Une approche multi-agent pour la conception de systèmes d'intelligence ambiante : un modèle formel intégrant planification et apprentissage

Ahmed Chawki Chaouche

Université Pierre et Marie Curie - Paris 6
ÉCOLE DOCTORALE EDITE
Laboratoire d'informatique Paris 6 - LIP6

en cotutelle avec

Université Abdelhamid Mehri - Constantine 2

T H È S E

pour obtenir le titre de

Docteur de l'Université Pierre et Marie Curie

et

Docteur de l'Université Abdelhamid Mehri

Spécialité : Informatique

Présentée et soutenue par
Ahmed-Chawki Chaouche

Une approche multi-agent pour la conception de systèmes d'intelligence ambiante : Un modèle formel intégrant planification et apprentissage

---

Thèse dirigée par
Amal El Fallah Serghrouchni (Directrice de thèse)
Djamel Eddine Saïdouni (Directeur de thèse)
Jean-Michel Ilié (Encadrant)

Soutenue publiquement le 14 Mai 2015

Devant le jury composé de :

Amal El Fallah Serghrouchni	Directrice de thèse	Professeur à l'Université Pierre et Marie Curie - Paris 6
Djamel Eddine Saïdouni	Co-directeur de thèse	Professeur à l'Université Abdelhamid Mehri - Constantine 2
Jean-Michel Ilié	Encadrant	Maitre de conférences hors classe à l'Université Paris Descartes
Zizette Boufaida	Rapporteur	Professeur à l'Université Abdelhamid Mehri - Constantine 2
Salima Hassas	Rapporteur	Professeur à l'Université Claude Bernard - Lyon 1
Maria Potop-Butucaru	Examinateur	Professeur à l'Université Pierre et Marie Curie - Paris 6
Okba Kazar	Examinateur	Professeur à l'Université Mohamed Khider - Biskra

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abdelhamid Mehri - Constantine 2
Faculté des Nouvelles Technologies de l'Information & de la Communication

en cotutelle avec

Université Pierre et Marie Curie - Paris 6

Année : 2015
N° d'ordre : .../.../.....
N° de série : .../.../.....

T H È S E

pour obtenir le titre de

Docteur de l'Université Abdelhamid Mehri

et

Docteur de l'Université Pierre et Marie Curie

Spécialité : Informatique

Une approche multi-agent pour la conception de systèmes d'intelligence ambiante : Un modèle formel intégrant planification et apprentissage

---

Présentée et soutenue par

Ahmed-Chawki Chaouche

Soutenue le 14 Mai 2015 devant le jury composé de :

Zizette Boufaida	Présidente	Professeur à l'Université Abdelhamid Mehri - Constantine 2
Djamel Eddine Saïdouni	Rapporteur	Professeur à l'Université Abdelhamid Mehri - Constantine 2
Amal El Fallah Seghrouchni	Co-rapporteur	Professeur à l'Université Pierre et Marie Curie - Paris 6
Maria Potop-Butucaru	Examinateur	Professeur à l'Université Pierre et Marie Curie - Paris 6
Salima Hassas	Examinateur	Professeur à l'Université Claude Bernard - Lyon 1
Okba Kazar	Examinateur	Professeur à l'Université Mohamed Khider - Biskra
Jean-Michel Ilié	Invité	Maitre de conférences hors classe à l'Université Paris Descartes

Débuté en Décembre 2011, cette thèse a été préparée dans le cadre d'une cotutelle, au sein du Laboratoire d'Informatique Paris 6 (LIP6) de l'Université Pierre et Marie Curie, France et le laboratoire Modélisation et Implémentation des Systèmes Complexes (MISC) de l'Université Abdelhamid Mehri Constantine 2, Algérie.

Remerciements

Ce modeste travail est le fruit d'une collaboration constructive entre l'Université Pierre et Marie Curie - Paris 6 (UPMC), et l'Université Abdelhamid Mehri - Constantine 2 (UAM), sous forme d'une cotutelle coté français dirigée par le professeur Amal El Fallah Seghrouchni et encadrée par le docteur Jean-Michel Ilié, chercheurs au Laboratoire d'informatique, Paris 6 (LIP6) de l'UPMC et coté algérien le professeur Djamel Eddine Saïdouni, chercheur au laboratoire d'informatique MISC de l'UAM, qui ont veillé au bon déroulement de la thèse jusqu'à son aboutissement.

Tout d'abord, je tiens à remercier en premier lieu mes deux rapporteurs Zizette Boufaida, professeur à l'Université Abdelhamid Mehri - Constantine 2, et Salima Hassas, professeur à l'Université Claude Bernard - Lyon 1, pour avoir évalué mon manuscrit.

Je remercie également Maria Potop-Butucaru, professeur à l'Université Pierre et Marie Curie - Paris 6 et Okba Kazar, professeur à l'Université Mohamed Khider - Biskra, d'avoir accepté de venir évaluer mon travail en tant que membre de mon jury.

Ma gratitude est grande envers mes directeurs de thèse le professeur Amal El Fallah Seghrouchni et le professeur Djamel Eddine Saïdouni pour avoir accepté de diriger ce travail, malgré leurs innombrables tâches. Je tiens à les remercier vivement pour leurs précieux conseils et leurs critiques constructives qui ont orienté cette recherche et pour toutes les facilités administratives ainsi que les démarches qui ont permis le bon déroulement de la cotutelle.

Nos remerciements s'adressent particulièrement au Docteur Jean-Michel Ilié, pour l'accueil, le suivi, les aides et encouragements apportées durant l'élaboration

de cette thèse, en mettant à notre disposition tous les moyens nécessaires à la réalisation de ce travail. Qu'il trouve ici l'expression de toute notre reconnaissance, en lui exprimant nos vifs remerciements.

Cette recherche doit également à ma famille ; ma mère, mon père, mon frère, ainsi que mon épouse, pour leur amour et encouragements qu'ils m'ont prodigués, qu'ils trouvent, ici, toute ma reconnaissance et l'expression de ma profonde gratitude.

Je pense vivement à tous ceux qui ont contribué, de près ou de loin, à l'accomplissement de cette thèse. Que tous, ici, reçoivent mon profond respect et ma reconnaissance.

Résumé

Ce travail présente une architecture logicielle concrète dédiée aux besoins et caractéristiques des systèmes d'Intelligence Ambiante (AmI). Le modèle comportemental proposé, appelé Higher-order Agent (HoA), capture simultanément l'évolution de l'état mental de l'agent ainsi que l'état de son plan d'actions. Les expressions du plan sont écrites et composées en utilisant un langage algébrique formel, nommé AgLOTOS. Les plans sont construits automatiquement et à la volée, comme un système de processus concurrents, déduits des intentions de l'agent et de ses préférences d'exécution.

Basé sur une sémantique de plans et d'actions concurrentes, un service de guidance est aussi proposé afin d'assister l'agent dans le choix de ses prochaines exécutions. Cette guidance permet d'améliorer la satisfaction des intentions de l'agent au regard des plans concurrents possibles et en fonction du contexte actuel de l'agent. La "localité" et le "temps" étant considérés comme des informations contextuelles clés dans l'activité de l'agent, nous les prenons en compte au travers de deux fonctions utilitaires originales conçues à partir des expériences des exécutions d'action et pouvant être combinées suivant les préférences stratégiques de l'agent.

La structure compositionnelle des expressions AgLOTOS est mise à profit pour permettre des révisions ciblées du plan de l'agent, Les révisions des sous-plans sont donc réalisées automatiquement en fonction des mises à jour apportées aux intentions, tout en maintenant la consistance du comportement de l'agent. Un cas d'étude est développé afin de montrer comment l'agent peut agir, même s'il subit des changements inattendus de son contexte, en fonction de ses expériences passées qui révèlent certains cas d'échecs.

Mots-clés : Intelligence ambiante, agents BDI, langage de planification formelle, sémantique opérationnelle, révision dynamique des plans, consistance des intentions, guidance spatio-temporelle.

Abstract

This work presents a concrete software architecture dedicated to ambient intelligence (AmI) features and requirements. The proposed behavioral model, called Higher-order Agent (HoA) captures the evolution of the mental representation of the agent and the one of its plan simultaneously. Plan expressions are written and composed using a formal algebraic language, namely AgLOTOS, so that plans are built automatically and on the fly, as a system of concurrent processes. Due to the compositional structure of AgLOTOS expressions, the updates of sub-plans are realized automatically accordingly to the revising of intentions, hence maintaining the consistency of the agent.

Based on a specific semantics, a guidance service is also proposed to assist the agent in its execution. This guidance allows to improve the satisfaction of the agent's intentions with respect to the possible concurrent plans and the current context of the agent. Adopting the idea that "location" and "time" are key stones information in the activity of the agent, we show how to enforce guidance by ordering the different possible plans. As a major contribution, we demonstrate two original utility functions that are designed from the past-experiences of the action executions, and that can be combined accordingly to the current balance policy of the agent. A use case scenario is developed to show how the agent can act, even if it suffers from unexpected changes of contexts, it does not have many experiences and whose past experiences reveals some failure cases.

Keywords: Ambient intelligence, BDI agent, formal planning language, operational semantics, dynamic plan revision, planning consistency, spatio-temporal guidance.

ملخص

إنّ هذه الأطروحة تعرض هندسة برمجيات مخصصة لتجسيد إحتياجات وخصائص الذكاء الإصطناعي المحيطي (Ambient Intelligent). إنّ النموذج السلوكي المقترح لوكيل ما، المسمّى Higher-order Agent، يجسّد تطوره الفكري وخطة عمله. بالفعل تتم كتابة عبارة الخطة وتألفيها بإستخدام اللغة الجبرية المنهجية، AgLOTOS، وبالتالي يتم بناء الخطط تلقائيا وبطريقة ديناميكية كعمليات في حالة تنافس. وعلاوة على ذلك، نظرا للهيكل التركيبي للعبارات AgLOTOS، مراجعة خطة الوكيل تطبّق أوتوماتيكيا إستنادا إلى التحديثات الواقعة على مجموعة نواياه وذلك للحفاظ على تناسق سلوكه مع فكره.

بناء على دلالات واضحة المعالم، نقترح أيضا خدمة توجيه متاحة لمساعدة الوكيل لتنفيذ أفعاله المستقبلية. هذا التوجيه يسمح بتحسين تحقيق نوايا الوكيل بدلالة خططه الممكنة وسياقه الحالي. تبنّي فكرة أنّ "الموقع" و "الوقت" هما المعلومتان السّياقيتان الرئيسيتان في نشاط الوكيل، تُبيّن لنا كيفية تعزيز توجيهه بجدولة خططه المحتملة. لهذا تمّ تصميم دالتين بناء على تجارب الأعمال السابقة ودمجهما وفقا لسياسة الوكيل المتعلّقة بالتّوازن بين أداء خططه ومدة تنفيذها في السياق المحتمل.

تمّ تطوير دراسة حالة لإظهار كيف يُمكن للوكيل أن يتصرّف، حتى في حالة تعرضه لتغيرات غير متوقعة في سِياقه، وِفقا لتجاربه الماضية التي تكشف عن بعض حالات الفشل التي يمكن وقوعها.

الكلمات المفتاحية: الذكاء الإصطناعي المحيطي، الوكيل BDI، لغة تخطيط منهجية، دلالات تنفيذية، مراجعة ديناميكية للخطط، إتّساق نوايا الوكيل، التوجيه الزماني-المكاني.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

CHAPITRE 1

INTRODUCTION

Les systèmes d'intelligence ambiante (dit AmI[1]) peuvent être vus comme des systèmes distribués où l'environnement contextuel, des entités actives qui les composent, varie de façon imprévisible. L'étude de tels systèmes est, de nos jours, inéluctable car elle permet d'envisager la construction de systèmes réellement robustes aux changements continus de l'environnement.

L'intelligence embarquée dans les systèmes AmI peut prendre en considération les utilisateurs et les assister afin de satisfaire leurs intentions et leurs préférences changeantes au cours du temps. Les industriels ne s'y sont pas trompés, en proposant des projets informatiques d'envergure, destinés à assister ou à aider les utilisateurs dans leurs espaces de vie. Ainsi, la domotique qui était historiquement une façon d'automatiser le fonctionnement d'une maison a évolué pour être au service des personnes résidentes, au cours de leurs différentes activités [Makonin et al., 2013].

Les limites technologiques étant repoussées toujours plus loin, le concept même de système d'information ou d'ordinateur change : à l'origine, l'activité est exclusivement centrée sur l'utilisateur et son poste de travail, désormais l'informatique

[1] AmI : Ambient Intelligence

devient un outil au service d'une intelligence ambiante. D'immense projets industriels sont lancés et certains d'entre eux sont déjà opérationnels. Ils ne se limitent pas à une assistance intelligente personnelle mais se déclinent à toutes les échelles d'espaces de vie des individus : smart-room (pièce), smart-home (maison), smart-campus (campus), smart-city (ville). La notion même de smart-world a été envisagée [InTech, 2011].

La conception de nouveaux systèmes ambiants va de paire avec les avancées vertigineuses de la technologie. L'Internet des objets peut parfois paraître comme un sujet abstrait [Weber et Weber, 2010]. Et pourtant, nombre d'applications concrètes existent déjà, intégrant de plus en plus d'objets électroniques du quotidien dans l'environnement numériques de l'utilisateur. À cet effet, l'édition 2011 de Smart-World a montré nombre d'acteurs – industriels, fabricants, éditeurs, chercheurs, étudiants – proposant leurs projets innovants : miroir interactif tactile connecté à Internet, imprimantes et pèse-personne connectés, robotique de service et humanoïdes domestiques, immotique et bâtiments intelligents.

Les limites applicatives de l'informatique ambiante ont été repoussées encore plus loin par l'avènement de la nano-informatique. L'humain lui même peut devenir *un objet connecté*. Ainsi pour prévenir des problèmes de santé, il est assisté au travers d'objets et d'ordinateurs parfois minuscules et connectés, qui lui sont attachés.

Tout notre mode de vie au quotidien en est bouleversé

Problématique

Les systèmes AmI, que nous considérons, sont très proches de leurs utilisateurs finaux. À cette fin, ils sont constitués d'objets connectés dont certains sont directement portés par les utilisateurs et considérés comme autant d'agents logiciels. De plus, ils permettent d'envisager des actions adaptées à chaque utilisateur et l'interaction personnalisée avec lui, s'agissant de prendre en considération ses intentions et ses préférences.

Les agents du système AmI, appelé agents AmI, doivent être construits pour être *pervasifs*, au sens de pouvoir agir sans de nécessaires interventions des utili-

sateurs. Cela permet de déplacer les informations de la périphérie vers le centre de l'attention humaine.

Dans un environnement en perpétuelle évolution, le problème majeur des agents ambiants est la reconnaissance de leurs contextes environnementaux. C'est une condition nécessaire pour concevoir des agents pro-actifs ayant un comportement rationnel. En effet, plusieurs travaux ont abordés la caractérisation du contexte de l'agent vis-à-vis de son environnement [Henricksen et Indulska, 2006; Olaru et al., 2013]. Selon Chen et Kotz [2000], cinq catégories de contextes ont été distinguées :

- Contexte de calcul : caractérise la disponibilité des ressources, la qualité du réseau et les informations relatives au calcul de l'agent,
- Contexte d'utilisateur : ce type de contexte concerne le profil de l'utilisateur, sa localité spatiale, son voisinage de proximité, ses relations sociales et même ses activités,
- Contexte physique : représenté par la température, les ressources énergétiques, les conditions de la circulation, les niveaux de bruit, etc.,
- Contexte du temps : concerne par exemple l'heure dans une journée, la date de l'année, les périodes et les durées des activités etc.,
- Contexte historique : lorsque les contextes cités précédemment sont enregistrés sur des périodes de temps, cela donne lieu à un contexte historique.

À travers cette recherche, nous nous intéressons à deux types de contexte qui nous semblent capitaux pour la modélisation des agents ambiants à savoir ; le contexte spatial et le contexte temporel. Proche de la vision de l'utilisateur, ces deux types de contexte représentent les paramètres les plus sûrs, stables, et souvent ceux auxquels on peut faire confiance. De plus, la combinaison entre l'espace et le temps permettra une abstraction des autres types de contextes. En effet, être au bon endroit et au bon moment suffit souvent pour réaliser certaines activités.

De nouveaux systèmes nécessitent de nouveaux besoins d'analyse comportementale

L'autonomie est un aspect fondamental des technologies automatisées d'assistance aux utilisateurs. La conception d'un agent revêt des exigences particulières imposées par les raisons suivantes :

- par rapport aux autres agents, celui-ci doit préserver la confidentialité des données de l'utilisateur assisté. Dans le cas contraire, la confiance perdue de l'utilisateur rendrait obsolète toute idée d'assistance,
- la connectivité du réseau peut s'avérer totalement imparfaite, induisant parfois l'impossibilité totale de s'appuyer sur un service distant ou un agent voisin,
- suivant certaines applications où la compétition l'emporte sur la coopération, l'autonomie s'avère préférable, du moins partiellement.

La mise en œuvre de l'autonomie de l'agent ambiant est fondée sur des mécanismes fondamentaux, variés, complexes et interactifs, en particulier, l'intelligence, la mémoire des expérience passées et la planification des actions à réaliser. Tous ces mécanismes dépendent des informations contextuelles, lesquels les rend en système ambiant, d'autant plus sensibles et complexes à appréhender, tels que :

1. **L'intelligence**, où la façon de déduire et de raisonner sur ses connaissances/croyances, a constitué le sujet de nombreux travaux.

   D'un point de vue théorique, le modèle BDI (Belief-Desire-Intention) [Rao et Georgeff, 1992] est conçu pour la modélisation du comportements des agents intelligents. Les *croyances* (Beliefs) de l'agent dépendent principalement du contexte et de son évolution. Les *désirs* (Desires) sont ainsi formés sur la base des croyances, et de logiques adaptées représentant les motivations de l'agent [Bratman, 1987; Cohen et Levesque, 1990]. En fonction de certains critères relevant de consistance ou au contraire de conflits entre les désirs, ou de critères plus abstraits tels que des préférences, l'agent s'engage sur certains désirs dans un mécanisme complexe, désirs résultant en des *intentions* (Intentions) à réaliser.

   D'un point de vue pratique, l'architecture BDI a été implémentée et utilisée avec succès dans beaucoup d'applications, parmi elles celles qui sont basées sur PRS (Practical Reasoning System) [Georgeff et Ingrand, 1989], on

retrouve le système du contrôle du trafic aérien OASIS à l'aéroport de Sydney en Australie [Ljungberg et Lucas, 1992] et le système du management de business SPOC (Signal Point of Contact) [Georgeff et Rao, 1996].

2. **La mémoire des expériences passées** peut contribuer à la qualité de services de l'agent, l'heuristique étant que l'on apprend positivement des échecs et des réussites passées. Pour certaines applications, un agent peut apprendre tout autant de lui même que des autres agents. Ainsi, dans le travail de Guerra-Hernandez et al. [2005] la qualité de travail de l'agent au sein d'un système coopératif, est largement amélioré par le partage d'information.

3. **La planification des actions** émerge de l'idée que la réalisation des intentions est concrétisée sous formes d'actions coordonnées, et doit tenir compte des dépendances entre les actions. La planification nécessite toutefois une certaine stabilité de l'environnement. Elle apporte, par contre, une vision à long terme du comportement de l'agent, qui peut être mise au profit de l'utilisateur sous forme d'une certaine *guidance contextuelle*. Plusieurs écoles se distinguent, incluant même la non planification :

    - *Les modèles d'action :* Les approches d'exécution, uniquement fondées sur un modèle d'action, sont souvent l'apanage d'agents très réactifs à leur contexte. Munis de capacité d'apprentissage, ces agents peu intelligents, sont capables d'adaptation aux changements de contexte. Dans Guivarch et al. [2013] par exemple, les applications ambiantes de domotique sont automatisées en fonction de caractéristiques environnantes et du comportement de l'utilisateur.
    - *Les modèles généraux de plans* adressent notamment les agents BDI, pour lesquels chaque plan reflète la façon de réaliser une intention de l'agent. Il s'agit alors de sélectionner le plan le plus adapté, parmi ceux qui se présentent. Dans [Nunes et Luck, 2014], cette sélection est caractérisée à la fois de manière structurelle (contribution d'un plan à différents "softgoal", comme la sécurité, le coût, ou la performance) et contextuelle (selon des préférences de l'utilisateur).
    - *Les modèles compositionnels de plans* correspondent à des agents BDI plus structurés, où les plans sont raffinés en sous plans. La structure hiérarchique induite pour le plan de l'agent permet d'envisager des traite-

ments plus concrets avec l'espoir d'une meilleure sélection générale du plan. Dans [Singh et al., 2010], pour une intention, la hiérarchie est issue d'un raffinement des buts en plans alternatifs de réalisation et de plans en sous buts à investiguer sous forme de (sous) plans. Les plans alternatifs auront l'avantage de renforcer l'autonomie et la robustesse de l'agent, quand certains plans ne seront pas réalisables dans le contexte considéré par l'agent. Cette approche, suppose souvent l'existence, a priori, de plans conçus lors de la spécification de l'agent, et inscrits dans une bibliothèque de plans opérationnels pour celui-ci.

Vers un modèle de planification des agents ambiants basé sur une architecture BDI

Approche de la thèse

Cette thèse poursuit l'objectif général de pourvoir les agents intelligents d'un mécanisme formel leur permettant d'adapter leur comportement en toute consistance en fonction de leur intentions, en réponse aux changements de contexte de toute nature. Face à l'impossibilité de définir, a priori, un système multi-agents, qui intègre l'ouverture et les changements de l'environnement, le parti sera pris de développer une théorie centrée sur l'agent. Il restera à l'intelligence de l'agent ambiant de se munir et exploiter des mécanismes de concertation avec les utilisateurs et de coordination avec les autres agents ambiants, ce qui définira a posteriori le système dynamique.

Le cycle de vie d'un agent ambiant, qui est proposé dans cette recherche, repose de façon cyclique, sur les trois points successifs suivants :

- un système de production automatique de plans, permettant de réaliser les intentions de l'agent en fonction du contexte ambiant de l'agent,
- un modèle d'exécution concurrente des actions des plans, permettant de sélectionner le meilleur plan d'exécution pour l'agent, tout en tenant compte des préférences de l'utilisateur,

- un retour de l'exécution des actions, permettant à l'agent d'opérer la révision de ses intentions et donc de ses plans.

Organisation du manuscrit

Ce manuscrit est organisé en deux parties distinctes.
La première est consacrée à un état de l'art sur les différents concepts de l'intelligence ambiante afin de permettre une bonne compréhension de notre contribution scientifique. Cette partie est composée de deux chapitres :

- Le chapitre 2 permet de donner au lecteur une vue d'ensemble sur les systèmes ambiants en mettant en évidence les agents et les systèmes multi-agents. L'état de l'art qui est présenté, traite les agents ambiants que nous considérons dans notre recherche ainsi que les travaux qui s'intéressent à la sensibilité au contexte des agents dans un environnement ambiant. Ensuite, pour prendre en compte l'intelligence de ces agents, nous présentons le paradigme BDI avec ses concepts de croyances, des désirs et d'intentions.
- Le chapitre 3 est consacré à la littérature qui s'intéresse à la spécification formelle des systèmes concurrents et mobiles. Nous décrivons particulièrement le langage algébrique LOTOS que nous étendrons dans le chapitre qui suit afin de spécifier le comportement des agents ambiants. Une discussion est engagée pour permettre de discerner les limites des langages actuels de spécification formelle dans le cadre de la planification des agents ambiants.

La seconde partie se veut une présentation des contributions de cette recherche. Elle est structurée en trois chapitres :

- Après une présentation générale de l'architecture logicielle de l'agent dans le chapitre 4, nous démontrons comment les évolutions mentales et comportementales de l'agent sont capturées par un modèle d'ordre supérieur appelé *Higher-order Agent (HoA)*. Dans ce contexte, nous proposons un langage algébrique, appelé *AgLOTOS*, consacré à la spécification et la construction du plan. Nous montrons aussi comment la sémantique modulaire d'AgLOTOS permet la révision des plans dynamiquement et à la volée, afin d'assurer

la cohérence de ces derniers avec les intentions de l'agent lorsqu'elles sont mises à jour.

- Le chapitre 5 introduit les techniques de guidance que nous proposons. Dans un premier temps, la sémantique d'AgLOTOS est enrichie afin de produire une structure appelée *Contextual Planning System (CPS)*, capturant les différentes traces d'actions possibles au regard des plans de l'agent. Fondé sur les principes du CPS, un mécanisme de guidance est proposé, permettant à l'agent de satisfaire de façon concurrente, le maximum de ses intentions. Cette guidance est fondée sur l'apprentissage des échecs et des succès des actions, tirant ainsi profit des expériences passées de l'agent.

- Enfin, dans le chapitre 6, nous enrichissons la guidance de l'agent par la prise en compte de son contexte spatio-temporel. Nous proposons un renforcement de la technique d'apprentissage par des stratégies temporelles adaptées. Nous désirons ainsi améliorer la performance de l'agent, sa précision ainsi que son efficacité, malgré les changements dynamiques qui peuvent intervenir dans son environnement.

Nous terminons ce manuscrit par le chapitre 7, remettant en lumière de la problématique, les contributions que nous apportons. Cela nous amènera à exposer différentes problématiques restant à résoudre, et quelques perspectives futures envisageables à court et moyen termes.

Première partie

Notions préliminaires : Systèmes d'intelligence ambiante

CHAPITRE 2

INTELLIGENCE AMBIANTE ET SYSTÈMES MULTI-AGENTS

L'intelligence ambiante (ou AmI) représente le troisième élan de l'informatique après l'ordinateur central (mainframe) et l'ordinateur personnel (PC). Depuis les années 2000, les appareils informatiques sont rendus invisibles et intégrés dans des terminaux électroniques dotés d'une capacité à communiquer entre eux, baptisés *objets connectés*.

L'AmI est une notion assez vague du système informatisé connecté avec un environnement réel et concret. Elle vise à mettre l'ordinateur à la portée du monde des hommes pour l'intégrer au point de disparaître, contrairement à la réalité virtuelle qui met l'individu à l'intérieur d'un monde créé par l'ordinateur.

L'informatisation de l'AmI dite *systèmes ambiants*, est à la convergence de trois domaines [Baskiotis et Ferey, 2003] :

- **l'informatique ubiquitaire (Ubiquitous Computing)**, consiste à intégrer les systèmes informatiques dans les objets connectés de la vie quotidienne,
- **La Communication ubiquitaire (Ubiquitous Communication)**, permet à

ces objets de communiquer non seulement entre eux mais aussi avec l'utilisateur,

- **L'interfaçage intelligent des utilisateurs (Intelligent User Interface)**, permet aux usagers de contrôler et interagir avec ces objets de manière intuitive.

Depuis quelques années, l'intelligence artificielle est intégrée à ces systèmes ubiquitaires pour les rendre intelligents, en vue d'analyser l'environnement réel et l'activité d'utilisateurs, permettant ainsi aux systèmes ambiants de réagir en contexte.

2.1 Contexte ambiant

Une des composantes de l'intelligence ambiante est une interprétation par le système de sa perception de l'environnement. Elle s'appuie sur la notion de contexte. Kokinov [1995] définit le contexte comme étant l'ensemble de toutes les entités (du système) qui influencent le comportement cognitif de l'humain (ou du système) à une occasion particulière. Il associe ensuite quatre propriétés au contexte :

- le contexte en tant que "état d'esprit",
- le contexte n'a pas de frontière clairement définie,
- le contexte est composé de l'association d'éléments pertinents,
- le contexte est dynamique.

Pascoe [1998] considère le contexte comme un concept subjectif défini par l'entité qui le perçoit. Pour une entité donnée, le contexte pourrait être sa localisation spatiale, tandis que pour une autre, il s'agira de sa localisation temporelle, ou bien de son état émotionnel.

2.2 Scénarii des applications AmI

De nombreux scénarii ont été proposés pour illustrer l'intérêt des applications ambiantes, comme par exemple dans [Ducatel et al., 2001] et [Friedewald et al., 2006]. On peut les classifier en 3 grands thèmes, entre (1) celui qui offre à l'utilisateur des

informations ambiantes et physiques, comme la température ou la qualité des ressources accessibles, (2) celui des informations et services à la personne offrant un support partiellement transparent ou intuitif, qui permet à l'utilisateur d'automatiser certaines parties de ses tâches quotidiennes, notamment les applications de la domotique [Guivarch et al., 2013], (3) celui de l'intelligence de groupes favorisant les interactions sociales entre les utilisateurs, sans nécessairement se préoccuper d'une personne [Grégoire et al., 2012]. Cette thèse entre dans le thème (2), avec l'ambition d'offrir un service rapproché de guidance planifiée, permettant à l'utilisateur d'envisager des tâches complexes dans un système ambiant, avec la possibilité de réviser certains choix en fonction des changements de contexte inattendus. Dans cet environnement, l'utilisateur est supposé pouvoir réaliser plusieurs tâches de façon concurrente et en contexte. Le lecteur notera que le thème (3) est naissant. Cela pourrait être, dans le futur, un autre sujet qui bénéficiera de la base formelle de notre travail.

Le scénario qui illustre cette thèse est relatif à un campus intelligent (*Smart-Campus*). Au travers celui-ci, nous introduirons des considérations techniques tout au long de recherche thèse, montrant comment automatiser la planification de tâches en fonction de désirs ou d'intentions spécifiés, et également comment cette planification sera modifiée en cas de changement contextuels inattendus.

Scénario "Smart-Campus"

Dans une certaine mesure, les campus peuvent être considérés comme des "petites villes". Les projets de villes intelligentes (Smart city) sont développés désormais dans le monde entier, se concentrant sur la durabilité environnementale, l'e-gouvernance, le transport, la santé, et d'autres activités du quotidien, au travers de réseaux favorisant l'échange des connaissances ou opinions.

L'objectif est donc, de concevoir un système d'agents ambiants qui assistent les utilisateurs dans leurs activités au sein d'un campus universitaire complexe. La communication du campus est soutenue par une infrastructure réseau. Chaque utilisateur est équipé par un appareil (ordiphone) embarquant une application smart-campus. Cette application prend la forme d'un agent assistant muni d'une interface graphique pour interagir avec l'utilisateur assisté. Le système d'agents mis en place est non seulement distribué mais ij est ouvert et dynamique. En effet, les agents entrent, sortent et se déplacent au gré de la volonté des utilisateurs.

Tout au long de son exploitation, le campus subit des restructurations et des modifications (bâtiments, locaux, services, chemins, ...), qui peuvent être parfois temporaires (travaux, mutation, maintenance, ...).

Les agents que nous considérons sont sensibles à leur contexte ambiant, notamment la localité spatiale dans l'espace du campus. Les agents ont la capacité de communiquer entre eux, ce qui permet des coopérations et des échanges de données au travers du système.

Le service que nous mettons, particulièrement, en valeur est un *service de guidance* permettant à un agent d'optimiser la réussite des objectifs qu'un utilisateur se donne.

Pour guider l'utilisateur dans un environnement dynamique et changeant, l'agent doit être doté de capacité d'intelligence et d'anticipation. La connaissance d'expériences passées pourrait lui être fort utile.

Il peut compter sur les capacités de l'utilisateur, notamment en le notifiant pour des déplacements ou des interactions hors système avec d'autres agents. Il sait aussi agir et prendre des décisions suivant les privilèges qui lui sont attribués, de façon autonome et pervasive. L'intervention de l'utilisateur n'est pas tout le temps requise, comme par exemple, le basculement vers le mode économie d'énergie (désactivation du WiFi, diminution de l'éclairage de l'écran) lorsque la capacité de la batterie est en deçà de 15% afin de préserver plus d'autonomie.

2.3 Caractérisation d'un SMA ambiant

Comme dans le scénario Smart-Campus de la section 2.2, un ensemble d'individus peut être représenté au niveau de l'informatique par un ensemble d'agents logiciels qui ont la capacité d'interagir entre eux et de percevoir leur environnement ambiant. Le système, ainsi composé par les agents, forme un Système Multi-Agents (SMA). Par extension, il est possible de greffer à ce système d'autres agents logiciels, notamment ceux qui font office de serveurs de services ou de données. On cite souvent quatre problématiques pour les SMA :

- la problématique de l'action s'interroge sur comment un ensemble d'agents autonomes peut agir dans un environnement partagé, et comment cet en-

vironnement interagit en retour avec les agents [Kesäiemi et Terziyan, 2011; Mili et Steiner, 2008].

- la modélisation de l'agent et de sa relation au monde ; le modèle cognitif de l'agent doit être capable de mettre en œuvre les actions qui répondent au mieux à ses objectifs. Cette capacité à la décision est liée à un *"état mental"* qui reflète les perceptions, les représentations, les croyances, ajoutées à un certain nombre de paramètres "psychiques" (désirs, tendances,...) de l'agent [Bratman, 1987; Rao et Georgeff, 1995].
- la problématique des interactions entre agents adressent des notions centrales de coordination et de coopération [Balke et al., 2014].
- la problématique de l'adaptation en termes d'adaptation du comportement de l'agent en fonction du contexte ou de son apprentissage. L'adaptation peut être purement individuelle ou partiellement collective [Guivarch et al., 2013].

Les systèmes ambiants considérés dans cette thèse font que les SMA ouvrent, en fait, des perspectives complémentaires, dont chacune d'elles constitue un axe de recherche en soi.

- **Ouverture du système :** Un nombre important des SMA considérés dans la littérature étaient des systèmes préconçus, ayant notamment un nombre fixe d'agents. Dans cette thèse, les agents peuvent entrer et sortir de l'espace alloué à leur évolution. Leur nombre effectif dans le système, à un instant donné, est donc dynamique.
- **Reconfiguration de l'environnement :** L'espace dans lequel évolue les agents et de façon plus générale les conditions contextuelles de chaque agent peuvent évoluer inopinément. Nous tenons à préciser que dans cette thèse, aucune hypothèse n'est prise en compte concernant la stabilité du système. Nous supposons, toutefois, que le système n'est pas un chaos, ce qui laisse supposer que celui-ci pourrait avoir des moments de stabilité ou de convergence, entrecoupés de changements contextuels non prévisibles et inattendus.
- **Autonomie des agents et relation avec les utilisateurs :** Les agents que nous considérons sont pro-actifs, ce que l'on peut considérer comme un critère

d'autonomie de fonctionnement. Toutefois, ils sont supposés faire assistance à un utilisateur, voire à plusieurs par extension, c'est pourquoi des interactions existent, nécessairement, entre agent et utilisateur. Dans cette thèse, de telles interactions sont prises en compte sous forme de préférences émises par l'utilisateur sur les (types) de choix qu'auraient à prendre un agent.

- **Hétérogénéité du système et relations de communication :** Dans les SMA, l'hétérogénéité des comportements ne provient pas du seul fait que les contextes environnementaux perçus par chaque agent sont différents. En fait, la question de l'hétérogénéité des agents a été posée à plusieurs reprises dans la littérature, on citera [Pérez et al., 2014].

  Dans notre recherche, nous supposons, de façon raisonnable, que les agents aient des ontologies communes afin de pouvoir interagir entre eux. Leur hétérogénéité est relative aux applications que les agents devront supporter. Face à cette complexité, celle-ci est souvent abstraite, dans la mesure où le comportement des utilisateurs peut être réduit à celui des agents. Dans cette thèse, les utilisateurs sont assimilés aux préférences qu'ils spécifient à leurs agents assistants et en retour aux notifications que ceux-ci peuvent servir aux utilisateurs.

2.4 Caractérisation des agents ambiants

Les agents qui composent un SMA ont des caractéristiques qui différent sur la complexité du système réel que les agents logiciels du SMA informatisent.

Selon Jennings et Wooldridge [1998], *"un agent est un programme informatique évoluant dans un environnement partagé et doté de comportements autonomes représentés par des actions, lui permettant d'atteindre dans cet environnement les objectifs, qui lui ont été fixés à sa conception"*.

Et d'après Ferber et Gutknecht [1998], *"un agent peut être défini comme une entité, physique ou abstraite, capable d'agir sur elle-même et sur son environnement, pouvant communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, connaissances et interactions avec les autres agents"*.

Disposant d'un ensemble de ressources, l'agent possède un ensemble de buts qu'il cherche à accomplir en exécutant ses propres plans. De plus, les agents doivent

être capables de réagir aux changements et perturbations de leur environnement.

Les agents que nous considérons satisfont les propriétés suivantes :

- **Autonomie :** l'agent agit sans l'intervention des humains ou des autres agents, et il contrôle ses actions en fonction de son état interne et de son environnement. En effet, chaque agent dispose de son propre contexte d'exécution, ses propres objectifs et son propre mécanisme de prise de décision.
- **Réactivité :** l'agent réactif est plutôt orienté comportement que connaissance, il ne répond qu'aux stimuli de son environnement.
- **Proactivité :** contrairement à un objectif dont l'action est simplement une réponse aux messages reçus, l'agent a une activité dirigée par son but.
- **Adaptation :** L'adaptabilité d'un agent implique qu'il est capable de réagir à des situations non prévues, de changer le cours de son exécution en conséquence et de mettre à jour ses connaissances par apprentissage en fonction des situations rencontrées.

Dans notre étude, nous définissons des agents AmI comme étant des entités autonomes, capables de percevoir et de capturer leurs contextes à travers des capteurs, et d'agir sur ces derniers à travers des comportements. Il est, cependant, fréquent que chaque agent du système possède un comportement totalement indépendant de celui des autres agents. Par ailleurs, la sensibilité de l'agent à son contexte nécessite que son comportement soit adapté à son contexte afin de mieux satisfaire les besoins des utilisateurs assistés.

Sensibilité au contexte

Schilit et al. [1994] ont été les premiers à introduire la notion de sensibilité au contexte, au travers de la conception d'un système d'*active map service*, proposant des informations à un utilisateur en fonction de sa localisation. Ils définissent alors la sensibilité au contexte comme la capacité des applications d'un utilisateur mobile à découvrir et à réagir aux changements survenant dans l'environnement dons lequel ils sont situés. Depuis, plusieurs définitions ont été proposées [Abowd et al., 1999; Brézillon et Brézillon, 2007; Henricksen et Indulska, 2006; Ryan et al.,

1998]. Dans cette thèse, nous optons pour celle de Korkea-Aho [2000], qui établit qu'un système est sensible au contexte s'il peut extraire, interpréter et utiliser des informations de contexte, et adapter sa fonctionnalité au contexte d'utilisation courant.

Nous pouvons remarquer que deux aspects principaux caractérisent la sensibilité au contexte. Le premier concerne la capacité à percevoir son contexte ; cette notion de perception/détection est présente dans toutes les définitions. Le second aspect porte sur la capacité à exploiter le contexte perçu. Être sensible au contexte ne consiste donc pas uniquement à percevoir son contexte, mais aussi à agir différemment en fonction de celui-ci.

La sensibilité au contexte est une question centrale dans le domaine de l'intelligence ambiante. Un comportement pro-actif et non-intrusif ne serait pas possible sans une bonne compréhension du contexte de l'utilisateur. Il s'agit donc de donner à un agent ambiant la capacité d'adapter dynamiquement sa fonctionnalité selon son contexte courant.

Les travaux qui s'intéressent à la sensibilité au contexte, dans les applications de l'intelligence ambiante, tournent généralement autour de deux aspects : d'une part, l'infrastructure permettant la capture et le traitement des informations contextuelles, d'autre part, la modélisation et le raisonnement sur ces informations contextuelles.

Dans le travail Olaru et al. [2011], un système d'espace hiérarchique a été proposé permettant de spécifier directement la localité de chaque déplacement de l'agent ainsi que d'autres éléments de son contexte. La vue globale du système est abstraite tandis que le contexte de l'agent est modélisé comme étant une structure dynamique sur lesquelles les activités de coordination sont capturées par une technique de filtrage (*Pattern-matching*). Par ailleurs, les auteurs ont mis en évidence plusieurs projets intéressants qui implémentent des systèmes AmI.

Dans la plateforme de domotique intelligente proposée par Guivarch et al. [2012], le contexte d'un utilisateur est perçu au sein de sa propre maison. L'utilisateur est assisté par un agent connecté , dont l'architecture est composée de 4 agents internes : un agent des données, un agent de l'utilisateur, un agent du contexte et enfin un agent contrôleur. De cette manière, le comportement de l'agent est capable d'évoluer en fonction des données collectées à l'aide de capteurs externes.

Par ailleurs, des techniques d'apprentissage sont appliquées, celles-ci ont démontré leurs apports dans un environnement dynamique.

2.5 Caractérisation des agents BDI

De nombreuses architectures multi-agents ont été proposées. Certaines d'entre elles, comme les machines à états finis ou les architectures motivationnelles, peuvent être très utiles pour la définition d'agents simples. Cependant, elles ne sont pas adaptées à la définition d'agents cognitifs complexes, connus pour leur pro-activité et leur anticipation, car leur pouvoir de représentation est très limité.

Le paradigme Belief-Desire-Intention (BDI) est l'une des approches majeures dédiées à la conception et à la construction des agents rationnels ainsi que des systèmes multi-agents [Cohen et Levesque, 1990; Rao et Georgeff, 1995].

Des travaux, comme [Shendarkar et al., 2008], ont montré l'intérêt d'utiliser un tel paradigme dans le contexte de la simulation. Les premiers fondements théoriques ont été décrits par les travaux de Georgeff et Ingrand [1989] : PRS[1]. Parmi les applications qui exploitent ces concepts, citons le système du contrôle du trafic aérien OASIS à l'aéroport de Sydney en Australie [Ljungberg et Lucas, 1992] et le système du management de business SPOC [Georgeff et Rao, 1996]. À notre connaissance, l'approche BDI est encore peu utilisée dans des applications concrètes, en raison de la complexité apparente des architectures proposées, par exemple JAM [Huber, 1999], au regard de la programmation intuitive (purement algorithmique) généralement admise pour spécifier le comportement des agents. Récemment, l'importance de l'approche BDI est réaffirmée au travers de plusieurs langages spécialisés, tels que Jadex [Pokahr et al., 2005], Jason [Bordini et al., 2007], 2APL [Dastani, 2008].

L'idée phare de l'approche BDI est de décrire l'état interne des agents, dit *agents BDI*, en termes d'attitudes mentales, et aussi de définir une architecture de contrôle grâce à laquelle l'agent peut sélectionner son plan [Müller, 1996]. Le mécanisme de raisonnement d'un agent BDI, déclenché par des événements perçus, se base sur les attitudes mentales de celui-ci qui sont représentées par les Croyances (Beliefs), les Désirs (Desires) et les Intentions (Intentions) :

[1]Le prototype PRS (Procedural Reasoning System) développé à Stanford Research Center

- Les croyances expriment ce que l'agent croît sur l'état courant de son environnement.
- Les désirs (ou buts) sont une notion abstraite spécifiant les préférences de l'agent. Leur caractéristique principale est qu'un agent peut avoir des désirs inconsistants et qu'il n'a donc pas à croire que ses désirs sont réalisables.
- Les intentions représentent les désirs que l'agent s'engage à réaliser. Cependant, étant limité par ses ressources, l'agent peut ne pas poursuivre tous ses désirs même si ces derniers sont consistants. Il est nécessaire alors qu'il choisisse un certain nombre de désirs pour lesquels il s'engage à réaliser.

L'agent a une représentation explicite de ses croyances, désirs et intentions. On dénote par B l'ensemble des croyances de l'agent, par D l'ensemble de ses désirs, et par I l'ensemble de ses intentions.

Les ensembles B, D et I peuvent être représentés au moyen de divers modèles de représentation de connaissances, par exemple en utilisant la logique des prédicats du premier ordre, la logique d'ordre supérieur, le modèle des règles de production, ou bien comme de simples structures de données [Florea et al., 2002].

Afin de bien comprendre la théorie du paradigme BDI proposée pour la première fois par Bratman [1987], voici un scénario réaliste qui l'explique :

"L'agent Pierre a la croyance que, si quelqu'un passe son temps à étudier, cette personne peut faire une thèse de doctorat. En plus, Pierre a le désir de voyager beaucoup, de faire une thèse de doctorat et d'obtenir un poste d'assistant à l'université. Le désir de voyager beaucoup n'est pas consistant avec les deux autres et Pierre, après réflexion, décide de choisir, parmi ces désirs inconsistants, les deux derniers. Comme il se rend compte qu'il ne peut pas réaliser ses deux désirs à la fois, il décide de faire d'abord une thèse de doctorat. En ce moment, Pierre a l'intention de faire une thèse et, normalement, il va utiliser tous ses moyens pour y parvenir. Il serait irrationnel de la part de Pierre, une fois sa décision prise, d'utiliser son temps et son énergie, notamment ses moyens, pour voyager autour du monde. En fixant ses intentions, Pierre a moins de choix à considérer car il a renoncé à faire le tour des agences de voyage pour trouver l'offre de voyage qui le satisferait au mieux."

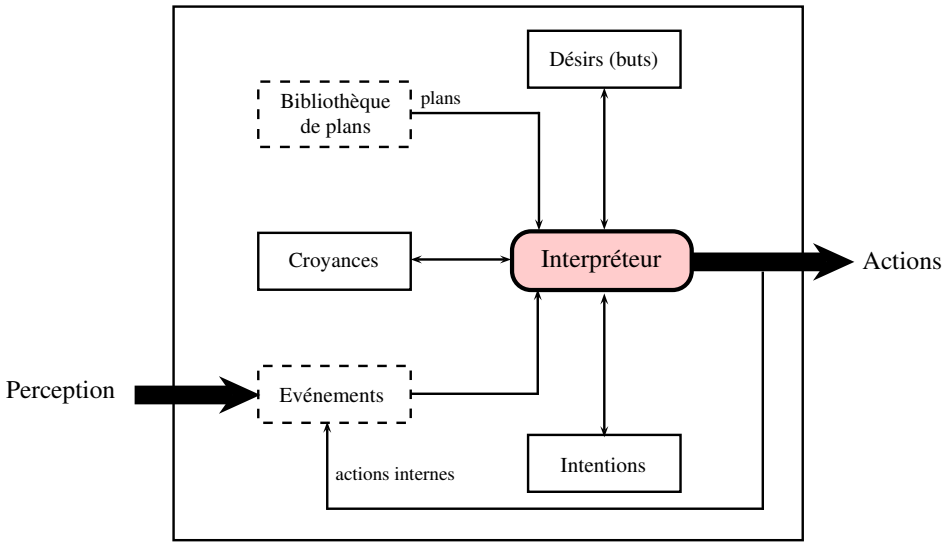

FIGURE 2.1: L'architecture d'un agent BDI

2.5.1 Architecture des agents BDI

Inspiré de [d'Inverno et al., 1997], la figure 2.1 illustre les principaux composants de l'architecture BDI. En effet, plusieurs travaux ont détaillé le fonctionnement de l'interpréteur BDI ; comme par exemple [Bratman et al., 1988; Pollack et Horty, 1999].

Le mécanisme de raisonnement de l'agent, basé sur ses Croyances (B), ses Désirs (D) et ses Intentions (I), est déclenché par les événements (Evt) perçus par celui-ci lui permettant de produire des plans d'actions consistants. Ces plans sont extraits à partir d'une *bibliothèque de plans partiels* (*Library of plans*), appelée $LibP$.

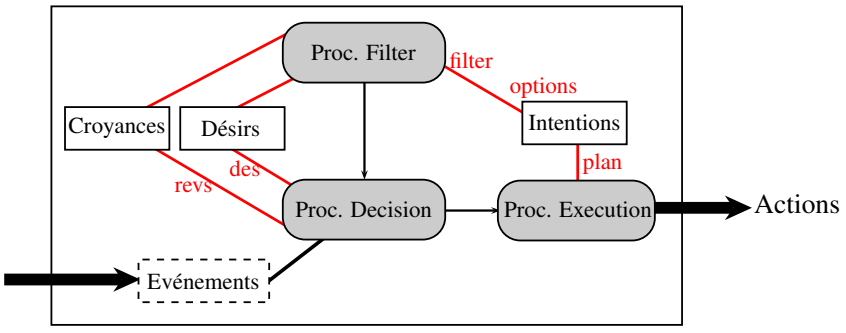

FIGURE 2.2: Une vue fonctionnelle de l'interpréteur BDI

La figure 2.2 illustre le schéma de fonctionnement de l'interpréteur associé à l'agent BDI, les fonctions suivantes décrivent son mécanisme de raisonnement :

- $revs : 2^B \times Evt \rightarrow 2^B$ est la fonction de révision des croyances de l'agent déclenchée à partir des nouvelles perceptions,
- $des : 2^B \times 2^D \times 2^I \rightarrow 2^D$ est la fonction responsable de la mise à jour des désirs de l'agent si ses croyances ou ses intentions changent afin de maintenir la consistance des désirs sélectionnés,
- $filter : 2^B \times 2^D \times 2^I \times LibP \rightarrow 2^I$ est la fonction qui décide des intentions à poursuivre parmi les options possibles, prenant en compte les nouvelles opportunités.
- $options : I \times LibP \rightarrow P$ est la fonction qui associe à chaque intention de l'agent, des plans partiels dédiés en utilisant la bibliothèque de plans $LibP$.
- $plan : 2^I \rightarrow P$ est la fonction qui permet de construire un plan exécutable à partir de plans partiels en utilisant la fonction $options$.

L'algorithme 1, conçu par Michael Wooldrige, décrit le mécanisme de raisonnement de l'agent BDI.

Algorithme 1 Mécanisme de raisonnement de l'agent BDI

1: **repeat**
2: obtenir nouvelles perceptions p ;
3: $B = revc(B, p)$;
4: $I = options(D, I)$
5: $D = des(B, D, I)$
6: $I = filtre(B, D, I)$
7: $P = plan(B, I)$
8: exécuter le plan P
9: **until** (vrai)

2.5.2 Les langages BDI

Dans la littérature, il existe de nombreux langages de programmation consacrés aux agents BDI [Hindriks, 2014], mettant en évidence différents aspects et modules de développement au sein de l'agent, tels que les buts, la planification pour

les agents et les organisations d'agents au niveau des SMA. Citons à cet effet, Jadex [Pokahr et al., 2005], Jason [Bordini et al., 2007], 2APL [Dastani, 2008] et JIAC [Lützenberger et al., 2013].

En 2006, les approches BDI centrées-agent émergent dans le but d'intégrer dans les comportements attendus, la dynamicité de l'environnement. En particulier, le modèle formel *CanPlan* proposé dans [Sardina et al., 2006], étend le langage Can de façon à augmenter la spécification des actions dans les plans, par la considération d'attitudes mentales de type BDI.

Les paradigmes mis en œuvre dans les langages les plus anciens sont repris [Hindriks, 2014].

- Un plan est conditionné par la spécification formelle et explicite de contextes d'application.
- Les plans sont décomposés en buts et les buts en plans.
- C'est le processus mental de l'agent qui manipule dans son ensemble les croyances, les désirs, les intentions et même les plans.

Tout semble fait pour que le programmeur ait à sa disposition un outil formel, de programmation globale des agents. Le comportement de l'agent est cependant limité aux connaissances que le concepteur peut inculquer au moment du développement de l'agent, en particulier les contextes d'application doivent être connus à l'avance. Ceci est une limitation fondamentale au regard des systèmes ambiants.

2.6 Planification des agents BDI

Les premières conceptions de systèmes distribués sous forme de SMA n'incluaient aucune anticipation dans le processus de planification, c'est pourquoi des approches BDI centrées-agent ont été proposées à partir des années 2000, pour faire face à la dynamicité des environnements dans lesquels sont plongés les agents.

Dans [Sardina et al., 2006], un modèle hiérarchique appelé Hierarchical Task Network (HTN) est proposé en cas d'échec d'un plan de l'agent, pour sélectionner le prochain plan à exécuter. L'approche d'exécution est en effet séquentielle. Le

prochain plan sélectionné sera basiquement l'un de ceux qui respectent les conditions contextuelles de l'agent. Ce sera soit une alternative du plan échoué soit un autre plan de la hiérarchie Goal-Plan. Dans [Shapiro et al., 2012], la sélection d'un plan alternatif a été spécifiée sur la base des échecs et succès des expériences d'exécution passées. Ainsi, il ne suffit pas qu'un plan soit applicable pour un contexte d'agent donné pour être sélectionner, il faut en plus qu'il ait de fortes chances d'être applicables, au regard de l'expérience. Toutefois, une sélection satisfaisante n'est obtenue qu'avec un nombre considérable d'expériences. Dans [Waters et al., 2014a], une approche complémentaire a été définie pour estimer la propagation des échecs dans la hiérarchie Goal-Plan, afin de sélectionner plus globalement du prochain plans à exécuter.

De façon orthogonale, la consistance mutuelle de plans d'un agent a été étudiée dans un but d'exécutions concurrentes. Le planificateur GraphPlan de [Meneguzzi et al., 2007] reprend le travail planification concurrente de [Blum et Furst, 1995] pour l'appliquer aux agents BDI. Une étude des conflits d'exécution des actions composants les plans est réalisée, sous contraintes de préconditions d'exécution des actions et d'effets de ces exécutions, Ainsi, le planificateur GraphPlan est capable de construire efficacement un plan global comme un flux d'actions réalisant à un sous-ensemble de désirs (buts), qui pourraient être exécutés en concurrence. Toutefois, GraphPlan ne traite que certains cas d'ordonnancements entre les actions, puisque l'exécution concurrente proposée suit une approche de pas temporels synchrones. Il n'est pas dit qu'une approche asynchrone rendent un résultat inférieur en termes de buts atteints, or les approches synchrones et asynchrones sont souvent incomparables.

2.7 Synthèse et discussion

Au cours de la dernière décennie, plusieurs projets ont été proposés traitant des systèmes AmI. En conjonction avec ces projets, un défi majeur était de comprendre comment les agents AmI pouvaient interagir et raisonner dans un environnement dynamique, un besoin important s'en est senti et qui est de modéliser la sensibilité de l'agent envers son contexte, ce qui a été abordé récemment dans [Guivarch et al., 2012; Olaru et al., 2013; Pi et al., 2013].

Dans les SMA, la programmation des agents s'est beaucoup concentrée sur

des systèmes dont la spécification formelle était un préalable à toutes démarches de programmation et de modélisation, avec un niveau d'abstraction permettant d'appréhender le système sous la forme unique d'un système d'agents. Les notions d'assistance aux utilisateurs et de guidance par les agents, n'ont été prises en compte qu'au travers du prisme de cette programmation.

A notre connaissance, il n'existe pas de travaux fondés sur les SMA, prenant en compte l'ensemble des critères d'ouverture et de dynamicité environnementale. Dans cette thèse, nous considérons comme un fait acquis, de tels environnement d'exécution. Toutefois, de façon raisonnable dans un monde informatisé, nous supposerons que les informations de terrains, comme la géo-localisation ou le voisinage des agents, sont accessibles à la volée et fiables. La proximité de l'utilisateur nous conduira à concrétiser les capacités abstraites conférées généralement de façon abstraite aux agents, comme l'autonomie et la mobilité.

L'architecture de raisonnement des agents BDI dans les SMA supporte, de façon rationnelle, la dynamicité des comportements nécessaires à l'agent. Ces agents sont donc de très bons candidats pour le développement informatique des systèmes ambiants, même si certaines limites sont généralement supposées, comme une spécification prédéfinie des objectifs et un support de plans d'exécution préétablis, sous forme de bibliothèques, pour la réalisation des intentions de l'agent. Cependant, du fait des changements de contexte qui peuvent être inattendus, voire inconnus, il apparaît nécessaire de reconsidérer les tâches de planification et de guidance des agents pour les adapter au mieux aux systèmes ambiants. Avant cette thèse, aucun travail ne montrait comment guider un agent en fonction de l'ensemble de ses intentions et des préférences de l'utilisateur. Aucun travail actuel ne montre comment réviser un plan en contexte à la volée, dans un système ambiant.

Chapitre 3

Modélisation formelle et langages de spécification

Ce chapitre introduit différents modèles algébriques de description formelle pouvant être utilisés pour spécifier le comportement des agents ambiants. Nous commençons par exposer les formalismes dédiés à la spécification des systèmes distribués et mobiles, tels que CCS, le π-calcul, le calcul des ambiants et LOTOS. À la fin de ce chapitre, une discussion sera prononcée sur les limites de ces modèles par rapport aux besoins des agent ambiants que nous considérons. Néanmoins, grâce à ses opérateurs, LOTOS pourrait être utilisé pour la spécification des plans d'actions de l'agent. Les (sous-)plans de celui-ci pourraient être vus comme des processus concurrents capables de se synchroniser. Nous donnons une description plus détaillée à LOTOS dont nous étudierons les propriétés et lequel nous étendrons par la suite.

Pour la première fois, une définition d'une algèbre, décrivant le comportement d'un système concurrent, a introduit le concept d'agents (ou de processus). Ces derniers peuvent être composés en utilisant des opérateurs de composition séquentielle, de choix, parallèle, etc. Par la suite, les mots processus et agent ont été utilisés indifféremment dans la littérature. Le mot processus signifie une exécu-

tion séquentielle d'instructions[1].

Le comportement d'un système est généralement constitué de processus et de données où les processus sont des mécanismes de contrôle pour la manipulation des données. Ce comportement tend à être composé de plusieurs processus qui sont exécutés simultanément lesquels s'échangent des données afin d'influencer le comportement.

3.1 Algèbres de processus

Une algèbre de processus se concentre sur la spécification et la manipulation des termes de processus induits par un ensemble d'opérateurs et des règles syntaxiques pour spécifier un processus utilisant des composants simples. La plupart des algèbres de processus contiennent des opérateurs de base pour construire des processus complexes. Une sémantique opérationnelle structurelle est utilisée pour donner formellement à chaque terme "processus" une représentation sémantique. Cette représentation est souvent exprimée sous forme d'un système de transitions. Les algèbres de processus imposent une logique équationnelle sur les termes de processus, tels que deux processus qui peuvent être assimilés seulement si leurs représentations sémantiques soient équivalentes selon une relation d'équivalence [Maarouk, 2011]. Une algèbre de processus peut être étendue par de nouveaux opérateurs, pour améliorer son expressivité ou pour faciliter la spécification du comportement du système.

Ces concepts fondamentaux des algèbres de processus, lorsqu'ils sont combinés avec des langages appropriés pour la description des types de données, fournissent des solutions techniques, en particulier par l'expressivité offerte aux utilisateurs et le large spectre de leurs applications.

Plusieurs algèbres de processus ont été utilisées pour spécifier et analyser des systèmes concurrents qui communiquent entre eux. Nous pouvons citer, à cet effet, CSP (Communicating Sequential Processes) de Hoare [1978], CCS (Calculus of Communicating Systems) de Milner [1982], ACP (Algebra of Communicating Processes) de Bergstra et Klop [1985] et LOTOS (Language Of Temporal Ordering Specifications) de Bolognesi et Brinksma [1987]. De telles algèbres offrent un

[1] terminologie standard dans le domaine des systèmes d'exploitation

excellent cadre pour la description des systèmes concurrents communicants, et ils sont bien équipés pour l'étude de leurs propriétés comportementales. Les logiques temporelles peuvent être utilisées pour exprimer formellement ces propriétés. Cependant, ces algèbres ne représentent qu'un fondement théorique et un formalisme conceptuel pour la spécification des systèmes informatiques.

3.1.1 CCS : Calculus of Communicating Systems

Le formalisme CCS [Milner, 1982] décrit les systèmes communicants comme des ensembles d'automates non-déterministes, appelés processus, qui interagissent par le biais de synchronisations. Le comportement d'un processus n'est décrit que partiellement par l'ensemble de ses traces (une trace est une suite de transitions) alors que la notion de bisimulation propre à cette théorie, permet de raffiner la notion d'égalité des processus.

Les actions, vues comme des canaux de communication, constituent l'alphabet de base des processus. On suppose un ensemble dénombrable d'actions λ parcouru par $a, b, c, ...$ chacune ayant un inverse unique $\overline{a}, \overline{b}, \overline{c}, ...$ La notion d'inverse est tels que l'action et son inverse constituent des actions pouvant se synchroniser si elles proviennent de deux processus communicants.

La syntaxe des processus CCS est définie comme suit :

$$\begin{array}{lll} P & ::= & nil \mid \lambda.P \mid P+P \mid (P \mid P) \mid P \setminus a \qquad Processus \\ \lambda & ::= & a?v \mid a!v \mid \tau \qquad\qquad\qquad\qquad\quad Action \end{array}$$

Soit P et Q deux processus, un processus P peut être soit le processus inerte nil, soit un processus de la forme $a.P$ est dit *préfixé* par une action a et de continuation P. La forme $P|Q$ exprime la concurrence entre les exécutions de P et Q. Le choix $P+Q$ est une façon d'obtenir un comportement non-déterministe : le processus $a.P+b.Q$ peut soit émettre sur le canal a et passer dans l'état P, soit émettre sur b et passer dans l'état Q. L'opérateur de restriction $P \setminus a$ permet de spécifier que l'usage de l'action a est restreint au processus P.

Dans CCS il existe trois types d'actions : $a?x$ correspond à la réception de n'importe qu'elle valeur x sur le canal a, $a!v$, représente l'émission de la valeur v sur le canal a et τ est l'action interne ou silencieuse.

La sémantique opérationnelle des processus CCS est donnée par un système de transitions étiquetées (LTS pour labelled transition system). Cette sémantique est détaillée dans [Maarouk, 2011].

3.1.2 Le π-calcul

Extension de CCS, le π-calcul [Milner et al., 1992] est un calcul de processus parallèles, qui permet de rendre compte de systèmes dont la topologie de communication change dynamiquement. Ce calcul est maintenant considéré comme le fondement de la programmation concurrente [Sangiorgi et Walker, 2001], au même titre que le λ-calcul [Church, 1985] qui est considéré comme le fondement de la programmation fonctionnelle. En effet, le π-calcul a pour vocation première de modéliser des échanges de messages entre des programmes s'exécutant en parallèle.

En particulier, Le π-calcul de base (appelé π-calcul monadique) permet de rendre compte de la notion de migration au cours de calcul et de décrire des systèmes de processus mobiles, c'est à dire, des systèmes dont le nombre de processus, ainsi que les liens de communication entre eux, peuvent varier au cours du temps : les noms des canaux sont transmis comme valeurs sur des canaux déjà existants, ce qui permet à un processus de recevoir un canal nouvellement crée et d'acquérir ainsi de nouvelles possibilités de communication.

Le π-calcul est principalement utilisé pour établir des preuves d'équivalence entre des modèles de systèmes distribués [Milner et al., 1992]. Nous présentons ce calcul de systèmes communicants dans lequel, non seulement les processus composant le système ont une structure dynamique, mais aussi ils peuvent porter l'information qui change les liens entre ces processus.

Formellement, les processus sont décrits par la grammaire suivante :

$$\begin{array}{lll} P & ::= & nil \mid \alpha.P \mid P+P \mid (P \mid P) \mid (\nu c)P \mid [a=b]P \qquad Processus \\ \alpha & ::= & a(x) \mid \overline{a}\langle v \rangle \mid \tau \qquad\qquad\qquad\qquad\qquad\qquad\quad Action \end{array}$$

Dans la syntaxe du π-calcul : nil désigne le processus nulle, $\alpha.P$ le préfixage d'action, $P|P$ la concurrence, $(\nu c)P$ la restriction et $[a=b]P$ le branchement conditionnel. Parmi les actions possibles, τ exprime une communication interne à P,

$a(x)$ est la réception par P du message x sur le canal a, et $\overline{a}\langle v\rangle$ l'émission par P sur le canal a le message v.

Par la suite, Milner [1993] a proposé le π-calcul polyadique afin de faciliter la manipulation des termes du π-calcul en utilisant une forme moins primitive pour la communication. Cette extension permet un passage simultané de plusieurs arguments sur un même canal. La syntaxe est, donc, la même que celle de la forme monadique, cependant, les changements suivants sont à considérer :

- Le processus $a(x).P$, attend qu'un tuple y soit transmis sur le canal a, puis dérive sur le processus P avec la substitution de x par y.
- Le processus $\overline{a}\langle x\rangle.P$, envoie le tuple x sur le canal a, ensuite dérive sur le processus P.

Il existe plusieurs variantes canoniques du π-calcul telles que la forme asynchrone, distribuée, etc. À cet effet, le lecteur pourra se référer à [Maarouk, 2011] pour plus de précisions.

3.1.3 Calcul des ambients

Le calcul des ambients, introduit par Cardelli et Gordon [1998], est un modèle pour la programmation mobile et distribuée, basée sur la notion de localité et de mobilité. C'est un modèle de calcul de processus utilisant la communication de noms, dans l'esprit du π-calcul.

Soit un ensemble dénombrable de noms $\mathcal{N}$ parcouru par $a, b, c, m, n, ...$ Les processus du calcul des ambients sont définis comme suit :

$$
\begin{array}{lll}
P & ::= & 0 \mid n[P] \mid (P|P) \mid M.P \mid !P \mid (\nu n)P \mid (n).P \mid \langle M\rangle \qquad Processus \\
M & ::= & in\ n \mid out\ n \mid open\ n \mid \epsilon \mid M.M' \qquad Capacité
\end{array}
$$

Les opérateurs de base du calcul des ambients sont présents : le processus inerte "O", la composition parallèle "$P|P$", la restriction de noms "$(\nu n)P$", la réplication "$!P$". Ils se comportent comme en π-calcul. Un ambient est noté "$n[P]$", où n est le nom de l'ambient et P est le processus exécuté à l'intérieur. Le mécanisme de communication est caractérisé par une communication locale anonyme :

$(x).P|\langle M\rangle \longrightarrow P\{x \leftarrow M\}$. Les actions sont appelées des capacités et sont notées $M.P$. Il existe trois types de capacités :

- $in\ m$: l'ambient veut devenir une sous-localité de m, lorsqu'il en est le frère,
- $out\ m$: l'ambient veut sortir de la localité m, lorsqu'il en est le fils,
- $open\ m$: l'ambient veut ouvrir la localité m, lorsqu'il en est le père. Tout le contenu de m (ambients et capacités) sera alors libéré dans l'ambient père,et l'ambient m disparaît.

De plus, $M.M'$ représente la combinaison de deux capacités dans un chemin, par exemple, $(out\ n.in\ m)$.

En général, les ambients sont organisés en structure d'arbre. La racine de l'arbre représente le réseau et les sous-ambients sont des entités mobiles logiques ou physiques. Les migrations sont locales et dépendent de la structure de l'arbre. Ce sont les capacités qui induisent la mobilité au sein de l'arbre ainsi construit. Il est important de remarquer que l'action des capacités est locale. Ainsi les mouvements dans deux branches distinctes de l'arbre sont totalement indépendants, ce qui confère au modèle son aspect distribué.

3.2 LOTOS : Language of Temporal Ordering Specification

3.2.1 Définition de LOTOS

LOTOS (ou Basic-LOTOS) [Brinksma, 1988; IEC 15437, 2001], est une algèbre de processus, standardisé par ISO, permettant d'exprimer la structure logique et temporelle des comportements. Pour spécifier le comportement d'un processus, LOTOS s'appuie sur le langage CCS qui est étendu par un mécanisme de synchronisation multiple héritant du langage CSP. Notons que chacune de ces algèbres se distinguent par la sémantique des opérateurs la constituant.

Le concept sous-jacent à LOTOS est que tout système peut être spécifié en exprimant les interactions constituant le comportement observable des composantes du système. En LOTOS, un système est vu comme un processus, qui peut être

composé en sous-processus capables de réaliser des actions internes et d'interagir avec d'autres processus. Une spécification LOTOS décrit, ainsi, un système par hiérarchie de processus.

Les définitions de processus sont exprimées en termes d'expressions comportementales qui sont à leurs tours construites par composition de sous-expressions en utilisant des opérateurs. Ainsi, les processus sont, en général, définis récursivement, et le rendez-vous multidirectionnel constitue le mécanisme de base pour la communication inter-processus. Les principales caractéristiques définissant LOTOS sont :

Puissance d'expression : LOTOS est capable d'exprimer un large rang de propriétés qui sont appropriées pour la description des services, des protocoles et des interfaces OSI.

Définition formelle : la syntaxe et la sémantique de ce langage se basent sur des définitions formelles et complètes. En particulier, le modèle sur lequel la sémantique du langage est basée qui supporte une théorie analytique pour la vérification et la validation.

Abstraction : les constructeurs de LOTOS représentent les concepts architecturaux adaptés à un niveau suffisamment élevé d'abstraction, où les détails d'implémentation ne sont pas exprimés, ce qui est en faveur d'une représentation précise des besoins.

Structure : cette algèbre offre des moyens pour structurer une spécification en une construction significative. Une bonne structuration assure une bonne lisibilité et une facilité de maintenance, et elle peut simplifier l'analyse.

Avec toutes ces caractéristiques, LOTOS apparait comme un langage adéquat à la modélisation comportementale d'un agent ambiant.

3.2.2 Syntaxe formelle de LOTOS

En LOTOS, le contrôle des programmes est décrit par des expressions algébriques appelées comportements. La synchronisation et la communication s'effectuent exclusivement par rendez-vous, sans partage de mémoire.

On appelle *opérateurs de comportement* les structures de contrôle utilisées dans le langage : composition séquentielle, composition parallèle, ... Dans la sous-section suivante ces opérateurs seront détaillés l'un après l'autre.

On appelle *expression comportementale* (ou plus simplement *expression*) un terme syntaxique obtenu par combinaison des opérateurs de comportement. Les expressions de comportement sont dénotées par le non-terminal $\mathcal{E}$.

On appelle *porte* (*gate*) une action vue comme un canal de communication permettant la synchronisation par rendez-vous et l'échange de valeurs entre plusieurs tâches qui se déroulent en parallèle.

On note $\mathcal{G}$ l'ensemble de tous les identificateurs de portes qui figurent dans une spécification LOTOS. Deux portes spéciales sont prédéfinies et qui n'appartiennent pas à $\mathcal{G}$:

- la porte invisible, notée "τ". Cette porte peut apparaître dans les expressions LOTOS,
- la porte de terminaison, notée "δ". Cette porte qui est utilisée pour notifier la terminaison avec succès d'un processus donnée, ne peut jamais être employée explicitement dans une expression LOTOS mais elle est utilisée dans la définition sémantique du langage.

Soit PN l'ensemble des variables de processus parcouru par X et soit $\mathcal{G}$ l'ensemble des portes (actions observables) parcouru par $a, b, ...$ L dénote tout sous-ensemble de $\mathcal{G}$. $\mathcal{E}$ parcouru par $E, F, ...$ dénote l'ensemble des expressions comportementales dont la syntaxe est :

$$
\begin{array}{lcll}
P & ::= & E & Processus \\
E & ::= & exit \mid stop & \\
 & & \mid a;E \mid E \odot E & (a \in \mathcal{O}) \\
 & & \mid hide\ L\ in\ E & \\
\odot & = & \{\ [\ ], |[L]|, |||, ||, \gg, [>\ \} &
\end{array}
$$

Étant donné un processus dont le nom est P et dont l'expression comportementale est E, la définition de P est exprimée par $P := E$. L'expression élémentaire

"$stop$" spécifie un comportement sans évolution possible et "$exit$" représente la terminaison avec succès d'un processus. Dans la syntaxe, l'ensemble $\odot$ représente les opérateurs du langage LOTOS qui sera détaillé dans la section suivante.

3.2.3 Les opérateurs de LOTOS

Les expressions élémentaires "$stop$" **et** "$exit$"

L'expression "$stop$" dénote un comportement inactif, qui ne propose aucune action à l'environnement ni aucune transition "τ" interne. "$stop$" permet de spécifier explicitement l'arrêt d'une expression comportementale. Mais "$stop$" peut aussi apparaître de manière implicite, lorsqu'une expression se bloque. Pour distinguer ces deux formes de terminaison, normale et anormale, une nouvelle expression est introduite. "$exit$" dénote un comportement qui se termine normalement. La terminaison avec succès s'exprime par le franchissement d'une transition "δ". De fait "$exit$" est équivalent à l'expression "$\delta; stop$".

L'opérateur de préfixage ";"

L'opérateur ";" permet de spécifier le rendez-vous. Si a est une action et E une expression comportementale, $a; E$ dénote le comportement qui offre l'action a et, une fois que celle-ci est exécutée, elle lance E. La notation ";" a une signification séquentielle : on dit que l'expression E est préfixée par l'action a.

L'opérateur de choix non-déterministe "[ ]"

L'opérateur "[ ]" permet de spécifier le choix non-déterministe. Si E et F sont deux expressions comportementales, $E[\,]F$ dénote le comportement qui peut exécuter soit E soit F.

les opérandes de "[ ]" doivent être des expressions comportementales et non des actions ; de même l'opérande gauche de ";" doit être une action et non une expression. Un exemple d'une écriture correcte d'une expression est :
$(a; b; stop)[\,](c; d; stop)$

Les opérateurs de parallélisme "||", "|||" **et** "|[L]|"

Les opérateurs qui ont été présentés, jusqu'ici, sont strictement séquentiels ; LOTOS comprend aussi des opérateurs parallèles. Si E et F sont deux expressions

comportementales et L un sous-ensemble de $\mathcal{G}$, $E|[L]|F$ dénote le comportement qui exécute E et F en parallèle. La synchronisation et la communication entre les opérandes E et F s'effectuent uniquement par rendez-vous sur les actions de l'ensemble $L \cup \{\delta\}$.

Lorsqu'un des opérandes veut effectuer une transition étiquetée par une action a de $L \cup \{\delta\}$, il doit attendre que l'autre opérande puisse en faire autant. Lorsque le rendez-vous est possible, les deux opérandes effectuent simultanément une même transition synchrone étiquetée par a ; puis ils reprennent chacun leur exécution.

En revanche si l'un des opérandes veut effectuer une transition étiquetée par une porte a qui n'appartient pas à $L \cup \{\delta\}$, alors il le fait indépendamment de l'autre opérande, de manière asynchrone.

En CCS, le rendez-vous entre deux portes complémentaires est facultatif ; on peut le rendre obligatoire en utilisant l'opérateur de restriction, mais on ne peut jamais l'interdire. En LOTOS si deux portes identiques sont composées de manière synchrone, le rendez-vous est obligatoire ; si elles sont composées de manière asynchrone, le rendez-vous est interdit.

Outre l'opérateur de synchronisation général "$|[L]|$" qui traduit la synchronisation sur toutes les actions $a \in L$ et "δ", LOTOS possède deux autres opérateurs de composition parallèle :

- le premier opérateur exprime la synchronisation sur aucune action, sauf "δ". La syntaxe $E|||F$, signifie que les deux expressions comportementales E et F sont exécutés de manière totalement indépendante, exceptée leur terminaison sur "δ", c'est à dire les deux expressions ne se synchronisent pas et ne communiquent pas entre elles. En revanche, elles sont capables d'interagir avec leur environnement commun.

- le second opérateur exprime la synchronisation sur toutes les actions, y compris "δ". Sa syntaxe est $E||F$ décrivant que les deux expressions comportementales E et F sont exécutées en parallèle de manière entièrement synchrone et doivent se synchroniser sur toutes leurs interactions.

L'opérateur de séquence " $\gg$ "

L'opérateur de préfixage ";" est asymétrique : son opérande gauche est une action alors que son opérande droit est une expression comportementale. LOTOS possède un autre opérateur de composition séquentielle dont les deux opérandes sont des expressions comportementales. Si E et F sont deux expressions, $E \gg F$ dénote le comportement qui exécute séquentiellement E puis F.

Intuitivement, la composition séquentielle est modélisée en LOTOS comme un cas particulier de composition parallèle. Les expressions E et F sont exécutés en parallèle mais F ne peut pas commencer avant que E ne se soit terminé. L'attente de F s'obtient par un rendez-vous sur "δ", rendez-vous qui devient possible dès que E exécute "$exit$". Si E boucle indéfiniment ou se bloque sans atteindre "$exit$", F ne sera jamais exécuté.

L'opérateur d'interruption "[> "

LOTOS dispose d'un opérateur permettant de spécifier l'interruption d'une expression comportementale par un autre. Si E et F sont deux expressions, $E[> F$ dénote le comportement qui exécute E mais qui peut abandonner à tout instant l'exécution de E pour commencer celle de F. Intuitivement, il s'agit d'un mécanisme d'interruption avec terminaison : E joue le rôle d'un traitement normal et F celui d'un traitement d'exception. Initialement, E est exécuté seul mais, tant qu'il n'a pas effectué de transition étiquetée par "δ", son exécution peut être interrompue au profit de celle de F. Si E se bloque avant d'atteindre une instruction "$exit$" alors F est inévitablement exécuté. Dans le cas contraire F peut très bien ne jamais être exécuté.

L'opérateur d'intériorisation "$hide$"

LOTOS possède un opérateur qui permet de cacher certaines actions d'une expression comportementale. Si L est un sous-ensemble d'actions observables et E une expression comportementale, "$hide\ L\ in\ E$" dénote l'expression E dont les actions de L sont intériorisées (c'est à dire renommées en "τ". Les actions de L ne sont visibles que dans E et deviennent inaptes à la synchronisation avec l'environnement. Vues de l'extérieur, ces interactions sont invisibles (puisqu'étiquetées par "τ"). Par exemple, "$hide\ a,d\ in\ (a;b;stop|||c;d;stop)$" est équivalent, vu de l'extérieur, à "$b;stop|||c;stop$".

3.2.4 Full-LOTOS

Full-LOTOS, est Basic-LOTOS étendu avec la possibilité d'échanger des valeurs lors des synchronisations entre processus. Contrairement à Basic-LOTOS, en full LOTOS, une action n'est pas simplement une porte de synchronisation, mais une porte avec plus des données échangées lors de la synchronisation. Les structures de ces données et les opérations associées sont définies par le langage Act-One [Ehrig et Mahr, 1985] qui formalise la spécification de types abstraits de données. Cette spécification se fera en définissant leurs propriétés essentielles et les opérations qu'une implémentation correcte du type doit assurer. Les valeurs en Full-LOTOS peuvent être :

- échangées entre processus lors d'une synchronisation,
- employées dans les prédicats de garde des processus,
- utilisées comme paramètres pour la définition des processus et pour l'instanciation de valeurs,
- associées à l'opérateur de choix généralisé,
- exportées lors d'une terminaison avec succès d'un processus,

TABLE 3.1: Les règles de synchronisation de deux processus P et Q en Full-LOTOS

P	Q	Conditions de synchronisation	Type d'interaction	Résultat
$a!v_1$	$a!v_2$	$valeur(v_1) = valeur(v_2)$	Concordance des valeurs	Synchronisation
$a!v$	$a?x:T$	$v \in T$	Passage de valeur	$x = valeur(v)$
$a?x:T$	$a?y:U$	$T = U$	Choix aléatoire d'une valeur	$x = y = v$ avec $v \in T$ ($v \in U$)

Avec l'ajout des données dans la spécification, deux processus P et Q peuvent se synchroniser selon trois cas de base, comme cela est illustré dans la table 3.1. Une action a peut :

- Offrir une valeur x ($a!x$),

- Accepter une valeur y $(a?y : type)$,
- Incorporer des prédicats qui conditionnent l'acceptation de valeurs.

3.3 Discussion

Dans ce chapitre, nous avons exploré plusieurs formalismes formels qui adaptent les besoins de la programmation distribuée et mobile. Nous avons vu que le formalisme CCS était le point de départ de ces formalismes, mais n'incluant pas la notion de mobilité. Par la suite le π-calcul a été défini pour remédier à ce manque. Depuis, plusieurs modèles ont été définis comme le Join-calcul [Fournet et al., 1996] proposant une communication asynchrone entre les processus. Cependant, aucun de ces modèles ne couvrent les besoins et les caractéristiques des agents ambiants du fait de leur rigidité. En effet, il ne sont pas adaptés aux systèmes ouverts et dynamiques. Ce qui nous a inspiré pour proposer le modèle le plus adapté pour la description de nos agents.

Dans les SMA, la concurrence entre les agents a été largement abordée où ces derniers devaient coopérer et collaborer afin d'atteindre leurs objectifs individuels ou communs. Par ailleurs, la plupart des travaux relatifs à la planification des agents, traitent uniquement les exécutions séquentielles des actions, à l'exception de quelques langages tel que CanPlan [Sardina et al., 2006] qui prend en considération la composition concurrente des (sous-)plans. Cependant, cette concurrence ne s'applique qu'aux actions et non pas aux intentions.

Notre objectif, dans cette thèse, est de proposer un modèle formel pour la spécification du comportement des agents intelligents et ambiants. Ce modèle doit être capable de capturer simultanément, d'une part, l'état mental de l'agent s'agissant de son raisonnement, ses connaissances et ses objectifs, et d'autre part, l'état comportemental de celui-ci qui se résume en plan d'actions.

De plus, la spécification du plan de l'agent doit prendre en considération la concurrence entre ses intentions et doit être assez flexible pour permettre une mise à jour à la volée de ces intentions, quand cela est nécessaire.

LOTOS apparaît comme un bon candidat pour cette spécification de plans particulièrement comme un modèle d'actions, mais il n'est pas adapté aux besoins des agents ambiants. En effet, il est nécessaire pour l'agent de prendre en consi-

dération son contexte et adapter son comportement en fonction. Des extensions de LOTOS existent mais ne semble pas être meilleures que ce dernier. Alors, nous proposons d'étendre LOTOS pour mieux modéliser les plans de l'agent ambiant.

Deuxième partie

Contributions : Modélisation des agents ambiants

Chapitre 4

Modèle formel pour la spécification des agents ambiants

Dans ce présent chapitre, nous nous intéressons à modéliser l'évolution comportementale de l'agent ambiant. Nous présentons un modèle d'agent d'ordre supérieur, appelé *Higher-order Agent - HoA*. Celui-ci décrie comment des agents ambiants, possédant des capacités mentales (BDI), qui sont sensibles à leurs contexte, peuvent évoluer et se déplacer dans un environnement ambiant. En outre, une architecture logicielle sous-jacente à ce modèle est proposée pour l'agent, tout en privilégiant une séparation nette entre son processus mental et celui de sa planification, qui est appelé *planning-process*.

Ensuite, afin de décrire les plans de l'agent, nous proposons un langage algébrique formel, nommé *AgLOTOS*, intégrant les aspects de modularité et de concurrence au niveau des sous-plans, ces derniers sont vus comme des processus. De plus, un mécanisme efficace de gestion des plans sera introduit afin de permettre à l'agent de construire un plan à partir de l'ensemble de ses intentions courantes.

L'agent est vu comme une entité dynamique changeant ses intentions tout au

long de son évolution. Le planning-process de l'agent doit être capable de réviser son plan en fonction de la mise à jour de ses intentions. Á cet effet, nous proposons un mécanisme de révision du plan de l'agent automatique et à la volée, qui est basé sur la modularité du langage AgLOTOS. Par ailleurs, la sémantique opérationnelle d'AgLOTOS prend en considération les échecs inattendus des actions, contribuant ainsi à la robustesse de l'agent vis-à-vis de son contexte dynamique et incertain.

4.1 L'architecture de l'agent AmI

La figure 4.1 décrit l'architecture de l'agent que nous proposons, appelée *Architecture d'agent d'ordre supérieur* (en anglais *Higher-order Agent architecture - HoA*). Contrairement aux architectures BDI traditionnelles, l'architecture HoA privilégie une séparation nette en quatre processus :

- le processus *Context-process* est en charge de traiter les informations contextuelles de l'agent. Il est déclenché d'une part par les perceptions de l'agent vis-à-vis de son environnement et d'autre part par les événements internes à l'agent provenant des exécutions de ses actions. À un niveau physique, le context-process est capable si une action s'est réalisée avec succès ou bien si elle a échoué.

- le processus *Mental-process* correspond au module de raisonnement de l'agent. Il est mis au courant des changements importants de l'environnement par les éventuelles notifications du context-process et en fonction de ces dernières il peut mettre à jour les croyances (B), les désirs (D) et les intentions (I) de l'agent.

- le processus *Planning-process* est sollicité par le mental-process et aidé par la bibliothèque de plans LibP, afin de produire principalement un plan d'actions pour l'agent à partir de l'ensemble de ses intentions. Par ailleurs, il offre des mécanismes de gestion dynamique des plans ainsi qu'un service de guidance contextuelle aidant l'agent dans la prise de ses décisions.

- le processus *Execution-process* est en charge de l'exécution des actions. Il est à l'écoute du planning-process où ce dernier lui offre une trace d'actions parmi celles qui sont possibles dans le plan de l'agent. De plus, l'execution-

process est capable d'enrichir les expériences de l'agent par les succès et les échecs des actions exécutées.

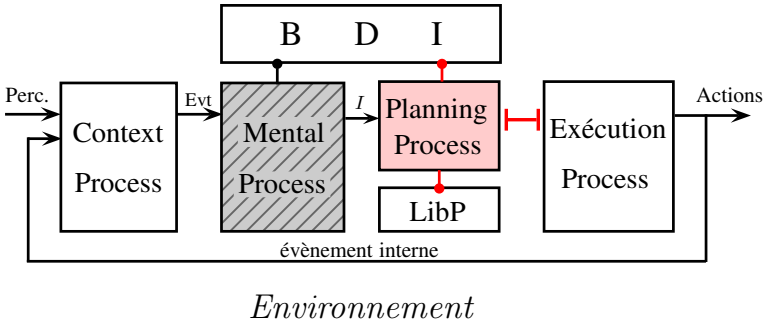

FIGURE 4.1: L'architecture d'ordre supérieur de l'agent (HoA)

4.2 Le modèle d'ordre supérieur de l'agent

Le modèle comportemental de l'agent se présente sous forme de deux principaux aspects : *(1)* Le raisonnement mental de l'agent et *(2)* l'évolution de son plan. Par conséquent, l'état de l'agent, nommé *configuration HoA*, est une paire composée d'un *état BDI* (*BDI state*) et d'un *état du plan* (*Planning state*). L'état BDI représente l'évolution mental de l'agent en terme de ses attitudes mentales B, D et I. Comme il est illustré dans la figure 4.2, ce sont les occurrences des évènements qui sont responsables de provoquer les changements des configurations de l'agent.

L'évolution de ces configurations est formellement représentée par un modèle comportemental d'ordre supérieur, appelé *modèle HoA* (*Higher-order Agent model*). Il est défini par un alphabet d'évènements notés $Evt = Evt_{Act} \cup Evt_{Perc}$ où ces derniers peuvent être déclenchés, d'une part par la terminaison des actions en cours d'exécution (Evt_{Act}), et d'autre part par des perceptions de l'environnement (Evt_{Perc}).

Définition 1. *(Le modèle HoA et la configuration HoA)*
Considérons n'importe quel agent AmI, soit $\mathcal{BDI}$ l'ensemble de tous les états BDI possibles, $\mathcal{P}$ l'ensemble de tous les plans possibles pouvant être construits à partir des états BDI et $\mathcal{PS}$ l'ensemble de tous les états évoluant dans un plan. En outre, $LibP$ représente la

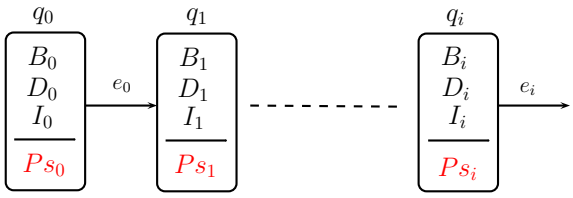

FIGURE 4.2: Le changement comportemental de l'agent

bibliothèque de plans. Le modèle d'ordre supérieur de l'agent est un système de transitions, représenté par un 6-uplet $\langle Q, q_0, \rightarrow, F_M, F_P, F_{PS} \rangle$*, avec :*

- Q *est l'ensemble des configurations HoA tels que chaque configuration* q *est une pair* $q = (bdi, ps)$ *où* bdi *et* ps *représentent respectivement l'état BDI et l'état du plan de l'agent dans* q*,*
- $q_0 \subseteq Q$ *est la configuration HoA initiale tel que* $q_0 = (bdi_0, ps_0)$*,*
- $\rightarrow \subseteq Q \times Evt \times Q$ *est l'ensemble des transitions indiquant les évolutions possibles d'une configuration HoA que de l'agent,*
- $F_M : Q \longrightarrow \mathcal{BDI}$ *associe un état BDI à chaque configuration HoA,*
- $F_P : \mathcal{BDI} \times LibP \longrightarrow \mathcal{P}$ *associe à chaque état BDI, un plan d'agent construit à partir de la bibliothèque LibP,*
- $F_{PS} : Q \longrightarrow \mathcal{PS}$ *associe un état du plan à chaque configuration HoA.*

L'état BDI est composé de trois ensembles de propositions, représentant les croyances, les désirs et les intentions de l'agent. Les intentions sont supposées être partiellement ordonnées par des poids. Cela permet au mental-process d'organiser ses intentions sélectionnées afin de résoudre les conflits possibles entre elles. Par ailleurs, en se basant sur le langage AgLOTOS, le plan est directement construit à partir des intentions et écrit en termes d'expressions AgLOTOS. En effet, Les états possibles du plan sont dérivés en utilisant la sémantique d'AgLOTOS à partir de l'expression du plan (voir Section 4.3).

Un exemple simple d'un scénario AmI

Soient Alice et Bob, deux agents au sein d'un système universitaire AmI. Ce système est considéré comme étant ouvert vu que les agents peuvent y entrer et sortir. Le fait que Bob soit entré dans le système, peut être perçu par Alice dans le cas où elle y est déjà. Étant donné que cette dernière est sensible à son contexte, elle peut tirer profit de cette information, en plus de ses propres informations. De plus, elle est capable de communiquer avec Bob à travers le système.

Soit $\ell_1, \ell_2 \in \Theta$ deux localités spatiales du système où les agents s'y trouvent. Le problème posé par Alice est qu'elle n'est pas capable de réaliser ses deux tâches suivantes au même moment : *(1)* se réunir avec Bob dans la localité ℓ_1, et *(2)* prendre ses copies d'examen qui sont dans ℓ_2. Clairement, les désirs d'Alice sont conflictuels puisqu'Alice ne peut pas être simultanément dans deux localités distinctes.

Table 4.1: Une évolution mentale possible d'Alice et Bob

Scénario d'Alice	
q_0^A	$B_0 = \{in(me, \ell_1), in(copies, \ell_2)\}$ $D_0 = \{meeting(Bob, \ell_1), getting\_copies(\ell_2)\}$ $I_0 = \{meeting(Bob, \ell_1)\}$
q_1^A	$B_1 = \{in(me, \ell_1), in(copies, \ell_2), in(Bob, \ell_2)\}$ $D_1 = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$ $I_1 = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$

Scénario de Bob	
q_0^B	$B_0 = \{in(me, \ell_2)\}$ $D_0 = \{waiting(v), meeting(Alice, \ell_1)\}$ $I_0 = \{waiting(v), meeting(Alice, \ell_1)\}$
q_1^B	$B_1 = \{in(me, \ell_2), in(copies, \ell_2)\}$ $D_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ $I_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$

La table 4.1 représente une évolution possible des configurations de ces deux agents. Les configurations initiales d'Alice et Bob sont respectivement q_0^A et q_0^B, tel que Alice est dans ℓ_1 et ayant deux désirs inconsistants, tandis que Bob est dans ℓ_2 ayant pour objectif de faire une réunion avec Alice. L'intention courante d'Alice est seulement de rencontrer Bob. Dans ce contexte, les attitudes BDI des agents sont simplement exprimées en utilisant des prédicats.

Le scénario AmI proposé évolue comme il est montré dans la figure 4.3 par les

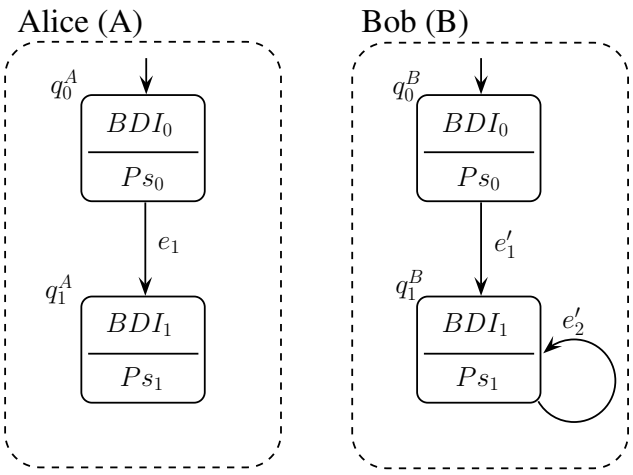

FIGURE 4.3: Représentation modulaire du scénario

modèles HoA d'Alice et Bob respectifs. En effet, les transitions possibles dérivent sur les configurations q_1^A et q_1^B, en fonction de l'évolution d'Alice et Bob. Effectivement, après avoir perçu que Bob est dans ℓ_2 (par exemple, $e_1 = perc(in(Bob, \ell_2))$), c'est à dire dans la même localité où se trouvent les copies d'examens, Alice enrichit ses croyances, ses désirs et ses intentions, dans le but de déléguer à Bob la tâche de lui ramener les copies d'examen. Par conséquent, son comportement évolue et donne la configuration q_1^A, où le plan construit consiste à demander à Bob de ramener les copies par envoi de message.

Á la réception du message d'Alice par Bob, l'évènement e'_1 est déclenché représentant la demande d'Alice. Il accepte (volontiers) de ramener les copies pour elle, alors Bob enrichit ses désirs par le nouveau $getting\_copies(\ell_2)$, sachant que ce dernier est consistant avec les anciens. Ainsi, l'état comportemental de Bob dérive vers la nouvelle configuration q_1^B. Le plan de Bob doit être révisé afin de satisfaire tous ses désirs à cette configuration, récupérer les copies d'Alice en premier ensuite il se déplace pour assister à la réunion.

Par ailleurs, un évènement qui n'est pas pertinent pour un agent, telle qu'une perception récurrente, est négligeable par celui-ci, par exemple, l'évènement e'_2 représente la perception de Bob qu'il est toujours dans la même localité ℓ_2.

La figure 4.4 illustre l'évolution résultante du système AmI de notre scénario. Notons qu'un état de ce système est composé des deux configurations d'Alice et Bob. De ce fait, nous abstrayons la communication en supposant que l'envoi

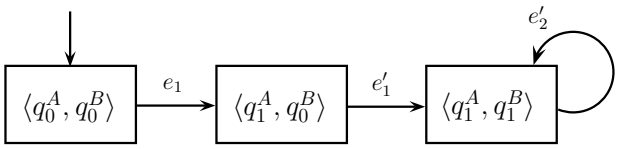

FIGURE 4.4: Une évolution possible du scénario

et la réception des messages sont assurés par des protocoles de communications fiables.

Dans la sections suivantes, le langage *AgLOTOS* sera introduit et utilisé afin de spécifier formellement ce scénario.

4.3 Le langage de spécification des plans : AgLOTOS

Dans cette section, nous proposons un langage de description algébrique pour la spécification des plans de l'agent. La structure du plan est présentée dans la figure 4.5 et les différentes notations associées à cette structure sont décrites dans la table 4.2. Le *plan d'agent*, s'accorde avec les niveaux suivants : *(1)* le plan d'agent est construit à partir de sous-plans appelés *plans d'intentions*. Ces derniers correspondent à l'achèvement des intentions de l'agent ; *(2)* chaque plan d'intention est une composition de plusieurs sous-plans alternatifs, appelés *plans élémentaires*, extraits à partir de la bibliothèque LibP. Cela permet à l'agent de les prendre en considération de différentes manières afin de réaliser l'intention correspondante. Aussi, nous assumons que la bibliothèque LibP est indexées par l'ensemble de toutes les intentions possibles auxquelles l'agent peut s'engager.

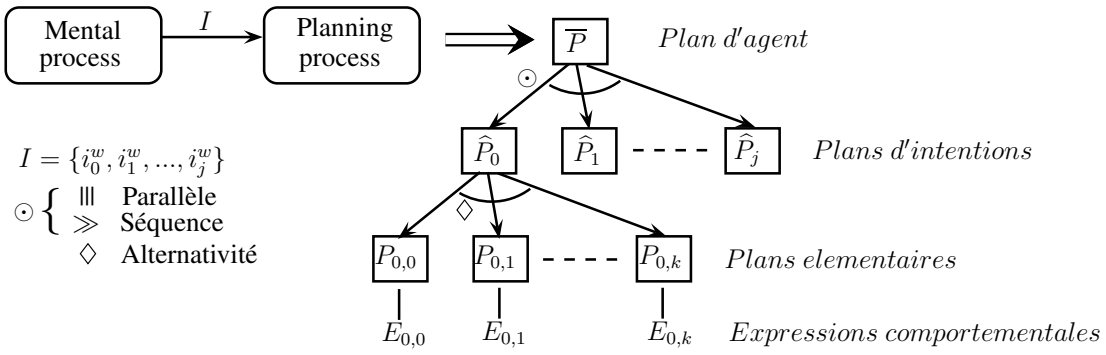

FIGURE 4.5: La structure du plan d'un agent

TABLE 4.2: Présentation synthétique des notations utilisées

Notation	Description
q	Configuration HoA
bdi	État BDI (BDI state)
ps	État du plan (Planning state)
E	Expression AgLOTOS
P	Plan élémentaire
$\widehat{P}$	Plan d'intention
$\overline{P}$	Plan d'agent
(E, P)	Configuration du plan élémentaire P
$(E, \widehat{P})$	Configuration du plan d'intention $\widehat{P}$
$[\overline{P}]$	Configuration du plan d'agent $\overline{P}$

4.3.1 La syntaxe et la sémantique informelle des plans AgLOTOS

Syntaxe des plans élémentaires

Les plans élémentaires d'un agent sont écrites en utilisant le langage AgLOTOS. Ce dernier est une extension du langage LOTOS[1] [Bolognesi et Brinksma, 1987], en offrant différentes manières d'exprimer la concurrence des actions au niveau des plans. La construction d'une expression AgLOTOS se base sur un ensemble fini d'actions observables, noté $\mathcal{O}$, qui est parcouru par $a, b, ...$

Soit L un sous-ensemble de $\mathcal{O}$ et $\mathcal{H} \subset \mathcal{O}$ l'ensemble des primitives AmI qui représentent la mobilité et la communication, sachant que :

- dans AgLOTOS, l'ensemble des actions est enrichi afin de prendre en compte les primitives AmI en les rendant observables : *(1)* un agent du système AmI est capable de percevoir l'entrée et la sortie d'un autre agent, *(2)* il peut se déplacer à travers les différentes localités, et *(3)* il peut communiquer avec d'autres agents qui sont visibles pour lui.
- une expression AgLOTOS se réfère aux informations contextuelles disponibles au niveau de l'état BDI courant de l'agent où : *(1)* Θ est l'ensemble fini des localités spatiales du système AmI, *(2)* Λ est un ensemble fini ou infini

[1] LOTOS : Language of Temporal Ordering Specification

d'agents avec lesquels l'agent peut communiquer, et *(3)* $\mathcal{M}$ est l'ensemble des messages possibles qu'il peut envoyer ou recevoir.

- la mobilité de l'agent est exprimée par la primitive $move(\ell)$ utilisée pour caractériser le déplacement de l'agent vers une localité donnée ℓ ($\ell \in \Theta$). La syntaxe des primitives de communication est inspirée de celle de π-calcul [Milner, 1999], cependant nous supposons une dynamicité totale de la communication fournie par les couches inférieures du système, donc sans une spécification prédéfinie des canaux. De ce fait, l'expression $x!(v)$ spécifie l'émission d'un message v ($v \in \mathcal{M}$) à l'agent x ($x \in \Lambda$) , tandis que, l'expression $x?(v)$ signifie qu'un message v est reçu depuis l'agent x.

Soit $Act = \mathcal{O} \cup \{\tau\}$, l'ensemble de toutes les actions, où $\tau \notin \mathcal{O}$ est l'action interne (ou silencieuse). Soit $\delta \notin Act$ un évènement particulier qui caractérise la terminaison d'un plan avec succès. Le langage AgLOTOS spécifie, pour chaque plan élémentaire, une paire (E, P) composée d'un nom pour l'identifier P et d'une expression AgLOTOS représentant son comportement E. Considérons que l'ensemble de tous les noms des plans élémentaires $\mathcal{P}$ est parcouru par $P, Q, ...$ et que l'ensemble de toutes les expressions comportementales possibles est noté $\mathcal{E}$, parcouru par $E, F, ...$ Les expressions AgLOTOS sont écrites en composant les actions par les opérateurs. La syntaxe d'un plan élémentaire P est défini par induction comme suit :

$$
\begin{array}{lcll}
P & ::= & E & Plan\ elementaire \\
E & ::= & \quad exit \mid stop & \\
 & & \mid a;E \mid E \odot E & (a \in \mathcal{O}) \\
 & & \mid hide\ L\ in\ E & \\
\mathcal{H} & ::= & \mid move(\ell) & (\mathcal{H} \subset \mathcal{O}, \ell \in \Theta) \\
 & & \mid x!(v) \mid x?(v) & (x \in \Lambda, v \in \mathcal{M}) \\
\odot & = & \{\ [\ ], |[L]|, |||, ||, \gg, [>\ \} &
\end{array}
$$

Maintenant, rappelons les opérateurs de LOTOS, l'expression élémentaire $stop$ spécifie un comportement sans évolution possible et $exit$ représente la terminaison d'un plan avec succès. Dans la syntaxe, l'ensemble $\odot$ représente les opérateurs : $E\ [\ ]\ E$ spécifie le choix non-déterministe, $hide\ L\ in\ E$ l'intériorisation des actions de L qui apparaissent dans E, $E \gg E$ la composition séquentielle et $E\ [>\ E$

l'interruption de l'expression gauche par l'expression de droite. La composition concurrente, dénotée $E|[L]|E$, peut modéliser la composition synchrone, $E||E$ si $L = \mathcal{O}$, et la composition asynchrone (parallélisme), $E|||E$ si $L = \emptyset$. De ce fait, le langage AgLOTOS fournit une riche expressivité dans laquelle une exécution séquentielle est vue comme un cas particulier.

La construction des plans d'agents à partir des intentions

La construction d'un plan d'agent nécessite des opérateurs spécifiques de AgLOTOS, tels que :

- au niveau du plan d'agent, l'opérateur de la *composition parallèle* $|||$ et celui de la *composition séquentielle* $\gg$ sont utilisés afin de construire le plan d'agent, en fonction des intentions de l'agent et leurs poids associés.
- l'opérateur de la *composition alternative*, dénoté $\Diamond$, permet de spécifier l'alternativité des plans élémentaires. En particulier, une intention est satisfaite si, et seulement si, au moins l'un de ses plans élémentaires est terminé avec succès.

Soit $\widehat{\mathcal{P}}$ l'ensemble des noms utilisés pour identifier les plans d'intentions possibles avec $\widehat{P} \in \widehat{\mathcal{P}}$ et soit $\overline{\mathcal{P}}$ l'ensemble des noms qualifiant les plans d'agents possibles avec $\overline{P} \in \overline{\mathcal{P}}$. La syntaxe des plans d'agents et d'intentions et définie récursivement comme suit :

$$\begin{array}{lcll} \overline{P} & ::= & \widehat{P} \mid \overline{P} ||| \overline{P} \mid \overline{P} \gg \overline{P} & Plan\ d'agent \\ \widehat{P} & ::= & P \mid \widehat{P} \Diamond \widehat{P} & Plan\ d'intention \end{array}$$

En fonction de l'ensemble des intentions I de l'agent, son plan d'agent est construit en deux étapes : *(1)* par un mécanisme d'extraction des plans élémentaires à partir de la bibliothèque de plans en utilisant la fonction $libp$, ensuite *(2)* ces plans élémentaires sont composés à l'aide des fonctions de composition $options$ et $plan$:

- $libp : \mathcal{I} \rightarrow 2^{\mathcal{P}}$, permet de gérer la bibliothèque des plans élémentaires. Cette fonction fournie pour chaque intention i, un ensemble de plans élémentaires instanciés dédiés à l'achèvement de l'intention i.

- $options : \mathcal{I} \rightarrow \widehat{\mathcal{P}}$, produit pour chaque intention $i \in \mathcal{I}$, un plan d'intention de la forme : $\widehat{P_i} = \Diamond_{P \in libp(i)} P$.
- $plan : 2^I \rightarrow \overline{\mathcal{P}}$, construit le plan d'agent final $\overline{P}$ à partir de l'ensemble des intentions I. En fonction de la façon auquel l'ensemble I est ordonné, les plans d'intentions résultants des différentes fonctions $\widehat{P_i} = options(i)$ tel que $i \in I$, sont composés en utilisant les opérateurs AgLOTOS $|||$ et $\gg$.

Selon notre approche, le mental-process a la charge d'étiqueter les différentes intentions dans I en utilisant une fonction de poids $weight : I \longrightarrow \mathbb{N}$. Cela permet au planning-process d'ordonnancer les plans d'intentions correspondants, produits par la fonction $options$. Les plans d'intentions ayant le même poids sont composés par l'opérateur de composition parallèle $|||$. En revanche, les plans d'intentions correspondants à des poids distincts sont ordonnés en appliquant l'opérateur de séquencement $\gg$.

Considérons par exemple, l'ensemble d'intentions $I = \{i_0^1, i_1^2, i_2^1, i_3^0\}$, tel que pour chaque intention, l'exposant représente la valeur de son poids. Sachant que $\widehat{P_0}, \widehat{P_1}, \widehat{P_2}, \widehat{P_3}$ représentent leurs plans d'intentions respectifs, le plan d'agent construit est : $plan(I) = \widehat{P_1} \gg (\widehat{P_0} ||| \widehat{P_2}) \gg \widehat{P_3}$.

4.3.2 La sémantique opérationnelle des plans AgLOTOS

La sémantique opérationnelle d'AgLOTOS, essentiellement dérivée de celle de Basic LOTOS [Bolognesi et Brinksma, 1987], est capable de capturer l'évolution des processus concurrents.

La sémantique des plans élémentaires

Une configuration (E, P) représente un processus identifié par P et une expression comportementale associée E. La table 4.3 rappelle la sémantique opérationnelle de Basic LOTOS qui permet de formaliser comment un processus peut évoluer à travers l'exécution des actions. Dans notre contexte, cette sémantique est reconsidérée afin de définir la sémantique des *plans élémentaires*, qui sont vus comme des processus. En particulier, la dernière règle spécifie comment chaque configuration (E, P) est dérivée vers (E', P) par l'exécution de l'action a. Par ailleurs, $P := E$

dénote que l'expression comportementale E est attribuée à P et $P \xrightarrow{a} E'$ qui représente l'évolution du plan P de E à E'.

TABLE 4.3: Les règles sémantiques des plans élémentaires

(Terminaison)	$\dfrac{}{exit \xrightarrow{\delta} stop}$
(Préfixage d'action)	$\dfrac{a \in \mathcal{O}}{a;E \xrightarrow{a} E}$
(Choix non-déterministe)	$\dfrac{E \xrightarrow{a} E'}{F \,[]\, E \xrightarrow{a} E' \qquad E \,[]\, F \xrightarrow{a} E'}$
(Concurrence)	$\dfrac{E \xrightarrow{a} E' \qquad a \notin L \cup \{\delta\}}{E \,\|[L]\|\, F \xrightarrow{a} E' \,\|[L]\|\, F} \qquad \dfrac{E \xrightarrow{a} E' \qquad a \notin L \cup \{\delta\}}{F \,\|[L]\|\, E \xrightarrow{a} F \,\|[L]\|\, E'}$ $\dfrac{E \xrightarrow{a} E' \qquad F \xrightarrow{a} F' \qquad a \in L \cup \{\delta\}}{E \,\|[L]\|\, F \xrightarrow{a} E' \,\|[L]\|\, F'}$
(Intériorisation)	$\dfrac{E \xrightarrow{a} E' \qquad a \notin L}{hideLinE \xrightarrow{a} hideLinE'} \qquad \dfrac{E \xrightarrow{a} E' \qquad a \in L}{hideLinE \xrightarrow{\tau} hideLinE'}$
(Séquence)	$\dfrac{E \xrightarrow{a} E' \qquad a \neq \delta}{E \gg F \xrightarrow{a} E' \gg F} \qquad \dfrac{E \xrightarrow{\delta} E'}{E \gg F \xrightarrow{\tau} F}$
(Interruption)	$\dfrac{E \xrightarrow{a} E' \qquad a \neq \delta}{E \,[>\, F \xrightarrow{a} E' \,[>\, F} \qquad \dfrac{E \xrightarrow{\delta} E'}{E \,[>\, F \xrightarrow{\delta} E'}$ $\dfrac{F \xrightarrow{a} F'}{E \,[>\, F \xrightarrow{a} F'}$
(Substitution)	$\dfrac{E \xrightarrow{a} E' \qquad a \notin \{a_1,...,a_n\}}{E[b_1/a_1,...,b_n/a_n] \xrightarrow{a} E'[b_1/a_1,...,b_n/a_n]}$ $\dfrac{E \xrightarrow{a} E' \qquad a = a_k \qquad 1 \leq k \leq n}{E[b_1/a_1,...,b_n/a_n] \xrightarrow{b_k} E'[b_1/a_1,...,b_n/a_n]}$
(Définition du plan)	$\dfrac{P := E \qquad E \xrightarrow{a} E'}{P \xrightarrow{a} E'}$

Dans ce qui suit, les règles de la sémantique opérationnelle de Basic-LOTOS sont brièvement expliquées :

- la règle de *(Terminaison)* caractérise l'occurrence de l'évènement δ capturant la terminaison de l'expression E,
- la règle de *(Préfixage d'action)* caractérise l'occurrence d'une action observable a,

- la règle du *(Choix non-déterministe)* exprime le choix entre l'exécution de l'expression E ou F,
- les règles de *(Concurrence)* représentent la concurrence entre deux expressions E et F, telles que : la règle *(Concurrence.a)* dénote l'exécution indépendante de l'action a dans l'expression E de manière asynchrone et parallèle, de la même manière que pour l'expression F dans la règle *(Concurrence.b)*. La règle *(Concurrence.c)* dénote la synchronisation entre les deux expressions E et F sur l'action a,
- la règle d'*(Intériorisation)* signifie que les actions qui appartiennent à L, dans l'expression E, sont cachées en dérivant sur l'action interne τ, c'est à dire elles ne sont visibles que dans E, et deviennent inaptes en dehors de E,
- la règle *(Séquence.a)* correspond à l'exécution séquentielle de l'expression E puis F, tandis que la règle *(Séquence.b)* dénote que l'exécution de F devient possible dès que E termine son exécution,
- les règles d'*(Interruption)* caractérisent l'interruption d'une expression par une autre. En effet, la règle *(Interruption.a)* signifie que l'expression E suit son exécution tant qu'elle n'est pas terminée, toutefois, son exécution peut être interrompue au profit de celle de F, ce qui est exprimé par la règle *(Interruption.c)*. La règle *(Interruption.b)* exprime le cas contraire où F n'est jamais exécuté du fait que E a terminé son exécution.
- les règles de *(Substitution)* représentent la remplacement des actions $a_1, a_2, ..., a_n$ par les actions $b_1, b_2, ..., b_n$ dans l'expression E.

La sémantique des plans d'intentions et des plans d'agents

La définition 2 ci-dessous spécifie comment l'expression d'un *plan d'agent* est construite par composition. En outre, l'expression comportementale du plan d'agent $\overline{P}$ est appelée *configuration du plan d'agent* et notée $[\overline{P}]$. Cette dernière est obtenue en composant les *configurations des plans d'intentions*, telle que $(E, \widehat{P})$ (voir la *Règle 2*), elles-mêmes construites à partir d'une composition alternative des *configurations de plans élémentaires*, telle que (E_k, P_k) (voir la *Règle 1*).

Définition 2. *(Représentation canonique du plan d'agent)*
Chaque configuration du plan d'agent $[\overline{P}]$ *a une représentation canonique définie par les deux règles suivantes :*

1. $$\frac{\overline{P}::=\widehat{P} \qquad \widehat{P}::=\Diamond^{k=1..n} P_k \qquad P_k::=E_k}{[\overline{P}]::=(\Diamond^{k=1..n} E_k,\ \widehat{P})}$$

2. $$\frac{\overline{P}::=\overline{P_1}\ \odot\ \overline{P_2} \qquad \odot\in\{|||,\gg\}}{[\overline{P}]::=[\overline{P_1}]\ \odot\ [\overline{P_2}]}$$

La table 4.4 présente les règles de la sémantique opérationnelle définissant les dérivations possibles à partir des états du plan (planning states) de l'agent. Ces règles sont appliquées sur une configuration HoA $q = (bdi, ps)$, où l'état du plan ps représente directement une configuration du plan d'agent, tel que $[\overline{P}]$. À chaque ligne de la table, il y a trois types de dérivations : les *règles (a)* illustrent les cas nominaux considérant l'exécution d'une action quelconque a ($a \in Act$), tandis que les *règles (b) et (c)* traitent les cas de terminaison d'un plan d'intention, $\widehat{P}$. Les *règles (b)* capturent la terminaison avec succès du plan d'intention concerné et les *règle (c)* capturent son échec de terminaison.

Soient le plan d'intention $\widehat{P}$, $\widehat{P}$ et $\neg\widehat{P}$ représentent respectivement la terminaison avec succès et celle avec échec du plan d'intention $\widehat{P}$. Par conséquent, si $\mathcal{PS}$ est l'ensemble de tous les états possibles du plan, alors la relation de transition entre les états du plan est un sous-ensemble de $\mathcal{PS} \times Act \times (\widehat{\mathcal{P}} \cup \neg\widehat{\mathcal{P}}) \times \mathcal{PS}$. Les transitions $(ps_1, a, \widehat{P}, ps_2)$ et $(ps_1, a, \neg\widehat{P}, ps_2)$ où $\widehat{P} \in \widehat{\mathcal{P}}$, provoquent des évènements internes informant le mental-process de la terminaison du plan d'intention $\widehat{P}$. Pour des raisons de clarté, la transition (ps_1, a, nil, ps_2) est simplement dénotée $ps_1 \xrightarrow{a} ps_2$, représentant l'exécution d'une action simple a (qui ne représente pas un évènement de terminaison).

- les deux premières lignes de la table concernent les règles de dérivation, d'une action, à partir d'une expression comportementale d'un plan d'intention $\widehat{P}$. Les *règles d'Action* illustrent les cas simples où E est l'expression d'un plan élémentaire appelé *expression élémentaire*, tandis que les *règles dites d'Alternativité* se focalisent sur l'exécution d'une alternativité d'expressions élémentaires, par exemple $\Diamond^{k=1..n} E_k$. La règle *(Alternativité.b)* capture la terminaison avec succès de $\widehat{P}$, par la dérivation de δ, alors que la règle *(Alterna-*

tivité.c) capture le cas d'échec, dans le cas où l'expression comportementale de $\widehat{P}$ est équivalente à $fail$. Dans notre approche, $fail$ représente le fait que l'exécution d'une expression comportementale E échoue à cause du contexte dynamique de l'agent.

Dans la règle *(Alternativité.a)*, l'expression comportementale $E = \Diamond^{k=1..n} E_k$ du $\widehat{P}$, est raffinée en utilisant la fonction $select$, qui permet de sélectionner une des expressions élémentaires parmi celles de E, par exemple $E_j := select(\widehat{P})$. L'opération d'alternativité est sémantiquement définie en introduisant un nouvel opérateur sémantique $\triangleright$, afin de prendre en compte cette sélection : $E_j \triangleright (\Diamond E_{k\neq j}^{k=1..n} E_k)$. Observons que l'opération $E \triangleright F$, dérive sur l'expression E si E réussit, et dérive sur F si E échoue.

- dans les deux dernières lignes de la table, les opérateurs de *séquence* et du *parallélisme* sont utilisés pour exprimer les compositions des configurations de plans d'intentions d'une manière séquentielle ou bien parallèle. Par ailleurs, les règles sont redéfinies sémantiquement afin de prendre en considération les cas d'échecs des plans d'intentions.

Formalisation du scénario

Reconsidérons le scénario de la section 4.2, la table 4.1 montre les évolutions possibles des configurations HoA de chacun des agents Alice et Bob séparément. Afin d'exprimer les configurations des plans d'agents, les actions sont écrites en termes de prédicats instanciés, par exemple $get\_copies(\ell_2)$. Les plans d'intentions sont composés à partir des plans élémentaires qui sont vus comme les processus concurrents, se terminant par $exit$.

Sachant que le mental-process d'un agent est capable d'ordonner son ensemble d'intentions, les intentions de Bob dans q_1^B, $I_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$, sont ordonnées tels que $weight(meeting(Alice, \ell_1)) = weight(getting\_copies(\ell_2))$. La configuration du plan d'agent correspondante à l'ensemble des intentions I_1 de Bob est : $[\overline{P_1}] = ((E_g, \widehat{P_g})|||(E_m, \widehat{P_m}))$, où $(E_g, \widehat{P_g})$ et $(E_m, \widehat{P_m})$ représentent les deux configurations des plans d'intentions ; la première configuration correspond à l'intention $getting\_copies(\ell_2)$ tandis que la deuxième correspond à $meeting(Alice, \ell_1)$, tel que $E_g = get\_copies(\ell_2); Alice!(confirm); exit$ et $E_m = move(\ell_1); meet(Alice); exit$.

TABLE 4.4: Les règles sémantiques des configurations des plans d'agents

(Action)	$(a)\ \dfrac{E \xrightarrow{a} E'}{(E,\widehat{P}) \xrightarrow{a} (E',\widehat{P})}$ $(b)\ \dfrac{E \xrightarrow{\delta} stop}{(E,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (stop,\widehat{P})}$ $(c)\ \dfrac{E \equiv fail}{(E,\widehat{P}) \xrightarrow[\neg\widehat{P}]{\tau} (stop,\widehat{P})}$
(Alternativité)	$(a)\ \dfrac{E_j \xrightarrow{a} E'_j \quad E_j := select(\widehat{P})}{(\Diamond^{k=1..n} E_k,\widehat{P}) \xrightarrow{a} (E'_j \triangleright (\Diamond^{k=1..n}_{k\neq j} E_k),\widehat{P})}$ $(b)\ \dfrac{E \xrightarrow{\delta} stop}{(E \triangleright F,\widehat{P}) \xrightarrow[\widehat{P}]{\tau} (stop,\widehat{P})}$ $(c)\ \dfrac{E \equiv fail \quad F \xrightarrow{a} F'}{(E \triangleright F,\widehat{P}) \xrightarrow{a} (F',\widehat{P})}$
(Séquence)	$(a)\ \dfrac{ps_1 \xrightarrow{a} ps'_1}{ps_1 \gg ps_2 \xrightarrow{a} ps'_1 \gg ps_2}$ $(b)\ \dfrac{ps_1 \xrightarrow[\widehat{P}]{\tau} ps'_1}{ps_1 \gg ps_2 \xrightarrow[\widehat{P}]{\tau} ps'_1 \gg ps_2}$ $(c)\ \dfrac{ps_1 \xrightarrow[\neg\widehat{P}]{\tau} ps'_1}{ps_1 \gg ps_2 \xrightarrow[\neg\widehat{P}]{\tau} ps'_1 \gg ps_2}$
(Parallélisme)	$(a)\ \dfrac{ps_1 \xrightarrow{a} ps'_1}{ps_1 \vert\vert\vert ps_2 \xrightarrow{a} ps'_1 \vert\vert\vert ps_2 \quad ps_2 \vert\vert\vert ps_1 \xrightarrow{a} ps_2 \vert\vert\vert ps'_1}$ $(b)\ \dfrac{ps_1 \xrightarrow[\widehat{P}]{\tau} ps'_1}{ps_1 \vert\vert\vert ps_2 \xrightarrow[\widehat{P}]{\tau} ps'_1 \vert\vert\vert ps_2 \quad ps_2 \vert\vert\vert ps_1 \xrightarrow[\widehat{P}]{\tau} ps_2 \vert\vert\vert ps'_1}$ $(c)\ \dfrac{ps_1 \xrightarrow[\neg\widehat{P}]{\tau} ps'_1}{ps_1 \vert\vert\vert ps_2 \xrightarrow[\neg\widehat{P}]{\tau} ps'_1 \vert\vert\vert ps_2 \quad ps_2 \vert\vert\vert ps_1 \xrightarrow[\neg\widehat{P}]{\tau} ps_2 \vert\vert\vert ps'_1}$

Un exemple d'exécution de Bob, dérivé à partir de son état initial du plan dans q_1^B, est comme suit : $((E_g,\widehat{P_g})\vert\vert\vert(E_m,\widehat{P_m})) \xrightarrow[\neg\widehat{P_g}]{\tau} (E_m,\widehat{P_m}) \xrightarrow{move(\ell_1)} (E'_m,\widehat{P_m}) \xrightarrow{meet} (E''_m,\widehat{P_m}) \xrightarrow[\widehat{P_m}]{\tau} (stop,\widehat{P_m})$.

exprimant que Bob échoue à ramener les copies mais cela ne l'empêche pas de se déplacer vers ℓ_1 afin d'assister à la réunion avec Alice.

Notons que les configurations des plans correspondant aux agents Alice et Bob dans q_1^A et q_1^B peuvent évoluer en fonction de leurs intentions mises à jour. La sec-

TABLE 4.5: Une formalisation possible des plans d'Alice et Bob

Scénario d'Alice	
q_0^A	$I_0 = \{meeting(Bob, \ell_1)\}$ $[\overline{P_0}] = (meet(Bob); exit, \widehat{P_m})$
q_1^A	$I_1 = \{meeting(Bob, \ell_1), asking(Bob, get\_copies(\ell_2))\}$ $[\overline{P_1}] = (Bob!(get\_copies(\ell_2)); exit, \widehat{P_a}) \gg (meet(Bob); exit, \widehat{P_m})$

Scénario de Bob	
q_0^B	$I_0 = \{waiting(v), meeting(Alice, \ell_1)\}$ $[\overline{P_0}] = (Alice?(v); exit, \widehat{P_w}) \;\|\|\|\; (move(\ell_1); meet(Alice); exit, \widehat{P_m})$
q_1^B	$I_1 = \{meeting(Alice, \ell_1), getting\_copies(\ell_2)\}$ $[\overline{P_1}] = (get\_copies(\ell_2); Alice!(confirm); exit, \widehat{P_g}) \|\|\| (move(\ell_1); meet(Alice); exit, \widehat{P_m})$

tion 4.4 enrichit la sémantique d'AgLOTOS par un mécanisme de révision automatique et à la volée du plan d'agent. Ce scénario sera repris comme illustration de ce mécanisme.

4.4 La révision dynamique des plans de l'agent

Dans notre architecture HoA vue dans la section 4.1, le planning-process est sensible aux changements des attitudes mentales de l'agent au niveau du mental-process telle que la mise à jour dynamique des intentions qui nécessite la révision du plan de l'agent. Évidement, le mental-process ne doit pas être sollicité à chaque changement du plan en progression. De ce fait, le mental-process est informé par le planning-process concernant la terminaison des sous-plans, en particulier les plans d'intentions, afin de préserver la consistance du comportement de l'agent. Dans ce contexte, nous supposons que les seules dépendances entre les intentions sont définies relativement à leurs pondérations, ce qui est primordial pour le mental-process.

Une approche simpliste consisterait à attendre la terminaison complète du plan de l'agent (signifiant que le planning-process atteindrait l'état final du plan courant), avant de prendre en compte la mise à jour de ses intentions. En revanche, dans ce travail, nous proposons une approche améliorée qui consiste à réviser dynamiquement le plan de l'agent en fonction des mises à jour appliquées aux intentions par le mental-process.

Ces révisions concernent l'ajout de nouveaux plans d'intentions, la suppres-

sion d'un des plans en progression restant à exécuter, ou bien le retardement d'un d'entre eux. Nous exploitons les avantages tirées de la nature compositionnelle d'AgLOTOS, qui permet au planning-process de gérer distinctivement les différents plans d'intentions au niveau du plan de l'agent. Nous rappelons que chaque état du plan se compose de configurations des plans d'intentions correspondantes aux intentions à réaliser.

4.4.1 Ajout et suppression dynamiques des plans d'intentions

Dans une configuration HoA donnée $q = (bdi, ps)$, $I(bdi)$ représente l'ensemble d'intentions dans cette configuration. Considérons qu'un état du plan $ps = [\overline{P}]$, tel que $[\overline{P}] = \odot_{i \in 1..n} (E_i, \widehat{P_i})$ représente la composition des configurations des plans d'intentions restantes à être exécutées, où chaque $\widehat{P_i}$ correspond au plan de l'intention i, E_i à son expression comportementale associée et $\odot \in \{|||, \gg\}$.

La mise à jour de l'ensemble d'intentions ainsi que le plan d'agent correspondant, se basent sur les principes suivants :

- en se basant sur la sémantique d'AgLOTOS, chaque terminaison d'un plan d'intention produit un évènement interne qui fait évoluer l'état BDI de l'agent, en particulier la mise à jour de l'ensemble d'intentions.
- Il est intuitive de construire des fonctions qui lient les intentions ($i \in I$) avec leurs plans d'intentions correspondants $(E, \widehat{P})$ et vice versa :
  - $remain : [\overline{\mathcal{P}}] \times I \rightarrow \mathcal{E} \times \widehat{\mathcal{P}}$ donne pour chaque intention $i \in I$, sa configuration de plan d'intention courante $(E, \widehat{P})$ à partir du plan d'agent $[\overline{P}] \in [\overline{\mathcal{P}}]$[2],
  - $index : \widehat{\mathcal{P}} \rightarrow I$ détermine pour chaque $\widehat{P}$ son intention $i \in I$. Il est à noter que $weight(index(\widehat{P_i}))$ donne le poids de l'intention i correspondante.

L'ajout et la suppression d'un plan d'intention sont formalisés respectivement par les deux fonctions suivantes : $add, remove : 2^{\mathcal{E} \times \widehat{\mathcal{P}}} \times \mathcal{I} \rightarrow [\overline{\mathcal{P}}]$, construisent une configuration d'un plan d'agent $[\overline{P}]$ à partir d'un ensemble de configurations de plans d'intentions.

[2] $[\overline{\mathcal{P}}]$ représente l'ensemble de toutes les configurations de plans d'agents possibles

L'ajout d'une intention

L'ajout d'une nouvelle intention k, sachant que la configuration de son plan d'intention est $(E_k, \widehat{P_k})$, se fait comme suit :

- ajouter l'intention k à l'ensemble I et mettre à jour la fonction $weight$ afin de prendre en compte le poids de k, ensuite
- construire la nouvelle configuration du plan de l'agent $\overline{\mathcal{P}}$, à partir de l'ensemble des configurations des plans d'intentions restantes $\cup_{i\in I}^{i\neq k} remain(\overline{P}, i)$, et les poids leurs correspondants ($weight(i)$) ainsi que de la configuration d'intention k en utilisant la fonction $options(k)$ (vue dans la section 4.3).

Formellement, soit ps l'état courant du plan de l'agent et soit k l'intention à ajouter à l'ensemble de ses intentions, l'état du plan résultant ps' après le processus de révision est exprimé par $ps' = add(\cup_{i\in I}^{i\neq k} remain(ps, i), k)$.

La suppression d'une intention

La suppression explicite d'une intention (non-terminée) k, de l'ensemble I, signifie que $(E_k, \widehat{P_k}) = remain([\overline{P}], k)$ doit être supprimée de $[\overline{P}]$. De la même façon que pour la fonction d'ajout, l'état du plan résultant ps' après la suppression de l'intention k est $ps' = remove(\cup_{i\in I} remain(ps, i), k)$.

Le ré-ordonnancement des intentions

En fonction des mises à jour appliquées à l'ensemble des intentions et plus particulièrement à leurs pondérations, la structure du plan d'agent est révisée sans être obligé de la reconstruire. Á cet effet, nous introduisons la fonction $reorder$: $2^{\mathcal{E}\times\widehat{\mathcal{P}}} \rightarrow [\overline{\mathcal{P}}]$, qui permet de réorganiser les configurations de plans d'intentions restantes $\cup_{i\in I}^{i\neq k} remain(\overline{P}, i)$ correspondant aux intentions réordonnées. Ainsi, l'état du plan résultant ps' après la réorganisation de la configuration représenté par ps est $ps' = reorder(\cup_{i\in I}^{i\neq k} remain(ps, i))$.

4.4.2 La révision à la volée du scénario

Reconsidérons une nouvelle fois le scénario, vu dans les tables 4.1 et 4.5 où les évolutions des configurations HoA de Alice et Bob peuvent être provoquées par leurs

nouvelles perceptions respectives. En effet, après avoir perçu que Bob se trouve dans la même localité ℓ_2 que les copies d'examen ($e_1 = perc(in(Bob, \ell_2))$), Alice enrichit ses croyances, ses désirs et ses intentions, visant à demander à Bob de l'aider en ramenant les copies avec lui. Par conséquent, son comportement évolue vers une nouvelle configuration HoA q_1^A, où le plan révisé suggère à Alice d'envoyer un message demandant à Bob de ramener les copies avec lui, $Bob!(get\_copies(\ell_2))$.

Notons que Bob est capable de recevoir n'importe quel message d'Alice, par exemple $Alice?(v)$ dans la configuration HoA q_0^B. sa réception du message déclenche un évènement au niveau de son mental-process.

Étant donné que dans ce scénario, Bob accepte de ramener les copies d'examen à Alice, il augmente ses croyances par la proposition $in(copies, \ell_2)$. Cela engendrera l'ajout, par son mental-process, d'une nouvelle intention $getting\_copies(\ell_2)$ tout en préservant la consistance de cette dernière avec celles déjà existantes et en cours de progression. Donc, la configuration HoA de Bob évolue en q_1^B, en particulier $I_1 = I_0 \cup \{getting\_copies(\ell_2)\} \setminus \{waiting(v)\}$, et son plan $[\overline{P_0}]$ est révisé en utilisant successivement les fonctions $remove$ et add :
$[\overline{P_0'}] = remove(\cup_{i \in I_0} remain(\overline{P_0}, i), waiting(v))$ puis
$[\overline{P_1}] = add(\cup_{i \in I_0} remain(\overline{P_0'}, i), getting\_copies(\ell_2))$.

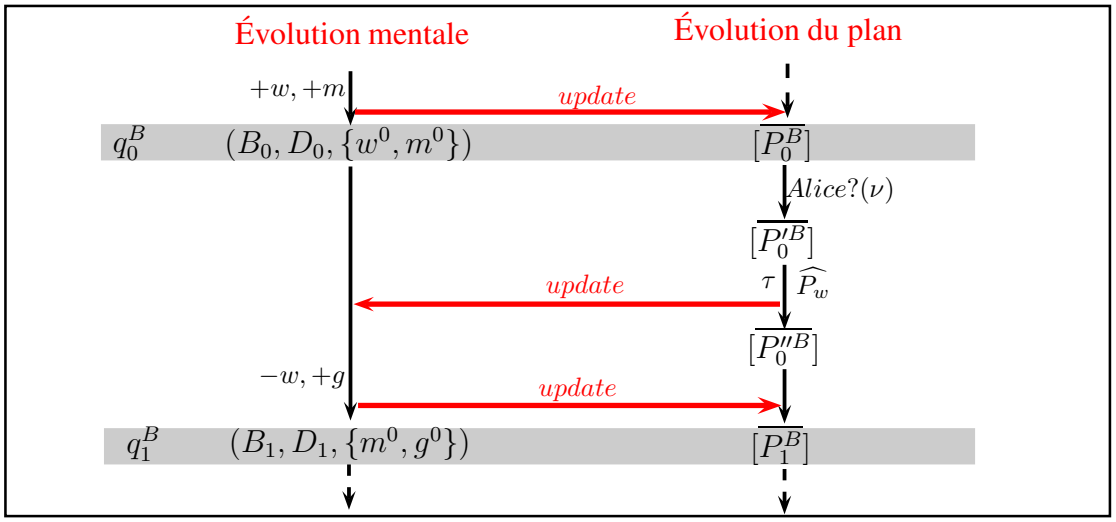

FIGURE 4.6: L'évolution HoA de l'agent Bob

Le diagramme de séquence de la figure 4.6 décrit le comportement de Bob en mettant l'accent sur les mises à jour mutuelles appliquées par le mental-process

et le planning-process. Cela permettra à l'agent Bob de synchroniser l'ensemble de ses intentions avec son plan d'exécution sous-jacent.

Pour simplifier, notons que w, m et g représentent respectivement les intentions $waiting(v)$, $meeting(Alice, \ell_1)$ et $getting\_copies(\ell_2)$. Par ailleurs, les signes + et − signifient respectivement l'ajout d'une intention et sa suppression.

Les flèches horizontales illustrent les mises à jour de cette partie du scénario. En premier lieu, le plan d'agent est construit du fait que w et m sont ajoutés à l'ensemble des intentions (avec des poids équivalents). Cela produit deux plans d'intentions concurrents, $[\overline{P_0^B}] = (E_w, \widehat{P_w})|||(E_m, \widehat{P_m})$. La réception du message d'Alice correspond à l'action $Alice?(v)$, suivie par le processus élémentaire $exit$ (caractérisé par l'évènement τ) indiquant la terminaison du plan d'intention $\widehat{P_w}$, donc l'état du plan résultant est $[\overline{P_0''^B}] = (E_m, \widehat{P_m})$. Puisque le mental-process est informé par des évènements internes, il décide finalement de mettre à jour ses intentions, avec $-w, +g$ où m et g ont le même poids. Ainsi, le planning-process, déclenché par le mental-process, révise le plan de l'agent en fonction de ses intentions, $[\overline{P_1^B}] = (E_g, \widehat{P_g})|||(E_m, \widehat{P_m})$.

4.5 Discussion

Le modèle comportemental HoA permet de capturer simultanément l'évolution de l'état mental de l'agent ainsi que son plan d'actions. Le planning-process est capable de sélectionner et composer les plans correspondant aux intentions de l'agent. Aussi, il est robuste aux changements dynamiques et imprévus du contexte de l'agent.

Dans la littérature, il existe de nombreux langages de programmation dédiés aux agents BDI [Hindriks, 2014], mettant en évidence les différents aspects et modules développés au sein de l'agent tels que les buts, la planification et l'organisation. Parmi ces langages, on trouve par exemple, Jadex [Pokahr et al., 2005], Jason [Bordini et al., 2007], 2APL [Dastani, 2008] et JIAC [Lützenberger et al., 2013]. Depuis l'année 2006, les approches BDI, centrées agent, émergent pour faire face à la dynamicité de l'environnement. En particulier, le modèle formel CanPlan, proposé dans [Sardina et al., 2006], il étend le langage Can afin de spécifier les actions dans les plans tout en intégrant les attitudes mentales (BDI) dans un même modèle.

Néanmoins, notre langage algébrique basé sur LOTOS, semble être plus expressif grâce à sa capacité à composer les (sous-)plans qui sont vus comme des processus concurrents. Il est également possible de gérer les échecs de ces plans d'actions au niveau du langage AgLOTOS et l'architecture HoA. En revanche, notre planning-process se comporte comme un service qui pourrait être intégré dans des architectures BDI, déjà existantes.

Dans notre approche, nous nous accordons à la nécessité de contrôler l'exécution des plans. Toutefois, nous privilégions une séparation claire entre le mental-process et le planning-process, où chacun a ses propres tâches respectives. De ce fait, les plans sont directement indexés par les intentions sans avoir le besoin de gérer les différents contextes de l'application. Le mental-process a la charge de filtrer, pondérer et réviser l'ensemble des intentions de l'agent, en prenant en considération les conditions les plus larges de son contexte. Par ailleurs, un contrôle de l'exécution qui se base sur les entrelacements des actions est envisagé pour permettre indirectement de contrôler la concurrence entre les plans. Il est également utile de réagir rapidement aux succès et aux échecs lors de l'exécution des actions.

La révision à la volée des plans, tout au long de l'évolution comportementale de l'agent, est en effet un domaine de recherche très pertinent dans les approches SMA, comme par exemple [Grant et al., 2010]. Aussi, Shapiro et al. [2012] ont proposé une approche pour la révision du plan à exécuter tout en tenant compte des informations contextuelles ainsi que de l'applicabilité du plan réalisant le but de l'agent.

Étant donné que notre technique de révision ne gère pas directement les informations contextuelles, il pourrait être intéressant d'intégrer l'une des propositions traditionnelles, dont le but est l'obtention d'une meilleure efficacité. Sur un temps restreint, il est fondamental quant à l'exécution des actions en vue de permettre à l'agent de traiter les échecs lors de l'exécution de celles-ci. En effet, l'échec éventuel d'une action provoque l'échec du plan correspondant et déclenche, en conséquence, la mise à jour de l'état mental de l'agent en fonction du changement de son contexte.

Chapitre 5

Aide à la guidance contextuelle

Notre objectif dans ce chapitre est d'aider le mental-process de l'agent dans ses décisions et de valider ses intentions concurrentes en fonction de son contexte prévu. Pour cela, le planning-process est augmenté par un service de guidance, exploitant les informations contextuelles de l'agent. Ce service de guidance est basé sur la construction d'un système de transitions, appelé *Système de planification contextuelle* (en anglais *Contextual Planning Systems - CPS*). La structure du CPS représente les différentes traces d'exécutions prédites, à partir d'une configuration HoA donnée.

5.1 Architecture interne du planning-process pour la guidance

Sachant que le planning-process que nous considérons est géré et contrôlé par le mental-process, la figure 5.1 décrit la vue fonctionnelle de ce planning-process, qui a la charge de construire un plan d'agent $\overline{P}$ et par la suite offrir une trace d'exécution (σ) à partir de l'ensemble I des intentions courantes de l'agent (). Le planning-process est composé de deux modules :

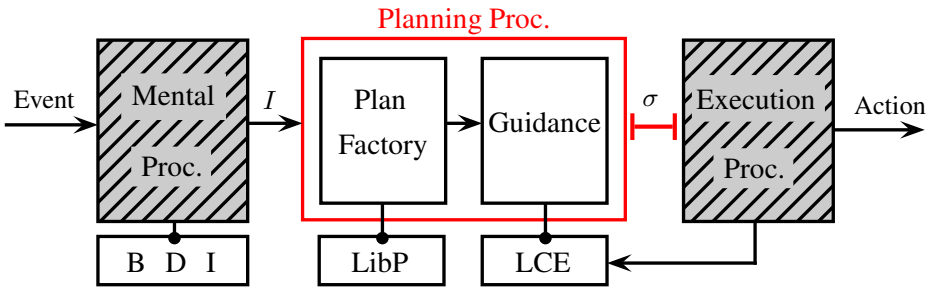

FIGURE 5.1: Vue architecturale de la guidance contextuelle

- le *constructeur des plans* (*Plan Factory*), utilisé précédemment dans la section 4.3, est responsable de la production du plan de l'agent. Ensuite, en se basant sur son contexte spatial courant, un CPS est généré afin de représenter tous les différents entrelacements d'actions dans le plan. En effet, ces entrelacements d'actions sont vus comme des traces possibles pouvant être exécutées afin de satisfaire les intentions de l'agent.
- le *module de Guidance* est capable d'enrichir le CPS par les expériences passées des actions, où ces dernières sont stockées dans un module appelé *Expériences contextuelles apprises* (*Learned Contextual Experiences (LCE)*). La structure résultante, appelée *CPS with Learning* (*CPS-L*), est utilisée pour capturer la trace optimale en respectant le contexte de l'agent.

5.2 Le système de planification contextuelle (CPS)

Considérons une configuration HoA donnée, le *CPS* de l'agent est défini contextuellement dans cette configuration, en fonction de sa localité et les terminaisons des différents plans d'intentions.

Soient q une configuration HoA de l'agent et $I(q)$ l'ensemble de ses intentions dans q. Les règles utilisées pour générer le CPS, prennent en compte trois types d'informations contextuelles dans chaque état du CPS, qui sont :

- la localité atteinte dans cet état,
- l'ensemble des plans d'intentions qui sont terminés, et

- globalement au niveau de tous les états du CPS, l'ensemble $\Lambda(q)$ des voisins connus actuellement par l'agent.

Définition 3. *(État CPS)*
Un état CPS est un triplet (ps, ℓ, T), où ps est l'état courant du plan de l'agent représenté par la configuration de son plan $[\overline{P}]$, ℓ correspond à la localité où l'agent se trouve, et T est le sous-ensemble des plans d'intentions qui sont terminés avec succès dans cet état.

Sémantique opérationnelle du CPS

TABLE 5.1: Les règles sémantiques des changements des états du CPS

(Action)	$(a)\ \dfrac{ps \xrightarrow{a} ps' \qquad a \in Act}{(ps,\ell,T) \xrightarrow{a} (ps',\ell,T)}$	$(b)\ \dfrac{ps \xrightarrow[\widehat{P}]{\tau} ps'}{(ps,\ell,T) \xrightarrow{\tau} (ps',\ell,T \cup \{\widehat{P}\})}$
(Mobilité)	$(a)\ \dfrac{ps \xrightarrow{move(\ell')} ps' \qquad \ell \neq \ell'}{(ps,\ell,T) \xrightarrow{move(\ell')} (ps',\ell',T)}$	$(b)\ \dfrac{ps \xrightarrow{move(\ell)} ps'}{(ps,\ell,T) \xrightarrow{\tau} (ps',\ell,T)}$
(Communication)	$(a)\ \dfrac{ps \xrightarrow{x!(v)} ps' \qquad x \in \Lambda}{(ps,\ell,T) \xrightarrow{x!(v)} (ps,\ell,T)}$	$(b)\ \dfrac{ps \xrightarrow{x?(v)} ps' \qquad x \in \Lambda}{(ps,\ell,T) \xrightarrow{x?(v)} (ps',\ell,T)}$

La table 5.1 met en évidence les règles de la sémantique opérationnelle, définissant les différents changements d'états possibles du CPS. Ces règles sont appliquées à partir de l'état CPS initial, $([\overline{P}], \ell, \emptyset)$ signifiant que l'agent est initialement à la localité ℓ, et la configuration de son plan d'agent est $[\overline{P}]$.

Les *règles d'Action* concernent la dérivation d'une action à exécuter au niveau d'un plan d'intention. La règle de *(Action.a)* illustre le cas d'une action simple, tandis que la règle *(Action.b)* représente le cas de terminaison avec succès d'un plan d'intention, où ce dernier est rajouté à l'ensemble T.

De plus, les *règles de Mobilité* capturent le déplacement de l'agent depuis la localité ℓ vers ℓ' et au niveau des *règles de Communication*, l'action d'envoi de message $x!(v)$ (respectivement l'action de réception $x?(v)$) est contrainte par la découverte de l'agent x dans le voisinage de notre agent.

Dans ce chapitre, nous proposons une approche prédictive et optimiste dans laquelle les cas d'échec sont écartés. Par conséquent, l'opérateur d'alternativité $\Diamond$ s'exprime par le simple choix non-déterministe $[\,] : \Diamond^{k=1..n} E_k \equiv [\,]^{k=1..n} E_k$. Ce qui permettra de considérer tous les plans élémentaires satisfaisant l'intention correspondante.

Définition 4. *(Contextual Planning System)*
Soit $q = (bdi, ps)$ une configuration HoA de l'agent, le système de planification contextuelle de q, noté $CPS(q)$, est une structure de kripke étiquetée $\langle S, s_0, Tr, \mathcal{L}, \mathcal{T} \rangle$ où :

- *S est l'ensemble de tous les états du CPS,*
- *$s_0 = (ps, \ell, \emptyset) \in S$ est l'état CPS initial, tels que $ps = [\overline{P}]$ représente l'état courant du plan de l'agent dans q et ℓ sa localité courante,*
- *$Tr \subseteq S \times Act \times S$ est l'ensemble des transitions étant de la forme $s \xrightarrow{a} s'$ tels que $s, s' \in S$ et $a \in Act$,*
- *$\mathcal{L} : S \rightarrow \Theta$ est la fonction étiquetant chaque état CPS par la localité prévue de l'agent dans cet état,*
- *$\mathcal{T} : S \rightarrow 2^{\widehat{\mathcal{P}}}$ est la fonction étiquetant chaque état CPS par les identités des plans d'intentions dont les terminaisons sont accumulés dans cet état.*

Au niveau du CPS, les transitions franchissables à partir d'un état CPS (source) représentent uniquement les actions qui sont réalisables dans cet état. Partant du même principe du langage de description STRIPS [Meneguzzi et al., 2007], les actions sont spécifiées par des prédicats instanciés soumis à des pré-conditions. En effet, ces dernières doivent être vérifiées au moment où l'on exécutera l'action. Dans ce travail, les préconditions ne concernent que les informations contextuelles connues à chaque état du CPS. Á cet effet, considérons $pre(a)$ la préconditions de l'action a, par exemple, $pre(a(\ell)) = \ell = \mathcal{L}(s)$ et plus particulièrement pour les action de communication, $pre(x!(v)) = pre(x?(v)) = (x \in \Lambda)$.

Remarque 5.2.1. *(Actions réalisables)*
Soit $pre(a)$ la pré-condition de l'exécution de l'action a. La transition $(s \xrightarrow{a} s') \in Tr$ est réalisable dans l'état s du $CPS(q)$ si et seulement si $pre(a) \subseteq \mathcal{L}(s) \cup \Lambda(q)$.

Reprenons la configuration du plan de l'agent Bob $[\overline{P_1}]$ dans q_1^B vu dans la table 4.5, où $[\overline{P_1}] = ((E_g, \widehat{P_g})|||(E_m, \widehat{P_m}))$. Une des traces possible dérivée à partir de l'état initial s_0 du $CPS(q_1^B)$ est :
$((E_g, \widehat{P_g})|||(E_m, \widehat{P_m}), \ell_2, \emptyset) \xrightarrow{getc} ((E'_g, \widehat{P_g})|||(E_m, \widehat{P_m}), \ell_2, \emptyset) \xrightarrow{\tau} ((E_m, \widehat{P_m}), \ell_2, \{\widehat{P_g}\}) \xrightarrow{move(\ell_1)}$
$((E'_m, \widehat{P_m}), \ell_1, \{\widehat{P_g}\}) \xrightarrow{meet} ((E''_m, \widehat{P_m}), \ell_1, \{\widehat{P_g}\}) \xrightarrow{\tau} (stop, \ell_1, \{\widehat{P_g}, \widehat{P_m}\})$.

5.3 La guidance contextuelle de l'agent

Dans un CPS, la fonction $\mathcal{T}(s)$ donne les terminaisons possibles des plans d'intentions dans l'état s, et par conséquent les intentions correspondantes. Afin de guider l'agent, le planning-process est capable de capturer dans le plan les traces du CPS maximisant le nombre de plans d'intentions de l'agent. Ces traces sont appelées *traces maximales*.

Définition 5. *(Trace maximale)*
Soient $\Sigma \subseteq 2^{Tr}$ l'ensemble de toutes les traces possibles de $CPS(q)$ et $\sigma \in \Sigma$ l'une de ces traces. La notion de trace maximale est basée sur la fonction $end : \Sigma \longrightarrow 2^{\widehat{\mathcal{P}}}$, spécifiant pour chaque trace σ, l'ensemble des plans d'intentions terminés. Une trace σ est dite maximale si et seulement si aucune trace σ' n'existe dans $CPS(q)$ tel que $|end(\sigma')| > |end(\sigma)|$. En effet, l'ensemble des traces maximales du CPS est noté Σ_{MAX}.

Par corrélation, une technique similaire peut être utilisée pour vérifier la consistance des intentions de l'agent. L'analyse d'une trace maximale σ dans $CPS(q)$ nous permet de capturer les deux cas extrêmes suivants :

- si $|end(\sigma)| = |I(q)|$, nous déduisons que les intentions de $I(q)$ sont consistantes,
- si $|end(\sigma)| = 0$, signifie qu'aucune intention n'est satisfaite, donc le plan de l'agent $\overline{P}$ dans q est inapproprié pour la réalisation de ses intentions.

5.3.1 Évaluation de quelques propriétés du plan

Le fait que le $CPS(q)$ est une structure de kripke, nous permet de vérifier quelques propriétés de la logique temporelle, telle que LTL (Linear Temporal Logic) [Gerth et al., 1996], CTL (Computational Tree Logic) [Emerson, 1990]. Grâce à l'étiquetage

contextuel des états du $CPS(q)$, les propriétés pouvant être vérifiées concernent la mobilité, la communication et les plans d'intentions terminés.

Ainsi, au niveau du $CPS(q_1^B)$ de la figure 5.2, si la formule CTL `AF(`ℓ_1`)` est vérifiée à `true`, cela signifie que l'agent franchira la localité ℓ_1 dans l'exécution de n'importe quelle trace possible du CPS. L'utilisation des techniques de model-checking est assez large puisque les propriétés possibles couvrent des différents types de propriétés. De ce fait, des services de model-checking basés sur le CPS pourraient renforcer la guidance contextuelle de l'agent.

5.3.2 Application de la guidance dans le scénario

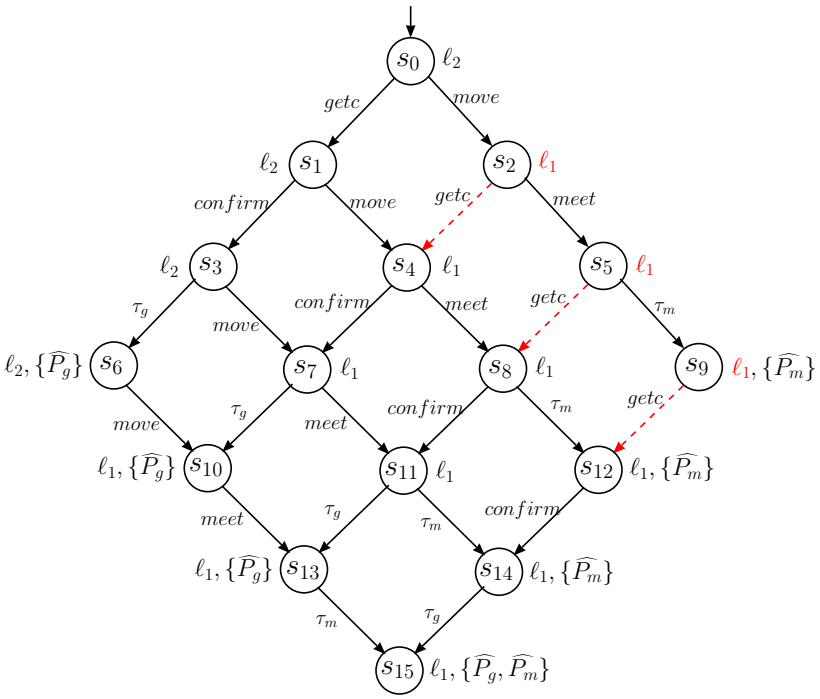

FIGURE 5.2: Le $CPS(q_1^B)$ correspondant au plan de l'agent Bob $\overline{P_1}$

Reprenons le scénario décrit dans la table 4.5 qui permet dà l'agent Bob de réaliser ses intentions de façon concurrente : $[\overline{P_1}] = ((E_g, \widehat{P_g})|||(E_m, \widehat{P_m}))$ représente la configuration du plan de l'agent Bob.

Le système de planification contextuelle $CPS(q_1^B)$ est illustré dans la figure 5.2 correspondant au plan de Bob dans sa configuration HoA q_1^B. Ce CPS est construit à partir de l'état initial $([\overline{P_1}], \ell_2, \emptyset)$. Dans la figure, les arcs en pointillés représentent

les transitions irréalisables à partir des états $s \in \{s_2, s_5, s_8\}$, car $pre(getC) = \{\ell_2\} \neq \mathcal{L}(s)$.

5.4 Discussion

Notre modèle centré agent est basé sur une gestion des plans, contrôlée étroitement à partir des attitudes mentales BDI. Cependant, contrairement aux approches antérieures, les intentions de l'agent peuvent être réalisées simultanément, à un niveau supérieur de la planification. Les conflits sont supposés être résolus par le mental-process de l'agent, par le biais d'une pondération des intentions. Par ailleurs, d'autres conflits peuvent surgir et être causés par le contexte changeant de l'agent, ainsi les informations contextuelles sont prises en compte lors de la construction du CPS.

Basée sur le CPS, la technique de guidance prend en compte l'ensemble des intentions de l'agent de façon concurrente lorsque cela est possible et voulu. La sémantique d'AgLOTOS est adaptée afin d'automatiser le processus de guidance qui est enrichi par un mécanisme d'apprentissage aidant l'agent à mieux prendre ses décisions, ce que nous traiterons dans le chapitre suivant.

Des travaux sur les dépendances entre les actions ont été déjà étudiés dans la littérature afin de mieux gérer l'activité de l'agent. En particulier, le planificateur GraphPlan est capable de construire un plan global comme un flux d'actions menant à un sous-ensemble de désirs (buts) qui pourraient être exécutés en concurrence [Meneguzzi et al., 2007]. Toutefois, GraphPlan ne traite que certains cas d'ordonnancements entre les actions, car il suit une approche de pas temporel. En revanche, notre approche prend en compte tous les cas des entrelacements possibles générés à partir de la sémantique d'AgLOTOS.

Chapitre 6

Guidance spatio-temporelle

Nous avons présenté au cours du chapitre précédent une technique de guidance basée la structure du CPS. Cette guidance est enrichie, dans ce présent chapitre, par une approche d'apprentissage prenant en compte les changements de l'environnement.

Dans le but d'améliorer la performance de l'exécution de l'agent, le planning-process, en relation avec l'execution-process, est capable de garder la trace des expériences passées de celui-ci. Ces expériences sont acquises lors de l'exécution des actions, afin d'apprendre sur le gain de leurs réutilisation, dans un contexte spatial et temporel identique. Contrairement à une manière simpliste consistant à offrir une action à chaque état du CPS, nous proposons une approche plus globale permettant d'évaluer le gain de toute une trace maximale afin de mieux les ordonner. Cela nous permet de construire un CPS enrichi, appelé *CPS with Learning* (*CPS-L* en abrégé). En guise de résultat, la meilleure des décisions stratégiques possibles qui est prise par l'agent est guidée par ses préférences.

6.1 L'apprentissage des actions et acquisition des données

La performance d'une action a est évaluée en fonction d'un certain contexte, par ailleurs, nous nous focalisons sur la localité de l'agent. Chaque exécution d'une action a dans une localité donnée est considéré comme une *expérience (d'action)*.

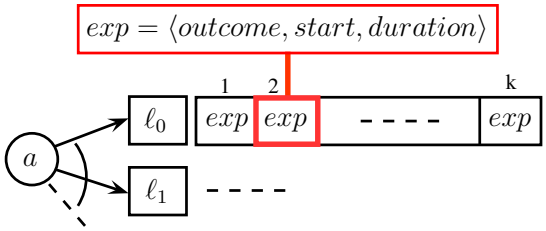

FIGURE 6.1: Expériences contextuelles (LCE) d'une action a

Définition 6. *(Expérience d'une action)*
Une expérience d'une action a dans une localité ℓ est un triplet $\langle outcome, start, duration \rangle$, où

- $outcome \in \{-1, 1\}$*, est le résultat d'une exécution passée de a, respectivement un échec ou un succès,*
- $start \in \mathbb{R}^+$*, est la date représentant le début de l'exécution de a,*
- $duration \in \mathbb{R}^+$*, est la durée d'exécution de a qui s'est terminée avec succès sinon indéfinie dans le cas inverse.*

La structure représentée dans la figure 6.1 est appelée *Expériences contextuelles apprises* (*en anglais, Learned Contextual Experiences - LCE*). Étant donné une action a, LCE_a représente des files FIFO des expériences de a, distinguant les différentes localités où l'action s'est exécutée. Les files LCE_a sont ordonnées par les dates $start$ des exécutions de a, sachant que la dernière expérience est celle qui est stockée dans la tête de la file.

Plus précisément, si une action a est exécutée dans une localité ℓ avec une certaine expérience exp, l'agent la rajoute dans la file, tel que $LCE_a(\ell) = LCE_a(\ell) \cup$

$\{exp\}$. En outre, k représente la taille effective de la file $LCE_a(\ell)$. Pour chaque expérience exp dans la file $LCE_a(\ell)$, nous notons par $index(exp)$ la position de exp dans cette file. Les trois composants d'une expérience exp sont respectivement dénotés par $exp.outcome$, $exp.start$, $exp.duration$.

6.2 Les stratégies de pertinence des données

Les stratégies spécifiées par l'agent sont données par la définition suivante et décrites dans ce qui suit.

Définition 7. *(Stratégies de l'agent)*
Les files de LCE sont paramétrées par la stratégie de l'agent $\mathcal{S} = \langle K, forget, M, \mathcal{C}, filter \rangle$ *tels que,*

- K *est la taille maximale des files,*
- $forget : 1..K \rightarrow \mathbb{R}^+$ *est la fonction d'oubli permettant d'associer un poids de pertinence à chaque expérience,*
- $M \leq K$ *est le nombre maximum des expériences filtrées,*
- $\mathcal{C}$ *est la classification périodique adoptée par les files, telle qu'une classification journalière (daily), hebdomadaire (weekly), mensuelle (monthly) ou bien annuelle (annually). C'est à dire, une opération de modulo est appliquée sur les dates de débuts des expériences afin de les ordonner en fonction de la classification adoptée,*
- $filter$ *est une fonction de filtrage par rapport au temps défini pour chaque file, donnant une sous-file de* M *expériences en fonction de la classification* $\mathcal{C}$*,*

6.2.1 La stratégie d'oubli

En raison de la dynamicité permanente des systèmes AmI, nous adoptons une heuristique qui considère les expériences les plus récentes comme étant les plus pertinentes. Chacune des files associées à une action est paramétrée par le couple $(forget, K)$ où f est une *fonction* dite *de pertinence* et K est la taille maximum de la file.

TABLE 6.1: $LCE_{getc}(\ell_2)$: LCE de l'action $getc$ dans la localité ℓ_2

$index$	1		5	6		10	11	12		18	k
$LCE_{getc}(\ell_2)$	...		A	B	...	C	D	E	...	F	...

$filter_{6,daily}(10)$

Expériences	$outcome$	$mod_{daily}(start)$	$duration$	$index$	$forget$
A	-1	8.29	-	5	0.20
B	1	9.50	3	6	0.16
C	1	9.05	2.10	10	0.10
D	1	10.78	3.40	11	0.09
E	-1	10.05	-	12	0.08
F	1	11.05	5.12	18	0.05

Au niveau de chaque file de LCE_a, la fonction d'oubli associe un *poids de pertinence* pour chaque expérience stockée dans la file. Un cas intéressant illustré dans la table 6.1 où la fonction d'oubli $forget(index) = \frac{1}{index}$ est utilisée, en considérant comme pertinente toute expérience stockée dans une position supérieure dans la file par rapport à une autre.

Par exemple, l'expérience C stockée dans la position 10 dans la file $LCE_{getc}(\ell_2)$ a un coefficient d'oubli $forget(10) = 0.10$, tandis que l'expérience F stockée dans la position 18 a un coefficient $forget(18) = 0.05$.

Observons que chaque file $LCE_a(\ell)$ est bornée par K éléments. Au-delà de K expériences, ces dernières sont oubliées, cela nous permet de remédier à l'explosion combinatoire des données stockées dans LCE. En effet, dans le cas où une file est pleine, l'ajout d'une nouvelle expérience dans la file provoque la suppression de la plus ancienne (celle qui est stockée à la position K de la file).

6.2.2 La stratégie de filtrage par le temps

Considérons une file $LCE_a(\ell)$, afin d'appliquer la sélection des expériences, leurs dates de débuts ainsi que la *date courante* '$date$' sont évaluées en utilisant la classification périodique $\mathcal{C}$. Á cet effet, nous introduisons la fonction $mod : \mathcal{C} \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$, tel que $mod_{\mathcal{C}}(date)$ correspond à la date courante modulo la classification périodique $\mathcal{C}$. Par ailleurs, la date peut être écrite sous forme textuelle ou bien comme une valeur réelle `'timestamp'`. Par exemple, si $date$ = `'Lundi 10 Février 2015,`

`10:00'`, alors $t = mod_{daily}(date) = 10$ tandis que $mod_{weekly}(date) =$`'Lundi 10'`. Le filtrage des expériences de la file permet de considérer uniquement celles qui ont les plus petits intervalles de temps avec la date courante $date$ suivant la classification périodique $\mathcal{C}$, tel que $|t - mod_{\mathcal{C}}(exp.start)|$. La fonction $filter_{M,\mathcal{C}}(t)$ appliquée à la file $LCE_a(\ell)$ permet de sélectionner M expériences. Cependant, dans le cas où M est supérieur à k, toutes les expériences de la file sont considérées.

Dans la table 6.1, le filtrage appliqué à $LCE_{getc}(\ell_2)$, est $filter_{6,daily}(10) =$ {A, B, C, D, E, F} signifiant que cette file est filtrée par les 6 expériences les plus proches de $t = mod_{daily}(date) = 10$, comme il est illustré dans la figure 6.2. Selon une vue graphique, en appliquant l'opération modulo sur la file, cette dernière est vue comme un ruban spiral, où les anneaux du ruban correspondent à des périodes successives, des jours dans notre exemple. Journalièrement parlant, les expériences A, B et C se sont produites avant t, tandis que celles de D, E et F se sont produites après t.

Dans cet exemple, nous avons pris comme stratégie, la configuration suivante : $\mathcal{S}_1 = (20, \frac{1}{index}, 6, daily, filter)$.

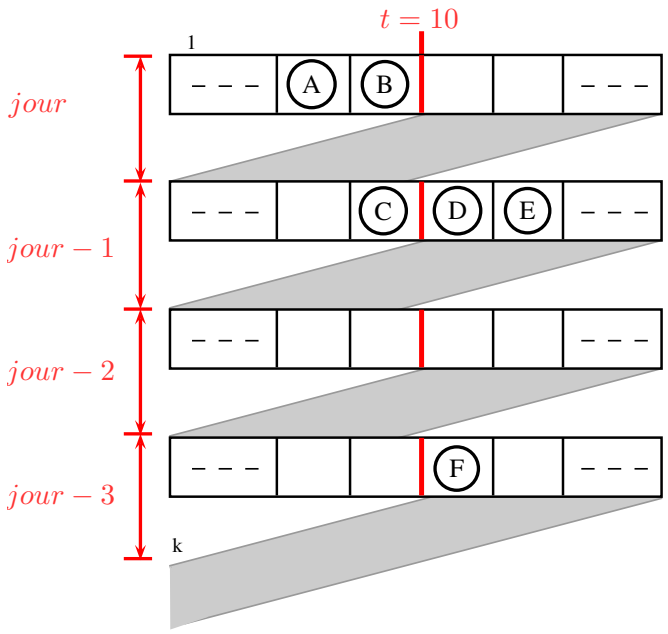

FIGURE 6.2: Un exemple d'une file filtrée $LCE_a(\ell)$

6.3 Calcul de la performance et de la durée prévues d'une action

Pour chaque file non-vide $LCE_a(\ell)$, la performance prévue $EP_a(\ell) \in [-1, 1]$ représente la performance de l'action a dans la localité ℓ, en se basant sur M expériences sélectionnées à partir de $LCE_a(\ell)$ tel que :

$$EP_a(\ell) = \frac{\sum_{exp}^{filter_{M,C}(t)} exp.outcome * forget(index(exp))}{\sum_{exp}^{filter_{M,C}(t)} forget(index(exp))}$$

Pour exécuter une action a dans ℓ, plus sa performance prévue $EP_a(\ell)$ est proche de '1', plus la chance de son succès est grande, tandis que plus $EP_a(\ell)$ est proche de -1', plus le risque de son échec est grand. Par ailleurs, $LCE_a(\ell) = \emptyset$ signifie que l'exécution de a dans ℓ n'a jamais été explorée. Ainsi, les actions non-explorées sont privilégiées par rapport à celles déjà explorées et ayant un gain $EP_a(\ell) < 0$.

Pour chaque file non-vide $LCE_a(\ell)$, la durée prévue $ED_a(\ell) \in \mathbb{R}^+$ correspond à la moyenne des durées effectives des M expériences sélectionnées, tel que :

$$ED_a(\ell) = \frac{\sum_{exp}^{filter_{M,C}(t)} exp.duration * forget(index(exp))}{\sum_{exp}^{filter_{M,C}(t)} forget(index(exp))}$$

Pour revenir à notre exemple, en prenant en considération $t = 10$, nous obtenons $EP_{getc}(\ell_2) = 0.19$ et $ED_{getc}(\ell_2) = 3.16$, (correspondant à `'3h10mn'`). Observons que l'expérience F a un faible impact sur $ED_{getc}(\ell_2)$ malgré son importante durée (5.12). Á cet effet, la fonction d'oubli, appliquée à sa 18ème position dans la file, la rend négligeable comparée aux autres expériences sélectionnées qui sont stockées à des positions inférieures.

6.4 La guidance spatio-temporelle à partir des expériences passées

6.4.1 Le CPS enrichi par l'apprentissage (CPS-L)

Héritant des traces maximales Σ_{MAX} du CPS, la structure du *CPS-L* est augmentée par les différentes valeurs $EP_a(\ell)$ et $ED_a(\ell)$ pour toute action a exécutée dans ℓ.

Définition 8. *(CPS enrichi par l'apprentissage)*
Soit Σ_{MAX} l'ensemble des traces maximales du CPS généré par le planning-process à partir du plan d'agent. Le CPS enrichi par l'apprentissage[1]*, appelé* CPS-L *est un triplet $\langle CPS, \mathcal{EP}, \mathcal{ED} \rangle$ tels que :*

- *$CPS = \langle S, s_0, Tr, \mathcal{L}, \mathcal{T} \rangle$ et $\Sigma_{MAX} \subseteq 2^{Tr}$ est l'ensemble des traces maximales du CPS.*
- *$\mathcal{EP}$ est une fonction définie dans Tr vers $[-1, 1]$, appliquée pour toute transition $tr = (s, a, s') \in \Sigma_{MAX}$ où $\mathcal{EP}(tr) = EP_a(\mathcal{L}(s))$,*
- *$\mathcal{ED}$ est une fonction définie dans Tr vers $\mathbb{R}^+$, appliquée pour toute transition $tr = (s, a, s') \in \Sigma_{MAX}$ où $\mathcal{ED}(tr) = ED_a(\mathcal{L}(s))$.*

Á partir du CPS-L, il est facile d'étendre les *valeurs de qualité prévues* des transitions à chaque trace maximale $\sigma \in \Sigma_{MAX}$.

$$QP(\sigma) = \frac{\sum_{tr}^{\sigma} \mathcal{EP}(tr)}{|\sigma|}$$

$$QD(\sigma) = \sum_{tr}^{\sigma} \mathcal{ED}(tr)$$

Afin de comparer les traces entre elles, nous proposons une normalisation entre $[-1, 1]$ appliquée à leurs valeurs de qualité, où '-1' représente la plus mauvaise qualité et '1' la meilleure. Cela est déjà fait pour QP, et dans le but de normaliser QD, nous considérons les valeurs extrêmes $QD_{min} = min(QD(\sigma) \mid \sigma \in \Sigma_{MAX})$

[1] CPS-L : Contextual Planning System with Learning

et $QD_{max} = max(QD(\sigma) \mid \sigma \in \Sigma_{MAX})$. Nous obtenons ainsi la formule calculant les *qualités de durées normalisées* :

$$NQD(\sigma) = 1 - \left(\frac{QD(\sigma) - QD_{min}}{QD_{max} - QD_{min}} * 2 \right)$$

Dans la figure 6.3 qui représente le CPS enrichi de la figure 5.2, les transitions des deux traces maximales σ_1 et σ_2, démarquées par des arcs gras, sont augmentées par les deux valeurs EP et ED. Par exemple, les valeurs $EP_{getc} = 0.19$ et $ED_{getc} = 3.16$ attachées à la transition $(s_0, getc, s_1)$, sont calculées à partir des expériences stockées dans $LCE_{getc}(\ell_2)$ de la table 6.1.

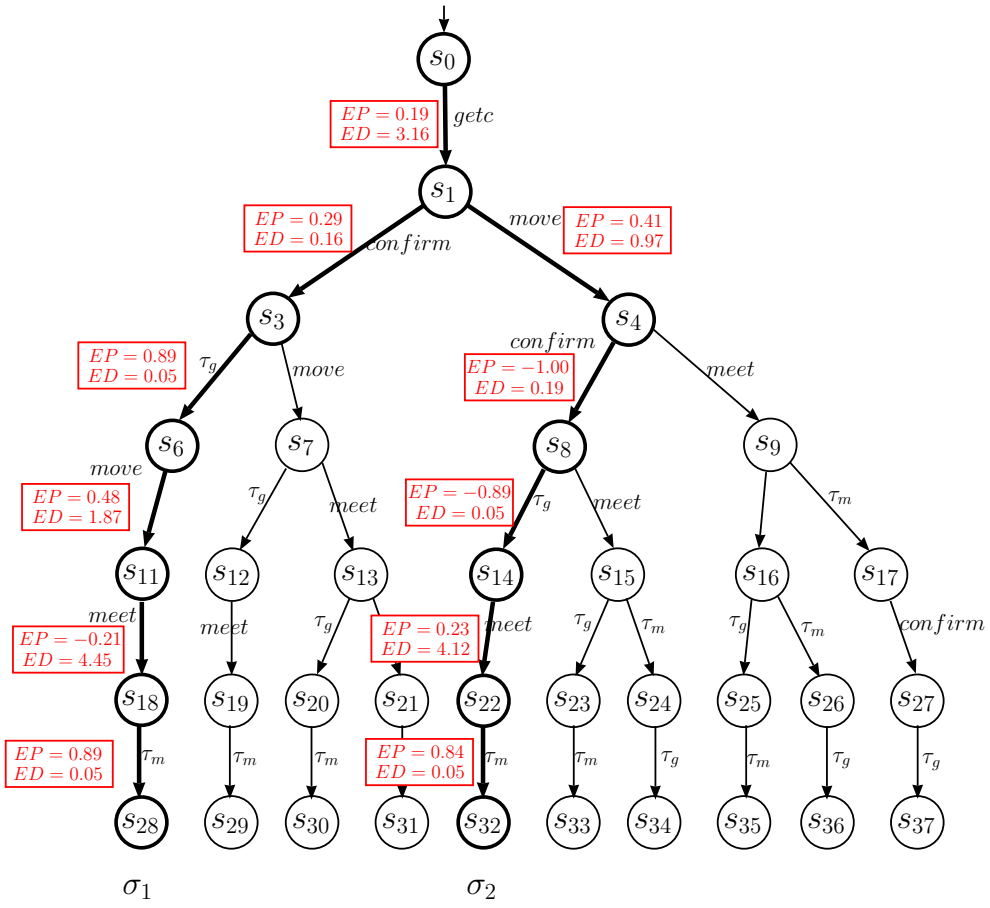

FIGURE 6.3: Les traces maximales ($CPS\text{-}L_B$) pour le plan $\overline{P_B}$

Comme il est montré dans la figure 6.4, les traces qui se trouvent dans la partie supérieure droite sont les meilleures, tandis que celles de la partie inférieure

gauche sont les plus mauvaises. En revanche, les deux autres parties de la figure (supérieure gauche et inférieure droite) représentent les traces qui ont une seule meilleure valeur parmi QP et NQD.

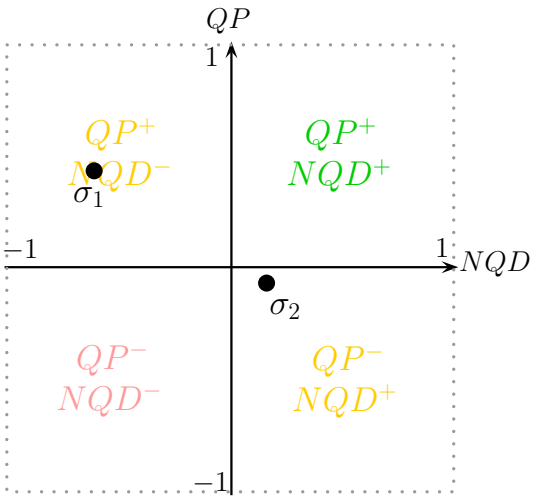

FIGURE 6.4: La balance entre $QP(\sigma)$ et $NQD(\sigma)$ des traces maximales

6.4.2 Les traces optimales du CPS-L

En plus de définir la stratégie $\mathcal{S}$, le mental-process est capable de spécifier une *balance de préférence* $\mathcal{B}$ entre les deux qualités caractérisant les traces (la performance et la durée normalisée). Cela permet de définir la trace (maximale) optimale à sélectionner, nommée σ_{opt}.

La balance est exprimée comme une proportion à appliquer sur les valeurs QP et NQD, afin d'obtenir une valeur de qualité globale comparable $Q_{\mathcal{B}}(\sigma) \in [-1, 1]$ pour toutes les traces maximales ($\sigma \in \Sigma_{MAX}$). Cette trace optimale σ_{opt} est offerte par le planning-process pour être exécutée au niveau de l'execution-process.

$$Q_{\mathcal{B}}(\sigma) = \mathcal{B}_{\mathcal{P}} * QP(\sigma) + \mathcal{B}_{\mathcal{D}} * NQD(\sigma)$$

La table 6.2 illustre les valeurs de qualité calculées pour les deux traces maximales σ_1 et σ_2 de la figure 6.3, sachant que pour toutes les traces maximales, nous obtenons $QD_{min} = 7.58$ et $QD_{max} = 10.23$. Á titre d'exemple, la balance $\mathcal{B}1$, adoptée par le mental-process, est représentée par la paire ($\mathcal{B}1_P = 0.30$, $\mathcal{B}1_D = 0.70$).

TABLE 6.2: Les valeurs de qualité calculées pour les traces maximales σ_1 et σ_2

Σ_{MAX}	QP	QD	NQD	$Q_{\mathcal{B}1}$	$Q_{\mathcal{B}2}$
σ_1	0.42	9.74	-0.63	-0.31	0.10
σ_2	-0.04	8.54	0.27	0.17	0.05

Étant donné que $Q_{\mathcal{B}1}(\sigma_2)$ est supérieure à $Q_{\mathcal{B}1}(\sigma_1)$, la trace maximale σ_2 est offerte à l'execution-process comme étant la trace optimale.

En revanche, en considérant la balance $\mathcal{B}2$ ($\mathcal{B}2_P = 0.70$, $\mathcal{B}2_D = 0.30$), c'est σ_2 qui est offerte du fait que $Q_{\mathcal{B}2}(\sigma_1) > Q_{\mathcal{B}2}(\sigma_0)$.

L'algorithme 2 synthétise notre processus de guidance en partant des intentions de l'agent définies par le mental-process, le planning-process construit sémantiquement le plan d'agent. Par la suite, un CPS est généré en utilisant une sémantique opérationnelle adaptée capturant toutes les traces possibles du plan de l'agent. Puis, le module de guidance ne prend en considération que les traces maximales afin de retenir que la plus optimale d'entre elles en fonction des expériences passées des actions.

Algorithme 2 Le processus de planification et de guidance de l'agent

1: **Prérequis :**
   I : ensemble d'intentions pondéré ;
2: construire un plan d'agent $\overline{P}$ en utilisant la fonction $plan(I)$;
3: générer le CPS à partir de $\overline{P}$;
4: extraire les traces maximales Σ_{MAX} à partir du CPS ;
5: sélectionner une trace σ parmi celles de Σ_{MAX} ;
6: offrir la trace σ ;

6.5 Discussion

Les techniques d'apprentissage permettent d'améliorer le comportement des agents intelligents grâce à leurs expériences passées, cela est appliqué à différents niveaux de l'agent. Dans un premier temps, l'apprentissage a été investi au niveau mental de l'agent afin d'optimiser la délibération BDI à partir des connaissances apprises, tel qu'il a été proposé par Merke et Riedmiller [2002], ou de produire de

nouveaux plans respectant certains objectifs, développé, entre autres, par Karim et al. [2006].

Ces techniques ont été utilisées aussi pour le renforcement de la sélection des plans parmi les alternatives possibles de l'agent. L'arbre de décision était introduit, particulièrement par Guerra-Hernandez et al. [2005] pour représenter les différents contextes dans lesquels l'agent peut s'y trouver. En effet, le comportement de l'agent est appris à partir des succès et des échecs de ses plans. L'idée de tirer profit des expériences passées, relatives aux actions exécutées, a été adaptée dans [Airiau et al., 2008]. Cela met en évidence une technique en ligne, basée sur la structure hiérarchique But-Plan (Goal-Plan), afin de sélectionner un plan alternatif, parmi ceux qui sont possibles, en fonction des échecs appris.

Le travail vu dans [Nunes et Luck, 2014] est consacré à la sélection des plans BDI. Les auteurs de cette recherche proposent d'évaluer la contribution de chaque plan en termes d'utilités et préférences, vues comme des paramètres d'optimisation. D'une manière similaire à ce travail, nous avons employé des utilités pour ordonner et sélectionner certaines traces maximales. Ces utilités correspondent à une stratégie multi-objectif basée sur des proportions de performances ainsi qu'un filtrage temporel des informations. Notons que les traces maximales du CPS-L dépendent plus des expériences réelles que de probabilités préétablies. En effet, ce travail montre comment l'agent tire partie de ses expériences passées, réussies ou échouées, pour améliorer ses prochaines exécutions.

Concernant les deux approches récentes [Nunes et Luck, 2014; Waters et al., 2014b], la validation des techniques proposées requiert un nombre important et considérable d'expériences relatives au calcul des probabilités puisque elles sont basées sur celles-ci. En revanche, notre technique est dirigée par la maximisation de la satisfaction des intentions, afin de guider l'agent à travers l'ensemble des traces caractérisant la concurrence des actions.

En se basant sur les réussites et les échecs des actions, déjà exécutées, Le CPS-L est vu comme un apprentissage par renforcement en vu de la sélection des plans élémentaires. En fait, la sélection d'une trace implique la sélection d'un plan alternatif pour chaque plan d'intention. Les files, stockant les expériences passées des actions, sont bornées en utilisant des stratégies d'oubli et de filtrage afin de faire face à la complexité combinatoire due à leur gestion et exploitation.

Récemment, dans [Guivarch et al., 2013] une technique d'apprentissage par renforcement consacrée au systèmes ambiants et domotiques a été introduite. La plate-forme proposée est capable d'apprendre les actions des humains en enregistrant un ensemble de perceptions pour chacune d'elles. En effet, la plate-forme réagit à la place des humains. Le fait que cela fonctionne par un rapprochement, donne quelques similitudes avec notre technique qui prend en compte les échecs d'actions, sans prendre en considération, toutefois, le plan de l'agent ou même son état mental.

Parmi les approches d'apprentissage alternatives basées sur un système probabiliste, le processus de décision markovienne (Markov Decision Process) a été présenté par Simari et Parsons [2006] pour l'apprentissage par renforcement dans les agents BDI ; tandis qu'une fonction de probabilité a été proposée par Singh et al. [2010] pour exprimer une certaine capacité d'explorer de nouveaux plans, au lieu d'exploiter des plans connus pour leurs réussites. Néanmoins, les deux approches sont basées sur un nombre très important d'expériences répétitives, ce qui est inapproprié pour les systèmes ambiants où les agents évoluent en temps réel dans un environnement dynamique. Aussi, l'assistanat des utilisateurs fournie un nombre très limité d'expériences.

Le fait que notre approche ne prenne en compte que les expériences les plus récentes des actions, ce qui semble appropriée aux systèmes ambiants, la fonction d'oubli appliquée aux expériences connues s'accorde avec les changements permanents dans l'environnement de l'agent.

Enfin, dans cette proposition, nous nous focalisons principalement sur l'aspect spatial et temporel lors des exécutions des actions. Les plans que nous considérons sont construits dynamiquement en raisonnant sur des actions situées ainsi que sur leur contexte (durée, début, localité, etc). Ce qui s'avère précieux à l'évolution de l'agent dans un environnement dynamique. Les traces maximales obtenues à partir du CPS dépendent non seulement des plans de l'agent, mais aussi de son contexte.

CHAPITRE 7

Conclusion

Cette thèse propose fondamentalement un modèle comportemental d'agent BDI intelligent, lui permettant d'évoluer dans un environnement ambiant et ouvert. La dynamicité de l'agent et sa sensibilité aux changements de contexte va inciter l'agent BDI à réviser ses intentions et suivre les évolutions qui en résultent.

L'axe principal de la thèse a été de définir la manière dont l'agent peut adapter son plan de façon formelle et optimale, en adéquation avec différents paramètres :

- les intentions de l'agent et ses priorités internes,
- les préférences de l'utilisateur sur la qualité à apporter en terme de sûreté ou de performances,
- des paramètres d'ajustement liés au contexte de l'agent et son analyse de l'environnement.

Contrairement aux travaux existants dans les SMA, les contributions de cette thèse sont fondées sur la base d'une séparation claire entre le processus mental et le processus de planification de l'agent. Nous avons pu élaborer une approche formelle dont les processus sont activés. Par contre, la gestion du contexte et l'in-

teraction concrète avec l'utilisateur, qui sortent du cadre d'étude de cette thèse, sont abstraites sous forme de considérations et de paramétrages généraux. Le modèle HoA qui est proposé, traduit la réalité des agents BDI ambiants qui évoluent non seulement par des perceptions environnementales mais par des évolutions des plans réalisant les intentions de l'agent.

En accord avec les principes fondamentaux des systèmes ambiants, cette thèse se différencie des travaux traditionnellement considérés pour les SMA. En particulier, les contextes des agents ne sont pas nécessairement prédéfinis et les changements de contexte sont inattendus. Ce travail s'appuie, cependant, sur des perceptions spatio-temporelles dont le but est d'abstraire raisonnablement les données inconnues.

Les contributions de cette thèse sont plurielles :

1) Un langage algébrique de spécification des plans d'agent

Le langage AgLOTOS et sa sémantique opérationnelle définissent formellement l'activité de l'agent suivant un modèle de plans concurrents décrits sous formes de processus concurrents LOTOS. Le fait que AgLOTOS soit structuré, il permet de façon tout à fait originale de capturer les relations de priorité entre intentions, les relations entre plans des intentions et actions opérationnelles dans les plans. Suivant le niveau, les opérateurs de parallélisme et de séquence sont observés d'un point de vue relationnel (de dépendance partielle entre intentions) ou opérationnel (concernant l'exécution des actions dans les plans élémentaires). Un opérateur algébrique a été formellement introduit pour spécifier une alternativité des plans destinée à renforcer la réalisation des plans d'intentions.

2) Une révision ciblée et automatique de la spécification

La structure des plans qui ressort des spécifications AgLOTOS a permis d'élaborer une technique de révision partielle des plans, tout en préservant la consistance des plans et des intentions considérées. La révision des plans est orchestrée par le processus mental, sous l'effet de nouvelles perceptions ou en fonction de la terminaison des plans. La structure hiérarchique proposée pour les règles de dérivation de la spécification algébrique permet, de façon formelle, de prendre en compte les échecs et les réussites des exécutions, afin d'impacter les intentions cor-

respondantes. Il est donc possible de faire évoluer les intentions ou leurs degrés de priorités à la volée.

3) Un modèle sémantique d'actions concurrentes exécutées en contexte

Le CPS est un système d'états-transitions automatiquement construit à partir d'une spécification AgLOTOS. Sa construction particulière est contrainte par le contexte spatial de l'agent. Le CPS offre ainsi une représentation exacte des capacités de réalisation des intentions, en concurrence et en contexte. La consistance des plans vis à vis des intentions est donc trivialement assurée. Ce modèle est orienté-action mais permet, par extension, de spécifier l'ensemble des plans possibles. Comparé à l'outil GraphPlan [Meneguzzi et al., 2007] qui se charge d'évaluer un sous ensemble de buts consistants à partir d'actions offertes par le contexte courant, le CPS offre la possibilité de connaître tous les sous ensembles consistants d'intentions, en fonction du contexte.

4) Un service de guidance centré sur la satisfaction des intentions de l'agent

Sur la base du CPS, un service original et automatique de guidance est proposé, maximisant la réalisation des intentions de l'agent en fonction de son contexte. Ce service de guidance constitue en fait un véritable oracle pour l'agent, pouvant pronostiquer la meilleure façon de réaliser toutes les intentions quand le contexte spatio-temporel le permet.

5) Un mécanisme d'apprentissage adapté aux changements de contexte

Un mécanisme d'apprentissage est proposé pour améliorer la guidance de l'agent. Il est fondé sur des informations spatio-temporelles concernant les actions, acquises des expériences d'exécution passées. Au regard des approches existantes, notre technique s'avère un compromis utile et adapté aux agents ambiants et aux changements de leur contexte. Contrairement aux approches des SMA non ambiants qui fondent leur apprentissage sur un modèle prédéfini de probabilités sur les plans, l'approche proposée reprend certains fondements du Q-Learning [Watkins, 1989] qui définit à la volée la qualité des actions offertes en chaque état, avec cependant des différences importantes. Du fait des changements de contexte qui peuvent s'avérer totalement inattendus, le processus d'évaluation ne peut plus être considéré comme markovien (sans mémoire). La procédure d'échantillon-

nage des expériences est fondée sur une connaissance explicite des expériences les plus pertinentes en fonction du contexte spatio-temporel de l'agent.

La sélection des expériences est traitée séparément de l'évaluation des gains estimés résultant d'exécution. De façon originale, cela permet de considérer tant les échecs que les réussites concernant l'évaluation de la pertinence d'une action. De plus, l'exploration des actions pour lequel il n'existe pas d'expérience pertinente sont considérées à un moyen terme entre les expériences connues pour être positives et celles négatives. D'un point de vue temporel, la sélection peut dépendre du moment d'exécution et de la qualité temporelle, possiblement périodique, des exécutions des actions. Enfin, l'évaluation du gain comparé des actions n'est considérée qu'à posteriori pour tenir compte des préférences de l'agent dans la sélection des meilleurs plans. Des compromis entre la sûreté ou la performance des plans ont été ainsi formalisés et automatisés.

Ouverture et perspectives

Les mécanismes proposés dans cette thèse sont non seulement opérationnels mais se veulent résiliants face aux changements de contexte. Étant centrés sur l'agent lui-même, l'agent ainsi construit pourra être exploité en tant qu'élément fondateur dans des configurations de systèmes multi-agents variés, qu'ils soient de nature hiérarchique ou simplement distribué.

Les résultats de cette thèse pourraient être étendus pour traiter des systèmes plus concrets. Une étude intéressante devra être menée pour tenir compte de variations inattendues des préférences de l'utilisateur et aussi rendre explicite certains paramètres contextuels complémentaires des critères spatio-temporels, par exemple les relations sociales entre les agents, ou entre les agents et les utilisateurs.

Index

Acronymes

2APL (Double) Agent-oriented Programming Language
ACP Algebra of Communicating Processes
AgLOTOS Agent LOTOS
AmI Ambient Intelligence (Intelligence Ambiante)
BDI Beliefs, Desires, Intentions
CCS Calculus of Communicating Systems
CPS Contextual Planning System
CPS-L Contextual Planning System with Learning
CSP Communicating Sequential Processes
CTL Computational Tree Logic
HoA Higher-order Agent
HTN Hierarchical Task Network
IAD Intelligence Artificielle Distribuée
LCE Learned Contextual Experiences
LibP Library of Plans
LOTOS Language of Temporal Ordering Specification
LTL Linear Temporal Logic
PRS Procedural Reasoning System
SMA Système multi-agent

Liste des publications

Saïdouni, D. E., Chaouche, A.-C. and Ilié, J.-M. (2012). On the Fly PSO Inspired Algorithm For Graph Distribution. In Proceedings of MISC'12, the 2nd International Symposium on Modelling and Implementation of Complex Systems, May 20-21, Constantine, Algeria, pages 103-110.

Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2013). Towards a MAS approach to model Ambient Systems. In the 8th NII-LIP6 Work-Shops on Multi-Agent and Distributed Systems, June 10-11, Paris, France.

Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2013). A Higher-order Agent Model for Ambient Systems. In EUSPN'13, the 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks, October 21-23, Niagara falls, Canada, volume 21 of Procedia Computer Science, pages 156-163. Elsevier. ISSN : 1877-0509.

Belala, N., Saïdouni, D. E., Boukharrou, R., Chaouche, A.-C., Seraoui, A. and Chachoua, A. (2013). Time Petri Nets with Action Duration : A True Concurrency Real-Time Model. International Journal of Embedded and Real-Time Communication Systems (IJERTCS), volume 4(2), pages 62-83, April-June. IGI Global. ISSN : 1947-3176.

Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2014). A Dynamical Plan Revising for Ambient Systems. In ANT'14, the 5th International Conference on Ambient Systems, Networks and Technologies, June 2-5, Hasselt, Belgium, volume 32 of Procedia Computer Science, pages 37-44. Elsevier. ISSN : 1877-0509.

Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2014). From Intentions to Plans : A Contextual Planning Guidance. In Camacho, D., Braubach, L., Venticinque, S. and Badica, C., Proceedings of IDC'2014, the 8th International Symposium on Intelligent Distributed Computing, September 3-5, Madrid, Spain, volume 570 of Studies in Computational Intelligence, pages 403-413. Springer. ISBN : 978-3-319-10421-8.

Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2014). A Higher order Agent Model with Contextual Management for Ambient Systems. In Transactions on Computational Collective Intelligence - TCCI XVI (Journal Subline), volume 8780, Lecture Notes in Computer Science, Kowalczyk, R. and Nguyen, N.-Th. (Eds.), pages 146-169. Springer. ISBN : 978-3-662-44870-0.

Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2014). Dealing with Dynamicity in Ambient Systems : A Formal Planning Approach. In the 10th NII-LIP6 WorkShops on Multi-Agent and Distributed Systems, September 24-25, Paris, France.

Boukharrou, R., Chaouche, A.-C., A., Ilié, J.-M. and Saïdouni, D. E. (2014). Contextual-Timed Planning Management for Ambient Systems. In ICTAI'14, the 26th IEEE International Conference on Tools with Artificial Intelligence, November 10-12, Limassol, Cyprus, pages 107-114. IEEE Computer Society. ISSN : 1082-3409.

Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2014). A Formal Approach for Contextual Planning Management : Application to Smart Campus Environment. In Bazzan, A. and Pichara, K., Proceedings of IBERAMIA'2014, the 14th Ibero-American Conference on Artificial Intelligence, November 24-27, Santiago, Chile, volume 8864 of Lecture Notes in Artificial Intelligence, pages 791-803. Springer. ISBN : 978-3-319-12026-3.

Boukharrou, R., Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2015). Dealing with Temporal Failure in Ambient Systems : A Dynamic Revision of Plans. In Journal of Ambient Intelligence and Humanized Computing (JAIHC), volume 6(3), pages 325-336. Springer. ISSN : 1868-5137.

Chaouche, A.-C., El Fallah-Seghrouchni, A., Ilié, J.-M. and Saïdouni, D. E. (2015). Spatio-Temporal Guidance for Ambient Agents. In Proceedings of CSCS'2014, the 20th International Conference on Control Systems and Computer Science, May 27-29, Bucharest, Romania. IEEE Computer Society. ISBN :978-1-4673-6140-8.

Bibliographie

Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, et Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC'99, pages 304–307, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-66550-1.

Stéphane Airiau, Lin Padgham, Sebastian Sardina, et Sandip Sen. Incorporating learning in BDI agents. In *Proceedings of ALAMAS - ALAg*, 2008.

Tina Balke, Frank Dignum, M. Birna van Riemsdijk, et Amit K. Chopra, editors. *Coordination, Organizations, Institutions, and Norms in Agent Systems IX - COIN 2013 International Workshops, COIN@AAMAS, St. Paul, MN, USA, May 6, 2013, COIN@PRIMA, Dunedin, New Zealand, December 3, 2013, Revised Selected Papers*, volume 8386 of *Lecture Notes in Computer Science*, 2014. Springer. ISBN 978-3-319-07313-2.

Nicolas Baskiotis et Nicolas Ferey. Introduction à l'ambient intelligence (ami), 2003. Cours.

J.A. Bergstra et J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(0) :77 – 121, 1985. ISSN 0304-3975.

Avrim L. Blum et Merrick L. Furst. Fast planning through planning graph analysis. *ARTIFICIAL INTELLIGENCE*, 90(1) :1636–1642, 1995.

Tommaso Bolognesi et Ed Brinksma. Introduction to the iso specification language lotos. *Computer Networks ISDN Systems*, 14(1) :25–59, March 1987. ISSN 0169-7552.

Rafael H. Bordini, Jomi Fred Hübner, et Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007. ISBN 0470029005.

M. E. Bratman. *Intentions, Plans and Practical Reason*. Harvard University Press, Cambridge, MA, USA, 1987.

Michael E. Bratman, David J. Israel, et Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4 :349–355, 1988.

E. Brinksma, editor. *ISO 8807, LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, 1988.

Juliette Brézillon et Patrick Brézillon. Context modeling : Context as a dressing of a focus. In Boicho Kokinov, DanielC. Richardson, ThomasR. Roth-Berghofer, et Laure Vieu, editors, *Modeling and Using Context*, volume 4635 of *Lecture Notes in Computer Science*, pages 136–149. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74254-8.

Luca Cardelli et AndrewD. Gordon. Mobile ambients. In Maurice Nivat, editor, *Foundations of Software Science and Computation Structures*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64300-5.

Guanling Chen et David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000.

Alonzo Church. *The Calculi of Lambda Conversion. (AM-6) (Annals of Mathematics Studies)*. Princeton University Press, Princeton, NJ, USA, 1985. ISBN 0691083940.

Philip R. Cohen et Hector J. Levesque. Intention is choice with commitment. *Artif. Intell.*, 42(2-3) :213–261, 1990.

Mehdi Dastani. 2apl : a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3), 2008. ISSN 1387-2532.

Mark d'Inverno, David Kinny, Michael Luck, et Michael Wooldridge. A formal specification of dmars. In Munindar P. Singh, Anand S. Rao, et Michael Wooldridge, editors, *ATAL*, volume 1365 of *Lecture Notes in Computer Science*, pages 155–176. Springer, 1997. ISBN 3-540-64162-9.

K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, et J. Burgelman. Scenarios for ambient intelligence in 2010, February 2001.

Hartmut Ehrig et B. Mahr. *Fundamentals of Algebraic Specification I*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1985. ISBN 0387137181.

E. A. Emerson. Temporal and Modal Logic. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 995–1072. Elsevier Science Publishers, North-Holland, 1990.

J. Ferber et O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 128–135, Jul 1998.

Adina Florea, Daniel Kayser, et Stefan Pentiuc. Cours : Agents intelligents, 2002. URL `http://turing.cs.pub.ro/auf2/html/chapters/chapter2/chapter_2_2_2.html`.

Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, et Didier Rémy. A calculus of mobile agents. In *Proceedings of the 7th International Conference on Concurrency Theory*, CONCUR '96, pages 406–421, London, UK, UK, 1996. Springer-Verlag. ISBN 3-540-61604-7.

Michael Friedewald, Elena Vildjiounaite, Yves Punie, et David Wright. The brave new world of ambient intelligence : An analysis of scenarios regarding privacy, identity and security issues. In JohnA. Clark, RichardF. Paige, FionaA.C. Polack, et PhillipJ. Brooke, editors, *Security in Pervasive Computing*, volume 3934 of *Lecture Notes in Computer Science*, pages 119–133. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33376-0.

Michael P. Georgeff et Francois Felix Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'89, pages 972–978, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

Michael Peter Georgeff et Anand S. Rao. A profile of the australian artificial intelligence institute. *IEEE Expert : Intelligent Systems and Their Applications*, 11(6) : 89–92, December 1996. ISSN 0885-9000.

Rob Gerth, Doron Peled, Moshe Y. Vardi, et Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification XV*, pages 3–18, London, UK, UK, 1996. Chapman & Hall, Ltd. ISBN 0-412-71620-8.

John Grant, Sarit Kraus, Donald Perlis, et Michael Wooldridge. Postulates for revising bdi structures. *Synthese*, 175(1) :39–62, 2010. ISSN 0039-7857.

Denis Grégoire, Arcangeli Jean-Paul, Noël Victor, Charles Triboulot, et Trouilhet Sylvie. Composition opportuniste et ascendante 'a base d'agents coopératifs. In *Journées francophones Mobilité et Ubiquité (UBIMOB 2012)*, page 196–209, 2012.

Alejandro Guerra-Hernandez, Amal El Fallah Seghrouchni, et Henry Soldano. Learning in BDI multi-agent systems. In *Computational Logic in Multi-Agent Systems*, volume 3259 of *LNCS*, pages 218–233. Springer, 2005. ISBN 978-3-540-24010-5.

Valérian Guivarch, Valérie Camps, et André Péninou. AMADEUS : an adaptive multi-agent system to learn a user's recurring actions in ambient systems. *ADCAIJ : Advances in Distributed Computing and Artificial Intelligence Journal*, 1(3), 2013. ISSN 2255-2863.

Valérian Guivarch, Valérie Camps, et André Péninou. Context awareness in ambient systems by an adaptive multi-agent approach. In Fabio Paternò, Boris Ruyter, Panos Markopoulos, Carmen Santoro, Evert Loenen, et Kris Luyten, editors, *Ambient Intelligence*, volume 7683 of *Lecture Notes in Computer Science*, pages 129–144. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34897-6.

Karen Henricksen et Jadwiga Indulska. Developing context-aware pervasive computing applications : Models and approach. *Pervasive Mob. Comput.*, 2(1) :37–64, February 2006. ISSN 1574-1192.

Koen Hindriks. Twenty years of engineering multiagent systems. In *Keynote of the EMAS workshop, part of AAMAS'14*, 2014.

C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8) :666–677, August 1978. ISSN 0001-0782.

Marcus J. Huber. Jam : A bdi-theoretic mobile agent architecture. In *Proceedings of the Third Annual Conference on Autonomous Agents*, AGENTS '99, pages 236–243, New York, NY, USA, 1999. ACM. ISBN 1-58113-066-X.

IEC 15437. International standard ISO, 2001. revised version of the LOTOS standard ISO8807.

InTech. Intelligence ambiante : évolution ou révolution ?, mai 2011. URL `http://www.inria.fr/centre/grenoble/agenda/seminaire-in-tech-intelligence-ambiante`.

N. R. Jennings et M. Wooldridge. Agent technology. chapter Applications of Intelligent Agents, pages 3–28. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998. ISBN 3-540-63591-2.

Samin Karim, Budhitama Subagdja, et Liz Sonenberg. Plans as products of learning. In *IAT'06*, pages 139–145, 2006. ISBN 0-7695-2748-5.

Joonas Kesäiemi et Vagan Terziyan. *Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies*, chapter Agent-Environment Interaction in MAS - Introduction and Survey. InTech, April 2011. ISBN 978-953-307-176-3.

Boicho Kokinov. A dynamic approach to context modeling. In P. Brezillon et S. Abu-Hakima, editors, *Proceedings of IJCAI'95 - Workshop on Modeling Context in Knowledge Representation and Reasoning*, volume 95, page 199–209, 1995.

Mari Korkea-Aho. Context-aware applications survey, 2000. URL `http://www.cse.tkk.fi/fi/opinnot/T-110.5190/2000/applications/context-aware.html`.

Magnus Ljungberg et Andrew Lucas. The oasis air traffic management system, 1992.

Marco Lützenberger, Tobias Küster, Thomas Konnerth, Alexander Thiele, Nils Masuch, Axel Heßler, Jan Keiser, Michael Burkhardt, Silvan Kaiser, et Sahin Albayrak. Jiac v : A MAS framework for industrial applications. In *Proceedings of AAMAS '13*, pages 1189–1190, 2013. ISBN 978-1-4503-1993-5.

Toufik Messaoud Maarouk. *Modèles formels pour la conception des systèmes temps réel*. PhD thesis, 2011.

S. Makonin, L. Bartram, et F. Popowich. A smarter smart home : Case studies of ambient intelligence. *Pervasive Computing, IEEE*, 12(1) :58–66, Jan 2013. ISSN 1536-1268.

Felipe Meneguzzi, Avelino Francisco Zorzo, Michael da Costa Móra, et Michael Luck. Incorporating planning into BDI agents. *Scalable Computing : Practice and Experience*, 8 :15–28, 2007.

A. Merke et M. Riedmiller. Karlsruhe brainstormers - a reinforcement learning approach to robotic soccer. In *RoboCup 2001 : Robot Soccer World Cup V*, volume 2377 of *LNCS*, pages 435–440. Springer, 2002. ISBN 978-3-540-43912-7.

R.Zalila Mili et Renee Steiner. Modeling agent-environment interactions in adaptive mas. In Danny Weyns, SvenA. Brueckner, et Yves Demazeau, editors, *Engineering Environment-Mediated Multi-Agent Systems*, volume 5049 of *Lecture Notes in Computer Science*, pages 135–147. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85028-1.

R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982. ISBN 0387102353.

R. Milner. *Communicating and Mobile Systems : The Pi-Calculus*. Cambridge University Press, Cambridge, UK, 1999.

Robin Milner. The polyadic pi-calculus : a tutorial. In F. L. Hamer, W. Brauer, et H. Schwichtenberg, editors, *Logic and Algebra of Specification*. Springer-Verlag, 1993.

Robin Milner, Joachim Parrow, et David Walker. A calculus of mobile processes, {II}. *Information and Computation*, 100(1) :41 – 77, 1992. ISSN 0890-5401.

Jörg P. Müller. *The Design of Intelligent Agents - A Layered Approach*, volume 1177 of *Lecture Notes in Computer Science*. Springer, 1996. ISBN 3-540-62003-6.

Ingrid Nunes et Michael Luck. Softgoal-based plan selection in model-driven bdi agents. In *AAMAS'14*, pages 749–756, 2014. ISBN 978-1-4503-2738-1.

Andrei Olaru, Adina Magda Florea, et Amal El Fallah Seghrouchni. Graphs and patterns for context-awareness. In *Ambient Intelligence - Software and Applications*, volume 92, pages 165–172. 2011. ISBN 978-3-642-19936-3.

Andrei Olaru, Adina Magda Florea, et Amal El Fallah Seghrouchni. A context-aware multi-agent system as a middleware for ambient intelligence. *MONET*, 18(3) :429–443, 2013.

Jason Pascoe. Adding generic contextual capabilities to wearable computers. In *Second International Symposium on Wearable Computers*, pages 92–99, Oct 1998.

Javier Bajo Pérez, Juan M. Corchado Rodríguez, Philippe Mathieu, Andrew Campbell, Alfonso Ortega, Emmanuel Adam, Elena Navarro, Sebastian Ahrndt, María N. Moreno, et Vicente Julián, editors. *Trends in Practical Applications of Heterogeneous Multi-Agent Systems*, volume 293 of *Advances in Intelligent Systems and Computing*, 2014. Springer. ISBN 978-3-319-07475-7.

Nayat Sánchez Pi, David Griol, Javier Carbó, et José M. Molina López. Evaluation of agents interactions in a context-aware system. *T. Computational Collective Intelligence*, 9 :79–97, 2013.

Alexander Pokahr, Lars Braubach, et Winfried Lamersdorf. Jadex : A bdi reasoning engine. In RafaelH. Bordini, Mehdi Dastani, Jürgen Dix, et Amal El Fallah Seghrouchni, editors, *Multi-Agent Programming*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 149–174. Springer US, 2005. ISBN 978-0-387-24568-3.

Martha E. Pollack et John F. Horty. Ther's more to life than making plans : Plan management in dynamic, multi-agent environments. *AI Magazine*, 20 :71–84, 1999.

A. S. Rao et M. Georgeff. BDI agents : from theory to practice. In *First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319, S. Francisco, CA, June 1995.

Anand S. Rao et Michael P. Georgeff. An abstract architecture for rational agents. In Bernhard Nebel, Charles Rich, et William R. Swartout, editors, *KR*, pages 439–449. Morgan Kaufmann, 1992. ISBN 1-55860-262-3.

N. S. Ryan, J. Pascoe, et D. R. Morse. Enhanced reality fieldwork : the context-aware archaeological assistant. In V. Gaffney, M. van Leusen, et S. Exxon, editors, *Computer Applications in Archaeology*, British Archaeological Reports, pages 182–196, Oxford, October 1998. Tempus Reparatum.

Davide Sangiorgi et David Walker. *PI-Calculus : A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA, 2001. ISBN 0521781779.

Sebastian Sardina, Lavindra de Silva, et Lin Padgham. Hierarchical planning in BDI agent programming languages : a formal approach. In *AAMAS '06*, pages 1001–1008, 2006. ISBN 1-59593-303-4.

B. Schilit, N. Adams, et R. Want. Context-aware computing applications. In *Workshop on Mobile Computing Systems and Applications*, pages 85–90, Dec 1994.

Steven Shapiro, Sebastian Sardina, John Thangarajah, Lawrence Cavedon, et Lin Padgham. Revising conflicting intention sets in BDI agents. In *AAMAS '12*, pages 1081–1088, 2012. ISBN 0-9817381-2-5, 978-0-9817381-2-3.

Ameya Shendarkar, Karthik Vasudevan, Seungho Lee, et Young-Jun Son. Crowd simulation for emergency response using {BDI} agents based on immersive virtual reality. *Simulation Modelling Practice and Theory*, 16(9) :1415 – 1429, 2008. ISSN 1569-190X.

Gerardo I. Simari et Simon Parsons. On the relationship between mdps and the bdi architecture. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 1041–1048, New York, NY, USA, 2006. ACM. ISBN 1-59593-303-4.

Dhirendra Singh, Sebastian Sardina, Lin Padgham, et Stéphane Airiau. Learning context conditions for BDI plan selection. In *AAMAS'10*, pages 325–332, 2010.

Smart-World. Conférence smart world 2011 : Une édition riche en concret, juin 2011. URL `http://www.conference-smartworld.com/`.

Max Waters, Lin Padgham, et Sebastian Sardina. Evaluating coverage based intention selection. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 957–964, Paris, France, May 2014a. IFAAMAS. Nominated for Jodi Best Student Paper award.

Max Waters, Lin Padgham, et Sebastian Sardina. Evaluating coverage based intention selection. In *AAMAS'14*, pages 957–964, 2014b. ISBN 978-1-4503-2738-1.

Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.

Rolf H. Weber et Romana Weber. *Internet of Things*. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-11709-1.