisonne avec une trajectographie estimée de son adversaire.

Une première étude, présentée dans le Chapitre 4, a porté sur la première technique utilisée à DCNS pour modéliser le comportement humain. Cette technique est basée sur la théorie des schémas. Elle décompose le comportement en entités indépendantes : les schémas. Cette méthode donne de bons résultats. Cependant, il ne nous a pas semblé judicieux de la réutiliser dans le cadre de la thèse pour différentes raisons. L'utilisation des automates à états finis permet difficilement d'ajouter de nouveaux comportements. Les schémas sont limités à un comportement cinématique. Enfin, tous les schémas activés étant pris en compte dans le calcul de la consigne de route du sous-marin, nous risquons d'obtenir des comportements incohérents.

Nous avons choisi de représenter les connaissances avec la logique non-monotone la plus utilisée : la logique des défauts (cf. Chapitre 5). Cette logique, à laquelle nous ajoutons une prise en compte du temps, nous permet de répondre aux exigences que nous nous étions fixées.

Elle nous permet de prendre en compte les informations incomplètes, incertaines et révisables dont l'opérateur dispose. Les défauts permettent de définir des règles générales, qui admettent des exceptions. En définissant les règles en logique des défauts, nous nous affranchissons de la représentation

des connaissances sous forme d'automate à états finis. De plus, de nouvelles règles de comportement peuvent être ajoutées, sans avoir à modifier les règles précédemment définies.

Ce travail étant demandé par la société DCNS pour des applications militaires, nous avons dû faire des choix techniques pour obtenir un programme robuste, rapide, interfaçable avec ATANOR et réutilisable par la suite par des utilisateurs non développeurs. Pour ces raisons, notre programme a été implémenté en langage Prolog. Nous avons choisi de définir les faits avec des formules et des clauses de Horn. Les clauses de Horn sont simplement implémentées en Prolog et facilement compréhensibles. L'ensemble des défauts est défini avec des défauts normaux. Nous obtenons un algorithme de calcul d'extension efficace et rapide.

Notre programme, interfacé avec l'atelier de simulation ATANOR, est utilisé pour des études. La modélisation du comportement a été validée par des sous-mariniers, qui retrouvent bien les actions engagées à la mer. De plus, cette modélisation est suffisamment robuste pour que le sous-marin mène à bien sa mission, malgré les mauvaises informations dont il dispose sur la trajectoire de l'ennemi.

Une fois les connaissances représentées en logique des défauts, nous abordons le problème de la sélection de l'action : comment choisir une extension parmi toutes celles calculées avec la logique des défauts. La méthode que nous proposons permet de prendre en compte l'importance de chaque défaut ainsi que le caractère de l'officier. Le choix final de l'extension est finalement guidé par le caractère de l'officier.

Pour réaliser cette thèse, nous avons fait des choix techniques (clauses de Horn, défauts normaux, prise en compte du temps ...). Ces choix conviennent à notre cas d'application et à nos règles de comportement. Cependant, ce travail pourrait être réutilisé pour d'autres cas d'application. Dans ce cas, il sera facile de généraliser notre programme. Nous pourrons utiliser d'autres clauses que les clauses de Horn et d'autres défauts que les défauts normaux. Nous prenons en compte le temps de manière discrète : à partir des informations connues à l'instant t, les actions possibles au temps $t+1$ sont proposées. Les actions aux temps $t+2$, $t+3$, ..., $t+n$ pourraient également être examinées. Cependant, nous aurions probablement à faire face à des problèmes de complexité algorithmique.

Nous avons programmé l'algorithme de calcul d'extension en Prolog. Nous obtenons un algorithme simple et efficace. Une autre approche possible pour calculer ces extensions est l'utilisation de la programmation par ensemble réponse (ou ASP) pour calculer les extensions. L'utilisation des ASP pourrait être une piste de recherche pour poursuivre ces travaux.

Notre méthode de sélection d'extension est basée sur des principes, qui sont des principes de bon sens, et qui peuvent facilement être généralisés à d'autres cas d'application. De même, la fonction de poids calculée pour chaque extension est une fonction simple : une somme pondérée. Ces fonctions de poids sont donc également généralisables. Cependant, il serait peut être intéressant de tester d'autres fonctions que la somme pondérée, comme, par exemple, l'intégrale de Choquet.

Bibliographie

[1] J.F. Allen, Towards a general theory of action and time, *Artificial Intelligence*, **23**(2), 123–154, (1984).

[2] J.M. Alliot and T. Schiex, *Intelligence artificielle et informatique théorique*, Cepadus, 2002.

[3] G. Amati, L.C Aiello, and F. Pirri, Definability and commonsense reasoning, *Artificial Intelligence*, **93**(1), 169–200, (1997).

[4] C. Anger, K. Konczak, and T. Linke, Nomore : Non-monotonic reasoning with logic programs, *Logics in Artificial Intelligence*, 521–524, (2002).

[5] M. Arbib, The metaphorical brains, *Artificial Intelligence*, **101**(1-2), 323–335, (1998).

[6] M.A. Arbib, Perceptual structures and distributed motor control, *Handbook of physiology, Section*, **2**, 1449–1480, (1981).

[7] M.A. Arbib and D.H. House, Depth and detours : an essay on visually guided behavior, in *Vision, brain, and cooperative computation*, p. 163. MIT Press, (1990).

[8] R.C. Arkin, Motor schema-based mobile robot navigation, in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 264–271, (1987).

[9] R.C. Arkin and T. Balch, AuRA : Principles and practice in review, *Journal of Experimental & Theoretical Artificial Intelligence*, **9**(2), 175–189, (1997).

[10] R.C. Arkin and T.Balch, Behavior-based formation control for multirobot teams, *IEEE transactions on robotics and automation*, **14**, 926–939, (1997).

[11] F. Baader and B. Hollunder, How to prefer more specific defaults in terminological default logic, in *International joint conference on artificial intelligence*, volume 13, pp. 669–669. Morgan Kaufman Publishers, (1993).

[12] F. Bacchus, J. Tenenderg, and J.A. Koomen, A non-reified temporal logic, *Artificial Intelligence*, **52**(1), 87–108, (1991).

[13] T. Balch, G.Boone, T. Collins, H. Forbes, and D. MacKenzie J.C. Santamaiia, Io, ganymede and callisto – a multiagent robot trash-collecting team, *AI Magazine*, **16**(2), (1995).

[14] S. Ben Mena, Introduction aux méthodes multicritères d'aide à la décision, *Biotechnol. Agron. Soc. Environ*, **4**(2), 83–93, (2000).

[15] R. Benayoun, J. De Montgolfier, J. Tergny, and O. Laritchev, Linear programming with multiple objective functions : Step method (STEM), *Mathematical Programming*, **1**(1), 366–375, (1971).

[16] F. Benhammadi, *La gestion des prÃľfÃ©rences en logique des dÃ©fauts = Handling Preference in default logic*, Ph.D. dissertation, UniversitÃ© d'Angers, 1999.

[17] B. Benhamou, T. Nabhani, and P. Siegel, Symétries dans les logiques non monotones, *JFPC'10*, (2010).

[18] P. Besnard and T. Schaub, Possible worlds semantics for default logics, *Fundamenta Informaticae*, **21**, 39–66, (1994).

[19] J. Botto, I. Toulgoat, and C Audoly, Modélisation de l'avantage tactique d'un sous-marin, *10ème Congrès Français d'Acoustique*, (2010).

[20] JP Brans, Mareschal. B., and Vincke. P., Promethee : A new family of outranking methods in multicriteria analysis, in *Operational research'84 : proceedings of the Tenth IFORS International Conference on Operational Research : actes de la Dixième Conférence internationale de recherche opérationnelle de l'IFORS, Washington, DC, USA, August 6-10, 1984*. North Holland, (1984).

[21] G. Brewka, Cumulative default logic : In defense of nonmonotonic inference rules, *Artificial Intelligence*, **50**(2), 183–205, (1991).

[22] G. Brewka, Adding priorities and specificity to default logic., in *Lecture Notes in Artificial Intelligence*, eds., L.Pereira and D.Pearce, pp. 247–260. European Workshop on Logics in Artificial Intelligence (JELIA'94), (1993).

[23] G. Brewka, Reasoning about priorities in default logic, in *Proceedings of the national conference on artificial intelligence*, pp. 940–940. John Wiley & Sons Ltd, (1994).

[24] G. Brewka and T. Eiter, Prioritizing default logic, *Intellectics and Computational Logic–Papers in Honor of Wolfgang Bibel*, 27–45, (2000).

[25] R. Brooks, A robust layered control system for a mobile robot, *IEEE journal of robotics and automation*, **2**(1), 14–23, (1985).

[26] G. Choquet, Theory of capacities, *Ann. Inst. Fourier*, **5**(131-295), 54, (1953).

[27] Alain Colmerauer, The birth of prolog, in *III, CACM Vol.33, No7*, pp. 37–52, (1993).

[28] M.O. Cordier and P.. Siegel, A temporal revision model for reasoning about world change., *Journal of intelligent systems*, **9**(1), 131–144, (1994).

[29] J.P Delahaye and M. Nivat, *Outils logiques pour l'intelligence artificielle*, Eyrolles Paris, 1986.

[30] J. Delgrande and T. Shaub, Expressing preferences in default logic, *Artificial Intelligence*, **123**(1-2), 41–87, (2000).

[31] J. Delgrande, T. Shaub, and H. Tompits, A classification and survey of preference handling approaches in non-monotonic reasoning, *Computational Intelligence*, **20**(2), 308–334, (2004).

[32] J. Delgrande, T. Shaub, and H. Tompits, A compilation of brewka and eiter's approach to prioritization, *Logics in Artificial Intelligence*, 376–390, (2009).

[33] J.P. Delgrande and T.H. Schaub, Compiling reasoning with and about preferences into default logic, in *International joint conference on artificial intelligence*, volume 15, pp. 168–175. Morgan Kaufman Publishers, (1997).

[34] C. Dirninger, N. Touraine, and S. Gautard, La discrétion Acoustique des sous-marins, *Acoustique & techniques*, **48**, 2–9, (2007).

[35] D. Dubois and H. Prade, La théorie des possibilités : applications à la représentation des connaissances en informatique, *Masson, Paris*, (1987).

[36] J. Engelfriet, *The Dynamics of Reasoning*, Ph.D. dissertation, Amsterdam : Vrije Universiteit, 1999.

[37] J. Engelfriet and J. Treur, An interpretation of default logic in minimal temporal epistemic logic, *Journal of Logic, Language and Information*, **7**(3), 369–388, (1998).

[38] J. Engelfriet and J. Treur, Branching Time Semantics for the Dynamics of Reasoning by Default, (2008).

[39] D.W. Etherington, A semantics for default logic, in *Proceedings 10th International Joint Conferences on Artificial Intelligence*, pp. 485–498. Citeseer, (1987).

[40] J. Ferber, *Les systèmes multi-agents : vers une intelligence collective*, InterEditions Paris, 1995.

[41] P.C. Fishburn. Utility theory for decision making, 1970.

[42] P.C. Fishburn, *The foundations of expected utility*, D Reidel Pub Co, 1982.

[43] F. Flacher, *Génération ascendante de coordination spatiale, vers une conception automatisée du contróle de coordination spatiale*, Ph.D. dissertation, Universitée de Paris 6, 2005.

[44] D. Gabbay, Intuitionistic basis for non-monotonic logic, in *6th Conference on Automated Deduction*, pp. 260–273. Springer, (1982).

[45] A. Galton, *Temporal logics and their applications*, Academic Press Professional, Inc. San Diego, CA, USA, 1987.

[46] A. Galton, Reified temporal theories and how to unreified them, in *Proceedings of IJCAI'91*, pp. 1177–1182. Citeseer, (1991).

[47] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, Clasp : a conflict-driven answer set solver, *Proceedings of logic programming and non-monotonic reasoning*, 260–265, (2007).

[48] M. Gelfond and V. Lifschitz, The stable model semantics for logic programming, in *In Proceedings of the 5th International Conference on Logic Programming*, pp. 1070–1080. Citeseer, (1988).

[49] AM Geoffrion, JS Dyer, and A. Feinberg, An interactive approach for multi-criterion optimization, with an application to the operation of an academic department, *Management Science*, **19**(4), 357–368, (1972).

[50] G. Gottlob, Complexity results for nonmonotonic logics, *Journal of Logic and Computation*, **2**(3), 397, (1992).

[51] M. Grabisch, The application of fuzzy integrals in multicriteria decision making, *European journal of operational research*, **89**(3), 445–456, (1996).

[52] D. Hanon, *Modèle déisionnel orienté comportement fondé sur le vote : Application à la navigation d'agents autonomes en environnement simulé*, Ph.D. dissertation, Universitée de Valenciennes et du Hainaut-Cambrésis, 2006.

[53] C. Hewitt, Viewing control structures as patterns of message passing, *Artificial Intelligence*, **8**(3), 323–374, (1977).

[54] T.R. Hostetler and J.K. Kearney, Strolling down the avenue with a few close friends, in *Third Irish Workshop on Computer graphics*, pp. 7–14. Citeseer, (2002).

[55] M.N. Huhns, Distributed Artificial Intelligence, (1987).

[56] E. Jacquet-Lagreze and J. Siskos, Assessing a set of additive utility functions for multicriteria decision-making, the UTA method, *European journal of operational research*, **10**(2), 151–164, (1982).

[57] Y. Katz and J. Golbeck, Social network-based trust in prioritized default logic, in *Proceedings of the national conference on artificial intelligence*, volume 21, p. 1345. Menlo Park, CA ; Cambridge, MA ; London ; AAAI Press ; MIT Press ; 1999, (2006).

[58] R.L. Keeney and H. Raiffa, *Decisions with multiple objectives : Preferences and value tradeoffs*, John Wiley & Sons, Inc., 1976.

[59] S. Konieczny, *Sur la logique du changement : révision et fusion de bases de connaissances*, Ph.D. dissertation, Universitée des Sciences et technologies de Lille, 1999.

[60] R. Kowalski and M. Sergot, A logic-based calculus of events, *New generation computing*, **4**(1), 67–95, (1986).

[61] S. Kripke, Semantical considerations on modal logic, *Acta Philosophica Fennica*, **16**, 83–94, (1963).

[62] S. Labidi and W. Lejouad, De l'intelligence artificielle distribuÃľe aux systÃĺmes multi-agents, *Rapport de recherche*, (1993).

[63] M. Lemaître, Déision multisource, notes de cours, (2007).

[64] F. Lin and Y. Shoham, A logic of knowledge and justified assumtions, *Artificial Intelligence*, **57**(2–3), 271–289, (1992).

[65] F. Lin and Y. Zhao, Assat : Computing answer sets of a logic program by sat solvers, *Artificial Intelligence*, **157**(1–2), 115–137, (2004).

[66] P. Maes, The dynamics of action selection, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 2, pp. 991–997. Citeseer, (1989).

[67] W. Marek, A. Nerode, and J. Remmel, A theory of nonmonotonic rule systems I, *Annals of Mathematics and Artificial Intelligence*, **1**(1), 241–273, (1990).

[68] W. Marek, A. Nerode, and J. Remmel, A theory of nonmonotonic rule systems II, *Annals of Mathematics and Artificial Intelligence*, **5**(2), 229–263, (1992).

[69] W. Marek, G.F. Schwartz, and M. Truszczynski, Modal nonmontonic logics : Ranges, characterization, computation, *Journal of the ACM*, **40**(4), 961–988, (1993).

[70] M.J. Mataric, *Interaction and intelligent behavior*, Ph.D. dissertation, Massachusetts institute of technology, 1994.

[71] L.Y. Maystre, J. Pictet, and J. Simos, *Méthodes multicritères ELECTRE : description, conseils pratiques et cas d'application à la gestion environnementale*, Presses polytechniques et universitaires romandes, 1994.

[72] J. McCarthy, Circumscription - a form of non-monotonic reasoning, *Artificial intelligence*, **13**(1-2), 27–39, (1980).

[73] D. McDermott, A temporal logic for reasoing about processus and plans, *Cognitive science*, **6**, 101–155, (1982).

[74] D. McDermott and J. Doyle, Non-montonic logic i., *Artificial intelligence*, **13**, 41–72, (1980).

[75] M. Minsky, Steps toward artificial intelligence, *Proceedings of the IRE*, **49**(1), 8–30, (1961).

[76] M. Minsky, A framework for representing knowledge, *AIM-306*, (1974).

[77] R.C. Moore, Semantical considerations on nonmonotonic logic, *Artificial Intelligence*, **25**(1), 75–94, (1985).

[78] G. Moreau, *Modélisation du comportement pour la simulation interactive : application au trafic routier multimodal*, Ph.D. dissertation, Universitée de Rennes 1, 1998.

[79] A. Nerode, JB Remmel, and VS Subrahmanian, Annotated nonmonotonic rule systems, *Theoretical Computer Science*, **171**(1-2), 111–146, (1997).

[80] P. Nicolas, L. Garcia, and I. Stéphan, Possibilistic stable models, in *International Joint Conference on Artificial Intelligence*, volume 19, p. 248. Citeseer, (2005).

[81] A.K. Pani and G.P. Bhattacharjee, Temporal representation and reasoning in artificial intelligence : A review, *Mathematical and Computer Modelling*, **34**(1-2), 55–80, (2001).

[82] CA. Petri, Fundamentals of a theory of asynchronous information flow, in *Information processing : proceedings of the IFIP Congress*, p. 386. North-Holland Pub. Co., (1963).

[83] P. Pirjanian, Behavior coordination mechanisms-state-of-the-art, *Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, Tech. Rep. IRIS-99-375*, (1999).

[84] P.Nicolas, L.Garcia, I.Stephan, and C.Lefèvre, Traitement de l'incertitude pour la programmation par ensemble réponses., in *RFIA, Tours, France*, (2006).

[85] A. Prior, Diodoram modalities, *Philosophical Quarterly*, **5**, 205–213, (1955).

[86] Jr. Prouty, *Displaying uncertainty : a comparison between submarine subject matter experts*, Ph.D. dissertation, Naval postgraduate school, Monterey, California, 2007.

[87] M.R Quillian, Semantic memory, *Semantic information processing*, 227–270, (1968).

[88] A. Ram, G. Boone, R. Arkin, and M. Pearce, Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation, *Adaptive Behavior*, **2**(3), 277, (1994).

[89] R. Reiter. On closed world databases. Logic and Databases, Gallaire and Minker eds, 1978.

[90] R. Reiter, A logic for default reasoning, *Artificial intelligence*, **13**(1–2), 81–132, (1980).

[91] C.W. Reynolds, Flocks, herds and schools : A distributed behavioral model, in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 25–34. ACM, (1987).

[92] J. Rintanen, On specificity in default logic, in *International joint conference on artificial intelligence*, volume 14, pp. 1474–1479. Morgan Kaufmann Publishers Inc., (1995).

[93] J. Rintanen, Lexicographic Ordering as a Basis of Priorities in Default Reasoning, (1996).

[94] J.K. Rosenblatt, *DAMN : A Distributed Architecture for Mobile Navigation*, Ph.D. dissertation, Universitée de Carnegie Mellon, Pittsburgh, 1996.

[95] M. Roubens, Preference relations on actions and criteria in multicriteria decision making, *European Journal of Operational Research*, **10**(1), 51–55, (1982).

[96] B. Roy, Critères multiples et modélisation des préféences : l'apport des relations de surclassement, *Revue d' Économie Politique*, (1974).

[97] B. Roy, Interactions et compromis : la procédure du point de mire, *Cah. Bel. Rech.Opér*, (1975).

[98] B. Roy, Vers une méthodologie générale d'aide à la décision, *Revue METRA*, **14**(3), 459–497, (1975).

[99] B. Roy, *Méthodologie multicritère d'aide à la décision*, Editions Economica, 1985.

[100] B. Roy, *Multicriteria Methodology for Decision Aiding*, Springer, 1996.

[101] B. Roy and B. Bertier, La Methode ELECTRE II : Une Méthode de Classement en Presence de Criteres Multiples, Note de Travail No. 142, *Direction Scientifique*, (1971).

[102] T.L. Saaty. The analytical hierarchy process, 1980.

[103] A. Scharlig, Décider sur plusieurs critères, *Panorama de l aide a la décision multicritère*, (1985).

[104] Y. Shoham, *Reasoning about change*, MIT Press Cambridge MA, 1988.

[105] E.H. Shortliffe, *Computer-based medical consultations, MYCIN*, Elsevier Publishing Company, 1976.

[106] P. Simons, I. Niemelá, and T. Soininen, Extending and implementing the stable model semantics, *Artificial Intelligence*, **138**(1-2), 181–234, (2002).

[107] L. Sombé, *Raisonnement sur des informations incomplètes en intelligence artificielle*, Teknea, 1989.

[108] VS Subrahmanian, On the semantics of quantitative logic programs, in *Proc. 4th IEEE Symp. on Logic Programming*, pp. 173–182, (1987).

[109] R. Sukthankar, *Situation awareness for tactical driving*, Ph.D. dissertation, The Robotics Institute, Carnegie Mellon University, Pittsburgh, 1997.

[110] A. Thayse, P. Gribomon, G. Louis, D. Snyers, P. Wodon, P. Gochet, E. Gregoire, E. Sanchez, and P. Delsarte, *Approche logique de l'intelligence artificielle*, Dunot, 1991.

[111] I. Toulgoat, Modélisation du processus de décision d'un opérateur dans les simulations de combat naval, in *Journées d'Intelligence Artificielle Fondamentales, Session doctorants,Marseille*, (2009).

[112] I. Toulgoat, J. Botto, Y. De lassus, and C Audoly, Modeling operator decision in underwater warfare performance simulations, in *Conference UDT, Cannes.*, (2009).

[113] I. Toulgoat, P. Siegel, and Y. Lacroix, Operator behavior modlling in a submarine, *CIMA 2010*, (2010).

[114] I. Toulgoat, P. Siegel, and Y. Lacroix, Simulation du comportement d'un opérateur en situation de combat naval, *JFPC'10*, (2010).

[115] I. Toulgoat, P. Siegel, Y. Lacroix, and J. Botto, Operator decision in naval action's simulations, in *13th International Workshop on Non-Monotonic Reasoning*, (2010).

[116] I. Toulgoat, P. Siegel, Y. Lacroix, and J. Botto, Operator decision modeling in a submarine, in *9th International Conference on Computer and IT Applications in the Maritime Industries*, pp. 65–75, (2010).

[117] A.M. Turing, Computing Machinery and intelligence, *Mind*, **59**(236), 433–460, (1950).

[118] T. Tyrrell, *Computational Mechanisms for Action Seletion*, Ph.D. dissertation, University of Edinburg, 1993.

[119] L. Vila, A survey on temporal reasoning in artificial intelligence, *AI Communications*, **7**(1), 4–28, (1994).

[120] P. Vincke, *L'aide multicritère à la décision*, Ellipses, 1989.

[121] R.R. Yager, On ordered weigthed averaging aggregation operators in multicriteria decision making, *IEEE transactions on Systems, Man and Cybernetics*, **18**(1), 183–190, (1988).

[122] L.A Zadeh, Fuzzy sets, *Information and control*, **8**(3), 338–353, (1965).

[123] L.A Zadeh, Fuzzy sets as a basis for the theory of possibility, *Fuzzy Sets and Systems*, **1**(1), 3–28, (1978).

Résumé :

Cette thèse porte sur la modélisation du comportement humain dans les simulations de combat naval. Au sein de l'entreprise DCNS, les simulations de combat naval permettent d'évaluer les performances opérationnelles des navires militaires, dans un scénario donné. Les simulations actuelles ne permettent pas de prendre en compte l'analyse et la décision d'un opérateur, qui peuvent parfois conduire à des réactions inattendues. Le but de cette thèse est donc de modéliser le comportement d'un opérateur pour les simulations de combats navals.

Pour représenter les connaissances, la logique non monotone la plus employée a été utilisée : la logique des défauts. Une prise en compte du temps a été ajoutée à cette logique des défauts. La logique des défauts va permettre de calculer des extensions. Chaque extension correspond à une action possible pour l'opérateur.

Une méthode qui permet de choisir une extension a été définie. Cette méthode simule la décision de l'opérateur et elle prend en compte le caractère de l'opérateur.
Mots-clés : Intelligence artificielle - Modélisation du comportement - Simulation de combat naval - Représentation des connaissances - Logique non monotone - Logique des défauts - Préférences - Aide multicritère à la décision

Abstract :

This thesis deals with the modelling of operator behavior in naval action simulations. At DCNS, simulations of naval action estimate the operational performance of warships or submarines for a given scenario. However, the current simulations do not take into account the analysis and the decision of an operator, which can produce unexpected reactions. The purpose of this thesis is to develop a system allowing to model the behavior of an operator in the naval action simulations.

To represent knowledge, the most widely known non-monotonic logic is used : the default logic. A consideration of time is added to this logic. The default logic allows to calculate extensions. Each extension represents a possible action for the operator.

A method to choose an extension is defined, which allows to simulate the decision of the operator and it handles the operator character.
Keywords : Artificial Intelligence - Behavior Modelling - Naval action's simulation - Knowledge representation - Non-monotonic logic - Default logic - Preferences - Multicriteria decision aid