View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Savoirs UdeS

RECONNAISSANCE DES BUTS D'UN AGENT À PARTIR D'UNE OBSERVATION PARTIELLE DE SES ACTIONS ET DES CONNAISSANCES STRATÉGIQUES DE SON ESPACE DE DÉCISION

par

Thierry Christian Kuate Kengne

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 27 mars 2017

Le 27 mars 2017

le jury a accepté le mémoire de Monsieur Thierry Christian Kuate Kengne dans sa version finale.

Membres du jury

Professeur Froduald Kabanza
Directeur
Département d'informatique

Professeur Richard St-Denis
Codirecteur
Département d'informatique

Professeur Gabriel Girard
Évaluateur interne
Département d'informatique

Professeure Hélène Pigot
Présidente rapporteuse
Département d'informatique

Sommaire

La capacité de reconnaître les intentions des autres est une composante essentielle non seulement de l'intelligence humaine mais aussi de l'intelligence artificielle dans plusieurs domaines d'application. Pour les algorithmes d'intelligence artificielle, reconnaître l'intention d'un agent à partir d'une observation partielle de ses actions demeure un défi de taille. Par exemple dans les jeux de stratégie en temps réel, on aimerait reconnaître les intentions de son adversaire afin de mieux contrer ses actions futures. En domotique, on voudrait une maison capable de comprendre et d'anticiper les intentions de ses habitants pour maximiser leur confort et les assister dans leurs activités quotidiennes. Dans le domaine de la sécurité informatique, un outil de détection des intrus doit pouvoir observer les actions des usagers du réseau et déceler ceux qui ont des intentions malveillantes. Ce mémoire de maîtrise propose d'aborder ce problème sous observabilité partielle par adaptation des méthodes utilisées dans l'analyse grammaticale probabiliste. L'approche probabiliste considérée utilise une grammaire hors contexte de multi-ensemble partiellement ordonnée et considère la poursuite de plusieurs buts simultanément, ordonnés ou non. Cela revient donc à faire de l'analyse grammaticale probabiliste avec plusieurs symboles de départ.

Mots-clés: intelligence artificielle ; reconnaissance de plan ; observation partielle ; raisonnement probabiliste

Remerciements

Je tiens à remercier les professeurs Froduald Kabanza et Richard St-Denis d'avoir accepté de diriger les travaux effectués au cours de ma maîtrise. Leurs suggestions et conseils m'ont permis de bien cerner les lignes directrices de ma recherche et de réorienter mes travaux lorsque ceux-ci s'en éloignaient. Je voudrais également remercier Francis Bisson, mon collègue du laboratoire PLANIART pour l'assistance inconditionnelle qu'il m'a accordée. J'espère avoir été en mesure de l'assister similairement dans son propre cheminement.

Je tiens à remercier mon papa, ma maman, mes frères et ma famille toute entière pour leurs sacrifices et encouragements. Ce travail est le fruit de votre amour, de votre éducation, de vos bénédictions et de vos prières.

J'adresse ma vive gratitude à l'Université de Sherbrooke et à la Faculté des sciences en particulier.

Abréviations

DOPLAR Decision-Oriented Plan Recognizer

GGPO Geib and Goldman's algorithm for Partial Observation

HTN Hierarchical Task Networks (Réseau de tâches hiérarchiques)

IA Intelligence artificielle

MDP Markov Decision Process (Processus de décision markovien)

PHATT Probabilistic Hostile Agent Task Tracker

POMDP Partially Observable MDP (MDP partiellement observable)

PROBE Provocation for the Recognition of Opponent BEhaviours

RB-OP Reconnaissance de buts avec observabilité partielle

SHOP2 Simple Hierarchical Ordered Planner 2

STR Stragégie en temps réel

YAPPR Yet Another Probabilistic Plan Recognizer

Table des matières

Liste des figures

Liste des tableaux

Introduction

Mise en contexte

L'intelligence artificielle est une discipline de l'informatique dont certains objectifs sont la création ou la simulation de processus cognitifs similaires à ceux de l'humain. Dans la conception d'entités intelligentes, la définition de la notion d'intelligence donne naissance à deux courants de pensée. Pour certains, un système intelligent est un système devant agir ou penser comme un humain (comportement vérifiable par le test de Turing, par exemple), alors que d'autres parlent plutôt d'un système devant agir ou penser rationnellement, c'est-à-dire obtenir les meilleurs résultats ou arriver aux meilleures conclusions avec l'information disponible.

La compréhension des intentions d'un agent à partir d'observations s'appelle la *reconnaissance de plan*. Le problème de reconnaissance de plan fut décrit pour la première fois en 1978 par Schmidt et al. [29]. Sur le plan conceptuel, la reconnaissance de plan est le problème inverse de la planification. En planification, le raisonnement se fait à partir d'un but pour trouver les actions permettant de l'atteindre, alors qu'en reconnaissance de plan, le raisonnement se fait à partir de l'observation d'actions pour inférer le but poursuivi. Tout comme les humains, ces approches considèrent que l'agent peut poursuivre un ou plusieurs buts simultanément, chaque but pouvant avoir des actions entrelacées ou pas.

Diverses applications de l'intelligence artificielle sont associées au domaine de la reconnaissance de plan. Par exemple dans les jeux vidéo, connaître les intentions de son adversaire permet à une intelligence artificielle de contrer ses actions futures [3, 30]. En robotique, un robot autonome désire comprendre et anticiper les intentions des agents avec qui il interagit, que ce soit des humains ou d'autres robots [12]. En do-

motique, on souhaite une maison capable de comprendre et d'anticiper les intentions de ses habitants pour maximiser leur confort et les assister dans leurs activités quotidiennes. Il s'agit en quelque sorte d'un habitat robotisé [22] et de tels concepts sont motivés par des besoins divers, par exemple l'assistance aux personnes avec des capacités cognitives réduites ou la réduction de la charge de travail pour des habitats complexes tels qu'une station spatiale. Dans le domaine de la sécurité informatique, un outil de détection des intrus doit pouvoir observer les actions des usagers du réseau et déceler ceux qui ont des intentions malveillantes [17, 20]. Une interface personne-machine doit avoir la capacité de reconnaître les intentions des usagers et d'anticiper leurs actions [14, 25].

Cependant, dans plusieurs applications, il est difficile de se trouver dans un environnement parfaitement observable. L'observabilité parfaite correspond à la situation où l'observateur voit tout avec précision. Il observe chacune des actions de l'agent sans aucun bruit et avec certitude. Ce type d'environnement est peu fréquent, mais il est néanmoins souvent utilisé dans des prototypes de recherche pour simplifier le problème.

L'observabilité partielle, qui prévaut dans la plupart des applications intéressantes, se définit de deux façons. D'une part, certaines zones de l'environnement peuvent ne pas être observées. Cela signifie qu'il est possible que l'on ne perçoive pas l'ensemble des actions de l'agent observé (interdiction d'accès à l'information). D'autre part, il est possible que certains capteurs de l'observateur soient imparfaits. Une partie de la séquence d'actions peut avoir été omise par l'un des capteurs (effacement d'action), un des capteurs peut avoir détecté une fausse action (insertion d'action non réalisée) ou alors un capteur défectueux peut avoir inversé la séquence des actions qu'il a détectée. Tout cela introduit du bruit et les observations manquantes ou fausses seront mal interprétées.

Les concepts de plan, but et intentions sont très proches, mais avec des nuances importantes [10]. Le désir correspond en quelque sorte à un but que l'on veut réaliser. L'intention par contre, est un choix d'actions qu'on s'est engagé à accomplir afin de combler son désir. L'intention est en quelque sorte une implémentation du désir et il représente à la fois un plan d'actions et un engagement dans l'exécution du plan. Ainsi, un agent peut avoir des désirs incohérents, mais il ne devrait pas avoir des

intentions incohérentes. Malgré ces nuances bien comprises entre ces concepts, pour simplifier l'exposé, il est fréquent dans la littérature de l'intelligence artificielle de les voir utilisés de façon interchangeable lorsqu'on aborde le problème de reconnaissance plan. Ce mémoire ne fait pas non plus de distinction terminologique entre ces concepts, bien qu'il porte en fait sur la reconnaissance des buts d'un agent.

Définition du problème

Une des approches souvent utilisée pour reconnaître les intentions d'un agent suppose une connaissance préalable et exhaustive des comportements possibles de l'agent observé. Ces comportements sont souvent représentés par une bibliothèque de plans potentiels que l'agent observé pourrait suivre et les buts correspondants. Étant donné une bibliothèque de plans et une séquence d'actions observées, l'objectif est d'obtenir une probabilité pour chaque but que l'agent observé tente de réaliser.

Afin de donner une formulation plus formelle, définissons $\mathcal{L}(P,B)$ comme étant la bibliothèque qui contient les plans $p_1,\ldots,p_n \in P$ associés aux buts $b_1,\ldots,b_n \in B$ que l'agent peut vouloir atteindre. À ce stade, il n'est pas important d'élaborer sur le formalisme mathématique précis pour décrire les plans et les buts. Étant donné une séquence d'actions $A = [a_1,\ldots,a_m]$, définissons aussi la fonction de masquage $\mathcal{O}$ qui donne une sous-séquence $O = [a_{i_1},\ldots,a_{i_l}]$, $1 \leq i_j \leq m$ et $1 \leq j \leq l \leq m$, obtenue en masquant certaines actions de A. Ainsi, $O = \mathcal{O}(A)$ décrit la séquence d'actions réellement observée lorsque A est exécutée. Le problème de la reconnaissance de buts pour lequel nous voulons définir un algorithme est formulé comme suit :

> Étant donné une bibliothèque de plans $\mathcal{L}(P,B)$, un sous-ensemble de buts $B' \subset B$, et une séquence d'actions observées O, on veut calculer la probabilité de chaque but $b \in B'$.

Il faut noter que dans la définition du but, l'ensemble d'actions réellement exécutées A, tel que $O = \mathcal{O}(A)$ demeure caché ou sous-entendu. Pour résoudre le problème, il faudra donc d'une façon ou d'une autre formuler (directement ou indirectement) des hypothèses sur A.

La formulation précédente du problème de reconnaissance de plan est en fait nettement plus restrictive que la formulation plus générale dans la littérature. La

formulation générale ne suppose pas que l'on dispose au préalable d'une bibliothèque de plans. Notre formulation du problème correspond directement à une famille d'approches de solutions parmi lesquelles celle abordée dans ce mémoire. Il s'agit des approches dite de reconnaissance de plan en passant par des bibliothèques de plans (ou en anglais, *plan-library-based plan recognition*) [2].

Différentes approches

Les approches de reconnaissance de plan présentes dans la littérature reflètent différentes approches mathématiques pour formaliser le problème, par exemple sous la forme d'inférence Bayesienne, d'analyse grammaticale ou de planification [2].

Indépendamment du cadre mathématique, différentes approches nécessitent un certain niveau de connaissances *a priori* sur les actions potentielles, voire les plans potentiels de l'agent [2].

Comme mentionné plus haut, l'approche étudiée dans ce mémoire fait partie de celles qui nécessitent la connaissance a priori des plans potentiels (et par conséquent aussi des actions potentielles) de l'agent. Plus spécifiquement, nous nous intéressons à un algorithme de Geib et Goldman [16] basé sur l'analyse grammaticale.

Dans cette approche, la bibliothèque de plans est spécifiée comme une grammaire hors-contexte probabiliste dont le langage généré représente l'ensemble des plans potentiels, avec des actions partiellement ordonnées. L'idée derrière cette grammaire est d'interpréter les règles de production comme des règles exécutables par l'agent, décrivant des recettes qu'il suivrait pour atteindre un but. On peut alors associer une distribution de probabilité aux règles de production. Cette distribution codifie alors le modèle d'exécution de l'agent, c'est-à-dire comment il fait ses choix dans l'exécution des règles.

Pour inférer une distribution de probabilité sur les buts poursuivis par l'agent, l'algorithme de Geib et Goldman procède d'abord par une génération de l'ensemble des hypothèses sur les modèles d'exécution cohérents avec la séquence d'observations. Une fois cet ensemble d'hypothèses calculé, la probabilité que l'agent poursuive un but particulier est alors simplement obtenue comme étant la proportion du nombre de modèles générés cohérents avec le but dans l'ensemble de tous les modèles cohérents

avec la séquence d'observations.

Objectif du mémoire

Geib et Goldman ont donné une description abstraite de leur algorithme dont ils ont fait un exposé théorique sommaire. Ils n'en ont pas fait d'implémentation et ont reporté des résultats empiriques. Ils ont aussi souligné que l'algorithme est d'une grande complexité théorique.

L'objectif de ce mémoire est de faire une évaluation empirique de l'algorithme, en évaluant son comportement sur des scénarios concrets. Cela passe par une description plus détaillée de l'algorithme, son implémentation, et son expérimentation sur des scénarios selon des métriques bien choisies.

Il faut noter que Geib et Goldman ont introduit un autre algorithme de reconnaissance de plan, appelé PHATT, cette fois-ci pour un environnement complétement observable [17]. Contrairement à leur algorithme pour observabilité partielle qui nous intéresse, ils ont décrit en détail PHATT et l'ont implémenté. Étant donné que leur version pour observabilité partielle est en fait une tentative de généralisation de ce PHATT, nous allons partir de PHATT pour notre implémentation.

Geib et Goldman n'ont pas donné de nom à leur version pour observabilité partielle. Pour faciliter l'exposé, dans ce mémoire nous allons nous y référer comme étant l'algorithme GGPO (*Geib and Goldman's algorithm for Partial Observation*).

Méthodologie

Pour évaluer l'algorithme GGPO, nous sommes passés par quatre étapes. Premièrement, nous en avons fait une description en pseudocode, plus détaillée que la description originale très succincte, purement textuelle faite par les auteurs. Ensuite, nous avons implémenté l'algorithme selon une approche orientée objet, ce qui implique le choix d'un langage, des structures de données et des patrons de conception. Troisièmement, nous avons mis en place un cadre de comparaison, incluant la définition des métriques et des scénarios, ainsi que des outils pour générer des jeux de données

à partir des scénarios, collecter et afficher les résultats de comparaison. Finalement, nous avons analysé les résultats.

Organisation du mémoire

Le présent mémoire est organisé en quatre chapitres en plus de la présente introduction. Le chapitre 1 présente les principales approches au problème tel que posé plus haut dans cette introduction. Le chapitre 2 présente la description détaillée de l'algorithme GGPO. Le chapitre 3 décrit l'implémentation et l'évaluation de l'algorithme. Le dernier chapitre est une conclusion.

Chapitre 1

État de l'art

Le problème de reconnaissance de plan à partir d'une bibliothèque de plans est formulé de différentes façons dans la littérature. Les approches purement **logiques** (symboliques) infèrent des hypothèses sans quantifier leur degré de vraisemblance. Les approches **probabilistes** produisent une distribution de probabilités sur les différentes hypothèses. On distingue aussi les approches **hybrides** qui visent à combiner les avantages des deux précédentes familles d'approches.

Peu importe l'approche, la supposition que l'on doive pouvoir décrire *a priori* une bibliothèque de plans est une limite fondamentale. Pour bien des domaines d'application, il n'est pas réaliste de supposer que les agents observés agissent toujours en suivant scrupuleusement un plan bien défini en avance, pas plus que supposer que l'on pourra décrire *a priori* de façon exhaustive et explicite tous les plans possibles. Par conséquent, pour bien de domaines d'application, la reconnaissance de plan par bibliothèque de plans présente des taux de reconnaissance corrects qui sont faibles.

Ce mémoire ne s'attaque pas à cette faiblesse. Comme précisé plus haut, le mémoire se limite à évaluer une des approches parmi tant d'autres. La revue de littérature très succincte ci-après permet de situer l'approche évaluée dans l'ensemble de l'état de l'art.

1.1 Approches logiques pures

Les approches logiques pures (symboliques) furent parmi les premières approches mises de l'avant pour faire de la reconnaissance de plan. Certaines d'entre elles sont abordées ici.

Kautz et Allen [23] ont abordé le problème de reconnaissance de plan comme un problème d'identification d'un ensemble minimal d'actions de haut niveau dans une taxonomie d'actions et de sous-actions prédéfinie (voir Figure 1.1) pour expliquer un ensemble d'actions observées. Le problème revient donc à calculer la couverture minimale du graphe de la bibliothèque de plans (la taxonomie d'actions) correspondant aux observations.

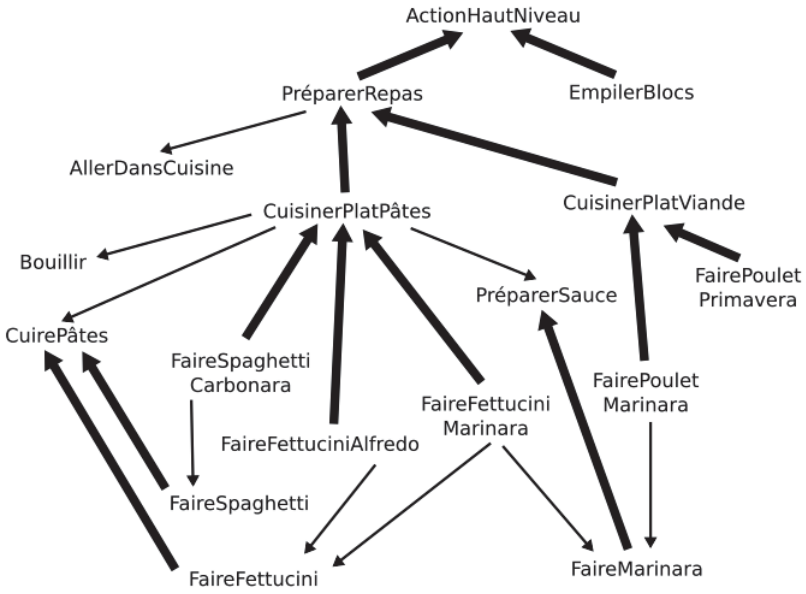


Figure 1.1 – *Taxonomie d'actions de Kautz et Allen [23]*

Avrahami-Zilberbrand et Kaminka [1] ont proposé un algorithme de reconnaissance de plan utilisant une bibliothèque de plans qui détaille de manière hiérarchique les plans possibles et les actions les composant. Cette approche tient compte des observations formées à partir de plusieurs caractéristiques, pas seulement celles qui correspondent à des activités plus ou moins abstraites.

1.2 Approches probabilistes pures

Les approches probabilistes permettent de donner une distribution de probabilité aux plans reconnus, ce que les approches logiques pures ne peuvent pas. En contre partie, les approches logiques permettent de décrire des comportements complexes plus facilement, étant donné le pouvoir d'expression de la logique du premier ordre plus grand que le langage des probabilités purement propositionnel.

Charniak et Goldman ont été les premiers à proposer une approche probabiliste pour la reconnaissance de plan [9]. Leur algorithme construit dynamiquement un réseau bayésien à partir des observations et d'une base de connaissances de faits sur le monde, spécifiés en logique du premier ordre. Dans le réseau bayésien, les nœuds racines correspondent aux hypothèses sur les buts de l'agent observé. Les probabilités conditionnelles de ces nœuds sont ensuite calculées (grâce au théorème de Bayes) afin de pouvoir comparer leur vraisemblance.

Pynadath et Wellman [27] ont proposé une architecture générale (réseau bayésien illustré à la Figure 1.2) pour la reconnaissance de plan, permettant cette fois-ci de modéliser, pas seulement les plans, mais la modélisation explicite des croyances de l'agent observé, ce qui permet de refléter son état mental et son impact sur ses prises de décision. Chaque composante de cette architecture est implémentée par un réseau bayésien dynamique permettant de capturer leur évolution dans le temps. L'architecture permet d'ajuster les liens entres les actions modélisées, de sorte à tenir compte de la nature du comportement d'un agent, qu'il soit amical, neutre ou même hostile ou trompeur.

Les calculs requis pour mettre à jour les réseaux bayésiens sont aussi généralement très coûteux, ces calculs sont fortement liés à la complexité des réseaux. Tout ceci rend difficile l'utilisation de ces approches dans un domaine complexe comme les jeux de stratégie en temps réel.

1.3 Approches hybrides

Les approches symboliques et probabilistes ayant chacune des limitations, plusieurs travaux récents s'appuient sur des approches hybrides, combinant ainsi les

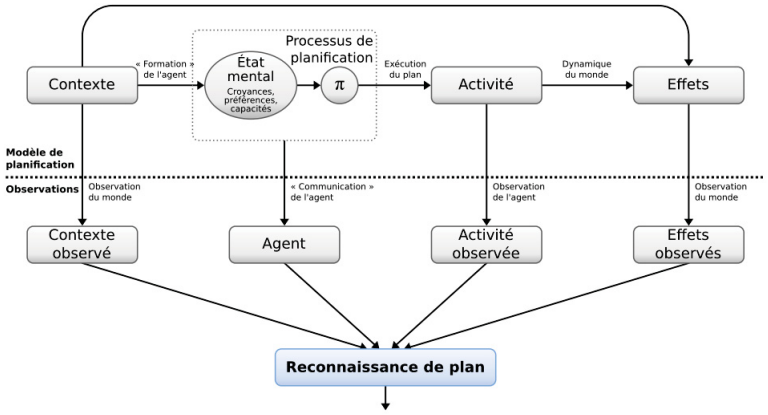


Figure 1.2 – *Architecture de reconnaissance de plan de Pynadath et Wellman [27]*

avantages des méthodes symboliques et probabilistes. Ici, nous voyons les approches qui utilisent une représentation hiérarchique des buts possibles de l'agent, appuyée par un modèle probabiliste.

Geib et al. [19, 18, 15, 17] ont développé une approche probabiliste qui utilise une grammaire hors contexte généralisée pour représenter la bibliothèque de plans. Ainsi, ils ont adapté des méthodes d'analyse grammaticale pour reconnaître automatiquement le plan d'un agent. Plus spécifiquement, l'analyse grammaticale est utilisée pour générer des hypothèses sur les plans poursuivis par un agent étant donné une séquence d'observations de ses actions. Une étape ultérieure se base sur l'ensemble d'hypothèses générées pour calculer une distribution de probabilités sur les buts poursuivis par l'agent.

Filion et al. ont proposé une amélioration de cette approche en introduisant des heuristiques rendant la génération d'hypothèses combinée avec l'inférence de la distribution des probabilités des buts plus efficace [13, 21].

Ces approches supposent une observation complète. Toutefois, dans bien de domaines, il n'est pas toujours possible d'avoir une séquence d'observations complète. Ainsi, Geib et Goldman [16] ont introduit un algorithme, GGPO, qui est en fait une tentative de généralisation de PHATT à l'observabilité partielle. Dans ce mémoire, nous évaluons l'algorithme GGPO en passant par une description plus détaillée de l'algorithme, son implémentation et son expérimentation sur des scénarios concrets.

Chapitre 2

Description de l'algorithme GGPO

Dans les approches de reconnaissance de plan utilisant des bibliothèques de plans, une manière de représenter la bibliothèque est d'utiliser une grammaire hors-contexte dont les terminaux représentent les actions et les règles de production représentent des recettes pour générer des plans. Les symboles dans la partie droite des règles de production sont partiellement ordonnés afin de modéliser des contraintes d'ordonnancement dans les plans. Afin de suivre la description de l'algorithme GGPO, nous décrivons le formalisme de représentation de la biliothèque de plans. Cette représentation n'est pas unique à l'approche décrite dans ce mémoire, elle est en fait adoptée de Filion [13] et de Geib et Goldman [17, 16].

L'algorithme prend en entrée une séquence d'observations, une bibliothèque de plans et des buts poursuivis sous forme d'une grammaire hors-contexte probabiliste et un modèle probabiliste d'exécution de cette bibliothèque, c'est-à-dire un modèle probabiliste pour la sélection de buts, de sous-buts et d'actions. En sortie, l'algorithme retourne des hypothèses sur les buts poursuivis par l'agent et une distribution de probabilité sur ces buts. Cette distribution est obtenue en inversant le modèle d'exécution, c'est-à-dire qu'à partir de la séquence d'observations et d'un ensemble d'hypothèses générées, l'algorithme calcule les probabilités *a posteriori* de chaque but de la bibliothèque.

Pour expliquer cet algorithme en détail, la section 2.1 décrit le formalisme de

représentation et la section 2.2 présente comment les hypothèses sont générées. La section 2.3 détaille comment obtenir la probabilité *a posteriori* d'un but. Pour terminer, la section 2.4 décrit le pseudocode de l'algorithme et indique brièvement la complexité de calcul.

2.1 Modélisation d'une bibliothèque de plans

2.1.1 Rappel d'une grammaire hors-contexte

Afin d'apprécier la différence entre une grammaire hors-contexte et une librairie de plans, il convient de rappeler la syntaxe des grammaires hors-contextes.

Définition 2.1.1 *Une grammaire hors contexte est un quadruple* $\langle \Sigma, \mathcal{V}, \mathcal{P}, \mathcal{S} \rangle$ *où* Σ *(l'alphabet) est un ensemble fini de symboles terminaux,* $\mathcal{V}$ *est un ensemble fini de variables (symboles non terminaux),* $\mathcal{P}$ *est un ensemble fini de règles de production et* S *est un élément de* $\mathcal{V}$*, le symbole de départ. Les ensembles* $\mathcal{V}$ *et* Σ *sont disjoints. Les règles de production* $\mathcal{R}$ *sont de la forme* $V \rightarrow w$*, où* $V \in \mathcal{V}$ *et* $w \in (\mathcal{V} \cup \Sigma)^*$*.*

La Figure 2.1 contient les règles de production d'une grammaire hors contexte pour un sous-ensemble d'expressions arithmétiques.

$$S \rightarrow S + S \tag{2.1}$$
$$S \rightarrow S \times S \tag{2.2}$$
$$S \rightarrow 0|1|2|3|4|5|6|7|8|9. \tag{2.3}$$

Figure 2.1 – *Règles de production d'une grammaire hors contexte*

Un mot est une séquence de terminaux. On dit qu'un mot est généré par la grammaire s'il peut être obtenu en partant du symbole de départ et en effectuant des substitutions des non terminaux par les terminaux en utilisant les règles, c'est-à-dire, en remplaçant chaque non terminal par la partie droite d'une règle de production ayant ce terminal comme partie gauche. Le langage généré par une grammaire est l'ensemble des mots généré par la grammaire. Ainsi la grammaire précédente génère

les mots qui sont des additions et des multiplications d'entiers naturels. Par exemple, $0+0$, $0*0$, $0+1$, $0*1$, ou $0*123+83*9$.

2.1.2 Bibliothèque de plans

Une bibliothèque de plans est une grammaire hors-contexte pour laquelle les terminaux représentent des actions et les mots des séquences d'actions—autrement dit, des plans. Ainsi, la librairie de plans est l'ensemble de plans générés par la grammaire. Dans ce cadre, les non-terminaux représentent des buts et le symbole de départ est un but principal (les autres étant des sous-buts). Cependant, contrairement à une grammaire hors-contexte, une bibliothèque de plans permet plusieurs symboles de départs reflétant le fait qu'un agent peut avoir, par exemple, deux buts principaux en même temps. De plus, dans une bibliothèque de plans les parties droites des règles de production sont partiellement ordonnées, reflétant le fait que l'ordre de certaines actions dans un plan n'a pas d'importance pour accomplir un but donné.

Définition 2.1.2 *Une bibliothèque de plans est un tuple* $\langle \Sigma, \mathcal{V}, \mathcal{P}, \mathcal{G} \rangle$ *où* Σ *est un ensemble fini de symboles terminaux (actions),* V *est un ensemble fini symboles non terminaux,* $\mathcal{G} \subseteq \mathcal{V}$ *est un ensemble de symboles de départ représentant les "buts possibles", et* $\mathcal{P}$ *est un ensemble de règles de production de la forme* $V \rightarrow \alpha : \phi$*, pour* $V \in \mathcal{V}$*, et* $\phi = \{(i,j) | \alpha[i] \prec \alpha[j]\}$ *est un ensemble de contraintes d'ordonnancement ; chaque contrainte signifie que le* $i^{ème}$ *symbole de* α *(*$\alpha[i]$*) doit précéder son* $j^{ème}$ *symbole (*$\alpha[j]$*).*

Un exemple de bibliothèque de plans est donné dans la figure 2.2 en utilisant une représentation graphique des règles de production sous la forme d'un arbre ET/OU. Les nœuds ET sont identifiables par un arc reliant les arêtes et les nœuds OU ne possèdent aucun arc. Ainsi, les feuilles de l'arbre représentent des symboles terminaux de Σ (les actions) alors que les nœuds internes et les racines représentent des symboles non terminaux de $\mathcal{V}$ (les buts et les sous-buts). Les buts principaux ($\mathcal{G}$) correspondent aux racines. Une règle de production $V \rightarrow w_1 \dots w_n$ correspond au sous-arbre formé du parent V et des enfants $w_1 \dots w_n$ avec un lien ET entre les deux, alors qu'une contrainte d'ordonnancement "w_i précède w_j" est représentée par une flèche allant de

w_i à w_j. Une règle alternative $V \rightarrow V_1 \mid \ldots \mid V_n$ correspond au sous-arbre formé du parent V et des enfants $V_1, \ldots, V_n$. Cela est équivalent aux n règles $V \rightarrow V_1, \ldots, V \rightarrow V_n$.

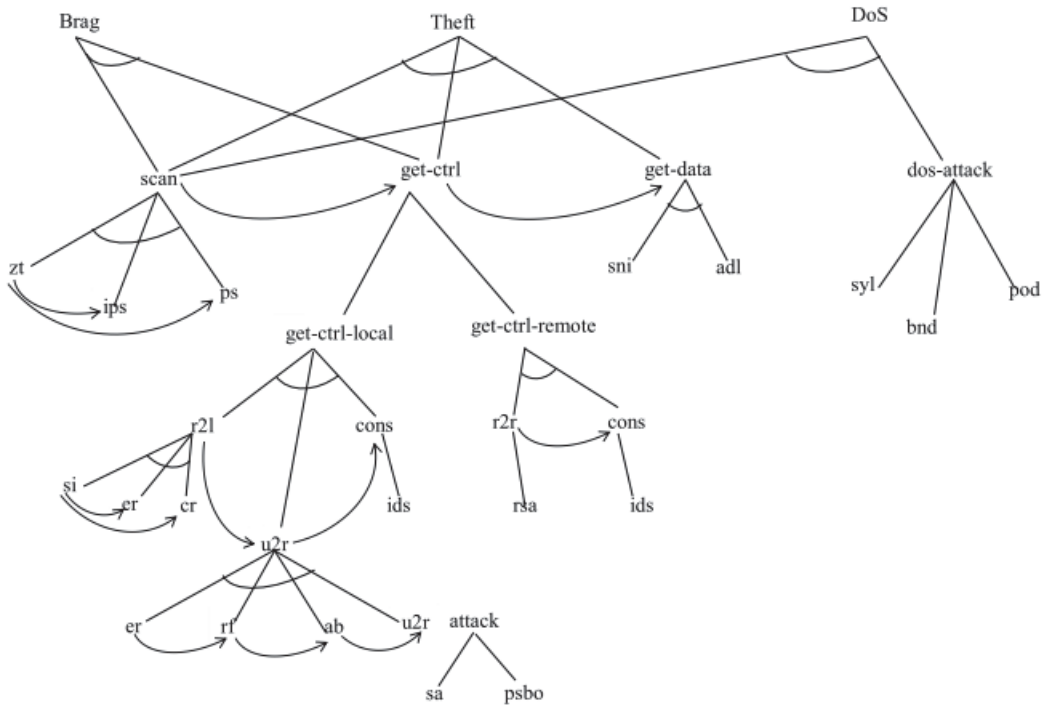


Figure 2.2 – *Un exemple de bibliothèque de plans [24]*

Cette bibliothèque décrit les plans possibles d'un agent qui pourrait tenter de faire une intrusion dans un réseau informatique. Dans ce cas, on suppose que l'intrus est motivé par l'un des trois buts **Brag** (il attaque le système pour se vanter de son succès auprès de ses pairs), **Theft** (il pénètre le système pour voler des informations) et **DoS** (il effectue une attaque par déni de service en consommant les ressources du système de sorte qu'il ne puisse plus être utilisé correctement). L'attaquant motivé par le but **Brag**, doit d'abord analyser les vulnérabilités (**scan**), ensuite il doit prendre le contrôle (**get-ctrl**). L'attaquant motivé par le but **Theft**, analyse les vulnérabilités, obtient le contrôle du système cible et enfin exploite ce contrôle pour voler les informations. L'attaque par **DoS** est réalisée en analysant les vulnérabilités et en effectuant l'attaque (**dos-attack**) proprement dite par exploitation des vulnérabili-

tés. On dispose de deux méthodes alternatives pour prendre le contrôle (**get-ctrl**) du système cible. Il est possible de le faire localement (**get-ctrl-local**) ou à distance (**get-ctrl-remote**).

2.1.3 Définitions et notations

Cette partie contient les définitions et notations utilisées dans la suite de ce mémoire.

Définition 2.1.3 *Étant donné une règle de production* $\rho = V \rightarrow \alpha : \phi$,$\alpha[i]$ *est un symbole plus à gauche de* ρ *s'il n'existe pas de* j *tel que* $(j,i) \in \phi$. *Soit* $\mathcal{L}(\rho)$, *l'ensemble des symboles plus à gauche de la règle* ρ, $\mathcal{R}(\rho)$ *désigne l'ensemble de ceux qui ne le sont pas, c'est-à-dire* $\mathcal{R}(\rho) = \{\alpha[1],\alpha[2],...,\alpha[l]\} - \mathcal{L}(\rho)$, *où* $l = |\alpha|$.

Définition 2.1.4 *Étant donné une bibliothèque* $\langle \Sigma,\mathcal{V},\mathcal{P},\mathcal{G} \rangle$ *et un symbole terminal* $\sigma \in \Sigma$, *on définit l'arbre le plus à gauche* T, *dérivant* σ, *comme un arbre tel que :*

1- chaque nœud de T *est marqué avec un symbole de* $\Sigma \cup \mathcal{V}$.

2- chaque nœud intérieur de T *est marqué avec un symbole de* $\mathcal{V}$.

3- si un nœud n *intérieur de* T, *marqué* A, *a des enfants étiquetés* $\beta_1,...,\beta_k$ *alors :*

- $\exists\ \rho \in \mathcal{P}$ *tel que* $\rho = V \rightarrow \beta_1...\beta_k : \phi$
- *le nœud* n *est aussi annoté avec* ρ.
- *aucun nœud marqué avec des symboles* $\mathcal{R}(\rho)$ *(ensemble des symboles qui ne sont pas plus à gauche), n'a des enfants.*
- *au plus un enfant (à gauche) de* n *a des enfants.*

4- il y a un nœud unique dans la frontière de T *marqué avec un symbole terminal et étiqueté* σ *: on l'appelle le pied de l'arbre* T *et on le note* ***foot(T)***.

Si T est un arbre de dérivation complet, tous les éléments dans ses feuilles sont des actions (symboles terminaux) sinon, l'arbre de dérivation T est partiel et on retrouve alors des variables parmi ses feuilles. Dans la Figure 2.3, l'arbre à gauche est un arbre de dérivation complet tandis qu'à droite on a un arbre partiel par rapport à la bibliothèque de plans de la Figure 2.2.

Définition 2.1.5 *Un ensemble d'arbres les plus à gauche est dit être générateur pour une bibliothèque de plans* $B = \langle \Sigma,\mathcal{V},\mathcal{P},\mathcal{G} \rangle$ *s'il contient tous les arbres les plus à gauche*

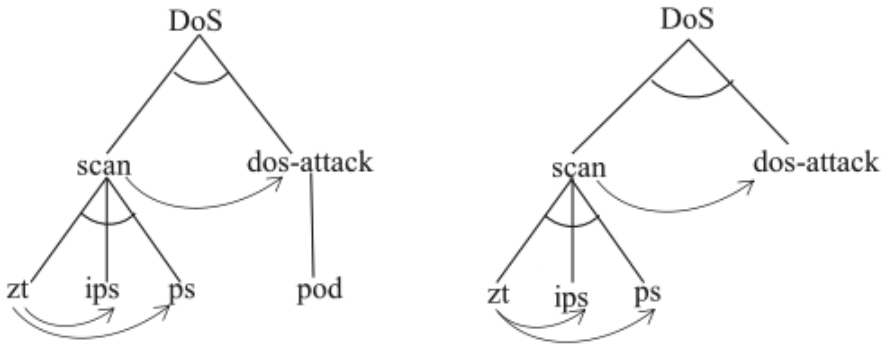


Figure 2.3 – *Arbre de dérivation complet et partiel*

dérivant une action dans Σ enracinée à une variable dans $\mathcal{G}$. L'ensemble générateur est noté $\mathcal{G}(B)$ et ses éléments sont des arbres générateurs.

Définition 2.1.6 *Étant donné un arbre T_{init} ayant un nœud non terminal m à sa frontière. Soit n le nœud parent de m, supposons que n soit étiqueté V et annoté avec la règle $V \rightarrow \beta_1...\beta_k : \phi$. Supposons que l'étiquette de m soit β_i $(1 \leq i \leq k)$ et supposons que nous avons un arbre le plus à gauche dont la racine est également marquée avec β_i, appelons le T_{β_i}. On dit que T_{β_i} peut être substitué à m dans T_{init} résultant en T_{res} seulement dans le cas où pour tout j tel que $(j,i) \in \phi$, la frontière du sous-arbre de T_{init} ancrée à l'enfant de n marqué β_j contient seulement des symboles terminaux et que T_{res} est l'arbre obtenu par substitution de β_i par T_{β_i}.*

Définition 2.1.7 *Étant donné une bibliothèque de plans B et un ensemble d'instances d'arbres de dérivation partielle $\mathcal{D}$, représentant les instances de plans de B, on définit l'ensemble de substitutions pour $\mathcal{D}$, représenté par $PS(\mathcal{D})$, comme un ensemble d'instances d'arbres inclus dans $\mathcal{G}(B)$, où chaque instance peut être substituée à un nœud dans un arbre de $\mathcal{D}$. Chaque instance d'arbre dans $PS(\mathcal{D})$ est indexée par l'arbre de $\mathcal{D}$ dans lequel il doit être substitué et par le symbole non terminal de l'arbre pour lequel il sera substitué.*

Définition 2.1.8 *Étant donné une bibliothèque de plans $B = \langle \Sigma, \mathcal{V}, \mathcal{P}, \mathcal{G} \rangle$, l'ensemble des arbres générateurs enracinés à un symbole non terminal de $\mathcal{G}$ est défini par :*

$$\mathcal{T} = \{T \in \mathcal{G}(B) | Racine(T) \in \mathcal{G}\},$$

où $Racine(T)$ est l'étiquette du nœud racine de l'arbre T.

Définition 2.1.9 *Les points d'attache en attente d'une hypothèse h (cf. définition 2.2.1) représentent les éléments de la frontière de l'hypothèse pour lesquels les actions devant les précéder ont été observées.*

Définition 2.1.10 *Les actions prêtes à être exécutées ou les actions prêtes tout court sont synonymes des points d'attache d'une hypothèse. Ce sont les actions qui sont prêtes à être exécutées (elles n'ont plus de prérequis) selon la séquence d'observations courante et compte tenu de la librairie de plans.*

Définition 2.1.11 *Pour un symbole non terminal v, on note $Fi(v)$ l'ensemble des actions pouvant être le premier élément d'une séquence d'actions accomplissant v.*

2.2 Génération d'hypothèses

Dans la génération d'hypothèses pour les actions non observées, il est important que chaque action non observée, mais réalisée par l'agent, soit cohérente avec ses buts.

Pour une séquence d'observations, des hypothèses sont générées itérativement pour l'ensemble de buts poursuivis par l'agent.

Définition 2.2.1 *Étant donné une bibliothèque de plans $B = \langle \Sigma, \mathcal{V}, \mathcal{P}, \mathcal{G} \rangle$, une hypothèse pour la séquence d'observations o est un couple $\langle \mathcal{D}, f \rangle$, où $\mathcal{D}$ est un ensemble fini d'arbres de dérivation avec contraintes d'ordonnancement défini par l'application de règles de P, de sorte que la racine de chaque arbre soit un élément de $\mathcal{G}$. Les racines désignent les buts à atteindre et les arbres de dérivation représentent les moyens de les atteindre. La fonction $f : o \rightarrow feuilles(\mathcal{D})$ associe chaque observation de o à une des feuilles des arbres de $\mathcal{D}$. Cette relation permet d'identifier les actions observées et leur ordre donné par les contraintes d'ordonnancement de B.*

Par exemple, à partir de la bibliothèque de plans de la Figure 2.2, une hypothèse possible pour la séquence d'observations **(zt,** t_1**), (ips,** t_2**), (zt,** t_3**), (ps,** t_4**), (pod,**

t_5) est donnée par la Figure 2.4. Les deux arbres de dérivation de cette figure sont des éléments de l'ensemble $\mathcal{D}$. Une flèche orientée indique une contrainte d'ordonnancement.

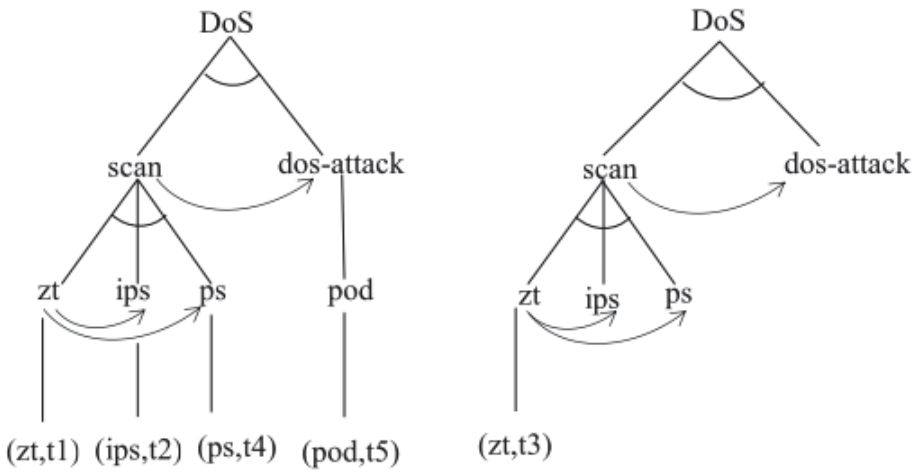


Figure 2.4 – *Une hypothèse*

Lors de la génération d'hypothèses, l'algorithme génère un ensemble d'hypothèses mutuellement exclusives permettant d'éviter ainsi la possibilité d'avoir des hypothèses redondantes. Les règles de production appliquées pour obtenir $\mathcal{D}$ doivent permettre l'ajout d'au moins une des actions observées dans chaque arbre de $\mathcal{D}$. Les règles de production qui ne sont pas nécessaires pour expliquer o à une itération donnée sont ignorées.

Pour générer un ensemble d'hypothèses, l'algorithme itère sur chaque observation, de manière à générer des arbres de dérivation compatibles avec cette observation et qui constituent donc des hypothèses pour les buts expliquant les observations. Ce faisant, on maintient une liste d'*actions prêtes* à être exécutées, c'est à dire, les actions qui seraient les prochaines à exécuter étant données la séquence d'observations et les hypothèses courantes.

L'algorithme tient aussi compte de la possibilité qu'une action pourrait être exécutée par l'agent, mais ne pas être observée. Intuitivement, cela reviendrait à prendre un élément de la liste d'actions prêtes et à l'insérer dans la séquence d'observation

si elle n'est pas observée. Ainsi, donc, pour chacune de ses possibilités, on génère des hypothèses cohérentes avec l'action non observée. Il peut toutefois en découler la génération d'un trop plein d'hypothèses. Pour éviter cela, on élimine les hypothèses en bas d'un certain seuil de probabilité, où le seuil est une valeur fixée *a priori*.

Nous verrons comment calculer la probabilité d'une hypothèse plus tard. D'abord, voyons comment on génère des hypothèses cohérentes avec l'ajout d'une nouvelle action. Le processus est le même, que ce soit dans le cas d'une action observée ou dans le cas d'une action non observée, mais ajoutée de façon hypothétique pour tenir compte qu'elle pourrait avoir été exécutée et non observée. Dans ce qui suit on suppose que les actions sont observées directement. Autrement dit, observation et action sont synonymes.

À chaque itération, un nouvel ensemble d'hypothèses est donc obtenu en intégrant une nouvelle action (observée ou hypothétique) aux hypothèses de l'itération précédente.

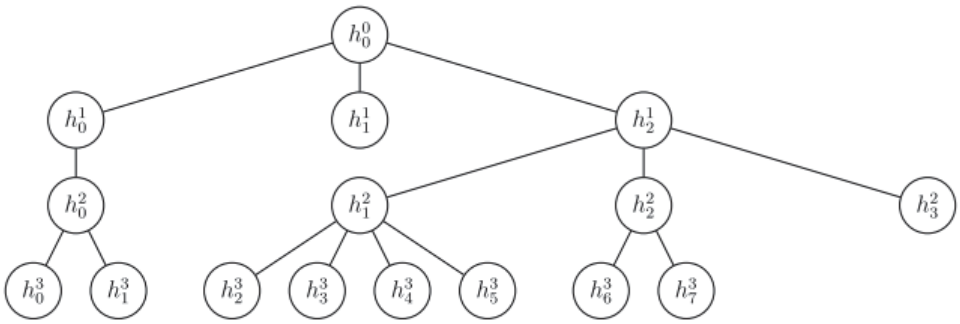
Étant donné une séquence o, notons o_i, le $i^{\text{ième}}$ élément de la séquence correspondant à la nouvelle action (observée ou hypothétique) et $\sigma_{i:j}$ la sous-séquence formée des éléments de o_i jusqu'à o_j inclusivement. Soit une hypothèse $h = \langle \mathcal{D}, f \rangle$ pour la séquence d'observations $\sigma_{1:i-1}$, le traitement fait à partir de l'action o_i permet de générer de nouvelles hypothèses expliquant la séquence $\sigma_{1:i}$, à la condition que σ_i soit compatible avec h. La génération de nouvelles hypothèses se fait en deux étapes.

Premièrement, l'observation est expliquée en l'associant à un des arbres de dérivation de $\mathcal{D}$. Une hypothèse est générée pour chaque action activée o_i. Une action activée est une action qui n'a pas encore été atteinte mais dont toutes les actions devant la précéder ont été atteintes. Il est possible qu'un des arbres doive être étendu en appliquant des règles de production afin d'obtenir une action activée. Chaque hypothèse générée est donc de la forme $\langle \mathcal{D}', f' \rangle$, où $\mathcal{D}'$ est le résultat de l'application des nouvelles règles de production, si nécessaire, à l'ensemble $\mathcal{D}$ original et f' est obtenue en étendant f pour associer o_i à l'action activée sélectionnée. Il est possible qu'il n'y ait aucune action activée correspondante à o_i, auquel cas aucune hypothèse n'est générée à cette étape.

Deuxièmement, l'observation est expliquée en ajoutant un nouvel arbre à l'hypothèse h_P. On vérifie d'abord si o_i peut être la première observation d'un but b. Une

nouvelle hypothèse est donc générée si $o_i \in Fi(b)$ (*cf.* définition 2.1.11). Pour obtenir l'ensemble d'arbres de dérivation $\mathcal{D}'$ de cette hypothèse, on ajoute à $\mathcal{D}$ l'arbre de dérivation partiel utilisé pour générer o_i.

Ce procédé de génération peut être représenté par un arbre d'hypothèses. Dans l'arbre donné à la fin de ce paragraphe, chaque nœud représente une hypothèse. Une arête entre deux hypothèses signifie que l'hypothèse enfant a été générée à partir de l'hypothèse parent. Pour une séquence d'observations o de longueur n, l'arbre est de profondeur $n+1$ et un nœud de profondeur i est une explication pour le préfixe $o_{1:i}$, par exemple o_1 à la profondeur 1, o_1o_2 à la profondeur 2 et $o_1o_2o_3$ à la profondeur 3. Les hypothèses de profondeur i représentent donc l'ensemble des hypothèses générées à la $i^{\text{ième}}$ itération de l'algorithme. Située à la profondeur 0, l'hypothèse racine h_0^0 de cet arbre représente l'hypothèse vide où l'on suppose que l'agent ne poursuit aucun but. Comme mentionné plus haut, les nœuds correspondant à des hypothèses en bas d'un certain seuil vont être coupés.



2.3 Modèle probabiliste

Cette section explore la théorie probabiliste derrière l'algorithme. La première sous-section détaille le modèle de prise de décision afin d'évaluer la vraisemblance d'une hypothèse et la deuxième sous-section détaille comment les probabilités *a posteriori* sont obtenues.

2.3.1 Probabilité d'une hypothèse

Toutes les hypothèses générées n'ont pas la même vraisemblance, certaines sont plus vraisemblables que d'autres. Le modèle de prise de décision conçu pour estimer la probabilité des hypothèses de l'algorithme suppose que l'agent observé agisse en sélectionnant un ensemble de buts à exécuter pour ensuite décomposer chaque but en sous-buts et finalement en actions (symboles terminaux) pour déterminer l'ordre d'exécution de ces actions. L'algorithme calcule donc la probabilité d'une hypothèse à partir de trois paradigmes et d'un terme qui tient compte de l'observabilité partielle. Les trois modèles de décision définis par GGPO, qui modélisent l'agent observé, sont détaillés dans les paragraphes suivants.

Le **modèle des intentions** dicte comment l'agent sélectionne les buts à atteindre. Cette sélection est modélisée par une distribution $P(\mathcal{B})$ pour un ensemble de buts $\mathcal{B}$. La probabilité *a priori* qu'un but soit poursuivi au moins une fois, notée $P(b)$, est fournie en entrée avec la bibliothèque de plans. On suppose que le nombre d'instances d'un même but suit une distribution géométrique. La probabilité de suivre exactement n fois le but b est donc $P(b)^n[1 - P(b)]$. Selon cette distribution, la sélection d'un but à atteindre est indépendante des buts précédemment choisis. L'approche étudiée permet théoriquement des modèles de probabilité plus sophistiqués supportant des dépendances entre les buts, mais cela pourrait augmenter le temps de calcul. Soit $N(b,\mathcal{B})$ le nombre d'instances de b dans l'ensemble $\mathcal{B}$, le modèle des intentions peut ainsi être exprimé par l'équation suivante :

$$P(\mathcal{B}) = \Pi_{b\in\mathcal{B}} P(b)^{N(b,\mathcal{B})}[1 - P(b)]. \tag{2.4}$$

Le **modèle de planification** dicte quelles actions sont utilisées pour atteindre les buts sélectionnés. Puisqu'on suppose que l'agent respecte la bibliothèque de plans, la sélection des actions se résume à choisir les règles de production afin de dériver un ensemble d'actions à partir de chaque but. Ce choix est modélisé par $P(\mathcal{D},\mathcal{B})$ la probabilité de dériver un ensemble de dérivation complète ou partielle, représenté par $\mathcal{D}$, à partir de l'ensemble de buts $\mathcal{B}$. On suppose que le choix d'une règle pour atteindre un symbole non terminal est indépendant des autres décisions de l'agent.

Chaque but peut ainsi conduire à des actions indépendamment des autres buts poursuivis par l'agent. Pour simplifier le calcul, on suppose aussi que toutes les options pour atteindre un symbole ont la même probabilité d'être choisies. Par exemple (*cf.* Figure 2.2), si un agent doit choisir entre utiliser **syl**, **bnd** ou **pod** pour l'action **dos-attack**, l'algorithme doit avoir une distribution de probabilités sur ces différents choix. Une distribution différente de celle de la distribution uniforme peut être utilisée à la condition que l'hypothèse d'indépendance entre les décisions soit respectée. Notons $mg(p)$ le membre gauche d'une règle de production $p \in P$ et $|v|$ [1] le nombre de règles de production ayant le symbole $v \in NT$ comme membre gauche. La probabilité de choisir une règle de production p ayant comme membre gauche v est exprimée comme suit :

$$P(p|v) = \begin{cases} \frac{1}{|v|}, \text{ si } v = mg(p); \\ 0, \text{ sinon.} \end{cases} \tag{2.5}$$

Puisque le choix d'une règle ne dépend que de son membre gauche, le modèle de planification peut être exprimé par le produit de la probabilité des règles de production utilisées pour générer les arbres de dérivation :

$$P(\mathcal{D},\mathcal{B}) = \begin{cases} \Pi_{p \in \text{règles}(\mathcal{D})} \frac{1}{|mg(p)|}, \text{ si } root(\mathcal{D}) = \mathcal{B}; \\ 0, \text{ sinon.} \end{cases} \tag{2.6}$$

Le **modèle d'entrelacement** définit dans quel ordre les actions du plan sont exécutées. La bibliothèque de plans permet une certaine liberté sur l'ordre d'exécution des actions puisque les plans sont partiellement ordonnés et qu'ils peuvent être entrelacés. La probabilité de choisir une séquence d'observations o s'écrit comme $P(o|\mathcal{D} \wedge \mathcal{B})$. Ce modèle suppose que l'agent observé choisisse chaque action o_i séparément à partir de l'ensemble des actions activées par les actions précédemment sélectionnées $o_{1:i-1}$. Notons l'ensemble des actions activées $AA(\mathcal{D},o)$. Afin de simplifier le modèle, on suppose que l'agent utilise une distribution uniforme. On obtient :

1. $|X|$ dénote la cardinalité de l'ensemble X.

$$P(o_i|\mathcal{D} \wedge o_{1:i-1}) = \begin{cases} \frac{1}{|AA(\mathcal{D},o_{1:i-1})|}, \text{ si } o_i \in AA(\mathcal{D},o_{1:i-1}); \\ 0, \text{ sinon.} \end{cases} \tag{2.7}$$

Le modèle d'entrelacement peut donc être exprimé par le produit de cette probabilité pour chaque élément de la séquence o :

$$P(o|\mathcal{D}) = \Pi_{i=1}^{|o|} P(o_i|\mathcal{D} \wedge o_{1:i-1}). \tag{2.8}$$

La probabilité *a priori* qu'une hypothèse $h = \langle \mathcal{D},f \rangle$ s'avère exacte pour une séquence d'observations peut s'écrire :

$$\begin{aligned} P(h) &= P(\mathcal{D} \wedge \phi) && (2.9) \\ &= P(\mathcal{D})P(\phi|\mathcal{D}) && (2.10) \\ &= P(root(\mathcal{D}))P(\mathcal{D}|root(\mathcal{D}))P(\phi|\mathcal{D}). && (2.11) \end{aligned}$$

La probabilité *a priori* qu'une hypothèse $h = \langle \mathcal{D},f \rangle$ s'avère exacte pour une séquence d'observations peut donc être exprimée à l'aide des trois modèles ci-dessus, comme le montre l'équation 2.11, où ϕ représente la séquence d'actions observées de sorte que $f(o_i) = \phi_i$. L'équation 2.10 est obtenue en appliquant la règle de multiplication bayésienne. L'équation 2.11 est obtenue en appliquant le théorème de Bayes et au fait que $P(root(\mathcal{D})|\mathcal{D}) = 1$. Les trois termes de l'équation 2.11 peuvent être calculés respectivement à partir des équations 2.4, 2.6 et 2.8.

On représente par $P(\neg a_l)$ la probabilité associée à une action a_l non observée (mais réalisée par l'agent). Si l'hypothèse h donnée contient L actions sensées avoir été observées mais qui ne l'ont pas été, la probabilité que ces actions aient été effectivement exécutées mais non observées est définie comme suit :

$$P_{Unobs(h)} = \Pi_{l=0}^{L} P(\neg a_l), 0 \leq l \leq L. \tag{2.12}$$

L'algorithme considère que chaque action dans l'ensemble d'attente peut être exécutée et non observée. Il doit donc générer les hypothèses dans lesquelles aucune action

non observée a été effectuée. Il doit également tenir compte du fait que la prochaine action choisie dans l'ensemble peut subir le même scénario. Cette caractéristique de l'algorithme GGPO constitue une très grande différence avec l'algorithme PHATT, car ce dernier termine à la fin du flux d'observations. Par contre avec le flux d'actions inobservables, l'algorithme GGPO génère des hypothèses probables. Pour qu'il puisse terminer, il est nécessaire d'indiquer une valeur seuil pour laquelle une hypothèse ne peut plus être acceptable (relativement aux actions non observées) dans le flux d'actions inobservables. L'algorithme rejette une hypothèse h dont la probabilité des actions non observées est inférieure à un hyper-paramètre τ.

$$P_{Unobs(h)} < \tau.$$

Considérons l'exemple suivant. Si τ est fixé à 0,6 et que la valeur de $P_{Unobs(h)} = 0{,}9$ dans l'exécution du flux, l'ajout d'une nouvelle hypothèse par rapport à une action non observée avec probabilité de 0,6 n'est pas possible car la nouvelle valeur de $P_{Unobs(h)} = 0{,}54$ $(0{,}9 \times 0{,}6) < \tau$ $(0{,}6)$. Par contre, celle avec une probabilité de 0,7 est acceptée puisque $P_{Unobs(h)} = 0{,}63$ $(0{,}9 \times 0{,}7) > \tau$.

De tout ce qui précède, la probabilité *a priori* qu'une hypothèse $h = \langle \mathcal{D}, f \rangle$ s'avère exacte pour une séquence d'observations est :

$$P(h) = P(root(\mathcal{D}))P(\mathcal{D}|root(\mathcal{D}))P(\phi|\mathcal{D})P_{Unobs(h)}. \tag{2.13}$$

2.3.2 Probabilité d'un but

La probabilité *a priori* de chaque but étant un paramètre de la bibliothèque de plans, il reste à déterminer la probabilité *a posteriori* que l'agent réalise le but $b \in \mathcal{B}$ étant donné la séquence d'observations o. Notons cette probabilité $P(b|o)$.

Dans ce contexte, l'univers Ω de l'espace probabiliste est composé des séquences d'exécution complètes qui respectent la bibliothèque de plans. Cet ensemble est infini puisqu'il n'y a pas de limites sur le nombre de buts exécutés par un agent. Notons Ω_o le sous-ensemble des séquences d'exécution de Ω pour lesquels o est un préfixe.

Considérons l'ensemble H_o comme l'ensemble des hypothèses générées par l'algorithme pour la séquence d'observations o. Chacune de ces hypothèses représente un

sous-ensemble de séquences d'exécution Ω_o. Les hypothèses étant mutuellement exclusives (par construction), chaque séquence d'exécution est associée à au plus une hypothèse de H_o. Pour la suite du raisonnement probabiliste, supposons que H_o couvre toutes les séquences d'exécution de Ω_o ; H_o forme donc une partition sur Ω_o. Sous cette hypothèse, on fait appel à la formule des probabilités totales[2], la probabilité *a posteriori* d'un but s'écrit donc :

$$P(b|o) = \sum_{h \in H_o} P(b|h)P(h|o). \tag{2.14}$$

Il nous faut donc déterminer les termes $P(b|h)$ et $P(h|o)$ afin d'obtenir la probabilité *a posteriori*.

Le terme $P(h|o)$ peut être obtenu par utilisation du théorème de Bayes. On a :

$$P(h|o) = \frac{P(o|h)P(h)}{P(o)}. \tag{2.15}$$

Pour le terme $P(b|h)$, si l'hypothèse h inclut le but b, alors $P(b|h)$ vaut 1 et 0 dans le cas contraire, puisqu'une hypothèse émet une supposition sur les buts poursuivis. On a donc :

$$P(b|h) = \begin{cases} 1, \text{ si } b \in root(h); \\ 0, \text{ sinon} \end{cases} \tag{2.16}$$

où $root(h)$ désigne l'ensemble des buts poursuivis par l'agent suivant l'hypothèse h.

Par reformulation de l'équation 2.14 sur les hypothèses non nulles, c'est-à-dire concordant avec le but, on obtient :

$$P(b|o) = \sum_{h \in H_o \wedge b \in root(h)} P(h|o). \tag{2.17}$$

Par utilisation de la formule des probabilités totales, la probabilité *a priori* d'une séquence d'observations o est :

2. Cette formule, qui peut être mise sous la forme $P(A|B) = \sum_{i \in I} P(A|B_i)P(B_i|B)$, est en fait une variante de la formule originale des probabilités totales qui est $P(A) = \sum_{i \in I} P(A|B_i)P(B_i)$.

$$P(o) = \sum_{h \in H_o} P(o|h)P(h). \tag{2.18}$$

La probabilité d'une séquence d'observations o étant donné une hypothèse h, matérialisée par le terme $P(o|h)$ dans l'équation 2.18, vaut 1, car chaque hypothèse h de l'ensemble H_o est une hypothèses expliquant la séquence o.

L'équation 2.18 devient donc :

$$P(o) = \sum_{h \in H_o} P(h). \tag{2.19}$$

D'après tout ce qui précède, on a donc :

$$P(b|o) = \frac{\sum_{h \ \in \ H_o \wedge b \in root(h)} P(h)}{\sum_{h \in H_o} P(h)} \tag{2.20}$$

De ce qui précède (sous-section 2.3.1 et sous-section 2.3.2), on dispose de l'information nécessaire pour calculer la probabilité d'un but.

2.4 Algorithme

Globalement, l'algorithme met à jour itérativement un ensemble d'hypothèses. Chaque itération considère une action observée de façon à étendre l'ensemble des hypothèses qui explique la séquence des observations examinée à la fin de cette itération.

L'algorithme 1 détaille le processus de génération d'hypothèses de GGPO. Il prend en entrée une grammaire probabiliste $G = \langle \Sigma, NT, R, P \rangle$, une séquence d'observations o et la valeur seuil τ utilisée pour le rejet des hypothèses non acceptables.

L'hypothèse $\langle \mathcal{D}, f \rangle$ est représentée par un tuple

$$(\langle \mathcal{D}, \{PS_1, ..., PS_n\}, P_{\mathcal{B}}, P_{\langle \mathcal{D}, \mathcal{B} \rangle}, P_{Unobs} \rangle)$$

dans lequel la fonction f est cachée dans les ensembles de substitutions PS_i ($1 \leq i \leq n$). On a :

- $\mathcal{D}$ qui représente l'ensemble des arbres de dérivation (*cf.* définition 2.2.1) ;

- $\{PS_1,...,PS_n\}$ qui représente les ensembles de substitutions dans la construction de $\mathcal{D}$, l'ensemble des arbres de dérivation après l'atteinte de n symboles de o ;
- $P_{\mathcal{B}}$ qui représente la probabilité *a priori* des buts poursuivis (modèle des intentions défini à l'équation 2.4) ;
- $P_{\langle \mathcal{D},\mathcal{B} \rangle}$ qui représente la probabilité conditionnelle du choix des règles (modèle de planification défini à l'équation 2.6) ;
- P_{Unobs} qui représente la probabilité associée aux actions non observées (*cf.* équation 2.12).

À chaque itération, toutes les hypothèses de l'ensemble H (ensemble des hypothèses générées) sont étendues pour intégrer une nouvelle observation et les nouvelles hypothèses générées forment l'ensemble d'hypothèses de l'itération suivante. Pour une observation o_i, l'algorithme étend chaque hypothèse en trois étapes, puis elle est acceptée ou rejetée suivant qu'elle satisfait ou non le critère d'acceptation.

Dans l'algorithme 1, l'ensemble d'hypothèses à une profondeur de l'arbre d'hypothèses est représenté par une file de façon à générer cet arbre en largeur d'abord. Les fonctions *Emptyqueue* (lignes 4, 7 et 23), *Nonemptyqueue* (ligne 6), *Enqueue* (lignes 4, 12, 20 et 32), *Dequeue* (ligne 8) sont les fonctions usuelles sur une file. Par exemple, *Enqueue*(h,H) ajoute l'hypothèse h à la queue de la file H. La file est initialisée avec l'hypothèse racine h_0^0 pour laquelle l'ensemble des arbres de dérivation et l'ensemble des substitutions sont vides (ligne 4). Les valeurs des probabilités sont initialisées à 1, car elles agissent comme valeurs initiales dans les produits de probabilités.

Une observation peut contribuer à un but qui fait déjà partie des arbres de dérivation. Dans ce cas, les lignes 10 à 13 étendent une hypothèse pour en produire de nouvelles en considérant chaque arbre de dérivation de PS_{i-1} ayant comme pied (foot) o_i. Une hypothèse enfant est donc générée pour l'action o_i dans l'ensemble des points d'attache en attente. Une observation peut aussi introduire un ou plusieurs arbres de dérivation. Les lignes 15 à 21 mettent à jour l'ensemble des arbres de dérivation $\mathcal{D}$. La fonction *BackpatchPS* (ligne 17) permet la mise à jour de chacun des ensembles de substitutions en leur ajoutant les arbres qui auraient dû être présents si le but introduit par T avait été connu plus tôt. Les lignes 24 à 33 étendent une hypothèse en considérant chaque arbre de dérivation de PS_i dont le pied a_i correspond à une

action qui n'est pas observée. Seules les hypothèses dont la probabilité des actions non observées ne dépasse par l'hyper-paramètre seuil τ sont conservées. Lorsque toutes les observations et hypothèses ont été traitées, l'ensemble H correspond aux hypothèses expliquant la séquence d'observations o.

Pour générer une hypothèse h correspondant à une action non observée, on procède comme suit. Pour chaque arbre d'analyse composant l'hypothèse (ligne 26), on considère chaque action non observée pouvant commencer une séquence d'actions dérivée de cet arbre (ligne 28), et on étend l'hypothèse avec cette action comme si elle avait été observée (ligne 29), pourvu que la probabilité des actions non observées ne soit pas supérieure ou égale au seuil τ (ligne 28). On met ensuite la probabilité des actions non observées à jour en tenant compte de cette nouvelle action.

L'algorithme 2 permet d'obtenir la probabilité *a posteriori* de chacun des buts à partir des hypothèses générées par l'algorithme 1. Les paramètres en entrée sont les hypothèses générées H, la séquence d'observations o et l'ensemble des buts. Les lignes 6 à 9 calculent la probabilité *a priori* d'une hypothèse selon l'équation 2.13. La probabilité *a posteriori* d'un but $P(b|o)$ est calculée à la ligne 12 selon l'équation 2.20 et affichée à la ligne 13.

Parmi les facteurs qui contribuent à la complexité du problème, on trouve les contraintes d'ordonnancement, le nombre de règles pour dériver un symbole non terminal, le nombre de membres droits des règles de production, le nombre de buts et la profondeur du plan. Le nombre d'hypothèses générées croît doublement exponentiellement avec le nombre de buts partageant un ensemble d'actions initiales.

Algorithme 1 Génération d'hypothèses

procedure EXPLAIN(G,$o_{1:n}$,τ)
  %% $G = \langle \Sigma, NT, P, R \rangle$, la grammaire probabiliste
  %% avec $\mathbb{P}_b(R)$ la distribution de probabilité sur les buts
  $H := Emptyqueue()$, $Enqueue(\langle \emptyset, \{\emptyset\}, 1, 1, 1 \rangle, H)$ %% hypothèse vide h_0^0
  loop FOR EACH $i := 1$ *to* n
    while $Nonemptyqueue(H)$ **do**
      $H' := Emptyqueue()$
      $\langle \mathcal{D}, \{PS_0, ..., PS_{i-1}\}, P_{\mathcal{B}}, P_{\langle \mathcal{D},\mathcal{B} \rangle}, P_{Unobs} \rangle := Dequeue(H)$
      %% Extension d'arbres par rapport à o_i
      loop FOR EACH $T \in PS_{i-1}$ SUCH THAT $foot(T) = o_i$
        $\mathcal{D}_{new} :=$ Substitute(T,$\mathcal{D}$) %% *cf.* définition 2.1.7
        $Enqueue(\langle \mathcal{D}_{new}, \{PS_0 ... PS_{i-1}, PS(\mathcal{D}_{new})\}, P_{\mathcal{B}}, P_{\langle \mathcal{D},\mathcal{B} \rangle}, P_{Unobs} \rangle, H')$
      end loop
      %% Introduction d'un nouvel arbre de $\mathcal{T}$ (*cf.* définition 2.1.8)
      loop FOR EACH $T \in \mathcal{T}$ SUCH THAT $foot(T) = o_i$
        $\mathcal{D}_{new} := \mathcal{D} \cup \{T\}$
        $\{PS'_0 ... PS'_{i-1}\} =$ BackpatchPS(root(T), $\{PS_0, ..., PS_{i-1}\}$)
        $P'_{\mathcal{B}} = P_{\mathcal{B}} \times \mathbb{P}_b(root(T))$ %% *cf.* équation 2.4
        $P'_{\langle \mathcal{D},\mathcal{B} \rangle} = P_{\langle \mathcal{D},\mathcal{B} \rangle} \times \Pi_{p \in \text{règles}(\mathcal{T})} \frac{1}{|mg(p)|}$ %% *cf.* équation 2.6
        $Enqueue(\langle \mathcal{D}_{new}, \{PS'_0 ... PS'_{i-1}, PS(\mathcal{D}_{new})\}, P'_{\mathcal{B}}, P'_{\langle \mathcal{D},\mathcal{B} \rangle}, P_{Unobs} \rangle, H')$
      end loop
      %% Hypothèses apportées par les actions non observées
      $H := Emptyqueue()$
      loop FOR EACH $\langle \mathcal{D}, \{PS_0, ..., PS_i\}, P_{\mathcal{B}}, P_{\langle \mathcal{D},\mathcal{B} \rangle}, P_{Unobs}, \rangle \in H'$
        $\mathcal{D}_{new} := \mathcal{D}$
        loop FOR EACH $T \in PS_i$
          $a_i = foot(T)$
          if $a_i = o_i$ *or* $P_{Unobs} \times P(\neg a_i) < \tau$ **Continue** %% *cf.* éq. 2.12
          $\mathcal{D}_{new} := \mathcal{D}_{new} \cup T$
          $P_{Unobs} = P_{Unobs} \times P(\neg a_i)$
        end loop
        $Enqueue(\langle \mathcal{D}_{new}, \{PS_0, ..., PS_i\}, P_{\mathcal{D}}, P_{\langle \mathcal{D},\mathcal{B} \rangle}, P_{Unobs} \rangle, H)$
      end loop
    end while
  end loop
  return H
end procedure

Algorithme 2 Calcul des probabilités

1: **procedure** COMPUTEPROB(H,o,R)
2: %% H, l'ensemble des hypothèses calculé par EXPLAIN
3: %% o, la séquence d'observations
4: %% R, les buts de la bibliothèque de plans
5:
6: **loop** FOR EACH $h = \langle \mathcal{D}, \{PS_0,...,PS_n\}, P_{\mathcal{B}}, P_{\langle \mathcal{D},\mathcal{B} \rangle}, P_{Unobs} \rangle \in H$
7: $P(o|\mathcal{D}) := \Pi_{i=1}^{n} \frac{1}{|PS_i|}$ %% *cf.* équation 2.8
8: $P(h) := P_{\mathcal{B}} \times P_{\langle \mathcal{D},\mathcal{B} \rangle} \times P(o|\mathcal{D}) \times P_{Unobs(h)}$ %% *cf.* équation 2.13
9: **end loop**
10:
11: **loop** FOR EACH $b \in R$
12: $P(b|o) := \frac{\sum_{h \in H \wedge b \in root(h)} P(h)}{\sum_{h \in H} P(h)}$ %% *cf.* équation 2.20
13: **print** $P(b|o)$
14: **end loop**
15: **end procedure**

Chapitre 3

Évaluation de l'algorithme

3.1 Développement

L'algorithme PHATT a été originalement implémenté en Common Lisp par Geib et Goldman. Cette version n'est pas disponible publiquement. L'algorithme GGPO a été implémenté en réutilisant les sources d'une implémentation de PHATT en Python, disponible au laboratoire PLANIART. Tous les tests ont été effectués en utilisant un interpréteur Python dans un environnement sous Ubuntu 14.04 sur un processeur à trois cœurs Intel Core i3-4030U (1.9 GHz).

Au chapitre 2, les fonctions qui correspondent aux fonctions usuelles utilisées dans les structures de données sur une file ont été implémentées par des listes. L'utilisation du langage Python offre de nombreux avantages de part le fait qu'il supporte la programmation fonctionnelle et orientée objet. Ceci a été utile dans la définition et la réutilisation de certaines classes, par exemple, pour représenter la bibliothèque de plans. Pour évaluer l'algorithme GGPO, on a eu recours à certaines métriques comme le temps CPU et le nombre d'hypothèses générées.

3.2 Évaluation de l'algorithme

L'évaluation se concentre sur la comparaison du nombre d'hypothèses générées, du nombre d'observations et du temps CPU mis pour expliquer les observations.

3.2.1 Échantillons de données

GGPO a été testé sur le corpus de plans Monroe, mis sur pied par l'Université de Rochester. Ce corpus est très utilisé pour les problèmes de reconnaissance de plan. Quelques auteurs [28, 31, 11, 32, 8, 7, 6] l'ont d'ailleurs déjà utilisée. *Monroe Plan Corpus* est une collections de 5000 sessions de plans générées automatiquement à l'aide d'un planificateur reposant sur SHOP2 (*Simple Hierarchical Ordered Planner 2*) [26], un système de planification indépendant du domaine, basé sur un réseau de tâches hiérarchiques (HTN). Chaque plan généré aléatoirement contient les sous-buts concourant à la réalisation d'un but principal. Le planificateur qui génère les plans prend des décisions non déterministes, il permet donc de générer un ensemble diversifié de plans (constitués de sous-buts ordonnés ou non) pour atteindre les buts principaux.

Le domaine Monroe (*cf.* Tableau 3.1) issu d'un domaine de gestion de catastrophes (intervention d'urgence) [5], a été légèrement modifié (suppression de la récursivité à gauche et des règles de production d'effacement dans la grammaire) par Bisson et al. [4]. La bibliothèque qui en résulte est constituée des paramètres définis au Tableau 3.2.

Nombre de sessions	5000
Symboles d'actions	10
Symboles de (sous) buts	28
Symboles de buts principaux	10
Moyenne de buts par session	9,6
Règles de production	46
Profondeur moyenne des (sous) buts	3,8
Profondeur maximale des (sous) buts	9

Tableau 3.1 – *Corpus Monroe*

Symboles d'actions	30
Symboles de buts	43
Symboles de buts principaux	10
Règles de production	91

Tableau 3.2 – *Domaine modifié du corpus Monroe*

La bibliothèque de plans utilisée est similaire à celle utilisée par Geib et al. [18]. Elle a une structure qui permet d'expliciter les différents paramètres qui interviennent dans la complexité du problème :

- nombre de buts ;
- nombre d'actions uniques ;
- contraintes d'ordonnancement ;
- niveau d'imbrication des règles de production (profondeur) ;
- nombre de membres droits des règles de production (facteur de branchement *ET*) ;
- nombre de règles de production pour dériver le même symbole non terminal (facteur de branchement *OU*).

Les plans de la bibliothèque sont structurés pour avoir un niveau de nœuds *ET* et *OU* en alternance. Les actions de chaque plan sont choisies aléatoirement parmi les actions de la bibliothèque. Pour chaque cas de test, une bibliothèque de plans respectant la structure décrite est générée, un des buts est donc dérivé en une séquence d'observations. Pour chaque choix de cette dérivation, une décision est prise de manière probabiliste en respectant les modèles définis au chapitre 2.

3.2.2 Métriques évaluées

Dans la sous-section 3.2.3, les graphiques présentés explicitent les grandeurs suivantes :

- le nombre d'hypothèses générées en fonction du nombre d'observations ;
- le temps CPU mis pour expliquer un nombre d'observations.

3.2.3 Résultats et commentaires

Les performances de GGPO et PHATT sont évaluées sur un problème de reconnaissance de buts et les résultats obtenus sont consignés dans les figures 3.1, 3.2 et 3.3. Il faut noter que dans les trois graphiques, l'axe vertical est représenté sous une échelle logarithmique (base 10).

La Figure 3.1 compare l'évolution du nombre d'hypothèses générées en fonction du nombre d'observations expliquées respectivement par GGPO et PHATT. Après avoir généré environ 82 584 hypothèses pour six observations expliquées, GGPO manque de mémoire pour continuer la génération d'hypothèses. Cela traduit le fait que les traitements lors de la génération des arbres d'hypothèses dans GGPO sont coûteux en mémoire et font face à l'explosion combinatoire. À titre d'illustration, le fichier obtenu à la troisième expérience de l'annexe A est d'environ 40 Mo et ralentissait l'ordinateur sur lequel étaient effectués les tests.

La figure 3.2 illustre les performances de GGPO et PHATT par rapport au temps CPU mis pour expliquer les observations. On constate que PHATT explique des observations pour la reconnaissance de buts de manière beaucoup plus rapide que GGPO. Par exemple, PHATT explique environ six observations pour un temps CPU de 2,2 s environ, alors que pour le même nombre d'observations, on a 318 s pour GGPO.

La Figure 3.3 montre l'évolution du nombre d'hypothèses générées par GGPO en fonction du nombre d'observations. On constate qu'à trois observations, on obtient 1 420 hypothèses générées. Mais à quatre observations, on obtient 736. Considérons la première expérience de l'annexe A, on suppose que dans la séquence d'entrée, on n'a pas l'observation *co*. La prise en compte de la nouvelle observation *co* peut ne pas créer de nouvelles hypothèses mais étendre celles qui existent déjà et encore là il faudrait que les hypothèses existantes puissent être combinées à la nouvelle observation. Autrement dit le nombre d'hypothèses ne croît pas toujours avec le nombre d'observations.

Ces figures confirment que la prise en compte d'actions non observées permet d'obtenir des résultats très proches de ceux obtenus lorsqu'on se trouve en situation d'observabilité totale mais de manière beaucoup plus lente. GGPO donne des résultats plus proches de ceux de PHATT avec beaucoup plus de temps, car il nécessite des calculs supplémentaires pour la génération de toutes les hypothèses.

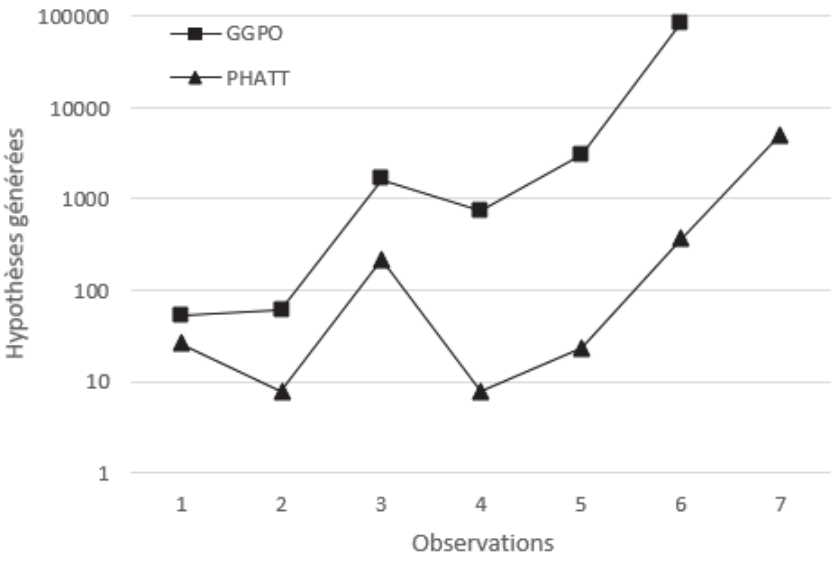


Figure 3.1 – *Comparaison du nombre d'hypothèses générées*

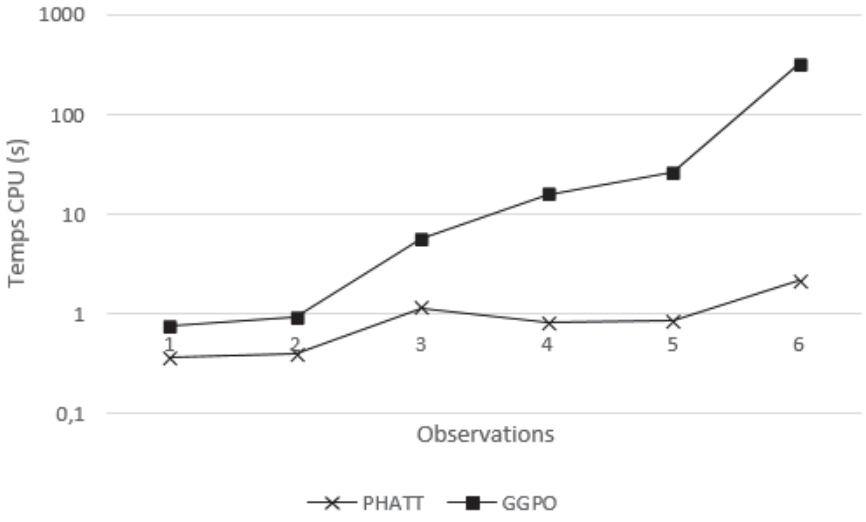


Figure 3.2 – *Analyse du temps pour expliquer les observations*

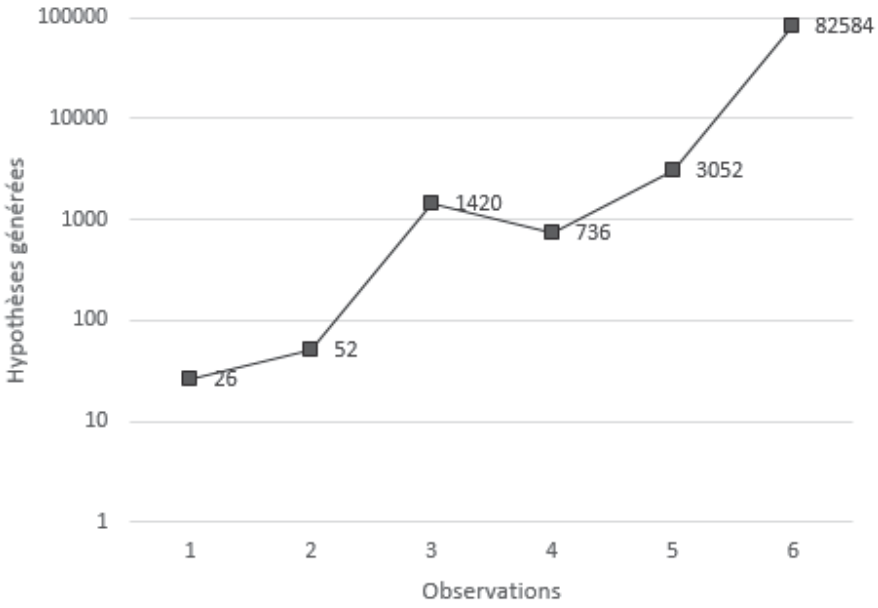


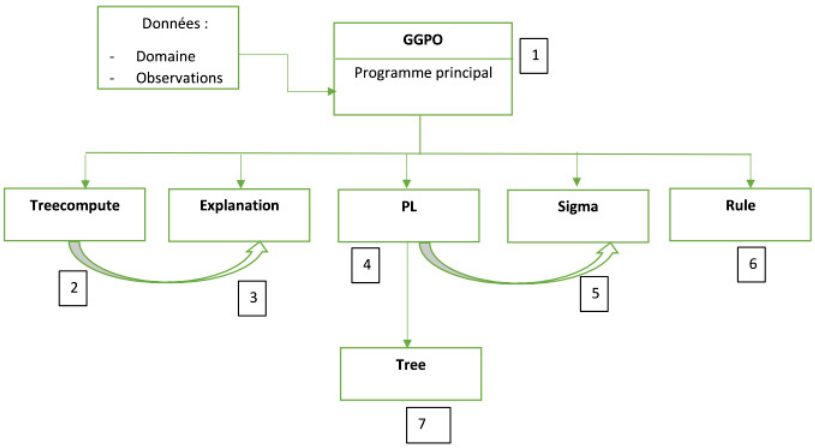
Figure 3.3 – *Évolution du nombre d'hypothèses en fonction des observations*

Conclusion

À partir d'une description succincte de l'algorithme de Geib et Goldman pour la reconnaissance de buts avec observabilité partielle, nous avons fourni une description plus détaillée et nous avons évalué cet algorithme. Plus particulièrement, nous avons fourni une description en pseudocode, puis nous avons implémenté l'algorithme GGPO. Nous avons défini un cadre de comparaison afin de procéder à l'évaluation de l'algorithme. Nous avons effectué des simulations avec l'algorithme PHATT pour la comparaison avec GGPO. L'analyse des résultats nous a permis de confirmer que le problème de reconnaissance de plan avec observabilité partielle est plus complexe que celui avec observabilité totale. Sur un même cas d'observabilité totale, GGPO arrive aux mêmes résultats que PHATT mais de manière beaucoup plus lente. Cependant, il existe encore des améliorations possibles. Par exemple, la détermination de la valeur seuil optimale utilisée pour accepter une hypothèse. Cela passe par des expérimentations avec différentes valeurs seuil. La mise à jour des arbres de dérivation lors de la génération des hypothèses, qui est un processus coûteux, nécessite sûrement une amélioration. Toutefois ces améliorations auront peu d'impact, étant donné que GGPO a une complexité doublement exponentielle. Par ailleurs, le seuil biaise les résultats car une hypothèse rejetée pourrait être ultérieurement utile pour expliquer une ou plusieurs observations.

Annexe A

Architecture du programme



1- GGPO (248 lc), classe principale, utilisée aussi pour la lecture des données.

2- Treecompute (144 lc), classe utilisée pour l'extension ou l'introduction d'arbres selon l'algorithme 1.

3- Explanation (128 lc), classe qui effectue le calcul des probabilités selon l'algorithme 2.

4- PL (220 lc), classe qui représente la librairie de plans et qui traite les arbres expliquant les observations.

5- Sigma (92 lc), classe utilisée pour parcourir les données afin d'utiliser l'alphabet.

6- Rule (115 lc), classe utilisée pour le traitement des contraintes sur les règles.

7- Tree (366 lc), classe contenant tous les traitements nécessaires à la mise à jour des arbres.

Au total 1313 lignes de codes.

Figure A.1 – *Architecture de GGPO*

Annexe B

Quelques données extraites d'expériences

B.1 Bibliothèque de plans

Le domaine Monroe (*cf.* Tableau 3.1) simplifié tel que défini au Tableau 3.2 est représenté par la bibliothèque suivante :

```
Plan Library:
Sigma=[<np />, <ep />, <dp />, <nv />, <ci />, <co />,
       <ld />, <ud />, <te />, <ti />, <ca />, <rw />,
       <sw />, <cb />, <cu />, <hu />, <pu />, <ro />,
       <py />, <pg />, <rh />, <sr />, <pc />, <pi />,
       <ht />, <uf />, <di />, <fi />, <pp />, <cz />]
NT=[<GT />, <SU />, <FW />, <CR />, <CW />, <CT />, <NS />,
    <PR />, <SB />, <QR />, <PH />, <FL />, <PA />, <CH />,
    <BR />, <UB />, <GE />, <RP />, <OH />, <CH />, <SC />,
    <TC />, <CW />, <TT />, <LT />, <RB />, <CX />, <DC />,
    <GT />, <FF />, <AF />, <RL />, <SP />, <TU />, <SO />,
    <OW />, <ET />, <SZ />, <EO />, <EE />, <DT />, <GI />,
    <GU />]
```

```
R=[<SU />, <FW />, <CR />, <CW />, <CT />, <PR />, <QR />,
   <PH />, <FL />, <PA />]

Rules= GT -> EO | [ ]
[ ]
SU -> GE EO GT | [ (GE,EO) (EO,GT) ]
[ ]
SU -> GE EO | [ (GE,EO) ]
[ ]
SU -> GE | [ ]
[ ]
SU -> EO GT | [ (EO,GT) ]
[ ]
SU -> EO | [ ]
[ ]
FW -> SO RP OW | [ (SO,RP) (RP,OW) ]
[ ]
CR -> BR CH UB | [ (BR,CH) (BR,UB) ]
[ ]
CW -> SC CW TC | [ (SC,CW) (CW,TC) ]
[ ]
CT -> SC LT TC | [ (SC,LT) (LT,TC) ]
[ ]
NS -> np | [ ]
[ ]
PR -> EO np ep NS dp | [ (EO,np) (np,ep) (ep,NS) (NS,dp) ]
[ ]
PR -> NS dp np ep | [ (NS,dp) (dp,np) (np,ep) ]
[ ]
SB -> sr | [ ]
[ ]
QR -> DC EO GT sr SB | [ (DC,EO) (EO,GT) (GT,sr) (sr,SB) ]
[ ]
QR -> DC EO sr SB | [ (DC,EO) (EO,sr) (sr,SB) ]
[ ]
```

```
QR -> DC sr SB | [ (DC,sr) (sr,SB) ]
[ ]
PH -> EO | [ ]
[ ]
PH -> GT rh | [ (GT,rh) ]
[ ]
FL -> EO GT RL | [ (EO,GT) (GT,RL) ]
[ ]
FL -> EO RL | [ (EO,RL) ]
[ ]
FL -> RL | [ ]
[ ]
PA -> EO ti | [ (EO,ti) ]
[ ]
PA -> ti | [ ]
[ ]
PA -> ET | [ ]
[ ]
CH -> ca | [ ]
[ ]
CH -> EO cz | [ (EO,cz) ]
[ ]
CH -> cz | [ ]
[ ]
BR -> SC EO | [ (SC,EO) ]
[ ]
BR -> SC | [ ]
[ ]
UB -> TC | [ ]
[ ]
GE -> GT | [ ]
[ ]
RP -> EO SC OH pp CH TC
    | [ (EO,SC) (SC,OH) (OH,pp) (pp,CH) (CH,TC) ]
```

```
[ ]
RP -> SC OH pp CH TC | [ (SC,OH) (OH,pp) (pp,CH) (CH,TC) ]
[ ]
OH -> EO di | [ (EO,di) ]
[ ]
OH -> di | [ ]
[ ]
CH -> EO fi | [ (EO,fi) ]
[ ]
CH -> fi | [ ]
[ ]
SC -> EO pc | [ (EO,pc) ]
[ ]
SC -> pc | [ ]
[ ]
TC -> EO pi | [ (EO,pi) ]
[ ]
TC -> pi | [ ]
[ ]
CW -> TT | [ ]
[ ]
TT -> EO ht GT uf | [ (EO,ht) (ht,GT) (GT,uf) ]
[ ]
TT -> ht EO uf | [ (ht,EO) (EO,uf) ]
[ ]
TT -> EO ht uf | [ (EO,ht) (ht,uf) ]
[ ]
TT -> ht uf | [ (ht,uf) ]
[ ]
LT -> EO cu RB | [ (EO,cu) (cu,RB) ]
[ ]
LT -> EO cu | [ (EO,cu) ]
[ ]
LT -> cu RB | [ (cu,RB) ]
```

```
[ ]
LT -> cu | [ ]
[ ]
RB -> EO cb | [ (EO,cb) ]
[ ]
RB -> cb | [ ]
[ ]
RB -> EO | [ ]
[ ]
CX -> ca | [ ]
[ ]
DC -> ca CX | [ (ca,CX) ]
[ ]
GT -> FF EO hu ro | [ (FF,EO) (EO,hu) (hu,ro) ]
[ ]
GT -> FF hu ro | [ (FF,hu) (hu,ro) ]
[ ]
FF -> EO AF GT pu | [ (EO,AF) (AF,GT) (GT,pu) ]
[ ]
FF -> AF EO pu | [ (AF,EO) (EO,pu) ]
[ ]
FF -> EO AF pu | [ (EO,AF) (AF,pu) ]
[ ]
FF -> AF pu | [ (AF,pu) ]
[ ]
FF -> EO AF | [ (EO,AF) ]
[ ]
FF -> AF | [ ]
[ ]
AF -> py pg | [ (py,pg) ]
[ ]
RL -> SP LT rw sw TU | [ (SP,LT) (LT,rw) (rw,sw) (sw,TU) ]
[ ]
RL -> SP rw sw TU | [ (SP,rw) (rw,sw) (sw,TU) ]
```

```
[ ]
SP -> ca | [ ]
[ ]
TU -> ca | [ ]
[ ]
SO -> ca | [ ]
[ ]
OW -> ca | [ ]
[ ]
ET -> EO te | [ (EO,te) ]
[ ]
ET -> te | [ ]
[ ]
SZ -> ET | [ ]
[ ]
EO -> DT EE | [ (DT,EE) ]
[ ]
EO -> DT | [ ]
[ ]
EE -> GI EO GU EE | [ (GI,EO) (EO,GU) (GU,EE) ]
[ ]
EE -> GI GU EE | [ (GI,GU) (GU,EE) ]
[ ]
EE -> GI EO GU | [ (GI,EO) (EO,GU) ]
[ ]
EE -> GI GU | [ (GI,GU) ]
[ ]
EE -> SZ GI EO GU EE | [ (SZ,GI) (GI,EO) (EO,GU) (GU,EE) ]
[ ]
EE -> SZ GI GU EE | [ (SZ,GI) (GI,GU) (GU,EE) ]
[ ]
EE -> SZ GI EO GU | [ (SZ,GI) (GI,EO) (EO,GU) ]
[ ]
EE -> SZ GU GI | [ (SZ,GU) (GU,GI) ]
```

```
[ ]
DT -> nv | [ ]
[ ]
GI -> ci | [ ]
[ ]
GI -> EO ld | [ (EO,ld) ]
[ ]
GI -> ld | [ ]
[ ]
GU -> co | [ ]
[ ]
GU -> EO ud | [ (EO,ud) ]
[ ]
GU -> ud | [ ]
[ ]
```

B.2 Résultats partiels d'expérimentation

Dans ce qui suit, le seuil d'élagage des hypothèses (on doit avoir $P_{Unobs(h)} \leq seuil$) a été fixé à 0,75 et la distribution des probabilités sur les buts de l'agent est la distribution uniforme.

B.2.1 Expérience 1

i. Séquence d'observations en entrée :
   `[<nv />, <ci />, <nv />, <co />]`

ii. Buts reconnus : CT, PR, SU, PA, PH, FL, CW avec pour probabilités respectives [0,19 ; 0,19 ; 0,22 ; 0,24 ; 0,28 ; 0,29 ; 0,30].

iii. Exemple d'une dérivation pour une hypothèse :

```
<Explanation prob="5.78703703704e-05">
<tree>
<PH />
 <GT />
  <EO />
```

```
   <DT />
<nv />
   <EE />
<GI />
 <ci />
<EO />
 <DT />
  <nv />
 <EE />
<GU />
 <co />
<EE />
 <rh />
</tree>
</Explanation>
```

Cette hypothèse est l'une des 736 hypothèses générées.

iv. Action non observée introduite : rh.

v. Temps d'exécution pour générer toutes les hypothèses : 14,15 s.

B.2.2 Expérience 2

i. Séquence d'observations en entrée :
`[<nv />, <ci />, <nv />, <co />, <pc />]`

ii. Buts reconnus : CT, PR, SU, PA, PH, FL, CW, CR avec pour probabilités respectives [0,58 ; 0,11 ; 0,25 ; 0,22 ; 0,22 ; 0,26 ; 0,60 ; 0,27].

iii. Exemple d'une dérivation pour une hypothèse :

```
<Explanation prob="0.0025">
<tree>
<PA />
 <ET />
  <EO />
   <DT />
<nv />
   <EE />
```

```
<GI />
 <ci />
<EO />
 <DT />
  <nv />
<GU />
 <co />
<EE />
  <te />
</tree>
<tree>
<CT />
 <SC />
  <pc />
 <LT />
 <TC />
</tree>
</Explanation>
```

Cette hypothèse est l'une des 3 052 hypothèses générées.

iv. Action non observée introduite : te.

v. Temps d'exécution pour générer toutes les hypothèses : 25,3 s.

B.2.3 Expérience 3

i. Séquence d'observations en entrée :

```
[<nv />, <ci />, <nv />, <co />, <pc />, <nv />]
```

ii. Buts reconnus : CT, PR, SU, PA, PH, FL, CW, CR avec pour probabilités respectives [0,32 ; 0,20 ; 0,34 ; 0,30 ; 0,41 ; 0,37 ; 0,71 ; 0,52].

iii. Exemple d'une dérivation pour une hypothèse :

```
<Explanation prob="1.25e-05">
<tree>
<CW />
 <TT />
  <EO />
```

```
   <DT />
<nv />
   <EE />
<GI />
 <ci />
<GU />
 <co />
  <ht />
  <GT />
  <uf />
</tree>
<tree>
<PA />
 <EO />
  <DT />
   <nv />
 <ti />
</tree>
<tree>
<CW />
 <SC />
  <pc />
 <CW />
 <TC />
</tree>
<tree>
<PH />
 <EO />
  <DT />
   <nv />
  <EE />
</tree>
</Explanation>
```

Cette hypothèse est l'une des 82 584 hypothèses générées.

iv. Actions non observées introduites : ti, ht, uf.

v. Temps d'exécution pour générer toutes les hypothèses : 318,22 s.

Bibliographie

[1] Dorit Avrahami-Zilberbrand et Gal A Kaminka.
« Fast and complete symbolic plan recognition ».
Dans *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 653–658, 2005.

[2] Francis Bisson.
« La reconnaissance de plan des adversaires ».
Mémoire de maîtrise, Université de Sherbrooke, Département d'informatique, Sherbrooke, Québec, Canada, 2012.

[3] Francis Bisson, Froduald Kabanza, Abder Rezak Benaskeur et Hengameh Irandoust.
« Provoking opponents to facilitate the recognition of their intentions ».
Dans *Proceedings of the AAAI Student Abstract and Poster Program*, 2011.

[4] Francis Bisson, Hugo Larochelle et Froduald Kabanza.
« Using a recursive neural network to learn an agent's decision model for plan recognition ».
Dans *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 918–924, 2015.

[5] Nate Blaylock et James Allen.
« Generating artificial corpora for plan recognition ».
Dans *International Conference on User Modeling*, pages 179–188, 2005.

[6] Nate Blaylock et James Allen.
« Fast hierarchical goal schema recognition ».

Dans *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 796–801, 2006.

[7] Nate BLAYLOCK et James ALLEN.
« Hierarchical instantiated goal recognition ».
Dans *Proceedings of the AAAI Workshop on Modeling Others from Observations*, pages 8–15, 2006.

[8] Nathan J. BLAYLOCK.
« *Towards tractable agent-based dialogue* ».
Thèse de doctorat, Dept. of Computer Science, University of Rochester, August 2004.

[9] Eugene CHARNIAK et Robert GOLDMAN.
« A probabilistic model of plan recognition ».
Dans *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 160–165, 1991.

[10] Philip R COHEN et Hector J LEVESQUE.
« Intention is choice with commitment ».
Artificial intelligence, 42:213–261, 1990.

[11] Tecuci DAN et Bruce PORTER.
« Memory based goal schema recognition ».
Dans *Proceedings of the 22nd Florida Artificial Intelligence Research Society Conference*, pages 111–116, 2009.

[12] Eric DEMEESTER, Alexander HÜNTEMANN, Dirk VANHOOYDONCK, Gerolf VANACKER, Hendrik VAN BRUSSEL et Marnix NUTTIN.
« User-adapted plan recognition and user-adapted shared control : A Bayesian approach to semi-autonomous wheelchair driving ».
Autonomous Robots, 24:193–211, 2008.

[13] Julien FILION.
« Reconnaissance de plan probabiliste par exploration partielle des hypothèses ».
Mémoire de maîtrise, Université de Sherbrooke, Département d'informatique, Sherbrooke, Québec, Canada, 2015.

[14] Ya'akov Gal, Swapna Reddy, Stuart M Shieber, Andee Rubin et Barbara J Grosz.
« Plan recognition in exploratory domains ».
Artificial intelligence, 176(1):2270–2290, 2012.

[15] Christopher W Geib.
« Delaying commitment in plan recognition using combinatory categorial grammars ».
Dans *International Joint Conference on Artificial Intelligence*, pages 1702–1707, 2009.

[16] Christopher W Geib et Robert P Goldman.
« Partial observability and probabilistic plan/goal recognition ».
Dans *Proceedings of the IJCAI Workshop on Modeling Others from Observations*, pages 418–425, 2005.

[17] Christopher W. Geib et Robert P. Goldman.
« A probabilistic plan recognition algorithm based on plan tree grammars ».
Artificial Intelligence, 173:1101–1132, 2009.

[18] Christopher W Geib, John Maraist et Robert P Goldman.
« A new probabilistic plan recognition algorithm based on string rewriting ».
Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 91–98, 2008.

[19] Robert P Goldman, Christopher W Geib et Christopher A Miller.
« A new model of plan recognition ».
Dans *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 245–254. Morgan Kaufmann Publishers Inc., 1999.

[20] Peter A Jarvis, Teresa F Lunt et Karen L Myers.
« Identifying terrorist activity with AI plan recognition technology ».
AI Magazine, 26:73–81, 2005.

[21] Froduald Kabanza, Julien Filion, Abder Rezak Benaskeur et Hengameh Irandoust.
« Controlling the hypothesis space in probabilistic plan recognition ».
Dans *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2306–2312, 2013.

[22] Henry KAUTZ, Oren ETZIONI, Dieter FOX, Dan WELD et Lokendra SHASTRI.
« Foundations of assisted cognition systems ».
Rapport Technique, University of Washington, Computer Science Department, 2003.

[23] Henry A KAUTZ et James F ALLEN.
« Generalized plan recognition ».
Dans *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 32–37, 1986.

[24] Alexander KOTT et William M. MCENEANEY.
Dans *Adversarial reasoning : computational approaches to reading the opponent's mind*, chapitre 1, pages 77–100.
Chapman & Hall/CRC, 2006.

[25] Neal LESH, Charles RICH et Candace L. SIDNER.
« Using plan recognition in human-computer collaboration ».
Dans *Proceedings of the Seventh International Conference UM99 User Modeling*, volume 407 de *CISM International Centre for Mechanical Sciences*, pages 23–32. Springer, 1999.

[26] Dana S NAU, Tsz-Chiu AU, Okhtay ILGHAMI, Ugur KUTER, J William MURDOCK, Dan WU et Fusun YAMAN.
« SHOP2 : An HTN planning system ».
Journal of Artificial Intelligence Research, 20:379–404, 2003.

[27] David V PYNADATH et Michael P WELLMAN.
« Accounting for context in plan recognition, with application to traffic monitoring ».
Dans *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 472–481, 1995.

[28] Sindhu RAGHAVAN et Raymond MOONEY.
« Bayesian abductive logic programs ».
Dans *Proceedings of the AAAI Workshop on Statistical Relational Artificial Intelligence*, pages 82–87, 2010.

[29] C. F. Schmidt, N. S. Sridharan et J. L. Goodson.
« The plan recognition problem : an intersection of psychology and artificial intelligence ».
Artificial Intelligence, 11:45–83, 1978.

[30] Gabriel Synnaeve et Pierre Bessière.
« A Bayesian model for plan recognition in RTS games applied to StarCraft ».
Dans *Proceedings of the 7th Artificial Intelligence and Interactive Digital Entertainment International Conference*, pages 79–84, 2011.

[31] Dan Tecuci.
« *A generic memory module for events* ».
Thèse de doctorat, Computer Sciences Department, University of Texas at Austin, August 2007.

[32] Dan Tecuci et Bruce W Porter.
« An episodic based approach to complex event processing ».
Dans *Proceedings of the AAAI Spring Symposium : Intelligent Event Processing*, pages 86–91, 2009.