La carte bayésienne : un modèle probabiliste hiérarchique pour la navigation en robotique mobile

Julien Diard

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N^o attribué par la bibliothèque
| / / / / / / / / / |

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : « Informatique : Systèmes et Communications »

préparée aux laboratoires GRAVIR et Leibniz, dans le cadre de **l'École Doctorale « Mathématiques, Sciences et Technologies de l'Information, Informatique »**

présentée et soutenue publiquement par

Julien DIARD

le 27/01/2003

Titre :

La carte bayésienne

Un modèle probabiliste hiérarchique pour la navigation en robotique mobile

Directeur de Thèse :

Pierre BESSIÈRE

Composition du jury :

M.	Augustin LUX	Président
M.	Richard WASHINGTON	Rapporteur
M.	René ZAPATA	Rapporteur
M.	Jacques DROULEZ	Examinateur
M.	Jean-Luc SCHWARTZ	Examinateur
M.	Pierre BESSIÈRE	Directeur de thèse
M.	Emmanuel MAZER	Co-directeur de thèse

Résumé

Qu'est-ce qu'une carte ? Quelle est son utilité ? Qu'est-ce qu'un lieu, un comportement ? Qu'est-ce que naviguer, se localiser et prédire, pour un robot mobile devant accomplir une tâche donnée ?

Ces questions n'ont pas de réponses uniques ou évidentes à ce jour, et restent centrales à de nombreux domaines de recherches.

La robotique, par exemple, souhaite y répondre en vue de la synthèse de systèmes sensori-moteurs performants. Les sciences cognitives placent ces questions comme essentielles à la compréhension des êtres vivants, de leurs compétences, et au-delà, de leurs intelligences.

Notre étude se situe à la croisée de ces disciplines. Nous étudions tout d'abord les méthodes probabilistes classiques (Localisation Markovienne, PDMPOs, MMCs, etc.), puis certaines approches dites « bio-inspirées » (Berthoz, Franz, Kuipers). Nous analysons les avantages et inconvénients respectifs de ces approches en les replaçant dans un cadre général de programmation des robots basé sur l'inférence bayésienne (PBR).

Nous proposons un formalisme original de modélisation probabiliste de l'interaction entre un robot et son environnement : la carte bayésienne.

Dans ce cadre, définir une carte revient à spécifier une distribution de probabilités particulière. Certaines des questions évoquées ci-dessus se ramènent alors à la résolution de problèmes d'inférence probabiliste.

Nous définissons des opérateurs d'assemblage de cartes bayésiennes, replaçant ainsi les notions de « hiérarchie de cartes » et de développement incrémental comme éléments centraux de notre approche, en accord avec les données biologiques. En appuyant l'ensemble de notre travail sur le formalisme bayésien, nous profitons d'une part d'une capacité de traitement unifié des incertitudes, et d'autre part, de fondations mathématiques claires et rigoureuses. Notre formalisme est illustré par des exemples implantés sur un robot mobile Koala.

Mots-clés : robotique mobile, modélisation probabiliste, navigation, cartographie, localisation.

Abstract

Title : The Bayesian map – A hierarchical probabilistic model for mobile robot navigation.

Summary : What is a map ? What is its utility ? What is a location, a behaviour ? What are navigation, localization and prediction for a mobile robot facing a given task ?

These questions have neither unique nor straightforward answer to this day, and are still the core of numerous research domains.

Robotics, for instance, aim at answering them for creating successful sensori-motor artefacts. Cognitive sciences use these questions as intermediate goals on the road to understanding living beings, their skills, and furthermore, their intelligence.

Our study lies between these two domains. We first study classical probabilistic approaches (Markov localization, POMDPs, HMMs, etc.), then some biomimetic approaches (Berthoz, Franz, Kuipers). We analyze their respective advantages and drawbacks in light of a general formalism for robot programming based on bayesian inference (BRP).

We propose a new probabilistic formalism for modelling the interaction between a robot and its environment : the Bayesian map.

In this framework, defining a map is done by specifying a particular probability distribution. Some of the questions above then amount to solving inference problems.

We define operators for putting maps together, so that « hierarchies of maps » and incremental development play a central role in our formalism, as in biomimetic approaches. By using the bayesian formalism, we also benefit both from a unified means of dealing with uncertainties, and from clear and rigorous mathematical foundations. Our formalism is illustrated by experiments that have been implemented on a Koala mobile robot.

Keywords : mobile robotics, probabilistic modelling, navigation, map building, localization.

Remerciements

Je tiens tout d'abord à exprimer ma très vive reconnaissance à Pierre Bessière et Emmanuel Mazer, mes deux directeurs de thèse, pour le soutien sans faille, les compétences scientifiques exceptionnelles, et surtout, pour m'avoir supporté.

Je souhaite également adresser ma plus vive gratitude aux membres de mon jury. Je remercie Richard Washington et René Zapata, mes rapporteurs, pour leur temps passé à travailler, eux aussi, sur ce document. Je remercie Jacques Droulez et Jean-Luc Schwartz pour l'intérêt et l'attention qu'ils ont porté à ce travail. Enfin, merci également à Augustin Lux pour m'avoir fait l'honneur de présider mon jury de thèse.

J'adresse mes remerciements à Philippe Jorrand et Nicolas Balacheff, directeurs successifs du laboratoire Leibniz au sein duquel j'ai commencé ce travail dans d'excellentes conditions, ainsi qu'à Claude Puech, directeur du laboratoire GRAVIR au sein duquel j'ai continué et terminé ce travail dans de non moins bonnes conditions. Je souhaite également remercier ici les différentes équipes administratives de ces deux laboratoires, et en particulier Véronique Roux et Anne Pasteur.

Je tiens aussi ici à remercier du fond du cœur tous les membres de l'équipe Laplace et du projet Sharp. Les discussions scientifiques passionnées ne sont pas incompatibles avec un sentiment d'amitié profonde.

Un grand merci également à tous mes amis et amies, de Grenoble et d'ailleurs. Merci en particulier à Cécile-Fleur, Armelle, Régis (de Progpulsion), Cyril, Odge, Brian, et surtout, un grand merci à Éva et à sa famille.

Je souhaite enfin remercier mon frère Jérôme, et toute ma famille. Que mes parents trouvent ici l'expression de toute ma gratitude. Je leur dédie humblement cette thèse.

À ma mère.

À mon père.

Table des matières

Table des figures

Chapitre 1

Introduction

$$
\begin{aligned}
P(BlesserHumain) &= 0 \\
P(Obéir \mid \neg BlesserHumain) &= 1 \\
P(SeProtéger \mid Obéir \wedge \neg BlesserHumain) &= 1
\end{aligned}
$$

Handbook of Robotics, 56^{e} édition, 2058.

1.1 Situation

Notre recherche se situe au carrefour de deux disciplines : la robotique autonome et l'inférence probabiliste. La robotique autonome pourrait se définir comme l'ambition de réaliser des systèmes sensori-moteurs complexes capables de travailler dans des environnements naturels. À contrario, la robotique industrielle se concentrerait sur des tâches où les conditions d'opération sont mieux maîtrisées. Dans cette acceptation du terme « naturel », on conçoit l'importance de l'incomplétude. Un monde « naturel » n'est t-il pas, en fin de compte, un monde où tout n'est pas dit ? Si l'on veut bien considérer cette posture philosophique, il faut alors se pencher sur le problème de cette incomplétude. La voie choisie ici est celle des probabilités bayésiennes. Plus spécifiquement, notre recherche contribue à un effort ayant pour objectif le développement d'un cadre formel permettant d'aborder l'incomplétude des modèles de représentation. La théorie bayésienne du raisonnement plausible a été retenue comme fondement théorique et la robotique autonome comme un des domaines d'applications possibles. En voulant construire des robots, notre équipe se dote d'une méthode de synthèse. Pour l'analyse, elle émet l'hypothèse d'un fonctionnement probabiliste des systèmes vivants.

Pour mettre en œuvre pratiquement cette vision et ses implications à l'analyse et à la synthèse des systèmes autonomes, nous nous intéressons, comme beaucoup d'autres, au domaine de la construction de cartes, de la localisation et de la navigation.

1.2 Objectif

Notre objectif est d'étudier la notion de carte en apportant des éléments de réponses aux questions suivantes :

Q1 Qu'est-ce qu'une *carte* ? Quelle est son utilité ? [1]

Q2 Qu'est-ce qu'un *lieu*, pour le robot ? Que sont les représentations *topologiques* et *métriques* ? Que sont les problèmes de *perceptual aliasing* et de *global connectivity* ?

Q3 Qu'est-ce que *naviguer* ? Qu'est-ce qu'un *comportement* ?

Q4 Qu'est-ce que *se localiser* ?

Q5 Qu'est-ce que *prédire* ?

Q6 Qu'est-ce que *générer une loi de contrôle ?*

Q7 Comment s'élaborent les cartes ? Comment en *spécifier* une ? Comment les *assembler* ? Comment les *apprendre* expérimentalement ?

1.3 Contribution

Notre première contribution est de proposer un formalisme basé sur la modélisation probabiliste : la carte bayésienne. Définir une carte revient à spécifier une distribution de probabilités particulière. Dans ce cadre, certaines des questions évoquées ci-dessus se ramènent à la résolution de problèmes d'inférence probabiliste.

Notre deuxième contribution est la définition d'opérateurs permettant d'assembler des cartes pour en créer de nouvelles. Cette opération nous permet de définir des hiérarchies de cartes bayésiennes. C'est, selon nous, notre contribution la plus importante.

1.4 Plan de lecture

Le Chapitre 2 introduit les éléments mathématiques et le formalisme de base. Le document s'articule ensuite autour de trois parties.

La première présente notre étude bibliographique. Elle est constituée de trois chapitres. Les deux premiers, Chapitres 3 et 4, étudient les approches probabilistes et les approches hiérarchiques de la représentation de l'espace et de la navigation en robotique. Le Chapitre 5 propose une synthèse de ces différentes approches, et analyse une série de problèmes clés liés à la construction de cartes et à leur utilisation.

Dans la seconde partie, nous présentons notre formalisme pour répondre à ces différents problèmes : la *carte bayésienne*. Le Chapitre 6 définit une carte bayésienne et son utilisation, et l'illustre sur un exemple. Dans le Chapitre 7, nous présentons l'éventail de possibilités laissées au programmeur pour la définition de cartes bayésiennes individuelles. Dans les

[1] Les labels (Q1, Q2, etc.) nous permettront de référer précisément à ces différentes questions, tout au long du document.

deux chapitres suivants, Chapitres 8 et 9, nous définissons deux opérateurs permettant d'assembler des cartes bayésiennes.

Enfin, la dernière partie présente les pistes de recherche envisagées pour l'avenir, et nos conclusions sur ce travail.

Chapitre 2

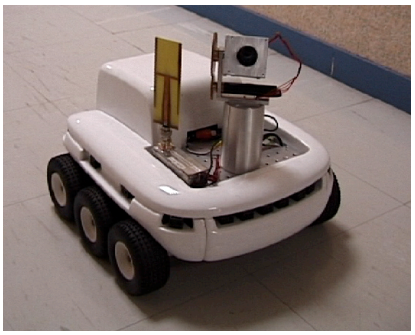
Environnement pratique et théorique

« Les représentations que nous construisons dans notre cerveau sont, nous le verrons, des objets physiques, des « modèles réduits » du monde extérieur et de notre propre monde intérieur. Ils ne peuvent prétendre à une description intégrale, à l'épuisement de la réalité du monde. Il existera toujours une marge d'incertitude, une frange de remise en question pour toute avancée de la connaissance scientifique. Est-ce une raison pour renoncer à en savoir plus ? »

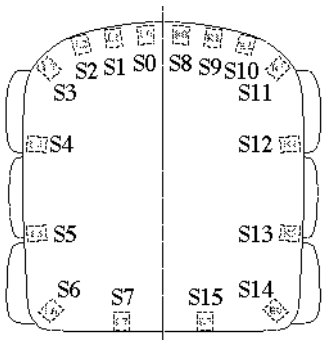
Jean-Pierre Changeux, *L'Homme de Vérité*, 2002, [Changeux02].

2.1 Présentation de la plate-forme expérimentale

Les différentes expériences présentées ont été réalisées avec un robot mobile développé à l'École Polytechnique Fédérale de Lausanne (EPFL) : le Koala (voir Figure 2.1(a)) [1].



(a) Le robot mobile Koala.



(b) L'emplacement des capteurs sur le Koala.

Fig. 2.1 – Photo et schéma du robot Koala.

[1] Une partie de ce descriptif du robot Koala est inspiré de [Coué00].

Le Koala (longueur 32 cm, largeur 32 cm, hauteur 20 cm, poids 3 kg dans sa configuration de base) dispose d'un processeur embarqué performant (M68331, 32 bits, cadencé a 16 MHz), associé à une EEPROM de 128 Koctets et une RAM statique de 1 Moctets. Le BIOS est le système bas niveau embarqué dans le robot. Il offre des capacités multi-tâches et permet la gestion de plusieurs modules logiciels : acquisition et conversion des données sensorielles, asservissement des moteurs, contrôle de la communication entre différents modules du robot et l'extérieur.

La partie motrice est constituée de deux blocs de trois roues latérales commandés indépendamment. Ils offrent en particulier au Koala la possibilité de tourner sur place. Le Koala est donc quasiment holonome. Chaque bloc de roues est entraîné par un moteur continu (M_g et M_d), associé à un réducteur à vis sans fin et à un codeur incrémental. Ces capteurs ($PosG$ et $PosD$) fournissent une information sur l'ego-mouvement du robot. À chaque moteur sont associés deux asservissements classiques de type PID permettant le contrôle du robot en vitesse ou en position.

Les ressources systèmes fournissent des ordres permettant de spécifier les valeurs des vitesses de rotation des moteurs gauche et droit (V_g et V_d). Nous avons préféré travailler avec des variables homogènes aux vitesses de rotation ($Vrot$) et de translation ($Vtrans$) du robot. Ces 4 variables sont reliées les unes aux autres par les relations :

$$V_g = Vtrans + Vrot$$
$$V_d = Vtrans - Vrot$$

Le Koala dispose d'un anneau de 16 capteurs infrarouges. Chaque capteur est composé d'un émetteur de lumière infrarouge et d'un récepteur. Ces capteurs permettent deux types de mesure : une de la luminosité ambiante en mode récepteur (mode lumimètre, variables Lm_i), et l'autre de la réflexion de la lumière par les obstacles en utilisant le mode émetteur et récepteur (mode proximètre, variables Px_i). Cette dernière mesure permet d'obtenir une information relative à la distance des obstacles pouvant varier entre 1 et 25 cm, dépendant fortement de leur nature (orientation, couleur, matière, etc.) et des conditions de luminosité ambiante. Le positionnement des capteurs sur le Koala est montré Figure 2.1(b).

Dans la suite de ce travail, nous ne considérerons principalement que les capteurs de proximétrie, excepté Section 8.2.1.3, où un « simulateur de nez » sera utilisé (et décrit).

Dans nos expériences, nous plaçons le robot dans un environnement simple, sorte d'enclos fermé constitué uniquement de murs et d'intersections (presque) perpendiculaires de murs. Ces murs sont faits de planches de bois hautes d'une quarantaine de centimètres, et sont facilement déplacées pour créer des configurations différentes. Notre zone d'expérimentation nous permet de créer des environnements d'environ 5 m sur 5 m. Il s'agit de reproduire un environnement similaire à une pièce d'appartement, sans les obstacles tels que les tables, les chaises, etc. Les obstacles de ce type seraient de toute façon difficilement perceptibles par les capteurs de proximétrie du robot.

2.2 Concepts de base

Nous présentons dans cette section le minimum théorique et mathématique pour la compréhension des expériences présentées dans les chapitres suivants. La suite de ce chapitre est principalement reprise de [DIARD99].

2.2.1 Définitions et notations

2.2.1.1 Propositions logiques

Les objets que nous manipulons sont des propositions de la logique classique. Elles possèdent donc une valeur de vérité, et peuvent êtres composées à l'aide des opérateurs usuels : $\mathcal{AB}$ dénote « les deux propositions $\mathcal{A}$ et $\mathcal{B}$ sont vraies » (conjonction ou produit logique), $\mathcal{A}+\mathcal{B}$ dénote « au moins une des deux propositions $\mathcal{A}$ ou $\mathcal{B}$ est vraie » (disjonction ou somme logique), enfin $\neg\mathcal{A}$ dénote « $\mathcal{A}$ est faux » (négation). Tout l'attirail de propriétés et de notations de l'algèbre booléenne est donc à notre disposition.

2.2.1.2 Variables

Soit V une variable, $\mathcal{D}_V = \{v_1, \ldots, v_k\}$ un domaine de valeurs. V est associée aux k propositions logiques $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$:

$$\begin{aligned} \mathcal{V}_1 &\equiv [V = v_1] \\ \mathcal{V}_2 &\equiv [V = v_2] \\ &\vdots \\ \mathcal{V}_k &\equiv [V = v_k]. \end{aligned}$$

Chacune de ces propositions indique la valeur prise par la variable ; elles sont donc exhaustives ($\mathcal{V}_1 + \cdots + \mathcal{V}_k = vrai$) et mutuellement exclusives ($\forall i, j, i \neq j, \mathcal{V}_i\mathcal{V}_j = faux$).

Ainsi, nous définirons les variables en donnant leur domaine $\mathcal{D}_V$ et le cardinal de ce domaine k_V. Par exemple, pour la variable motrice $Vrot$:

$$Vrot \ : \mathcal{D}_{Vrot} = [-50, +50], k_{Vrot} = 101.$$

En présence de deux variables V_1 et V_2 de domaines $\mathcal{D}_{V_1}$ et $\mathcal{D}_{V_2}$, nous pouvons construire la nouvelle variable $V = V_1 \wedge V_2$, associée à toutes les propositions logiques du type :

$$\mathcal{V}_{1_i}\mathcal{V}_{2_j} \equiv [V_1 = v_{1_i}][V_2 = v_{2_j}],$$

i et j variant dans les domaines respectifs des variables. Ces propositions dénotent « la variable $\mathcal{V}_1$ a pour valeur v_{1_i} et la variable $\mathcal{V}_2$ a pour valeur v_{2_j} » ; nous remarquons qu'elles restent exhaustives et mutuellement exclusives[2].

[2]En revanche, cela ne se passe pas aussi bien pour la disjonction, car les propositions ne sont plus alors mutuellement exclusives.

2.2.1.3 Probabilité

La notion de plausibilité intervient lorsque nous ne sommes pas en mesure de connaître de manière certaine la valeur de vérité d'une proposition. Intuitivement, nous définissons la plausibilité comme le degré de certitude que nous avons dans la véracité d'une proposition. Ce degré de certitude est fortement lié à un ensemble de connaissances dont on dispose et sur la base desquelles nous souhaitons juger de la véracité d'une proposition.

Les deux articles [BDL+98a] et [BDL+98b] présentent de manière très détaillée la formalisation de ces concepts que nous utilisons. Celle-ci se base sur la théorie du raisonnement plausible, « Probability as Logic », proposée par la physicien E. T. Jaynes [Jaynes95]. Le fondement de cette théorie est le théorème de Cox, qui montre comment passer de la notion intuitive de plausibilité à la notion mathématique de probabilité, au moyen de quelques désidérata de base.

La probabilité d'une certaine proposition $\mathcal{A}$ sera dans ce cadre toujours dépendante du fait qu'une autre, ou qu'un ensemble d'autres propositions soient vraies (faisant référence aux connaissances de base mentionnées plus haut, qui nous permettent d'assigner une probabilité à $\mathcal{A}$). Nous noterons donc $P(\mathcal{A} \mid \mathcal{B})$ pour la probabilité conditionnelle que $\mathcal{A}$ soit vraie, sachant que $\mathcal{B}$ est vraie (ou probabilité conditionnelle de $\mathcal{A}$ sachant $\mathcal{B}$).

En pratique, nous considérons que la probabilité d'une proposition $\mathcal{A}$ ne peut être définie que relativement à une connaissance d'arrière plan (ou connaissance préalable) c. Nous noterons donc $P(\mathcal{A} \mid c)$ pour la probabilité que $\mathcal{A}$ soit vraie conditionnellement à toutes les évidences dont on dispose et que l'on résume dans c. C'est en ce sens que nous parlerons de « probabilité subjective ». Toute probabilité qui n'est conditionnée que par c est appelée alors « probabilité a priori », par opposition aux probabilités qui pourront être conditionnées à la fois par c et par un ensemble de données D, par exemple.

Dans l'intention d'éviter des problèmes impossibles, nous ne nous permettrons pas de raisonner à partir de prémisses impossibles ou mutuellement contradictoires. Par conséquent, nous ne chercherons pas à définir $P(\mathcal{A} \mid \mathcal{B}\ \mathcal{C})$ lorsque $\mathcal{B}$ et $\mathcal{C}$ sont des propositions contradictoires. Quand de telles notations se présenteront, il sera implicitement compris que $\mathcal{B}$ et $\mathcal{C}$ sont compatibles.

L'expression $P(V \mid c)$ dénote la distribution de probabilité attribuée a priori à la variable V, c'est-à-dire aux différentes valeurs que peut prendre cette variable. Dans le cas où V est une variable discrète dont le domaine de valeurs est $\mathcal{D}_V$, nous notons $P(V \mid c)$ l'ensemble des probabilités $P(\mathcal{V}_i \mid c)$:

$$\forall v_i \in \mathcal{D}_V, P(\mathcal{V}_i \mid c) = P([V = v_i] \mid c).$$

Sachant qu'une variable ne peut prendre qu'une et une seule valeur à la fois, les k_V propositions logiques $\mathcal{V}_i$ associées à une variable V sont exhaustives et mutuellement exclusives :

$$P(\mathcal{V}_i\ \mathcal{V}_j \mid c) = P(\mathcal{V}_i \mid c)\delta_{ij}$$

$$\sum_{1 \leq i \leq k_v} P(\mathcal{V}_i \mid c) = 1.$$

2.2.2 Règles de calcul

Les deux règles fondamentales sur lesquelles repose l'inférence probabiliste [3] sont la *règle du produit* et la *règle de normalisation*. De ces deux règles nous pouvons en dériver d'autres, qui rendront les calculs plus aisés. Dans ce document nous ne faisons qu'énoncer ces règles, les dérivations permettant d'y aboutir sont présentes dans [JAYNES95].

2.2.2.1 [R1] : Règle du produit

La règle du produit permet d'exprimer la probabilité conjointe de deux termes (propositions ou variables) comme le produit de deux probabilités élémentaires. Dans le cas de propositions, [R1] prend la forme :

$$\begin{aligned} P(\mathcal{A}\ \mathcal{B} \mid c) &= P(\mathcal{A} \mid c)P(\mathcal{B} \mid \mathcal{A}\ c) \\ &= P(\mathcal{B} \mid c)P(\mathcal{A} \mid \mathcal{B}\ c). \end{aligned} \tag{2.1}$$

Les deux formes possibles proviennent bien évidemment de la commutativité de la conjonction logique. Une dérivation élémentaire permet d'arriver à la forme de [R1] pour les variables :

$$\begin{aligned} P(X\ Y \mid c) &= P(X \mid c)P(Y \mid X\ c) \\ &= P(Y \mid c)P(X \mid Y\ c). \end{aligned} \tag{2.2}$$

Cette règle est également appelée le *théorème de Bayes*.

2.2.2.2 [R2] : Règle de normalisation

La règle de normalisation est la deuxième règle fondamentale nécessaire pour l'inférence. De façon informelle, elle exprime le fait que la somme des probabilités de tous les cas possibles d'un terme vaut 1. Dans le cas des propositions, [R2] prend la forme :

$$P(\mathcal{A} \mid c) + P(\neg\mathcal{A} \mid c) = 1, \tag{2.3}$$

alors que pour les variables on écrit :

$$\forall X\ : \mathcal{D}_X, k_X, \sum_X P(X \mid c) = 1. \tag{2.4}$$

2.2.2.3 Autres règles dérivables des précédentes

Tout comme en logique où les opérateurs de conjonction et de négation permettent d'exprimer toute expression logique, les deux règles [R1] et [R2] forment la base sur laquelle

[3] Nous utilisons ici la définition de [HECKERMAN96] : « En général, le calcul d'une probabilité particulière au vu d'un modèle est connu sous le nom d'inférence probabiliste. »

tous les calculs de probabilités se dérivent. En particulier, de ces deux règles, nous pouvons dériver la *règle de la somme* :

$$\begin{aligned} &\forall X \ : \mathcal{D}_X, k_X, \forall Y \ : \mathcal{D}_Y, k_Y, \forall x_i \in \mathcal{D}_X, \forall y_i \in \mathcal{D}_Y, \\ &P(\mathcal{X}_i + \mathcal{Y}_i \mid c) = P(\mathcal{X}_i \mid c) + P(\mathcal{Y}_i \mid c) - P(\mathcal{X}_i \ \mathcal{Y}_i \mid c), \end{aligned} \tag{2.5}$$

où, rappelons le, $\mathcal{X}_i$ dénote la proposition $[X = x_i]$ et $\mathcal{Y}_i$ dénote $[Y = y_i]$. Nous pouvons aussi dériver la *règle de marginalisation* :

$$\forall X \ : \mathcal{D}_X, k_X, \forall Y \ : \mathcal{D}_Y, k_Y, \sum_X P(X \ Y \mid c) = P(Y \mid c). \tag{2.6}$$

2.2.3 Méthode

Dans cette section, nous présentons la méthode PBR (pour Programmation Bayésienne des Robots) que nous suivons pour « programmer » le Koala dans les expériences suivantes. Ces programmes se basent sur un objet formel, la *description* [Lebeltel99, LBDM03]. Programmer un robot à l'aide d'une description se décompose en trois phases :

- la phase de spécification des connaissances préalables c ;
- la phase d'identification des valeurs des paramètres des distributions de probabilité ;
- enfin, la phase d'utilisation de la description.

Nous allons maintenant définir le concept de description et décrire plus précisément chacune de ces trois phases. Tout au long de cette définition, nous utiliserons en exemple la programmation sur le Koala d'un comportement relativement simple d'évitement d'obstacles. Le comportement désiré est défini par la seule contrainte suivante : le robot doit tourner en présence d'obstacles. Cela veut notamment dire que l'évolution du robot ne doit pas être contrainte s'il n'y a pas d'obstacles (on doit donc constater un mouvement aléatoire).

2.2.3.1 Description (définition)

Une description est dénotée formellement par la probabilité conjointe $P(V_1 \ \ldots \ V_n \mid D \ c)$ d'un ensemble de variables $V_1, \ \ldots, \ V_n$, déterminée au vu des connaissances préalables c spécifiées par le programmeur et d'un ensemble de données expérimentales D. Ainsi, l'ensemble des connaissances devant êtres fournies se trouve circonscrit par les connaissances préalables c et le jeu de données D.

2.2.3.2 Spécification

La phase de spécification est la partie la plus délicate du travail du programmeur. Au cours de cette phase, il doit énoncer clairement et *explicitement* les connaissances dont il est à l'origine et celles qui résultent d'un processus adaptatif dépendant d'un jeu particulier de données expérimentales. Ces connaissances se subdivisent en trois : le choix des variables pertinentes, l'expression des dépendances entre les variables retenues sous la forme d'un

produit de distributions élémentaires, et enfin la forme paramétrique associée à chacune de ces distributions.

Connaissances préalables structurelles : le choix des variables pertinentes Nous appelons *connaissances préalables structurelles* les connaissances permettant de définir l'ensemble des variables $V_1, \ldots, V_n$ pour la description, et de spécifier pour chacune d'elles son domaine de variation $\mathcal{D}_{V_i}$ et le nombre k_{V_i} d'états possibles. Toutes les autres variables sont ainsi supposées non pertinentes pour le problème considéré.

En robotique, nous pouvons classer ces variables naturellement en trois sous-ensembles, les variables sensorielles (par exemple les Px_i pour le Koala), les variables motrices (par exemple $Vrot$), et enfin les variables internes, qui permettent de coder les états internes du robot.

Dans notre exemple d'évitement d'obstacles pour le Koala, nous décidons de ne pas garder les nombreuses variables sensorielles à notre disposition : nous faisons tout d'abord le choix de n'utiliser ni la caméra, ni les capteurs de lumière ; de plus, nous décidons de résumer l'information contenue dans les seize variables Px_0 à Px_{15} par deux variables, Dir et $Prox$, obtenues par les pré-traitements suivants :

$$\begin{aligned} Dir &= \left\lfloor \frac{\begin{array}{c} 90(Px_{12} - Px_4) + 45(Px_{11} - Px_3) + 14(Px_{10} - Px_2) \\ +7(Px_9 - Px_1) + (Px_8 - Px_0) \end{array}}{9\left(\begin{array}{c} Px_4 + Px_3 + Px_2 + Px_1 + Px_0 \\ +Px_8 + Px_9 + Px_{10} + Px_{11} + Px_{12} \end{array}\right) + 1} \right\rfloor, \\ Prox &= \lfloor \max(Px_4, Px_3, Px_2, Px_1, Px_0, Px_8, Px_9, Px_{10}, Px_{11}, Px_{12})/64 \rfloor . \end{aligned}$$

Ces variables correspondent approximativement à la direction et la proximité de l'obstacle le plus proche. Sur le plan moteur, nous décidons que notre robot se déplacera à vitesse de translation constante, ce qui ne nous laisse plus qu'un degré de liberté, la variable $Vrot$. Enfin, nous décidons de ne pas avoir de variables internes. Présentons donc les variables retenues selon notre notation :

- Dir : $\mathcal{D}_{Dir} = [-10, +10], k_{Dir} = 21$,
- $Prox$: $\mathcal{D}_{Prox} = [0, 15], k_{Prox} = 16$,
- $Vrot$: $\mathcal{D}_{Vrot} = [-50, +50], k_{Vrot} = 101$.

Connaissances préalables de dépendance : le choix d'une décomposition de la distribution conjointe Comme énoncé précédemment, la description sur les variables $V_1, \ldots, V_n$ a pour but la définition de la distribution conjointe $P(V_1 \cdots V_n \mid D\ c)$. Cette forme mathématique est une distribution de probabilité sur n dimensions, qui n'est souvent pas facile à spécifier. La règle du produit [R1] nous permet de décomposer cette expression, en l'exprimant sous forme de produit de distributions.

Dans notre exemple, si l'on considère les trois variables retenues précédemment, nous

avons à notre disposition bien des décompositions (voir Section 7.2) :

$$\begin{aligned} & P(Dir\ Prox\ Vrot \mid D\ c) \\ &= P(Dir \mid D\ c)P(Prox\ Vrot \mid Dir\ D\ c) \\ &= P(Vrot \mid D\ c)P(Dir\ Prox \mid Vrot\ D\ c) \\ &= P(Dir \mid D\ c)P(Prox \mid Dir\ D\ c)P(Vrot \mid Dir\ Prox\ D\ c) \\ &= \dots \end{aligned}$$

C'est la dernière expression ci-dessus que nous allons choisir dans notre exemple.

Dans cette seconde étape de la phase de spécification, nous allons simplifier davantage l'expression choisie par des *hypothèses d'indépendance conditionnelle*, qui permettent de réduire fortement les dimensions des termes sur lesquelles ces hypothèses portent.

Par exemple, nous n'avons pas de raisons de penser – ou décidons d'ignorer ce fait pour simplifier – que la proximité des objets que le robot rencontrera dépend de leur direction. Nous écrivons alors $P(Prox \mid Dir\ D\ c) = P(Prox \mid D\ c)$, ce qui nous donne :

$$P(Dir\ Prox\ Vrot \mid D\ c) = P(Dir \mid D\ c)P(Prox \mid D\ c)P(Vrot \mid Dir\ Prox\ D\ c).$$

Connaissances préalables d'observation : le choix des formes paramétriques Il faut maintenant associer, à chacun des termes apparaissant dans la décomposition choisie à l'étape précédente, une forme paramétrique. Par ces choix, nous allons fournir des a priori sur les valeurs des distributions de probabilité et la manière dont ces valeurs seront modifiées par l'expérience. Ce dernier point est composé d'un ensemble de valeurs initiales pour les paramètres, et d'un mécanisme de mise à jour au vu de données expérimentales (ce mécanisme peut éventuellement être vide si l'on veut figer la distribution lors de la présente phase de spécification. Une description dans laquelle tous les termes sont ainsi fixés est appelée *spécification (description) a priori*. Dans ce document, nous n'évoquerons que quelques formes paramétriques simples :

Loi uniforme C'est une forme paramétrique à un paramètre, qui est le nombre de cas de la variable sur laquelle porte cette distribution. Cette loi indique l'équiprobabilité de tous les cas ; elle sera choisie en général pour modéliser notre ignorance sur un phénomène. Dans notre formalisme, la définition des variables est statique, donc cette loi n'évoluera pas en fonction des données expérimentales. Pour une variable $V : \mathcal{D}_V, k_V$, nous notons :

$$\begin{aligned} P(V \mid D\ c) = P(V \mid c) &= \mathbf{U}_{k_V}(V) \\ \forall v_i \in \mathcal{D}_V, P([V = v_i] \mid c) &= \frac{1}{k_V}. \end{aligned}$$

Loi Dirac Cette forme paramétrique classique possède un paramètre, qui est une valeur particulière n de la variable sur laquelle elle porte : pour cette valeur donnée, la probabilité est 1, et 0 pour toutes les autres valeurs. Nous utiliserons la notation

standard :

$$\begin{aligned} P(V \mid D\ c) &= \boldsymbol{\delta}_n(V) \\ \forall v_i \in \mathcal{D}_V, P([V = v_i] \mid D\ c) &= \begin{cases} 1 \text{ si } v_i = n, \\ 0 \text{ sinon.} \end{cases} \end{aligned}$$

Loi normale ou gaussienne Cette forme paramétrique classique correspond au choix de ne mémoriser des données expérimentales que la moyenne des valeurs rencontrées et leur écart type. Nous notons :

$$\begin{aligned} P(V \mid D\ c) &= \mathbf{G}_{\mu,\sigma}(V) \\ \forall v_i \in \mathcal{D}_V, P([V = v_i] \mid c) &= \int_{v_i-\varepsilon}^{v_i+\varepsilon} \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(v-\mu)^2}{2\sigma^2}} dv, \end{aligned}$$

où μ est la moyenne de V dans les données expérimentales D, et σ est l'écart type de V dans D.

Loi de succession de Laplace Cette forme a autant de paramètres que de cas de la variable sur laquelle elle porte ; ces paramètres se calculent facilement au vu des données expérimentales. La distribution résultat ressemble fortement à une normalisation du nombre d'occurrences n_i de chaque cas dans les données expérimentales. Nous notons :

$$\begin{aligned} P(V \mid D\ c) &= \mathbf{L}_{n_1,\ldots,n_{k_V}}(V) \\ \forall v_i \in \mathcal{D}_V, P([V = v_i] \mid c) &= \frac{n_i + 1}{N + k_V}, \end{aligned}$$

où n_i est le nombre de fois où $[V = v_i]$ dans les données D, et N est le nombre total de données dans D.

Dans le cas où la loi de succession de Laplace est définie sur un terme de la forme $P(V \mid W\ D\ c)$ (ayant une variable probabiliste en partie droite), le nombre de paramètres à définir devient $k_V * k_W$. Nous noterons :

$$P(V \mid W\ D\ c) = \mathbf{L}_{n_1,\ldots,n_{k_V * k_W}}(V).$$

Question à une autre description Cette forme paramétrique, sans paramètres libres, consiste à poser une question probabiliste à une autre description c' pour obtenir un terme (voir Section 2.2.3.4 pour la définition d'une question probabiliste). Nous notons :

$$\begin{aligned} P(V \mid D\ c) &= P(V \mid c) = P(V \mid c') \\ \forall v_i \in \mathcal{D}_V, P([V = v_i] \mid c) &= P([V = v_i] \mid c'). \end{aligned}$$

Dans notre exemple courant d'évitement d'obstacles, nous définissons $P(Dir \mid D\ c)$ et $P(Prox \mid D\ c)$ comme étant des lois uniformes sur leurs domaines, pour indiquer que nous n'avons pas d'informations particulières ni sur la direction, ni sur la proximité des objets que nous rencontrerons. En revanche, le terme $P(Vrot \mid Dir\ Prox\ D\ c)$ est défini comme étant, pour chaque cas de $Dir, Prox$, une loi gaussienne dont la moyenne et l'écart type seront déterminés par les données expérimentales D.

Récapitulatif Cette première étape dans la définition d'une description, la phase de spécification, est maintenant terminée. Nous y avons inclus trois types de connaissances, les connaissances préalables structurelles, les connaissances préalables de dépendance, et enfin les connaissances préalables d'observation. Ce sont toutes ces connaissances que nous groupons sous le terme *connaissances préalables*, et que nous notons par la proposition c dans nos expressions mathématiques. L'on comprend maintenant que cette variable doive, dans notre cadre, toujours apparaître en partie droite des probabilités. En effet, différentes descriptions seront identifiées par ces connaissances préalables ; on pourrait par exemple définir une autre description d'évitement d'obstacles, dans laquelle nous exprimerions d'autres choix, que nous résumerions par la proposition c_2. Dans la suite de ce document, lorsqu'il n'y aura pas d'ambiguïté possible, et afin d'alléger les notations, nous omettrons cette proposition en partie droite des probabilités.

2.2.3.3 Identification

Pour achever la définition de notre description, il faut encore fournir les données expérimentales D afin de fixer les valeurs numériques des paramètres définis auparavant. C'est cette phase qui constitue la période d'*apprentissage* [4] du robot. Plusieurs méthodes sont disponibles :

Pas d'expérimentation Dans ce cas, les paramètres ne changent pas, et l'on se trouve dans le cas d'une spécification a priori.

Programme professeur On peut, pour fournir des données, faire évoluer le robot sous le contrôle d'un programme quelconque, de type Braitenberg ou autre. Il suffit alors d'enregistrer les données pendant l'évolution du robot pour créer le jeu de données D dont nous nous servons ensuite pour compléter notre description. L'outil formel que nous présentons peut alors être considéré comme un langage dans lequel n'importe quel programme robotique peut être traduit. Le succès de cette traduction dépendra bien sûr de la pertinence des choix faits pendant la phase de spécification.

Télé-opération Nous pouvons également piloter le robot, en essayant de lui fournir des exemples d'un même phénomène. Piloter le robot consiste simplement à fournir pendant son évolution des valeurs pour ses variables motrices. Ce pilotage peut se faire en utilisant un joystick, mais on peut aussi imaginer d'autres méthodes, par interaction avec un programme graphique, par commande vocale, par imitation d'un robot professeur [Raspail02], etc.

Dans notre exemple, nous choisissons de piloter le Koala au joystick. Pratiquement, il s'agit de le faire évoluer dans un environnement, et de lui faire éviter les obstacles, tout en contraignant le moins possible son comportement loin de ceux-ci (pour correspondre à la définition en français du comportement voulu). Expérimentalement, pour ce cas précis, une télé-opération de quelques dizaines de secondes sera nécessaire, sous réserve d'explorer relativement bien l'espace des configurations sensorielles.

[4] Pour une excellente introduction à l'apprentissage dans le domaine de l'informatique, voir [Mitchell97].

2.2.3.4 Utilisation

Au terme des phases de spécification et d'identification, nous disposons d'une description complètement définie. La phase d'utilisation va consister à mettre en œuvre les descriptions par le biais de questions probabilistes.

Question (définition) Poser une question consiste à chercher la distribution de probabilité d'un certain nombre de variables $\mathcal{E}_q$ de la description, connaissant les valeurs d'autres variables $\mathcal{E}_c$, et éventuellement ignorant les valeurs d'un troisième groupe de variables $\mathcal{E}_i$. Une question probabiliste est donc n'importe quelle expression de la forme :

$$P(V_k \ \ldots \ V_l \mid [V_m = v_{m_i}] \ \ldots \ [V_n = v_{n_j}] \ D \ c), \tag{2.7}$$

où $\mathcal{E}_q = \{V_k, \ldots, V_l\} \neq \emptyset$, $\mathcal{E}_c = \{V_m, \ldots, V_n\}$, et $\mathcal{E}_i = \{V_o, \ldots, V_p\}$ est l'ensemble des variables n'apparaissant ni dans $\mathcal{E}_q$, ni dans $\mathcal{E}_c$. Ces trois ensembles doivent bien sûr former une partition de l'ensemble des variables considérées pour que la question ait un sens.

Si l'on veut restituer l'évitement d'obstacle que nous avons appris dans la phase précédente, il est nécessaire de pouvoir tirer un état de connaissance sur la commande motrice $Vrot$, sachant les valeurs courantes d et p des variables sensorielles Dir et $Prox$. La question posée est donc :

$$P(Vrot \mid [Dir = d] \ [Prox = p] \ D \ c).$$

Inférence La connaissance de la distribution conjointe $P(V_1 \ \cdots \ V_n \mid D \ c)$ et l'application des règles du produit (Équation 2.2) et de la marginalisation (Équation 2.6) nous permet de répondre à toute question de la forme 2.7. Appliquons d'abord la règle du produit :

$$\begin{aligned} &P(V_k \ \ldots \ V_l \mid [V_m = v_{m_i}] \ \ldots \ [V_n = v_{n_j}] \ D \ c) = \\ &\quad \frac{P(V_k \ \ldots \ V_l \ [V_m = v_{m_i}] \ \ldots \ [V_n = v_{n_j}] \mid D \ c)}{P([V_m = v_{m_i}] \ \ldots \ [V_n = v_{n_j}] \mid D \ c)}. \end{aligned} \tag{2.8}$$

En appliquant maintenant la règle de marginalisation, nous pouvons exprimer le dénominateur et le numérateur de ce quotient en fonction de la distribution conjointe que l'on sait calculer :

$$\begin{aligned} &\frac{P(V_k \ \ldots \ V_l \ [V_m = v_{m_i}] \ \ldots \ [V_n = v_{n_j}] \mid D \ c)}{P([V_m = v_{m_i}] \ \ldots \ [V_n = v_{n_j}] \mid D \ c)} = \\ &\quad \frac{\sum_{V_o \ldots V_p} P(V_k \ \ldots \ V_l \ [V_m = v_{m_i}] \ \ldots \ [V_n = v_{n_j}] \ V_o \ \ldots \ V_p \mid D \ c)}{\sum_{V_k \ldots V_l V_o \ldots V_p} P(V_k \ \ldots \ V_l \ [V_m = v_{m_i}] \ \ldots \ [V_n = v_{n_j}] \ V_o \ \ldots \ V_p \mid D \ c)}. \end{aligned} \tag{2.9}$$

Cette méthode de résolution d'une question est purement symbolique et ne tient pas compte des simplifications que peuvent faire apparaître les connaissances préalables fournies lors de la phase de spécification ainsi que les modes de décision que l'on veut appliquer par la suite. Notons tout de même ici le cas particulier qui survient lorsque la question

probabiliste posée à une description est un terme de la décomposition. Par exemple, supposons qu'une description c, portant sur les variables X_1, X_2, X_3 choisisse la décomposition suivante :

$$P(X_1\ X_2\ X_3 \mid c) = P(X_1 \mid c)P(X_2 \mid X_1\ c)P(X_3 \mid X_1\ c).$$

Posons à cette description la question $P(X_2 \mid X_1\ c)$. Nous commençons l'inférence en nous ramenant à la décomposition de la conjointe, puis simplifions symboliquement [5] :

$$\begin{aligned}
P(X_2 \mid X_1\ c) &= \frac{\sum_{X_3} P(X_1 \mid c)P(X_2 \mid X_1\ c)P(X_3 \mid X_1\ c)}{\sum_{X_2\ X_3} P(X_1 \mid c)P(X_2 \mid X_1\ c)P(X_3 \mid X_1\ c)} \\
&= \frac{P(X_1 \mid c)P(X_2 \mid X_1\ c)\sum_{X_3} P(X_3 \mid X_1\ c)}{P(X_1 \mid c)\sum_{X_2} P(X_2 \mid X_1\ c)\sum_{X_3} P(X_3 \mid X_1\ c))} \\
&= \frac{P(X_1 \mid c)P(X_2 \mid X_1\ c)}{P(X_1 \mid c)} \\
P(X_2 \mid X_1\ c) &= P(X_2 \mid X_1\ c).
\end{aligned}$$

Remarquons que ce type d'inférence est toujours valide : si la question est un terme qui apparaît dans la décomposition, l'inférence restitue ce terme.

Il a été développé dans l'équipe un moteur d'inférence probabiliste qui permet de répondre à toute question probabiliste. Ce moteur a pour ambition d'être capable d'automatiser les raisonnements probabilistes. Il peut être, en cela, comparé aux nombreux moteurs d'inférence qui permettent d'automatiser le raisonnement logique (tels Prolog, LogLisp ou Lolita). Pour une question donnée, l'expression correspondante est tout d'abord simplifiée symboliquement par le moteur ; sa valeur numérique est ensuite estimée. La complexité du calcul numérique vient évidemment des sommes. Il a été développé dans le moteur une méthode très performante d'estimation de ces sommes qui fait notamment appel à un algorithme d'optimisation de type « génétique », et à une représentation des distributions sous forme très compacte et efficace, inspirée des Octree.

Dans l'exemple d'évitement d'obstacles, nous pouvons nous passer du moteur d'inférence pour répondre à la question posée. En effet, cette question est un terme qui apparaît directement dans la décomposition choisie à l'étape des connaissances préalables de dépendance. Il suffit donc, pour avoir la réponse voulue, de lire la forme mathématique de ce terme. Cependant, la même description permet, grâce au mécanisme d'inférence, de répondre à bien d'autres questions :

- $P(Vrot \mid [Prox = p]\ D\ c)$ si le module de calcul du pré-traitement correspondant à Dir tombait en panne (mode sensoriel dégradé),
- $P(Dir \mid [Prox = p]\ D\ c)$ si on voulait remédier à la panne précédente en inférant les valeurs probables de Dir,
- $P(Prox \mid [Dir = d]\ [Vrot = v]\ D\ c)$ si on voulait prévoir la valeur de $Prox$ selon le pilotage donné, pour la comparer à la valeur fournie par le module de calcul dédié (diagnostique de panne, prévision sensorielle),
- etc.

[5] Un exemple détaillé d'inférence sera développé Annexe A.

Décision Le résultat de l'inférence fournit une distribution de probabilité sur les variables recherchées. Rendre effectif le résultat d'une question va consister à choisir les valeurs particulières pour ces variables. De nombreuses stratégies sont ici imaginables, mais les plus simples restent de choisir les valeurs correspondant au maximum de probabilité, ou encore de tirer les valeurs aléatoirement selon la distribution obtenue.

C'est cette seconde stratégie qui est employée dans notre exemple d'évitement d'obstacles, et ce afin de restituer tout l'état de connaissance des formes gaussiennes utilisées (moyenne et écart type) ; choisir le maximum de probabilité aurait pour conséquence de ne prendre en compte que la moyenne des gaussiennes, sans traduire à l'exécution les écarts types. C'est aussi cette stratégie qui sera employée dans le reste de ce travail.

2.2.3.5 Résumé de la méthode PBR

Programme
- Description
  - Spécification
    - Variables Pertinentes
      - $Dir \ : \mathcal{D}_{Dir} = [-10, +10], k_{Dir} = 21$
      - $Prox \ : \mathcal{D}_{Prox} = [0, 15], k_{Prox} = 16$
      - $Vrot \ : \mathcal{D}_{Vrot} = [-50, +50], k_{Vrot} = 101$
    - Décomposition
      - $P(Dir\ Prox\ Vrot \mid D\ c) =$
        $P(Dir \mid D\ c)P(Prox \mid D\ c)P(Vrot \mid Dir\ Prox\ D\ c)$
    - Formes Paramétriques
      - $P(Dir \mid D\ c) = \mathbf{U}_{21}(Dir)$
      - $P(Prox \mid D\ c) = \mathbf{U}_{16}(Prox)$
      - $P(Vrot \mid Dir\ Prox\ D\ c) = \mathbf{G}_{\mu(Dir,Prox),\sigma(Dir,Prox)}(Vrot)$
  - Identification : Apprentissage par télé-opération
- Question : $P(Vrot \mid [Dir = d]\ [Prox = p])$

Fig. 2.2: Structure d'un programme PBR et exemple de l'évitement d'obstacles.

Nous montrons Figure 2.2 le programme bayésien correspondant à notre exemple d'évitement d'obstacle. Sur cette Figure, nous rappelons d'une part la structure générique d'un programme bayésien dans le formalisme PBR (les accolades à gauche), et d'autre part les choix spécifiques à cet exemple (les contenus de chaque rubrique). Afin d'alléger les notations dans le reste de ce document, nous omettrons les symboles de données D et de connaissances préalables c en parties droites des termes probabilistes, lorsque cela ne générera pas d'ambiguïté.

Première partie

Étude bibliographique

Chapitre 3

Approches probabilistes

> « [...] la supériorité de l'homme tient justement dans sa capacité à juger, à raisonner sur des données floues, incomplètes, à évoluer dans un monde ouvert et changeant et non sur les soixante-quatre cases d'un échiquier. »
>
> Jacques Droulez, Alain Berthoz, René Zapata, *Prédiction et programmation des mouvements*, [DBZ99].

Notre étude bibliographique n'a pas la prétention de couvrir les approches de navigation et localisation en robotique au sens large. Nous renvoyons le lecteur à l'ouvrage de Borenstein, Everett et Feng [BEF95] [1] pour une introduction assez complète, qui donne une vue d'ensemble du domaine. Dans notre travail, nous nous restreindrons d'une part à quelques approches probabilistes de la modélisation de l'environnement, et d'autre part aux approches bio-inspirées de la navigation en robotique.

3.1 Relations entre formalismes probabilistes

3.1.1 Choix des approches présentées

Les approches probabilistes sont devenues inévitables en robotique mobile, grâce à leur capacité à traiter des données sensori-motrices incomplètes et incertaines. Bien sûr, de nombreux formalismes différents sont nés dans les trente dernières années.

Certains auteurs se sont intéressés aux liens existant entre ces approches. Ils ont le plus souvent constaté que ces approches différaient par les hypothèses qu'elles faisaient, mais parfois, ils ont reconnu des approches identiques, mais appelées différemment. Il serait illusoire de vouloir dresser une liste exhaustive des multiples noms que portent certains de ces formalismes. Il est en revanche plus aisé, et plus intéressant, d'étudier leurs relations, en les réécrivant dans un cadre commun.

[1] Ce livre est épuisé, mais un rapport technique identique [BEF96] est disponible sur le site de Johann Borenstein, http ://www-personal.engin.umich.edu/~johannb/.

Ainsi, Padhraic Smyth [SHJ97, Smyth98] replace par exemple les Filtres de Kalman et les Modèles de Markov Cachés dans le cadre des Réseaux Bayésiens. Kevin Murphy [Murphy02], lui, replace ces deux approches, et un grand nombre de leurs variantes, dans le cadres des réseaux bayésiens dynamiques. Il traite en détail des capacités de modélisation, d'inférence, et d'apprentissage de chacune de ses approches. Pour un bon résumé de cette partie de sa thèse, voir [Murphy01] [2]. Enfin, les travaux de Ghahramani [Ghahramani98, Ghahramani01, RG99] concernent les liens entre les Réseaux Bayésiens, les Réseaux Bayésiens Dynamiques, les Modèles de Markov Cachés, et certaines variantes de Filtres de Kalman.

Dans ce chapitre, nous reprendrons l'ensemble des approches citées ci-dessus, y ajouterons les Chaînes de Markov, les Filtres Bayésiens et les Processus de Décision Markoviens Partiellement Observables, et les réécrirons dans le formalisme PBR.

Cette réécriture aura tout d'abord l'avantage d'utiliser une notation et un cadre de présentation unique, ce qui est assez rare dans la littérature. De plus, elle permettra d'offrir au lecteur une vue d'ensemble de ces formalismes et de leurs relations. En explicitant les hypothèses faites par chacune des approches, nous remettrons également en perspective leurs nombreuses variantes, ce qui aidera le lecteur intéressé à en imaginer de nouvelles (nous définirons et utiliserons par exemple une extension de la localisation markovienne au Chapitre 8).

La liste des approches que nous traiterons donc dans ce chapitre est donnée Table 3.1, ainsi que leurs traductions en anglais (pour les lecteurs qui – comme nous – sont plus habitués aux acronymes anglais).

Noms français	Acronymes	Noms anglais	Acronymes
Programmation bayésienne des robots	PBR	Bayesian robot programming	BRP
Carte bayésienne	CB	Bayesian map	BM
Réseau bayésien	RB	Bayesian network	BN
Réseau bayésien dynamique	RBD	Dynamic bayesian network	DBN
Chaîne de Markov	CM	Markov chain	MC
Filtre bayésien	FB	Bayes filter	BF
Modèle de Markov caché	MMC	Hidden Markov model	HMM
Localisation markovienne	Loc Markov	Markov localization	Markov Loc
Filtre de Kalman	FK	Kalman filter	KF
Processus de décision markovien partiellement observable	PDMPO	Partially observable Markov decision process	POMDP
Processus de décision markovien (totalement observable)	PDM	(Fully observable) Markov decision process	(FO)MDP

Tab. 3.1 – Traductions des noms des formalismes abordés dans ce chapitre.

[2] Également disponible sur le site http ://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html.

3.1.2 Choix d'un ordre de présentation

Nous présenterons les différents formalismes en allant du plus général au plus spécifique. La mesure de « généralité » que nous avons choisie tient compte du nombre de contraintes posées par chaque approche : une approche sera plus générale qu'une autre si elle fait moins d'hypothèses. Cependant, le nombre d'hypothèses n'est pas un critère suffisant, car certaines hypothèses ne sont pas de nature comparable (par exemple, spécialiser une même approche en fixant un choix de décomposition ou un choix de variables) : ainsi, nous n'obtiendrons pas un ordre total, mais partiel sur l'espace des approches probabilistes possibles.

Cette relation « X est plus général que Y » entre les formalismes cités est présentée Figure 3.1. Nous y faisons apparaître notre contribution, la Carte Bayésienne, en pointillés.

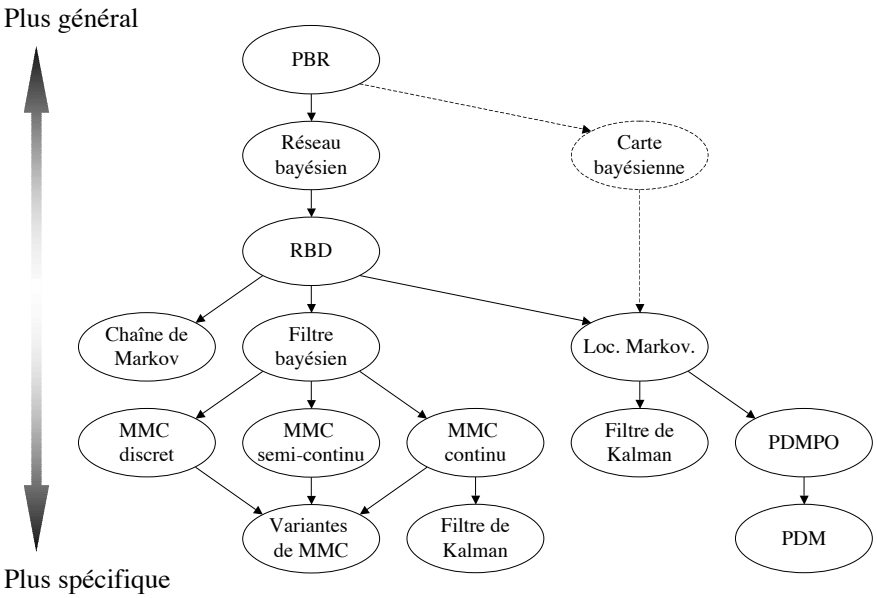


FIG. 3.1: Formalismes de modélisation probabiliste, et leur classement, du plus général (en haut), au plus spécifique (en bas). Les acronymes utilisés sont définis Table 3.1.

Bien sûr, les formalismes les plus généraux auront comme avantage un plus grand pouvoir d'expression (plus de modèles probabilistes en sont des instances). En revanche, la spécialisation permet d'utiliser des solutions efficaces aux problèmes d'inférence et d'apprentissage. Par exemple, les Filtres de Kalman permettent une inférence probabiliste exacte, grâce à une solution analytique. Les modèles de Markov cachés sont pour leur part célèbres, car leur choix de structure permet un apprentissage du modèle entier par l'algorithme de Baum-Welch. Dans notre travail, nous choisissons de restreindre notre analyse aux capacités de modélisation de chaque formalisme, sans développer ces méthodes d'inférence et d'apprentissage traditionnellement associées à chacun d'entre eux. En ef-

fet, ces méthodes constitueraient des domaines d'étude à part entière. Ne souhaitant donc pas détailler les inférences faites par chaque approche, lorsque nous les réécrirons dans le formalisme PBR, nous n'y détaillerons que la description correspondante, sans définir les questions probabilistes posées aux modèles.

La suite de ce chapitre reprendra donc chacune de ces approches brièvement, de la plus générale à la plus spécifique ; pour chacune d'entre elles, nous les présenterons intuitivement, en fournissant au lecteur intéressé quelques pointeurs de lecture, et rappellerons leurs définitions formelles, en les réécrivant dans le formalisme PBR.

3.2 Réseaux bayésiens

Un réseau bayésien [Pearl91, Heckerman96, LS88] est un modèle probabiliste portant sur n variables quelconques, $X_1, \ldots, X_n$, définissant la conjointe $P(X_1 \ldots X_n)$, par une décomposition satisfaisant la propriété suivante :

$$P(X_1 \ldots X_n) = \prod_{i=1}^{n} P(X_i |\ Pa_i),$$

où Pa_i est un sous-ensemble de $\{X_1, \ldots, X_{i-1}\}$.

Nous comprenons donc qu'il s'agit ici de n'apporter une contrainte que sur le type de décomposition possible. Les variables, leurs sémantiques, et les formes paramétriques peuvent être quelconques. C'est ce que nous résumons Figure 3.2.

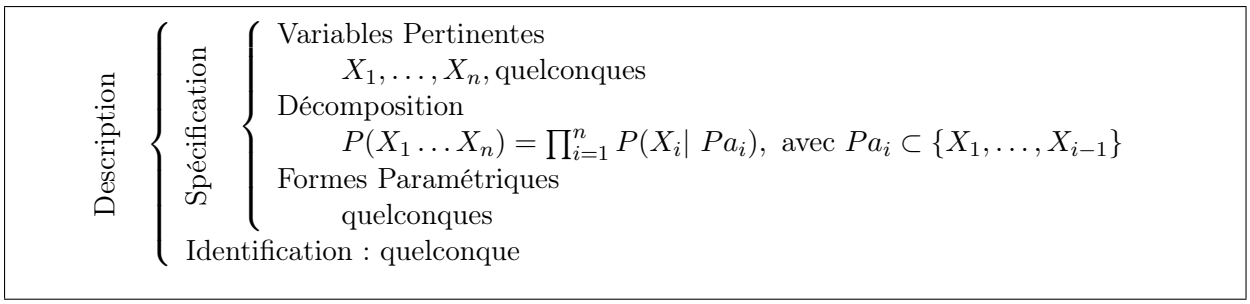


Fig. 3.2: Le formalisme des réseaux bayésiens exprimé en PBR.

Se restreindre à de telles décompositions permet d'assurer qu'un réseau bayésien se représente par un graphe, en associant un nœud à chaque variable X_i, et en établissant des arcs orientés de toutes les variables de Pa_i vers X_i. Cette représentation graphique présente de nombreux avantages, dont une meilleure explicabilité du système, ou une plus grande facilité à définir le modèle pour des experts non informaticiens par exemple.

Ce graphe permet aussi d'exploiter les « voisinages » des variables, dans la propagation des calculs lors de l'inférence sur le modèle. Cela permet notamment l'écriture d'algorithmes d'inférences exacts et efficaces, ainsi que la visualisation des flots d'information au sein du graphe.

Malheureusement, en restreignant la forme de la décomposition de la distribution conjointe, on limite aussi le pouvoir d'expression de ce formalisme (voir [PEARL91, SHJ97]), car toutes les dépendances conditionnelles ne sont plus représentables. En revanche, le formalisme PBR n'impose aucune condition sur la forme de la décomposition, et garde le pouvoir d'expression maximum. Par exemple, il y est possible d'écrire la décomposition suivante :

$$P(X\ Y\ Z \mid C) = P(X\ Y \mid C)P(Z \mid Y\ C).$$

Dans les réseaux bayésiens, la seule solution pour faire apparaître deux variables X et Y en partie gauche, comme dans $P(X\ Y \mid C)$, est de les agréger en une seule, $\langle XY \rangle$. Le problème est qu'il est ensuite impossible de les dissocier, et l'on ne peut plus écrire $P(Z \mid Y\ C)$, mais seulement $P(Z \mid \langle XY \rangle\ C)$. Nous venons donc d'exhiber une hypothèse d'indépendance conditionnelle particulière exprimable en PBR, mais pas dans le cadre d'un réseau bayésien. Nous avons également montré précédemment que tout réseau bayésien était définissable en PBR ; nous concluons donc que le formalisme PBR est plus général que les réseaux bayésiens.

3.3 Réseaux Bayésiens Dynamiques

Les réseaux bayésiens dynamiques [BELLOT02, BMR97, DK89, KKR95, MURPHY02, RUSSELL98] sont une spécialisation des réseaux bayésiens à la modélisation de phénomènes temporels. Ils supposent :

- que le temps se représente de manière discrète, $t = 1, \ldots, T$;
- que les nœuds du réseau bayésien peuvent être regroupés par « tranches » temporelles, qui contiennent N variables identiques : $Y_t^1, \ldots, Y_t^N$ du temps t ;
- que les variables Y_t^i d'une tranche t ne dépendent que d'autres variables de la même tranche ou de variables de la tranche précédente $t-1$;
- enfin, que les distributions associées à ces dépendances sont indépendantes de la tranche t considérée.

Ces hypothèses sont classiques : le fait qu'une tranche ne dépende que de la précédente est couramment appelée *hypothèse de Markov d'ordre 1* (d'ordre n pour le cas où une tranche dépend des n précédentes) ; le fait que les distributions de probabilités ne varient pas selon t est *l'hypothèse de stationnarité* (les modèles sont parfois également dits « invariants au temps » [3] ou encore « homogènes »).

Ces hypothèses permettent de définir un modèle *local* (temporellement) $P(Y_t^i \mid Pa_i)$, qui décrit les dépendances entre les variables Y_t^i et leurs parents Pa_i, indépendamment de l'indice de temps, puis de les réutiliser pour tout temps t, afin de définir le modèle *global*,

[3] *Time-invariant.*

suivant la dérivation [Murphy02] :

$$\begin{aligned} & P(Y_0^0 \ \ldots \ Y_0^N \ Y_1^0 \ \ldots \ Y_1^N \ \ldots \ Y_T^0 \ \ldots \ Y_T^N) \\ &= \ P(Y_0^0 \ \ldots \ Y_0^N) \prod_{t=1}^{T} \prod_{i=1}^{N} P(Y_t^i \mid Y_{t-1}^0, \ldots, Y_{t-1}^N, Y_t^0, \ldots, Y_t^{i-1}) \\ &= \ P(Y_0^0 \ \ldots \ Y_0^N) \prod_{t=1}^{T} \prod_{i=1}^{N} P(Y_t^i \mid Pa_i), \end{aligned}$$

où Pa_i est un sous-ensemble de $\{Y_{t-1}^0, \ldots, Y_{t-1}^N, Y_t^0, \ldots, Y_t^{i-1}\}$. Il apparaît alors clairement que les modèles locaux $P(Y_t^i \mid Pa_i)$ ne sont pas suffisants, et qu'il faut aussi définir l'a priori sur la tranche de temps initiale $t = 0$ ($P(Y_0^0 \ \ldots \ Y_0^N)$), pour rendre cette description globale fonctionnelle.

Nous comprenons donc que le modèle local de dépendance est un réseau bayésien, et qu'il s'agit ici de « l'enchaîner » dans le temps. Nous reviendrons sur la relation entre le modèle global et le modèle local dans le cadre des chaînes de Markov, où les écritures seront plus simples (Section 3.4).

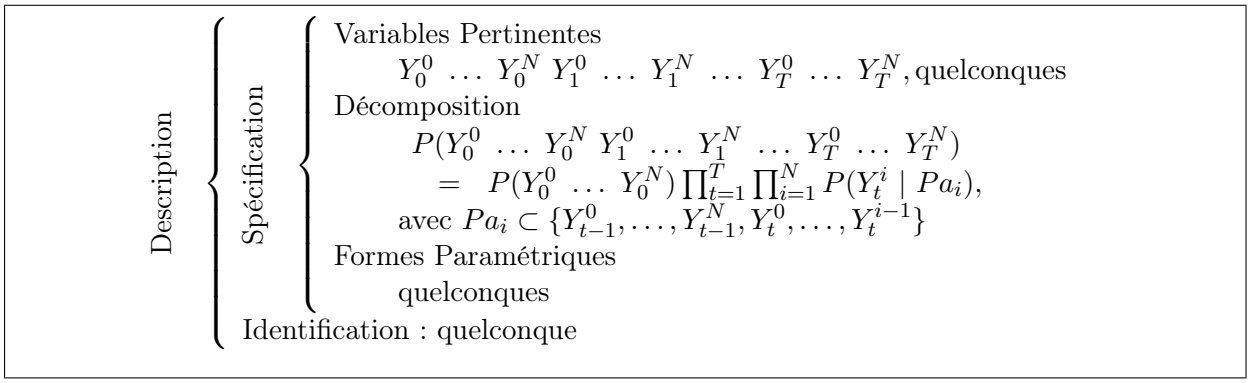


Fig. 3.3: Le formalisme des réseaux bayésiens dynamiques exprimé en PBR.

Nous réécrivons maintenant la structure du modèle global dans notre formalisme PBR, Figure 3.3 (la réécriture du modèle local est identique à celle des réseaux bayésiens).

3.4 Chaînes de Markov

Les chaînes de Markov [Rabiner89] sont l'exemple le plus simple de réseau bayésien dynamique. Il s'agit en effet ici de n'étudier que l'évolution d'une variable d'état discrète L_t unique. En ce sens, elles peuvent être considérées comme des filtres bayésiens (voir Section 3.6) dont l'état serait directement observable (donc des modèles de Markov « observables »). En appliquant une hypothèse de Markov de premier ordre, nous obtenons la

décomposition suivante pour le modèle global :

$$\begin{aligned} P(L_0 \dots L_t) &= P(L_0)P(L_1 \mid L_0)P(L_2 \mid L_1\ L_0)\dots P(L_t \mid L_{t-1}\ \dots\ L_0) \\ &= P(L_0)P(L_1 \mid L_0)P(L_2 \mid L_1)\dots P(L_t \mid L_{t-1}) \\ P(L_0 \dots L_t) &= P(L_0)\prod_{i=1}^{t} P(L_i \mid L_{i-1}). \end{aligned}$$

La simplicité de ce modèle n'empêche pas L_t d'être une conjonction de variables $L_t^1 \wedge \dots \wedge L_t^N$, comme dans les réseaux bayésiens dynamiques. Mais le fait de suivre la décomposition ci-dessus force l'écriture d'un modèle de transition de la forme $P(L_t^1\ \dots\ L_t^N \mid L_{t-1}^1\ \dots\ L_{t-1}^N)$, qui est plus contrainte que dans le formalisme des réseaux bayésiens dynamiques.

Nous disposons maintenant d'une décomposition pour le modèle global, il faut encore définir les formes paramétriques pour tous les termes y apparaissant. Le premier terme, $P(L_0)$, est l'a priori sur l'état discret du système, nous lui associons une loi de succession de Laplace. Quant aux termes $P(L_i \mid L_{i-1})$, chacun est une question probabiliste à un modèle local, qui porte sur deux variables, L et L'.

Dans la littérature, ce modèle local consiste uniquement en la définition du terme voulu, comme par exemple, dans notre cas des chaînes de Markov, $P(L' \mid L)$. Or, dans notre formalisme PBR, si nous distinguons les modèles globaux et locaux, chacun doivent définir la distribution conjointe sur l'ensemble de leurs variables. Dans les chaînes de Markov, cela implique que le modèle local définisse $P(L'\ L)$. Par exemple en choisissant la décomposition $P(L'\ L) = P(L)P(L' \mid L)$, et en définissant $P(L' \mid L)$ comme étant le modèle de transition qui nous intéresse. Mais il reste alors à définir $P(L)$. En pratique, cela n'est cependant pas utile, car le modèle local ne sera utilisé que pour en tirer $P(L' \mid L)$, et pour intégrer cette connaissance dans le modèle global.

Dans le modèle global d'ailleurs, les connaissances sur $P(L)$ ne sont pas quelconques. En effet, elles sont tirées de l'estimation récursive de l'état, $P(L_t \mid L_{t-1}\ \dots\ L_0)$. *Par abus de langage*, nous nous permettrons ainsi, lors de la définition du modèle local, d'attribuer au terme $P(L)$ une forme paramétrique dite « de récurrence ». Cette forme paramétrique, dans un modèle probabiliste, indiquera

- que ce modèle est un modèle local ;
- qu'on l'enchaînera dans le temps, par l'utilisation d'un modèle global (qui en général ne sera pas détaillé) ;
- que le terme sur lequel porte cette forme paramétrique sera le résultat de l'estimation récursive de cette variable au pas de temps précédent ;
- que le modèle global inclura une définition de l'a priori initial correspondant à ce terme.

Nous réécrivons le modèle global des chaînes de Markov dans notre formalisme PBR, Figure 3.4, et le modèle local Figure 3.5, où l'on voit apparaître la forme paramétrique de récurrence. Dans la suite de ce chapitre, nous ne définirons plus que les modèles locaux, la relation entre ces modèles et les modèles globaux étant implicite.

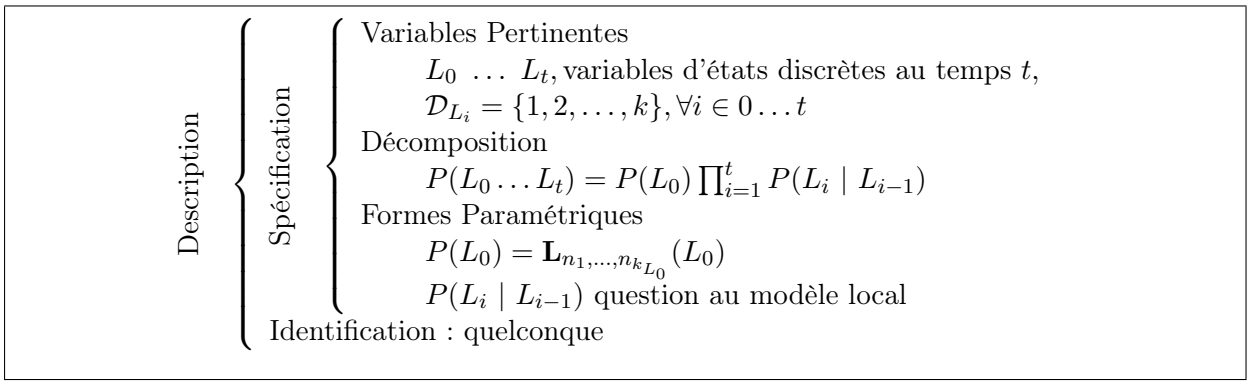


Fig. 3.4: Le formalisme des chaînes de Markov exprimé en PBR (modèle global).

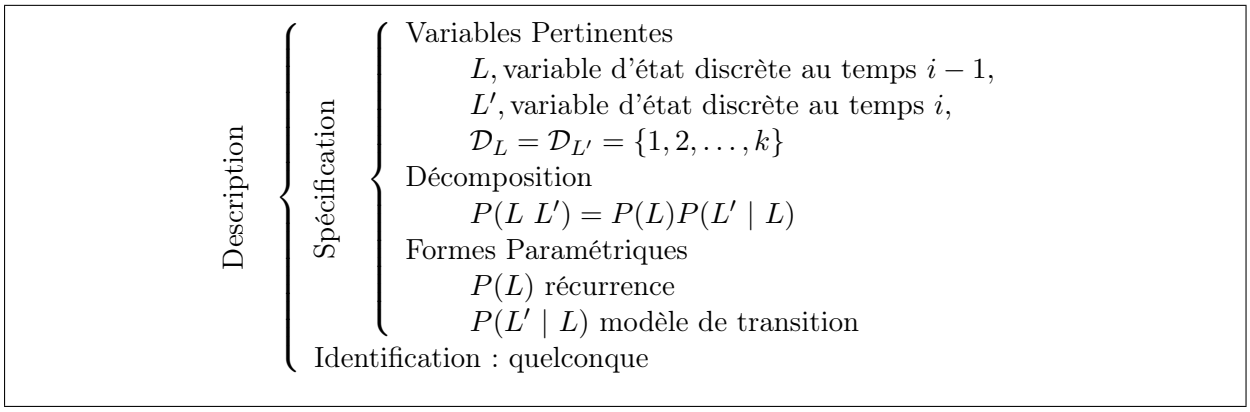


Fig. 3.5: Le formalisme des chaînes de Markov exprimé en PBR (modèle local).

3.5 Filtres Bayésiens

Nous regroupons sous le terme « filtre bayésien » [SBC99] tous les réseaux bayésiens dynamiques qui n'ont qu'une variable d'état, et qu'une variable d'observation P_t. Dans le cadre des filtres bayésiens, nous ne faisons pas d'hypothèses sur la nature des variables (discrètes ou réelles) ; lorsque nous en ferons, nous parlerons alors de modèles de Markov cachés. Les filtres bayésiens supposent de plus la décomposition suivante :

$$P(L_t \ L_{t-1} \ P) = P(L_{t-1})P(L_t \mid L_{t-1})P(P \mid L_t).$$

Le terme $P(L_{t-1})$ établit la récurrence, le terme $P(L_t \mid L_{t-1})$ est appelé « modèle de transition », et $P(P \mid L_t)$ est le « modèle d'observation ».

Nous résumons ces informations Figure 3.6.

3.6 Modèles de Markov Cachés

Un modèle de Markov caché [Aycard98, Rabiner89, RJ93, SK97, Shatkay98] est une spécialisation des filtres bayésiens, qui consiste à émettre une hypothèse sur la nature des variables.

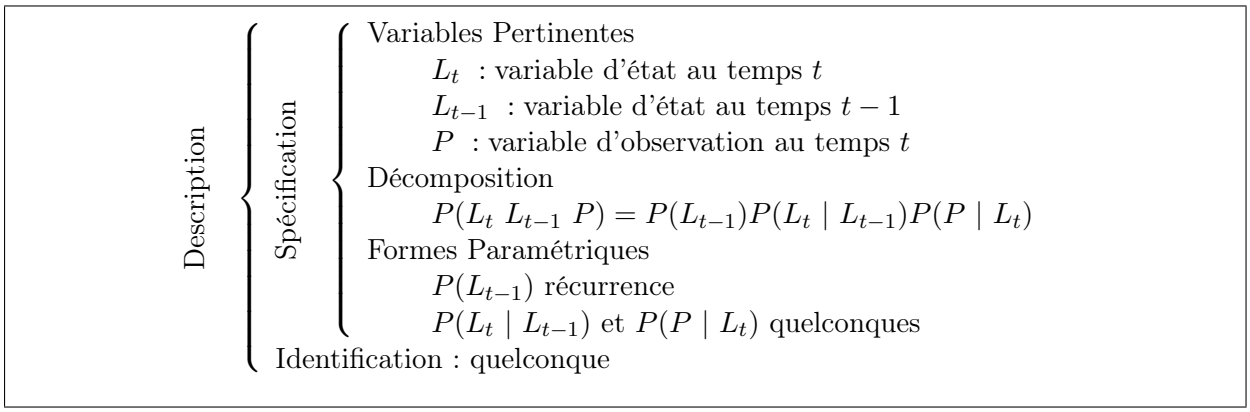


Fig. 3.6: Le formalisme des filtres bayésiens exprimé en PBR.

Dans le MMC « standard », les trois variables L_{t-1}, L_t et P sont supposées discrètes. Dans ce cas, les modèles de transition et d'observation prennent la forme de matrices, et sont donc respectivement appelés « matrice de transition » [4] et « matrice d'observation ». Le terme $P(L_{t-1})$ étant une récurrence, cela implique l'existence de l'*a priori* initial $P(L_0)$, qui lui aussi prend alors la forme d'une matrice. Cette *a priori* est, en général, noté π.

Lorsque la variable de perception P est supposée continue, on parle alors de MMC semi-continu [Rabiner89, Murphy02]. Dans ce cas, le modèle d'observation est en général soit un modèle gaussien, soit une mixture de distributions gaussiennes.

Les modèles de Markov cachés ont été utilisés avec succès dans des domaines variés (reconnaissance de la parole, séquençage du génome, etc.), mais aussi dans la navigation robotique [Aycard98, AMS01]. Ils doivent notamment leur succès à l'existence d'algorithmes d'inférence adaptés à leur structure de dépendance : l'algorithme de Viterbi estime la séquence la plus probable d'états au vu d'une séquence d'observations, et l'algorithme de Baum-Welch permet d'identifier les modèles d'observation et de transition à partir de données expérimentales.

Nous présentons la réécriture du formalisme des MMC (discrets) dans la notation PBR Figure 3.7.

3.7 Spécialisations des MMCs

3.7.1 Spécialisation de la structure de dépendance

Kevin Murphy, dans sa thèse [Murphy02], présente de nombreuses variantes des modèles de Markov cachés, en les replaçant dans le cadre des réseaux bayésiens dynamiques. Ces variantes ont pour point commun d'utiliser des structures de dépendances particulières. Parmi ces nombreuses variantes, notons :

– les « auto-regressive HMMs », où les variables d'observations aux temps $t-1$ et t ne sont plus considérées indépendantes, conditionnellement à la connaissance de

[4] Ou « matrice stochastique ».

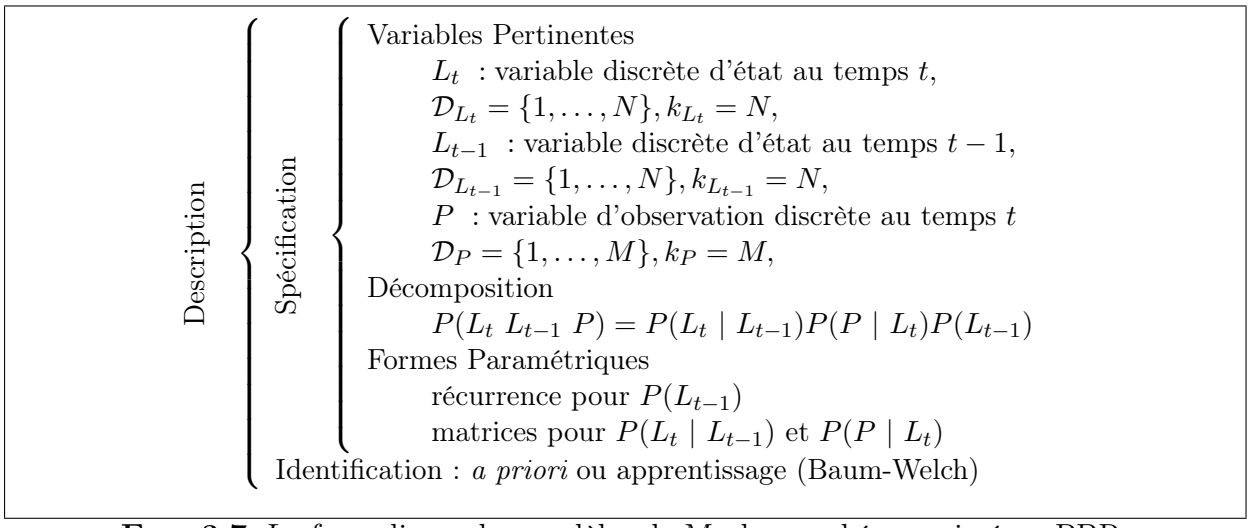


Fig. 3.7: Le formalisme des modèles de Markov cachés exprimé en PBR.

l'état L_t. Dans la décomposition, le modèle d'observation $P(P \mid L_t)$ devient donc $P(P_t \mid L_t\ P_{t-1})$;

- les « factorial HMMs », où plusieurs chaînes de Markov (sur les variables d'états $X^1, \ldots, X^n$), supposées indépendantes, ne génèrent qu'une séquence unique d'observations P. Dans ce formalisme, il y a plusieurs modèles de transitions ($P(X_t^1 \mid X_{t-1}^1)$, $\ldots$, $P(X_t^n \mid X_{t-1}^n)$), et un unique modèle d'observation conditionné par toutes les variables d'états ($P(P \mid X_t^1\ \ldots\ X_t^n)$) ;
- les « coupled HMMs », où l'on représente l'influence que peut avoir un modèle de Markov caché sur un autre, en supposant que la dépendance s'établit entre les variables d'états X_t^1 et X_t^2. Les modèles de transition deviennent $P(X_t^1 \mid X_{t-1}^1\ X_{t-1}^2)$ et $P(X_t^2 \mid X_{t-1}^1\ X_{t-1}^2)$. Bien sûr, cette notion est généralisable au cas de n modèles de Markov cachés couplés.

Nous aborderons les « input-output HMMs » Section 3.8, et les modèles hiérarchiques à base de MMC Section 4.1.

3.7.2 Spécialisation des formes paramétriques : les Filtres de Kalman

Spécialisons les modèles de Markov cachés en faisant les hypothèses suivantes :

- supposons que les variables d'état et d'observation soient continues ;
- supposons que les modèles de transition $P(L_t \mid L_{t-1})$ et d'observation $P(P \mid L_t)$ soient gaussiens ;
- supposons que les fonctions $\mu_O(L_t), \sigma_O(L_t)$ et $\mu_T(L_{t-1}), \sigma_T(L_{t-1})$ qui définissent les moyennes et écart-types de ces modèles gaussiens soient linéaires [5] ;
- supposons enfin que l'a priori initial $P(L_0)$ soit gaussien.

[5] Si elles ne sont pas linéaires, on en obtient une approximation linéaire par développement de Taylor, et on parle alors de Filtre de Kalman Étendu (FKÉ).

Nous obtenons alors le célèbre « Filtre de Kalman » [BBM94, WB95]. Le point essentiel à retenir, au sujet de ce modèle, est que ces hypothèses rendent possible l'inférence exacte d'une distribution de probabilités sur les états au vu des observations, grâce à une solution analytique et récursive. Un des désavantages est que cette distribution de probabilités sur les états est de forme gaussienne, donc unimodale. Elle ne permet donc de ne considérer qu'une hypothèse unique de localisation. On parle alors de suivi d'hypothèse [6], par opposition à la localisation globale (ou multi-hypothèses).

Les Filtres de Kalman ont été utilisés avec succès en robotique dans de nombreux systèmes : voir par exemple [Crowley89, LDC92].

Nous présentons la réécriture du formalisme des Filtres de Kalman dans la notation PBR Figure 3.8.

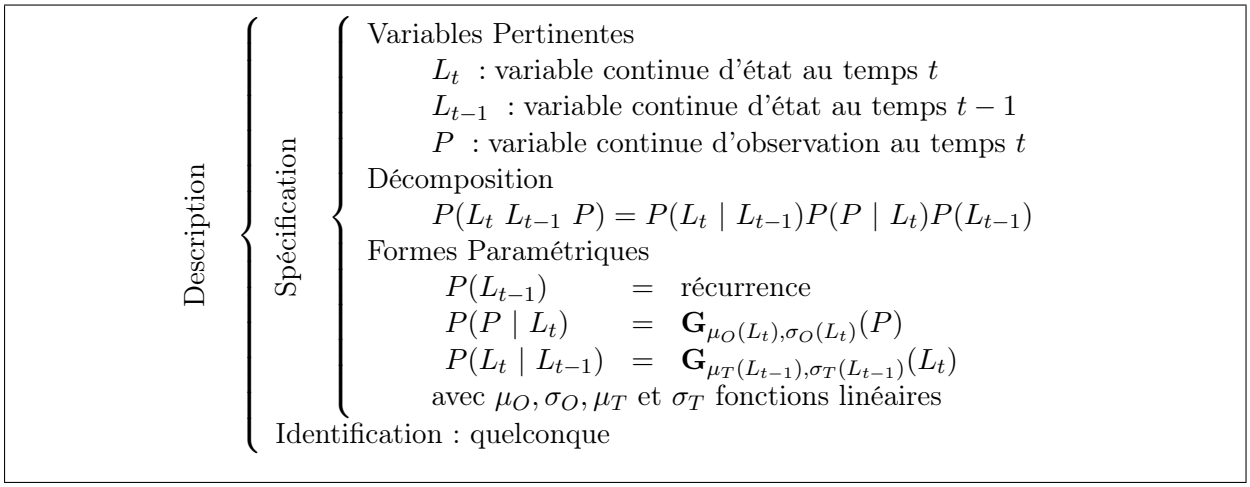


Fig. 3.8: Le formalisme des filtres de Kalman exprimé en PBR.

3.8 Localisation Markovienne

Nous avons vu jusqu'à maintenant deux spécialisations possible des réseaux bayésiens dynamiques : les chaînes de Markov (modèles $P(L_t\ L_{t-1}) = P(L_{t-1})P(L_t \mid L_{t-1})$), et les filtres bayésiens (modèles $P(P\ L_t\ L_{t-1}) = P(L_{t-1})P(L_t \mid L_{t-1})P(P \mid L_t)$). Continuons à enrichir ces formalismes, en ajoutant une variable d'action A.

Dans la littérature, cette variable d'action est intégrée dans le modèle de transition $P(L_t \mid L_{t-1})$, qui devient $P(L_t \mid A\ L_{t-1})$, et s'appelle donc parfois « modèle d'action ». Formellement, il faut également définir le terme $P(A)$. Mais la variable A se trouve, dans la littérature, toujours en partie droite des questions probabilistes. Ainsi, le terme $P([A = a])$ est toujours assimilable à la constante de normalisation, et les auteurs se passent de la définition de ce terme.

Ce formalisme s'appelle parfois input-output HMM [BF95, CN94, Ghahramani01, MJ96], mais nous l'avons étudié plus en détails sous le nom de localisation markovienne

[6] *Hypothesis tracking.*

[ALC98, BFHS96, Thrun00, TBF98]. Sous ce nom, il a été principalement développé et utilisé à l'Université de Carnegie Mellon, bien que récemment, les mêmes auteurs semblent décidés à l'appeler « filtre bayésien » [Thrun02]. Bien sûr, il ne faut pas confondre ces filtres bayésiens avec ceux que nous avons défini Section 3.5, qui, eux, ne contiennent pas de variable d'action A.

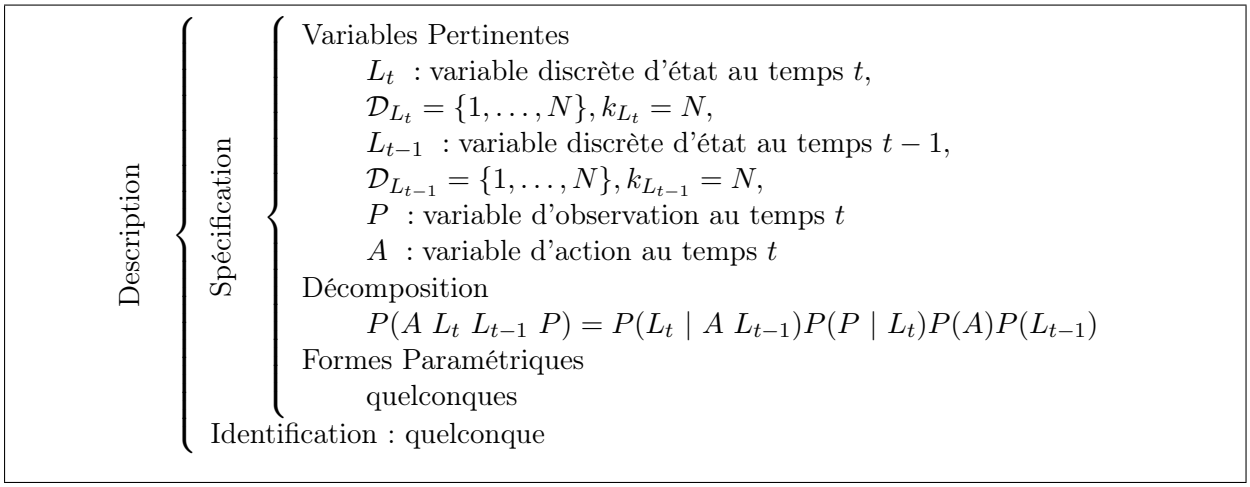


Fig. 3.9: Le formalisme de la localisation markovienne exprimé en PBR.

Sous le nom de « localisation markovienne », ce formalisme a connu des implantations sur des robots déployés dans des expositions, avec les deux robots Rhino [TBB+98, BCF+99] et Minerva [TBB+99a, TBB+99b], ce qui a également suscité de nombreuses améliorations et extensions :

- pour la navigation active (choisir une trajectoire et l'orientation des capteurs pour maximiser le gain d'information, et ainsi, réduire les risques de se perdre) [FBT98, RBFT99] ;
- pour la localisation en environnements dynamiques (filtrer les données qui ne correspondent pas à la lecture attendue – mais on perd alors la capacité de localisation globale) [FBT99] ;
- pour la localisation de robots effectuant une tâche en collaboration (lorsqu'un robot en voit un autre, il peut l'aider à se localiser) [FBKT00] ;
- pour le calcul efficace de l'inférence, des méthodes de Monte Carlo ont été associées au formalisme de la localisation markovienne (représentation des distributions par des nuages de points) [FBDT99, TFB00] ;
- et enfin, pour la construction de cartes en trois dimensions (basées sur des capteurs lasers planaires) [TBF00, Thrun02].

Ces nombreuses améliorations sont à l'origine du succès pratique de ce formalisme, notamment les variantes dites de Monte Carlo. En effet, elles rendent l'inférence réalisable en temps de calcul réduit (et même en temps *any-time*), ce qui permet une implantation à bord du robot.

Nous présentons la réécriture du formalisme de la localisation markovienne dans la notation PBR Figure 3.9.

3.9 Spécialisations de la Localisation Markovienne

3.9.1 Spécialisation des formes paramétriques : les Filtres de Kalman

Il existe une variante des Filtres de Kalman que nous avons défini Section 3.7.2, qui consiste à ajouter une variable d'action A, tout en conservant les hypothèses linéaires-gaussiennes. Ce formalisme conserve le nom de « Filtre de Kalman », et se place, dans notre analyse, comme une spécialisation de la localisation markovienne.

Nous présentons la description PBR associée Figure 3.10.

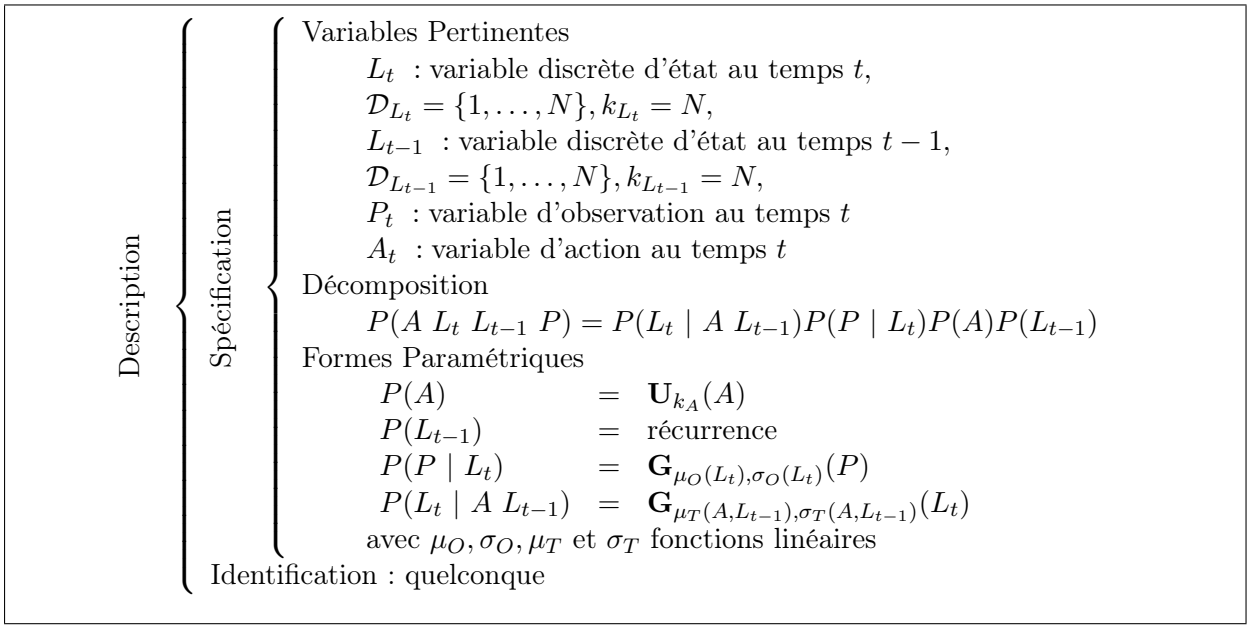


Fig. 3.10: Le formalisme des filtres de Kalman (avec variable d'action) exprimé en PBR.

3.9.2 Définition d'une fonction de récompense : PDMPO et PDM

Les Processus de Décision Markovien Partiellement Observables (PDMPO) [CKK96, KLC98, KS96, SK95], sont basés sur un modèle probabiliste identique [7] à de la localisation markovienne, et on ajoute à ce modèle une notion de coût ou de récompense. Formellement, il peut s'agir d'une fonction $R : L_t \rightarrow \mathbb{R}$, qui associe à chaque état une récompense, ou bien une fonction $C : L_t \times A \rightarrow \mathbb{R}$, qui associe un coût à l'exécution des actions [BDH99]. Parfois, et cette écriture est plus générale, ces deux fonctions sont assemblées en une seule, $R : L_t \times A \rightarrow \mathbb{R}$ [KLC98].

Cette fonction de récompense est la base des algorithmes de planification développés dans les PDMPOs. Les récompenses servent à définir les buts et les endroits à éviter de

[7] On trouve parfois des variantes, notamment en ce qui concerne le modèle d'observation : il peut prendre la forme $P(P \mid A\ L_t)$ [PT02], ou même $P(P \mid A\ L_t\ L_{t-1})$ [BDH99].

l'environnement. Le but de l'algorithme de planification est alors de maximiser une certaine mesure, calculée à partir de la fonction de récompense. En général, c'est le gain moyen *discounted* qui est choisi comme mesure : $E(\sum_{t=0}^{\infty} \gamma^t R_t)$, où γ est un facteur de *discount*, R_t la récompense obtenue au temps t, et $E(.)$ est l'espérance mathématique.

Il faut donc déterminer, pour chaque état, quelle action est à choisir qui permettra de maximiser cette mesure. Malheureusement, l'état du système n'est pas directement accessible, et on ne dispose que d'une connaissance sur cet état, par la distribution de probabilité $P(L_t)$. La fonction $\pi : P(L_t) \rightarrow A$, qui associe à chaque distribution sur L_t une action, est appelée politique [8]. Plusieurs algorithmes existent pour trouver la meilleure politique, les plus connus étant *value iteration* et *policy iteration*. Ces algorithmes sont itératifs : ils améliorent à chaque iteration la politique π, et s'arrêtent après convergence numérique de la solution.

Le problème majeur de ces algorithmes est le temps de calcul exponentiel (en la taille du domaine de L_t) pour arriver à une solution exacte ; en effet, une politique est une fonction sur l'espace des distributions de probabilités possibles ! Ceci réduit considérablement la taille des environnements sur lesquels le POMDP peut être défini : les approches les plus récentes (2002) sont inapplicables au delà d'une centaine d'états [PT02].

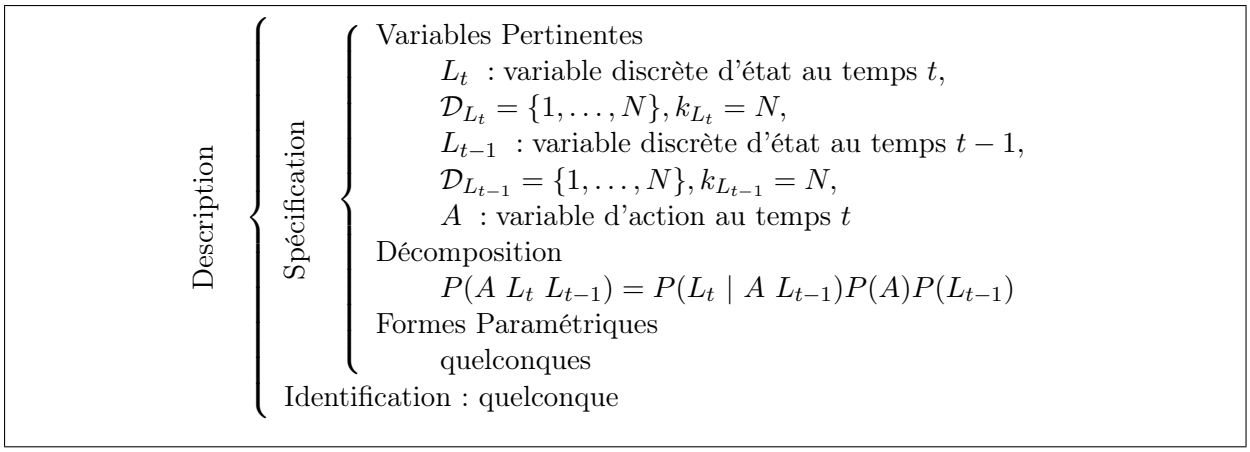


Fig. 3.11: Le formalisme des PDM exprimé en PBR. À ce modèle probabiliste, il faut ajouter la définition de la fonction $R : L_t \times A \rightarrow \mathbb{R}$ de récompense.

Plusieurs solutions existent à ce problème calculatoire ; nous n'en retiendrons que deux.

La première consiste à hiérarchiser des modèles de type PDMPOs, ce qui se comprend très bien intuitivement : souvent, il vaut mieux résoudre plusieurs problèmes de petite taille qu'un seul problème de grande taille. Nous développerons plus en détail ces approches Section 4.1.

La seconde solution au problème calculatoire posé par un PDMPO consiste à supposer que l'état du modèle est complètement observable. On parle alors de Processus de Décision Markovien Totalement Observable, ou, plus simplement, de Processus de Décision Mar-

[8] *Policy.*

kovien (PDM) [9]. Dans ces modèles, la variable d'observation et le modèle d'observation disparaissent du modèle probabiliste, et la fonction de politique devient une fonction sur les états, $\pi : L_t \rightarrow A$ (voir Figure 3.11). L'application d'un algorithme tel que *value iteration* devient abordable, mais nécessite tout de même un temps quadratique en la taille du domaine de L_t. Pour des espaces de grande taille, cela justifie d'utiliser des algorithmes approximés, ou de se rabattre une fois de plus sur des modèles hiérarchiques.

3.10 Questions « fil conducteur »

Pour conclure, nous reprenons nos questions « fil conducteur », et examinons les réponses classiquement fournies par les travaux mentionnés dans ce chapitre. Cela nous permettra également d'aborder certains points de ces approches qui sont plus liés à leur mise en œuvre qu'aux formalismes eux-mêmes.

Q2. Qu'est-ce qu'un lieu, pour le robot ? Classiquement, les systèmes de robotique mobile utilisant des cartes se classent en deux catégories, selon le type de la représentation de l'environnement : la variable d'état L_t peut avoir une sémantique *métrique* ou *topologique* (voir [Filliat01] pour un bon résumé de ces alternatives). Les approches citées dans ce chapitre n'échappent pas à la règle.

Q2. Que sont les représentations *topologiques* et *métriques* ? Il existe deux approches principales pour des variables d'états de nature métrique : les grilles d'occupation, ou les primitives géométriques [10]. Deux exemples représentatifs et bien documentés dans la littérature de ces approches sont les grilles d'occupation probabilistes de Thrun [Thrun98b], et les cartes basées sur des ensembles de droites infinies de Tomatis [TNAS01]. Comme noté dans de nombreuses analyses des approches métriques, celles-ci permettent une représentation de bonne précision (très bonne dans le cas des primitives géométriques), mais sont coûteuses en ressources (temps de calcul et mémoire), et s'accommodent mal de la dynamique des environnements. Elles peuvent également être difficile à mettre en œuvre, notamment lors de la fermeture de cycles dans l'environnement.

Le fait que la précision des approches métriques est grande se traduit par une variable d'état L_t ayant un grand domaine de valeur. Dans le cas des approches métriques à base de grilles d'occupation, L_t est typiquement la *pose* du robot, c'est-à-dire à la fois une position x, y et une orientation θ. Si la grille possède des cases de 50 cm de côté, que l'on souhaite une résolution angulaire de 10 °, et que l'environnement à cartographier mesure 50 m × 50 m, le cardinal du domaine de L_t est 360 000. Cela est bien au delà des capacités de planification dans les processus de décision markovien !

Ces formalismes (PDM et PDMPO) sont donc en général associés à des représentations topologiques, où ne sont représentés que quelques endroits d'intérêt (typiquement faci-

[9] À l'opposé, si le robot ne perçoit rien de son état, il est en *boucle ouverte*, et l'on parle alors de Processus de Décision Markovien Non-Observable (PDMNO) [BDH99].

[10] *Features*.

lement distinguables), reliés par des notions d'adjacence (typiquement un graphe). Bien sûr, formellement, les cartes métriques sont un cas particulier des cartes topologiques, car une carte métrique encode elle aussi des endroits d'intérêt et leurs relations spatiales. Mais en pratique, on parle de représentation topologique lorsque la variable d'état est de « faible cardinalité ». On trouve cependant de grandes variations quant à la cardinalité de la variable d'état dans les approches topologiques. On rencontre parfois les qualificatifs « denses » et « épars » [11] pour préciser le degré d'éloignement dans l'environnement des nœuds du graphe topologique.

Le choix d'un formalisme (PDMPO, HMM, etc.) pour un système robotique est orthogonal au choix d'une représentation [12] (topologique, métrique à base de grille d'occupation, etc.). Les associations citées ci-dessus, entre les approches et les représentations, sont plus de nature « historique », ou liées à des contraintes calculatoires.

Q2. Que sont les problèmes de perceptual aliasing et de global connectivity ?
Le *perceptual aliasing*, dans sa définition la plus courante, consiste à disposer d'une représentation interne telle que deux lieux *différents* soient associés aux mêmes perceptions. Il devient alors impossible de localiser instantanément le robot au seul vu d'une donnée capteur, car celle-ci peut correspondre à plusieurs lieux distincts. Le problème inverse est le problème de *global connectivity* : il intervient lorsqu'un même lieu peut générer des perceptions différentes. Il porte ce nom car il est lié à la fermeture de cycles dans l'environnement. En effet, lorsqu'un robot parcourt un cycle de l'environnement, il revient à son point de départ, mais ses données odométriques ne lui indiquent pas ce fait, car elles ont accumulé des erreurs le long du parcours. Si les capteurs sont bruités, ou si l'environnement a légèrement changé, ou si le robot ne reprend pas exactement la même trajectoire qu'au premier passage dans cette section de l'environnement, alors ses lectures capteurs ne seront pas les mêmes.

Ces deux problèmes ont des solutions diverses dans la littérature. Le perceptual aliasing se résout :

- soit en utilisant des capteurs suffisamment riches pour être discriminants sur tous l'environnement (caméra, capteurs lasers) ;
- soit en modifiant l'environnement (amers artificiels, codes barres) ;
- soit en utilisant l'historique de perception du robot pour désambiguïser sa position réelle (l'histoire du robot peut être assimilée à un capteur supplémentaire) ;
- soit enfin en utilisant une approche permettant de maintenir plusieurs hypothèses de localisation, et en attendant que le robot rencontre une donnée perceptive discriminante.

La global connectivity se résout, au moment de la cartographie de l'environnement :

- dans le cas des approches probabilistes, par des algorithmes de type EM, qui rétablissent la structure topologique réelle de l'environnement (retrouvent les cycles) ;
- par des heuristiques de navigation, qui « testent » des hypothèses d'égalité entre

[11] *Dense* ou *sparse*.
[12] Filtres de Kalman mis à part, car ils font l'hypothèse d'une variable d'état continue.

lieux, en explorant le voisinage de la position actuelle du robot, et en la comparant au voisinage du lieu précédemment rencontré.

Q4. Qu'est-ce que se localiser ? Cette capacité est bien analysée dans toutes les approches citées, car l'estimation de l'état au vu des données sensorielles est leur but premier. Il s'agit donc, dans une formulation probabiliste un peu générale, d'estimer par inférence une distribution de probabilités sur la variable d'état, sachant les variables perceptives (et/ou d'actions) : ceci peut se traduire, dans le modèle local, par la question $P(L_t \mid A_t\ P_t)$.

Q7. Comment apprendre expérimentalement les cartes ? La construction automatique de cartes est un domaine qui a généré de nombreuses recherches en robotique mobile. Ce domaine a récemment reçu le nom de SLAM, pour *Simultaneous Localization And Mapping* en anglais, car il s'agit de construire une carte au fur et à mesure de l'exploration de l'environnement, tout en se localisant dans la carte partielle déjà construite. Dans ce cadre, résoudre le problème de global connectivity afin d'obtenir une carte topologiquement correcte est crucial, mais rarement considéré. La communauté réalise ces derniers temps que le choix d'une stratégie d'exploration peut considérablement aider à la résolution de ce problème.

Q1. Qu'est-ce qu'une carte ? Les formalismes cités sont des modèles liant variables d'états, et/ou d'observation, et/ou d'actions. Les distributions de probabilités sur ces variables représentent les connaissances que le système a de son environnement. En ce sens, ces distributions sont la *carte* de cet environnement.

Cependant, peu d'auteurs choisissent cette définition du mot « carte ». En effet, les connaissances probabilistes sont trop riches pour être appréhendées intuitivement par l'utilisateur. En conséquence, le mot « carte » désigne plutôt la partie des connaissances probabilistes que l'on peut afficher sous forme graphique. Soit le modèle probabiliste mémorise explicitement un ensemble de primitives géométriques facilement projetable en deux dimensions, soit cette projection est plus implicite, et traitée par une routine d'affichage (par exemple, dans le cas des grilles d'occupation).

Chapitre 4

Approches hiérarchiques

> « Les ouvrières semblent apprendre les probabilités de trouver du nectar, sur la base de la fréquence avec laquelle elles rencontrent différents types d'états récompensants, et débutent leur vie de butineuse sans en avoir aucune estimation préalable. »
>
> L. A. Real, *Animal choice behavior and the evolution of cognitive architecture*, Science, n° 253, 1991, p. 980–986 [Real91] ; traduction en français dans [Damasio95].

Dans ce chapitre, nous présentons quelques approches de la navigation et de la modélisation de l'environnement qui se basent sur des *hiérarchies*. Nous développerons tout d'abord les approches probabilistes, qui sont des variantes des modèles présentés au Chapitre 3, puis les approches dites « bio-inspirées », ou « bio-mimétiques », qui ont pour caractéristique principale de s'inspirer du vivant.

4.1 Modèles hiérarchiques probabilistes

Les formalismes du chapitre précédent n'incluent pas de hiérarchie : rien n'y est dit sur la possibilité d'emboîter des modèles d'environnements d'échelles différentes. Bien sûr, cela est normal si l'on considère que ces approches visent à une généralité, et notamment une indépendance par rapport au domaine d'application. Mais dans le domaine de la robotique, le besoin d'approches hiérarchiques de la représentation de l'environnement pour la résolution des tâches de navigation se fait de plus en plus sentir. Ainsi, des efforts se sont développés très récemment en ce sens. Nous en recensons ici quelques uns, et les classons en deux catégories :

- nous traiterons tout d'abord des modèles hiérarchiques basés sur un seul formalisme, comme par exemple les modèles de Markov cachés hiérarchiques et les processus de décisions markoviens (partiellement observables ou non) hiérarchiques ;
- nous aborderons ensuite les modèles hiérarchiques basés sur plusieurs formalismes, comme les cartes à base de filtres de Kalman et de localisation markovienne de Tomatis, et les cartes métriques-topologiques de Thrun.

4.1.1 Modèles de Markov Cachés Hiérarchiques

Les modèles de Markov cachés hiérarchiques tirent leur inspiration du côté « automate » des modèles de Markov cachés [1]. Ainsi, la structure de dépendance est formellement la même que celle présentée Section 3.6, mais c'est l'espace des états $\mathcal{D}_{L_t}$ qui prend une structure hiérarchique. En particulier, il s'agit de permettre à un état d'un MMC « d'inclure » dans un de ses états un autre MMC : à l'exécution, arriver à cet état provoque un appel au MMC inclus, appel qui retourne lorsque celui-ci atteint un de ses états finals. Ce MMC de plus bas niveau d'abstraction peut également inclure lui-même d'autres MMC, et ainsi de suite, récursivement.

Bien évidemment, ce formalisme tient plus de l'automate « probabilisé » que du modèle probabiliste réellement hiérarchique. En effet, l'utilisation des concepts d'exécution et d'état final n'ont que peu de sens d'un point de vue probabiliste, où le système n'est jamais *dans un état*, mais où le système connaît une distribution de probabilités sur les états. Dans un modèle probabiliste, il est ainsi impossible *d'être* dans un état particulier, on peut au mieux avoir *une grande probabilité d'être* dans un état particulier.

Il existe plusieurs variantes du formalisme des MMCH. Par exemple, dans sa thèse, Aycard emboîte des modèles de Markov cachés portant sur des variables d'états de sémantiques différentes [Aycard98], au contraire des MMCH où la structure hiérarchique est contenue dans le modèle de transition portant sur une seule variable d'état. Toutes ces variantes s'appuient sur la notion d'état final des MMC.

4.1.2 Abstraction d'actions dans les PDM

Beaucoup d'auteurs ont noté que les processus de décision markoviens sont limités dès que l'espace d'état est grand. Dans le cadre de la navigation robotique, cette limitation se traduit en une limitation sur la taille de l'environnement que l'on peut représenter.

Une solution consiste à emboîter hiérarchiquement des PDM plus simples. Plusieurs méthodes sont envisageables pour créer ces PDM emboîtés [HMK+98, LK01b]. La plus courante consiste à partir d'un PDM représentant l'environnement entier, à en tirer des macro-états (régions) et/ou des macro-actions (sous-tâches), et à planifier dans ce PDM réduit.

Effectivement, les auteurs améliorent ainsi les capacités de représentation de l'environnement des PDM, mais se heurtent aux problèmes

- de trouver automatiquement une bonne décomposition de l'espace en régions, c'est-à-dire de trouver un moyen de générer automatiquement une bonne abstraction de l'interaction entre le robot et son environnement ;
- et de générer, dans l'espace de base initial, un plan optimal. La représentation de base étant métrique, cette optimalité prend le sens métrique, c'est-à-dire de générer la plus courte trajectoire.

[1] Une présentation rapide de ce formalisme se trouve dans la thèse de Kevin Murphy [Murphy02], nous renvoyons le lecteur aux références qu'il mentionne.

Certaines variantes très récentes de ces travaux visent à développer des formalismes applicables en pratique dans des systèmes robotiques réels. Or, les processus de décisions markoviens, de par leur hypothèse d'observabilité de l'état, sont peu applicables. Ainsi, certains travaux étendent les concepts de macro-actions et macro-états aux PDMPOs [PT02].

4.1.3 Filtre de Kalman et PDMPO

Tomatis note que les formalismes probabilistes, et leurs utilisations classiques en navigation robotique, ont des caractéristiques qui peuvent être complémentaires dans une application de navigation en intérieur structuré (bureaux et couloirs, par exemple) [TNAS01]. Il propose alors d'utiliser des formalismes différents dans les différentes régions de l'environnement. En effet, dans la classe de tâches qu'il considère, le robot a besoin d'une navigation de grande précision dans les bureaux : les filtres de Kalman sont donc le formalisme approprié, car ils font l'hypothèse d'un état L_t continu. Un filtre de Kalman est donc utilisé par bureau. En revanche, dans les couloirs, la tâche consiste à aller de la porte d'un bureau à l'autre : les erreurs de navigation peuvent être plus importantes, tant qu'elle n'amènent pas le robot à se tromper de porte. Il utilise donc, pour représenter le couloir, un PDMPO, avec des transitions pour représenter l'arrivée dans un bureau.

Le problème principal de ce modèle est qu'il ne s'agit pas réellement d'une hiérarchie de représentation à différents niveaux, car l'auteur n'inclut pas la possibilité d'exécuter simultanément les deux modèles : soit le robot sait qu'il se trouve dans le couloir, et l'estimation de sa position est faite dans le PDMPO, soit il sait se trouver dans un bureau particulier, et exécute alors le cycle de localisation dans le filtre de Kalman décrivant ce bureau. Le problème devient crucial à la transition entre le couloir et un bureau : dans le PDMPO, le robot peut maintenir plusieurs hypothèses sur sa localisation, mais quand il rentre dans un bureau, il ne peut activer qu'un seul filtre de Kalman. Si le robot se trompe à cet instant, et active un filtre de Kalman qui ne correspond pas au bureau où entre réellement le robot, la mission du robot sera un échec.

Dans une vraie hiérarchie de représentation, la multi-modalité du niveau PDMPO serait conservée, et permettrait au système de mettre en compétition les filtres de Kalman des bureaux les plus probables. Dans notre approche, une telle hiérarchie de modèles pourra être développée grâce à notre opérateur d'abstraction de cartes (voir Chapitre 9).

4.1.4 Modèle probabiliste et graphe topologique

Une autre approche possible pour l'emboîtement de modèles consiste à laisser les couches de bas niveau d'abstraction traiter les incertitudes, et exprimer les couches de plus haut niveau dans des formalismes déterministes. Dans le cadre de la navigation, il s'agit par exemple de tirer des graphes de modèles de type localisation markovienne métrique [Thrun98b] ou processus de décision markovien [LK01a, LK02]. Ces approches espèrent ainsi pouvoir s'affranchir de la complexité de la gestion des incertitudes aux plus haut niveaux d'abstraction. Ainsi, si l'environnement peut être représenté par un graphe, il devient facile et peu coûteux de planifier une trajectoire (par l'algorithme A* par exemple),

voire même d'optimiser l'ordre dans lequel les sous-tâches doivent se résoudre (par les heuristiques traitant du problème du voyageur de commerce [LK01a]).

Le problème principal de ces approches est similaire au cas précédent de Tomatis : est-il vraiment possible de s'affranchir des incertitudes, à quelque niveau d'abstraction que ce soit, lorsque le système est plongé dans le monde physique ? Le fait même que les approches citées ici se cantonnent aux environnements statiques, ou dont la dynamique est très restreinte, tend à prouver que non.

4.2 Modèles hiérarchiques bio-inspirés

Nous basons cette section principalement sur les deux excellents articles synoptiques de Franz et Mallot [FM00], et de Trullier *et al.* [TWBM97]. Nous invitons le lecteur intéressé à s'y reporter, ainsi qu'aux références bibliographiques s'y trouvant.

Les approches bio-inspirées [2] critiquent très tôt dans leur analyse la conception classique de la navigation robotique, héritée de la navigation maritime. Celle-ci peut être résumée par les questions de Levitt et Lawton [LL90] : « Où suis-je ? », « Où sont les autres endroits par rapport à moi ? », et « Comment aller à d'autres endroits à partir d'ici ? » [3], ou, alternativement, par celles de Leonard et Durrant-Whyte [LD91] : « Où suis-je ? », « Où vais-je ? », et « Comment y aller ? » [4].

Ces questions font référence à une conception de la navigation qui implique un modèle global de l'environnement, qui permet

- de se localiser (« Où suis-je ? ») ;
- d'inférer des relations spatiales entre la position reconnue et d'autres lieux (« Où sont les autres endroits par rapport à moi ? ») ;
- de déduire de ces relations une suite d'actions à exécuter pour se déplacer dans l'environnement (« Comment aller à d'autres endroits à partir d'ici ? »).

Ces compétences constituent les deux premières phases du cycle de calcul « perception – planification – action », très classique en robotique.

Dans le monde du vivant, l'existence d'une représentation globale, unique, permettant de résoudre ces trois sous-problèmes est douteuse. Cela paraît évident pour des animaux comme les abeilles ou les fourmis, mais est également discutable pour l'être humain. Il semble en effet que ses capacités de navigation se basent sur un type de représentation interne de son environnement (*carte cognitive*), mais la nature, le nombre et la complexité de ces représentations sont encore largement débattues [Berthoz97].

Les approches bio-inspirées préfèrent, elles, supposer d'emblée une *hiérarchie* de représentations.

[2] Dans la suite, nous emploierons les termes « bio-inspiré » et « bio-mimétique » de façon interchangeable.

[3] Nos traductions des formulations originelles en anglais « *Where am I ?* », « *Where are the other places with respect to me ?* », et « *How do I get to other places from here ?* »

[4] De nouveau nos traductions de l'anglais « *Where am I ?* », « *Where am I going ?* », et « *How should I get there ?* »

Il s'agit donc de décomposer le problème de la navigation en sous-problèmes, et l'on en vient à la question de la sémantique de la représentation de chaque couche de la hiérarchie. Nous avons abordé plus haut le modèle hiérarchique de [Thrun98b] : celui-ci extrait un graphe déterministe d'une grille d'occupation métrique à grain fin. La représentation métrique est la représentation de bas niveau d'abstraction, sur laquelle repose une représentation topologique. Ce choix est également celui suivi par de nombreuses approches hiérarchiques, probabilistes ou non [HMK+98, KSLO96, LK01b, MAB98, SO98].

Les approches bio-mimétiques remettent en cause cet ordre hiérarchique. Elles remarquent par exemple que les animaux possédant des compétences de raccourci et de détour, qui requièrent une représentation métrique, sont des animaux évolués comme les chiens ou les singes. En particulier, ces compétences semblent être absentes chez tous les non-vertébrés, ce qui n'empêche ni les abeilles ni les fourmis de réaliser des tâches complexes dans des environnements de grande taille. De plus, la représentation topologique de l'espace semble être la première créée par l'enfant dans son développement.

Les hiérarchies proposées par les approches bio-mimétiques placent donc les représentations métriques en haut de la hiérarchie, à un niveau élevé de représentation, et de compétence. Elles sont également associées à des représentations d'environnements de grandes tailles. On retrouve d'ailleurs cette correspondance à tous les niveaux de la hiérarchie : les plus basses couches sont associées à des compétences de navigation locales, et on y représente des environnements de petites tailles (sous *l'horizon perceptif* du robot, c'est-à-dire dans les limites de ce que peut percevoir le robot à un instant donné).

En particulier, Kuipers propose une hiérarchie de représentations, nommée SSH pour *Spatial Semantic Hierarchy* [Kuipers96, Kuipers00], qui a pour ambition de créer un système robotique performant

- en s'inspirant de la cognition humaine ;
- en gagnant en robustesse par l'empilement de connaissances partielles ;
- en choisissant des représentations ancrées dans l'interaction sensori-motrice du robot avec son environnement ;
- en basant son système sur une représentation topologique, et en repoussant l'inclusion d'une représentation métrique à la fin du développement de son système robotique (si nécessaire).

Cette hiérarchie est étonnamment proche de celles synthétisées par les deux articles synoptiques sur les approches bio-mimétiques en robotique [FM00, TWBM97]. Dans la suite de cette section, nous allons décrire les différents niveaux ou « couches » hiérarchiques de la SSH [Kuipers00], en notant les différences et similarités avec ces deux articles synoptiques.

4.2.1 Couche « contrôle »

Cette couche, la plus basse de la SSH, consiste en un ensemble de comportements réactifs. Ces lois de contrôles sont déduites d'équations différentielles, et sont de deux types.

Remontées de gradients Les gradients sont générés par une mesure de « particula-

rité » [5] appliquée aux données sensorielles. Remonter ce gradient fait aboutir le robot à un maximum local, qui définit un état discret de l'environnement « localement particulier » [6]. Ces états correspondent donc à des lieux de l'environnement tels qu'il existe une mesure dont ils sont un maximum. La mesure est applicable dans un voisinage autour du maximum. Par exemple, dans [KUIPERS00], lorsque le robot arrive dans une intersection, une mesure de particularité basée sur la distance aux trois obstacles les plus proches est utilisée pour amener le robot au milieu de l'intersection.

Suivis de trajectoires Ces comportements (suivi de mur, rester au milieu du couloir, etc.), permettent de passer d'un état discret au voisinage d'un autre. Passer d'état discret à état discret de cette manière permet d'éliminer l'accumulation d'erreurs odométriques, sans recourir à une carte métrique ni même à un référentiel global.

David Pierce, dans sa thèse [PK97], explore les possibilités de sélection expérimentale de gradients, et de comportements de suivis de trajectoires, par la détection d'invariants dans le flot sensori-moteur. Il s'agit ainsi de faire le lien entre les capteurs et actionneurs du robot et les compétences du niveau « contrôle » (*ancrer les représentations*).

Dans [FM00, TWBM97], cette couche hiérarchique se nomme la couche de « guidage » [7].

Toutes les compétences décrites dans cette couche ne se basent que sur les données sensorielles immédiates du robot. En ce sens, ces compétences sont locales : on dit alors qu'elle restent sous « l'horizon perceptif » du robot. Dans les couches suivantes, on s'intéresse à la mémorisation de plusieurs lieux que le robot ne peut pas percevoir en même temps, et de leurs relations spatiales. Toutes les couches suivantes sont rassemblées sous le nom de « way-finding » par [FM00] et [TWBM97].

4.2.2 Couche « causale »

Cette couche de la SSH est une abstraction déterministe et discrète du niveau inférieur, où :

- les lois de contrôle sont abstraites en application d'« actions » A ;
- et les états localement particuliers sont représentés par des « vues » [8] V, où l'on mémorise les données sensorielles associées à ce lieu de l'environnement. Cette représentation donne éventuellement naissance à du perceptual aliasing (sans que Kuipers ne traite en détail du sujet).

Cette couche s'exprime en logique du premier ordre. Un *schéma* est un triplet $\langle V, A, V' \rangle$, qui a deux sens :

- un sens déclaratif : « partir de la vue V et appliquer l'action A amène au bout d'un certain temps à la vue V' » ;
- et un sens procédural : « si le robot se trouve en V, il doit appliquer l'action A.

[5] *Distinctiveness.*
[6] *Locally distinctive state.*
[7] *Guidance.*
[8] *Views.*

Une liste de vues qui « s'enchaînent », c'est-à-dire dont le point d'arrivée de l'une est le point de départ de la suivante, permet de définir une « routine ». Les routines permettent de mémoriser de longs chemins, comme par exemple les résultats d'un calcul de planification, ou des routes fréquemment employées.

Cependant, comme la logique du premier ordre n'est pas capable de représenter les incertitudes nées de la navigation du robot, Kuipers *exige* que les comportements de la couche inférieure de « contrôle » *assurent* que le robot passe d'un état discret à un autre, toujours le même. Pour nous, cette « contrainte » n'est pas envisageable, ne serait-ce que si l'on considère les environnements dynamiques : comment assurer qu'un comportement amènera le robot à l'intérieur d'un bureau, si sa porte est soudain fermée ?

Cette couche est équivalente, de par son aspect procédural, à la couche *recognition-triggered response* de [FM00, TWBM97], où la reconnaissance d'un lieu V est associée à une action A. C'est dans cette couche que se placent les systèmes de navigation à base de champs de potentiels. Cependant, elles ne sont pas tout à fait équivalentes à la couche causale de la SSH, car cette dernière représente de plus les vues d'arrivées (dans ses triplets $\langle V, A, V' \rangle$). Cela permet de raisonner (planifier), et de se détacher d'un but unique, ce que les champs de potentiels ne peuvent pas faire (on les classe d'ailleurs dans les approches « orientées vers un but » [9]).

4.2.3 Couche « topologique »

On crée à ce niveau une carte topologique globale, avec des nœuds et des liens, respectivement appelés *endroits* et *chemins* [10]. Ils sont définis par un ensemble de propriétés exprimées par des règles de la logique, universellement quantifiées. Par exemple, pour établir le lien entre vues et endroits, on utilise la règle $at(v, p)$, qui indique qu'une vue v est présente à l'endroit p. Étant donné un ensemble de règles de ce type, un processus *d'abduction* trouve le nombre minimum d'endroits et de chemins, à partir de la base de données de vues et d'actions.

Il s'agit donc de structurer spatialement les vues et actions, par des prédicats indiquant qu'une vue se trouve à un endroit (voir plus haut), mais aussi qu'une vue est présente sur un chemin, à son début ou à sa fin, qu'un chemin est orienté, etc.

On trouve aussi à ce niveau la définition de régions : les endroits sont des points de l'environnement, les chemins sont des sous-espaces de dimension 1, et les régions sont de dimension 2. Ces régions sont également définies par des prédicats, utilisant les chemins comme délimiteurs spatiaux.

Cette couche traite également du problème de global connectivity, grâce à une heuristique de navigation, la « procédure de répétition » [11]. Lorsque le robot explore son environnement, et qu'il arrive à un lieu non-encore connu, le processus d'abduction crée un nouvel endroit p'. Pour tester si cet endroit p' n'est pas le même que p qui se trouve déjà

[9] *Goal-oriented.*
[10] *Places* et *paths.*
[11] *Rehearsal procedure.*

dans la carte, le robot explore les voisins de l'endroit p'. Il compare ce qu'il perçoit avec les voisins prédits par les connaissances sur p : si au bout d'un temps borné d'exploration les prédictions sont toutes en accord avec les observations, alors $p = p'$, et le graphe topologique est modifié en conséquence. Une fois de plus, cette heuristique, de par sa nature déterministe, s'accommode *a priori* mal d'un environnement dynamique : si le voisinage de p a changé d'aspect depuis le dernier passage du robot, il lui sera impossible de reconnaître p' comme un endroit déjà visité. Les conséquences sur la correction de la carte topologique d'une telle erreur ne sont pas mentionnés dans [KUIPERS00].

Remarquons que Kuipers ne précise malheureusement pas comment sélectionner cette hypothèse particulière $p \stackrel{?}{=} p'$, du moins ni dans [KUIPERS96] ni dans [KUIPERS00]. Cependant, le nombre d'hypothèses éventuelles à tester est réduit, car on ne possède qu'un petit nombre d'endroits dans l'environnement. Ceci est un énorme avantage sur les approches métriques ou topologiques denses, où le nombre de lieux représentés peut être très grand. Vu sous un autre angle, on profite ici de la discrétisation de l'environnement de la couche contrôle en vues, qui a géré les erreurs d'odométrie (voir Section 4.2.1). Les deux problèmes d'accumulation de ces erreurs et de la global connectivity sont ainsi découplés dans la SSH.

La carte topologique obtenue est utilisée pour la planification dans l'environnement, par un simple algorithme de recherche dans le graphe, par A* ou l'algorithme de Dijkstra.

Finalement, la carte topologique peut être utilisée pour créer d'autres cartes topologiques de plus haut niveau d'abstraction. Notamment, les régions sont abstraites en endroits de plus haut niveau d'abstraction. Nous obtenons ainsi un empilement hiérarchiques de cartes topologiques. Les cartes de plus faible granularité (le plus de détails), en bas de cette hiérarchie, sont ancrées dans la couche causale. Les cartes de plus grande granularité (le moins de détails), en haut de la hiérarchie, sont plus efficaces pour la planification, car leurs zones sont moins nombreuses pour couvrir le même espace. Cependant, ce gain se paye en *optimalité* (au sens du plus court chemin géométrique) du chemin planifié.

Cette couche est également appelée « topologique » par [FM00] et [TWBM97].

4.2.4 Couche « métrique »

La dernière couche est la couche métrique, dans laquelle le graphe topologique est projeté dans un référentiel unique. Nous avons déjà détaillé plus haut certaines des critiques faites à l'égard de cette représentation dans le cadre des approches bio-mimétiques. En plus de ces critiques, Kuipers note que ce niveau est plus une possibilité technique (difficile à obtenir), qu'un réel pré-requis à la navigation robotique. D'ailleurs, relativement peu de systèmes de navigation bio-mimétique incluent une couche métrique [FM00, TWBM97].

4.3 Questions « fil conducteur »

Pour conclure, nous reprenons nos questions « fil conducteur », et examinons les réponses classiquement fournies par les travaux bio-inspirés mentionnés dans ce chapitre.

Q1. Qu'est-ce qu'une carte ? Le terme de « carte » fait ici principalement référence à la notion de « carte cognitive ». Le but principal des approches bio-inspirées est d'apporter des éléments de définition de ce terme. Nous en avons détaillé un aspect principal dans ce chapitre : il ne s'agit pas d'une représentation unique, mais d'un *ensemble de représentations*, vraisemblablement organisées de manière hiérarchique. Ces hiérarchies sont à la fois des hiérarchies de compétence de navigation, d'échelle de l'environnement représenté, d'échelle de temps associé aux déplacements, et de complexité des représentations. Ce dernier aspect repousse les représentations métriques hors du domaine du requis, vers le domaine du perfectionnement. Il est possible de naviguer dans des environnements complexes sans la moindre notion métrique, mais elle devient nécessaire pour accomplir certaines tâches (raccourcis et détour).

Q2. Qu'est-ce qu'un lieu, pour le robot ? Ainsi, dans une telle hiérarchie, un lieu ne sera plus une seule coordonnées d'un référentiel unique, mais un ensemble de valeurs de variables de différentes sémantiques.

Q2. Que sont les représentations *topologiques* et *métriques* ? Dans ce contexte, les représentations topologiques et métriques sont clairement séparées, et associées à des *compétences envisagées* pour le robot. Il ne s'agit plus seulement de possibilités techniques associées à des capteurs performants, ou des grandes capacités de calcul, ou encore des environnements particuliers, comme souvent dans le cadre de la plupart des approches classiques du Chapitre 3.

Q3. Qu'est-ce que naviguer ? Pour naviguer, un robot utilisant un système hiérarchique devra utiliser en même temps toutes les abstractions à sa disposition. Il pourra par exemple planifier à un niveau élevé d'abstraction tout en exécutant déjà un plan partiel de bas niveau d'abstraction, tout en estimant sa position dans toutes ses représentations (localisation). Le contexte, c'est-à-dire les informations venant de plus hautes couches, pourront lui servir pour désambiguïser sa localisation dans un certaine couche, aidant ainsi à la résolution du problème de perceptual aliasing. Ses compétences de navigation, tirées de plus bas niveau d'abstraction, pourront aussi contribuer à la résolution du problème de global connectivity (procédure de répétition).

Q7. Comment spécifier une carte ? Nous l'avons vu, la première étape dans la définition des systèmes bio-mimétiques, c'est la définition des différentes couches de la hiérarchie. Il apparaît clairement que cette définition est à la charge du concepteur du système. Dans le domaine du vivant, déterminer si l'établissement des niveaux hiérarchiques de représentation de l'espace est de nature phylogénétique ou ontogénétique est une question intéressante.

Chapitre 5

Synthèse de notre étude bibliographique

> « Je crois qu'on s'instruit *contre quelque chose*, peut-être même *contre quelqu'un*, et déjà contre *contre soi-même*. C'est ce qui donne, à mes yeux, tant d'importance à la raison *polémique*. »
>
> Gaston Bachelard, *L'engagement rationaliste*, Presses Universitaires de France, Paris, 1972 [Bachelard72].

Nous sommes maintenant prêts à reprendre la liste de questions établie en introduction de ce document, et à résumer les réponses que nous souhaitons, pour notre part, y apporter.

Une seule question n'a pas été abordée jusqu'ici par les approches que nous avons étudiées : c'est celle de l'utilité d'une carte (elle est rarement traitée en détail, mais est parfois abordée dans certains paragraphes de conclusion, voir par exemple [Thrun02]).

En effet, la plupart des auteurs [1] font, très tôt dans la modélisation, une séparation entre la localisation (dans la carte) et son utilisation (planification, contrôle d'exécution de plan). Ils supposent alors que la phase d'utilisation nécessitera une description très fine de l'environnement. Ainsi, les représentations métriques fines ont été le but de nombreuses recherches, et ont gagné en popularité et en intérêt notamment grâce à l'arrivée de capteurs précis permettant leur réalisation. Cependant, les auteurs traitant à la fois de la carte et de son utilisation (dans le cadre de produits de démonstration au grand public notamment), se rendent vite compte qu'une représentation trop précise est inutilisable en pratique : trop d'espace mémoire requis, trop de temps de calcul associé, le tout difficilement optimisable. Certains auteurs décident donc de *dégrader* la carte métrique (qu'ils ont eu tant de mal à construire précédemment), et en extraient des cartes topologiques par exemple [Thrun98b]. Cet étrange retour en arrière, vers des cartes plus simples, conduit naturellement à replacer la question de l'utilité (l'utilisation) de la carte au centre de notre approche.

Ce constat est d'ailleurs également renforcé par de nombreuses remarques provenant des approches bio-inspirées [FM00, Kuipers00], ou même de certains domaines des sciences cognitives, comme la neurophysiologie [Berthoz97].

[1] D'autres ne considèrent même pas la question comme valide, car l'abstraction métrique, cartésienne du monde leur paraît tellement *naturelle*, qu'il est absurde de la remettre en cause... Nous n'aborderons pas plus avant ce débat philosophique entre instrumentalistes et réalistes ici.

5.1 À quoi sert une carte ?

Développons cette question, en la reformulant. Quelle est la *bonne* représentation de l'interaction entre le robot et son environnement, pour réaliser sa tâche ? Dans la phrase précédente, comment caractériser ce qu'est une « bonne » représentation ? Cette représentation est elle unique (*la* bonne) ou existe-t-il plusieurs bonnes représentations ? Comment caractériser l'ensemble des environnements dans lesquels évolue le robot ? Comment caractériser la tâche à accomplir ? Quelle sont les particularités de la structure physique du robot (capteurs, actionneurs, agencement matériel), et comment influent-elles sur le choix de la représentation ?

Nous souhaitons replacer ces questions comme éléments fondamentaux du travail du programmeur roboticien. Il s'agira donc pour lui d'essayer de définir une *abstraction du monde pertinente* pour faire accomplir à un robot défini un ensemble de tâches définies dans une classe définie d'environnements. Chacune des définitions ci-dessus (du robot, de la tâche à accomplir, et de la classe d'environnements) pouvant se faire :

- en extension : en listant les éléments d'un ensemble fini, ce qui est difficilement applicable dans le cas de la définition des environnements où le robot résout la tâche avec succès, par exemple ;
- en compréhension : en donnant des propriétés à satisfaire pour appartenir à l'ensemble, ce qui est plus envisageable ;
- ou approximativement : en donnant quelques exemples, en expliquant pourquoi cela marche sur ces exemples, et en laissant la généralité émerger : c'est souvent cette dernière option que l'on rencontre dans les articles qui décrivent des systèmes robotiques.

Par « abstraction du monde *pertinente* », nous voulons dire que le choix de la représentation de l'interaction entre le robot et l'environnement est nécessairement un compromis établi au vu de nombreuses contraintes :

- la contrainte de finesse de représentation, pour accomplir des tâches complexes dans des environnements variés ;
- les contraintes de temps de calcul et d'utilisation de l'espace mémoire disponible, pour raisonner en temps réel sur le matériel informatique disponible (à bord du robot, ou déporté) ;
- la contrainte de généralité, pour réutiliser la même carte sur des robots différents ;
- les contraintes dues à l'appareillage sensori-moteur utilisé, pour choisir des variables qui ont du sens pour le robot ;
- les contraintes dues à l'interaction avec un programmeur (phase de conception) ou un utilisateur non-programmeur (phase d'exploitation), pour choisir des variables qui ont du sens pour un interlocuteur humain (explicabilité) ;
- la contrainte de temps de développement du programme ;
- etc.

Dans les exemples que nous présenterons dans la suite de ce document, nous définirons ainsi clairement le problème posé au programmeur roboticien (nous), en définissant

systématiquement :

- le robot utilisé (partie matérielle : ses capteurs, ses actionneurs, la puissance de calcul à bord, etc.) ;
- la classe d'environnements auxquels le robot peut être confronté ;
- la tâche à accomplir [2].

La travail d'un programmeur roboticien est donc de remplir la partie logicielle du robot, par le biais de cartes bayésiennes, pour nos diverses tâches de navigation, en choisissant un représentation du monde pertinente pour le robot et répondant à toutes les contraintes ci-dessus. Nous négligeons la partie logicielle dédiée aux protocoles de communications, ou encore aux couches sensori-motrices de bas niveaux, comme les PID, mais idéalement, leur influence sur la résolution d'un problème robotique ne sont pas à ignorer (certains de ces effets sont d'ailleurs pernicieux, et souvent cruciaux le jour de démonstrations, ce que nombre de roboticiens apprennent à leurs dépends).

Cette définition initiale du problème à résoudre en termes aussi précis que possible est rarement présente dans la littérature. Dans la plupart des articles de notre connaissance, les auteurs ont plutôt tendance à essayer de résoudre une tâche exagérément générale, comme par exemple « naviguer d'un point quelconque A à un point quelconque B de l'environnement, lequel n'est caractérisé par aucune hypothèse ». Or en pratique, un tel but est irréalisable (ne serait ce que pour les cas extrêmes – difficile de cartographier un environnement possédant des escaliers pour un robot à roues, par exemple). On trouve donc souvent, au fil des articles, des constats d'échec honnêtes au meilleur des cas, ou alors des hypothèses s'ajoutent subrepticement sur certaines caractéristiques de l'environnement : l'une des plus classiques est celle d'angle minimum perçu pour considérer comme non-colinéaires deux segments proches détectés par le robot [Van Zwynsvoorde00]. Si l'environnement possède réellement deux murs s'intersectant avec un angle en dessous de ce seuil, ils ne pourront pas être représentés par le robot.

Nous souhaitons, au contraire, expliciter autant que faire se peut les hypothèses simplificatrices que nous utilisons dans le modèle du monde, afin de mieux identifier les raisons de réussite ou d'échec de nos expériences : c'est un des atouts que nous confère l'approche PBR. Elle nous force à énoncer nos choix de modélisation, notamment lorsqu'on émet des hypothèses d'indépendances conditionnelles, lorsqu'on choisit des variables plutôt que d'autres, etc.

Les quatre éléments

$$\langle \text{robot (matériel)}, \text{robot (logiciel)}, \text{environnement}, \text{tâche} \rangle$$

sont en relations fortes. Nous avons déjà noté plus haut l'influence que les environnements, les tâches, et la structure du robot pouvaient avoir sur le choix de la représentation du monde que le robot allait avoir. Cependant, toutes les autres influences imaginables entre ces quatres éléments ne sont pas à négliger. Citons par exemple :

[2] En pratique, les définitions du robot et du type d'environnement que nous considérons seront les mêmes pour toutes nos expériences, et se trouvent Section 2.1.

- Avila-Garciá, qui étudie la relation entre la complexité de l'environnement et la performance de résolution d'une tâche pour des architectures de contrôle ne variant que par leur mécanisme d'arbitrage entre comportements [AHC02] ;
- Kim, qui étudie la relation entre la quantité de mémoire allouée à un automate programmé par évolution et la performance à la résolution d'une tâche de navigation dans un environnement simulé [KH02] ;
- Bongard, qui étudie l'effet de caractéristiques morphologiques d'agents simulés (taille, masse, et configuration des segments du corps) sur la performance de locomotion de ces agents [BP02] ;
- Damasio, qui note pour sa part la relation entre la complexité et la taille des différentes régions du cerveau d'un animal, et la complexité et le « caractère imprévisible » des environnements auxquels il est confronté [Damasio95].

Dans le cadre de nos expériences, puisque nous désirons principalement illustrer les choix possibles de représentations du monde en fonction de la tâche de navigation à accomplir, nous essayons de nous affranchir des influences des deux autres éléments du problème :

- le robot (matériel) : nous avons choisi un appareillage sensori-moteur simple, en restreignant essentiellement nos capteurs aux seuls capteurs de proximité du robot Koala, capteurs qui ne sont pas très performants (au contraire d'un télémètre laser, par exemple) ;
- l'environnement : ici encore, nous nous restreignons à des environnements simples, dont nous maîtriserons la complexité.

Ainsi, nous pensons être dans de bonnes conditions pour apporter un éclairage sain sur la relation entre le choix d'une représentation du monde, et le succès ou l'échec de la réalisation d'une tâche. Un exemple flagrant d'une telle dépendance sera présenté dans le premier exemple de carte bayésienne, Section 6.4. Dans cet exemple, nous définirons une carte bayésienne, et nommerons une première tâche que le robot *peut* effectuer grâce à cette carte, et une seconde tâche (à peine plus complexe) que le robot *ne peut pas* effectuer avec cette même carte.

5.2 Les autres questions « fil conducteur »

Nous sommes maintenant parés pour donner quelques amorces de réponses aux questions restantes : certaines affirmations faites dans ce chapitre seront illustrées plus avant dans les expériences qui suivent, nous indiquerons dans ces cas les expériences correspondantes.

Nous souhaitons organiser nos réponses en fonction de trois remarques qui émergent grâce à la pratique du formalisme de Programmation Bayésienne des Robots.

5.2.1 Une forme paramétrique peut être une question probabiliste

Nous avons vu Section 2.2.3.2 qu'un terme d'une description c^1 pouvait être défini comme étant une question probabiliste posée à une autre description c^2. Cela permet de

décomposer un problème robotique en sous-problèmes. Ainsi, programmer une tâche robotique, c'est tout d'abord imaginer une architecture de modules, puis les programmer individuellement, et enfin, les agencer. Pour la description complète d'une telle architecture de contrôle possédant 42 variables et 4 niveaux hiérarchiques, dans le cadre de la programmation d'une tâche de surveillance, voir [Lebeltel99].

Il en va de même pour la navigation robotique à base de cartes. La première étape de programmation est une analyse de la tâche de navigation à résoudre, pour imaginer des représentations intermédiaires, éventuellement hiérarchisées. Certains de ces modèles intermédiaires peuvent être des sous-cartes, éventuellement au sens spatial (cartes de sous-parties de l'environnement) ou au sens des tâches (cartes permettant de résoudre des sous-tâches). Cette première étape de modélisation est très rare dans les approches probabilistes, et répandue dans les approches bio-inspirées.

Cependant, les approches bio-inspirées que nous avons étudié expriment leurs sous-modules dans des formalismes différents. Il devient alors difficile des mettre en relation ces sous-modules, et de justifier la consistance de leurs méthodes de communications. Cependant, cette diversité des formalismes utilisés est considérée comme un avantage par Kuipers, qui affirme [Kuipers00, p. 4] :

> La multiplicité des représentations dans la SSH confère à l'agent plus de pouvoir d'expression pour traiter les connaissances incomplètes, et plus de robustesse pour gérer les incertitudes sensori-motrices et les contraintes de calcul, qu'une quelconque représentation unique. [3]

Nous prenons comme point de départ de notre approche que le meilleur formalisme pour traiter l'incertitude et l'incomplétude en robotique est le formalisme probabiliste (voir [BDL+98a, BDL+98b]). Cette approche bénéficie notamment de fondements mathématiques clairs et rigoureux. Les distributions de probabilités y sont l'outil unique d'énonciation et de manipulation de connaissances. Ainsi, dans le formalisme PBR, toutes les communications entre modules s'effectuent par le biais de questions probabilistes, qui sont des distributions de probabilités, et bénéficient ainsi d'une sémantique claire. À la lumière de ces considérations, nous reprenons à notre compte les termes de Kuipers, en les modifiant un peu :

> L'*unicité* des représentations dans la *PBR* confère à l'agent plus de pouvoir d'expression pour traiter les connaissances incomplètes, et plus de robustesse pour gérer les incertitudes sensori-motrices et les contraintes de calcul, *que n'importe quelle autre* représentation.

Q7. Comment spécifier une carte ? Ainsi, programmer une carte, c'est tout d'abord spécifier la tâche de navigation à résoudre, puis (si elle est trop compliquée) la décomposer.

[3] Notre traduction de :

> *The multiplicity of representation in the SSH gives the agent more expressive power for incomplete knowledge, and more robustness in coping with sensorimotor uncertainties and computational limitations, compared with any individual representation.*

Il s'agit d'imaginer ou de recopier du vivant des niveaux de détails (ou de compétence) intermédiaires, pertinents. Chacun de ces niveaux peut ensuite donner lieu à la spécification et la programmation de sous-cartes. L'étape finale consiste à intégrer ces sous-cartes dans une architecture globale.

5.2.2 La modélisation commence par le choix des variables

Étant donné une décomposition du problème global de navigation en sous-problèmes, il faut ensuite programmer chacun des sous-modules.

La première étape dans la modélisation d'un tel sous-module, dans le formalisme PBR, c'est le choix des variables (voir Section 2.2.3.2). Dans le cas de cartes, il faut choisir des variables de perception et d'action, mais aussi une variable de lieu. Il faut donc trouver un ensemble de lieux *pertinents* pour la résolution de la tâche (voir Section 5.1 pour notre définition du terme « pertinent » dans ce contexte).

La sémantique de ces lieux, qu'elle soit topologique ou métrique, est une considération secondaire. Cela ne veut pas dire que le choix d'une représentation topologique ou métrique n'a pas d'importance, nous souhaitons au contraire replacer ce choix comme primordial, *au vu de la tâche à résoudre* (et de l'environnement, et de la structure du robot, etc.).

Représenter l'environnement par une variable de lieux « pertinente » implique également que ce modèle n'ait pas de perceptual aliasing tel que la tâche ne puisse être résolue. En effet, selon nous, le perceptual aliasing n'est pas une caractéristique intrinsèque de l'environnement : c'est une propriété de la *représentation* de l'environnement. La variable de lieux choisie prend-elle parfois la même valeur à des endroits géométriquement distincts ? Si oui, cette représentation souffre par définition de perceptual aliasing. Mais, ces endroits géométriquement distincts, représentés par la même valeur interne, doivent-ils *absolument*, pour résoudre la tâche, être associés à des ordres moteurs différents ? Si oui, la résolution de la tâche avec cette variable de lieux est impossible. Le perceptual aliasing, dans ce cas, est une réelle limitation. Dans le cas contraire, les ambiguïtés nées du perceptual aliasing n'influent pas la résolution de la tâche, et cette représentation, bien qu'ambiguë, est adéquate pour la résolution de la tâche. Dans ce cas, le perceptual aliasing n'est pas un problème.

La situation est symétrique en ce qui concerne le problème dual du perceptual aliasing, c'est-à-dire la global connectivity : l'analyse de l'importance de cet effet lors du choix d'une représentation n'a selon nous de sens que vis-à-vis d'une tâche à résoudre.

De plus, dans le formalisme PBR, le choix des variables précède le choix d'une structure de dépendance (voir Section 2.2.3.2). Ainsi le choix d'une décomposition probabiliste particulière est elle aussi secondaire. Or, les principales approches probabilistes (Localisation Markovienne, HMM, etc.) se caractérisent principalement par le choix d'une décomposition, laissant libre le choix de la sémantique de la variable d'état. Nous préférons dans ce travail repousser ce choix au second plan. Dans les exemples que nous présenterons, nous ne nous restreindrons ainsi pas à une structure de dépendance prédéfinie.

Q2. Qu'est-ce qu'un lieu, pour le robot ? Dans ce cadre, un lieu pourra avoir des sémantiques variées selon les sous-cartes. Dans nos exemples, nous définirons ainsi des

cartes topologiques (voir par exemple, Section 8.2.1.3, ou Section E.2), des cartes métriques de faible granularité (Section E.1), certaines de ces cartes définiront des zones connexes dans l'environnement (Section C), mais cette propriété sera parfois violée (Section 6.3).

5.2.3 La phase de description est indépendante de la phase d'utilisation

Continuons de parcourir la méthode PBR, pour arriver à la fin de la phase de description. Une fois cette phase achevée, le modèle est complet, dans le sens où toutes les connaissances sont fournies, et la distribution conjointe est entièrement définie : la partie *déclarative* du programme est terminée. La phase d'utilisation, qui vient ensuite, consiste à restituer ces connaissances par le biais de questions probabilistes : cette phase constitue la partie *procédurale* du programme probabiliste (voir Section 2.2.3.4).

Autrement dit, si les programmes en question sont des cartes, ce que l'on va en faire est indépendant de la façon dont elles ont été définies. Le programmeur a donc toute latitude pour découpler les choix faits dans les deux phases, déclaratives et procédurales. Une fois encore, cette indépendance cache en réalité une cause commune : la tâche à résoudre. Le programmeur pourra ainsi choisir une décomposition particulière, et une question particulière, au vu du problème de navigation qui lui est posé.

Dans les approches probabilistes du Chapitre 3, cette « indépendance » n'est pas satisfaite. En effet, la première étape classique dans la modélisation de séquences temporelles, c'est de classer les variables en variables observées (perception et action), et variables cachées (état). Voir par exemple à ce titre la thèse de Murphy [Murphy02], mais les exemples dans la littérature sont innombrables. Dans cette optique, les termes qui apparaissent dans les structures des différents formalismes sont déjà choisis en fonction de l'inférence future. Par exemple, les modèles d'observations de la forme $P(P_t \mid L_t)$ ont la propriété d'être facilement intégrés au calcul de la localisation $P(L_t \mid P_t)$. Cela est en contradiction avec l'indépendance des phases déclaratives et procédurales notée plus haut : les approches probabilistes classiques sont déjà des choix de modélisation *orientés vers* l'utilisation des modèles.

Q4.5.6. Qu'est-ce que se localiser, prédire, générer une loi de contrôle ? Dans notre cadre de programmation de tâches de navigation, ce sont des *compétences*, des choses que le robot *fait*, donc participant de la phase procédurale. Ces trois termes, « localiser », « prédire », et « contrôler », apparaîtront donc dans notre formalisme du côté « inférence », et non du côté « description ». Il en sera de même pour la notion « d'appliquer un comportement ».

Q3. Qu'est-ce que naviguer ? « Naviguer » sera pour nous relatif à l'application de ces compétences dans le monde. Naviguer sera « résoudre le problème d'aller à un endroit de l'environnement ». Cela sera donc le résultat d'une phase de calcul, sur un ensemble de connaissances, et fera donc également partie de la phase procédurale.

Q1. Qu'est-ce qu'une carte ? Une *carte* sera, au contraire, l'ensemble des connaissances qui permettent de naviguer. Il s'agira donc de la partie déclarative d'un programme, c'est-à-dire une description. Les seules contraintes que nous définirons porteront sur le choix des variables : nous supposerons disposer de variables de perception, d'action, et de lieux. En revanche, nous ne contraindrons pas la sémantique de cette dernière.

Ne définissant pas de contraintes sur le choix de la décomposition, notre formalisme sera plus général que les approches tels que la localisation markovienne et les filtres de Kalman (lorsqu'ils incluent une variable d'action) ; cela justifie la position de notre contribution, la carte bayésienne, sur la Figure 3.1.

5.3 Résumé

En conclusion, nous souhaitons résumer brièvement les forces et faiblesses des approches probabilistes et bio-inspirées que nous avons étudiées dans notre étude bibliographique, et replacer dans ce cadre notre proposition. Les approches probabilistes :

- utilisent un formalisme clair et rigoureux pour le traitement des incertitudes ;
- ont été largement étudiées dans le cadre de la navigation robotique à base de cartes, et ont donné naissance à de nombreux systèmes robotiques ;
- imposent en général un type de décomposition, une structure de dépendance, et ce quelle que soit la sémantique de la variable d'état considérée ;
- sont classiquement associées à des modèles « plats » (non-hiérarchiques) ;
- sont classiquement associées à des modèles métriques de haute résolution, espérant atteindre une localisation très précise dans l'espace cartésien.

Pour leur part, les approches bio-mimétiques :

- intègrent naturellement la notion de hiérarchie de représentations et de compétences ;
- proposent une réflexion sur la notion de « carte » (définitions parfois hétérogènes regroupées sous le terme de « carte cognitive »), et sur la sémantique des représentations utilisées par le robot (topologique ou métrique par exemple) ;
- souffrent d'un manque de réalisations techniques de grande ampleur ;
- souffrent d'une hétérogénéité des formalismes utilisés, impliquant des problèmes de consistance et de communication entre modules.

Notre formalisme, la carte bayésienne, se situe à la croisée de ces deux domaines :

- nous ne contraindrons pas notre approche par un choix de décomposition, préférant contraindre un choix de variables – mais pas de leur sémantique – en vue d'une modélisation de l'interaction entre le robot et son environnement pour la résolution d'une tâche de navigation ;
- nous ne contraindrons pas notre approche à des cartes monolithiques, et définirons au contraire des opérateurs à la sémantique claire permettant d'assembler des sous-cartes, donnant naissance à des hiérarchies de cartes bayésiennes ;
- en utilisant le formalisme probabiliste, nous bénéficierons de fondements mathématiques rigoureux, notamment en ce qui concerne la communication entre les sous-cartes d'une architecture globale de navigation.

Deuxième partie

La carte bayésienne

Chapitre 6

Carte bayésienne : définition et exemple

> « . . . En cet Empire, l'Art de la Cartographie fut poussé à une telle Perfection que la Carte d'une seule Province occupait toute une Ville et la Carte de l'Empire toute une Province. Avec le temps, ces Cartes Démesurées cessèrent de donner satisfaction et les Collèges des Cartographes levèrent une Carte de l'Empire qui avait le Format de l'Empire et qui coïncidait avec lui, point par point. Moins passionnées pour l'Étude de la Cartographie, les Générations Suivantes réfléchirent que cette Carte Dilatée était inutile et, non sans impiété, ils l'abandonnèrent à l'Inclémence du Soleil et des Hivers. Dans les déserts de l'Ouest, subsistent des Ruines très abîmées de la Carte. »
>
> Suárez Miranda, *Viajes de Varones Prudentes, livre IV, chap. XIV*, Lérida 1658 ; *cité par* Jorge Luis Borges, *Histoire universelle de l'infamie, « Et cætera »*, Paris, Bourgeois, 1985 ; *cité par* Umberto Eco, *Comment voyager avec un saumon. Nouveaux pastiches et postiches*, Paris, Grasset, 1997, p. 221–229 [Eco97].

6.1 Définition de la carte bayésienne

6.1.1 Définition

Une carte bayésienne c est une description (au sens de la PBR) portant sur la distribution conjointe $P(P\ L_t\ L_{t'}\ A)$, où

- P une variable de perception (le robot lit les valeurs de cette variable à partir de ses capteurs physiques),
- L_t une variable de *lieux* au temps t,
- $L_{t'}$ une variable de même domaine que L_t, au temps t' (sans perte de généralité, posons $t' > t$).
- et A est une variable d'action (le robot donne des consignes sur cette variable).

Pour qu'une carte soit utilisable en pratique, il faut de plus que la description probabiliste permette la création de *comportements*. Nous appelons *comportement élémentaire*

toute question de la forme $P(A_i \mid X)$, où A_i est une partie de A [1], et X une partie des variables de la carte (sauf A_i). Un comportement peut ne pas être « élémentaire », par exemple si sa réalisation implique l'alternance temporelle de plusieurs comportements, ou la combinaison de comportements avec d'autres connaissances, etc.

Un « guide » pour s'assurer qu'une carte puisse créer des comportements, c'est qu'elle réponde de manière *pertinente* aux trois questions

- de localisation $P(L_t \mid P)$;
- de prédiction $P(L_{t'} \mid A\ L_t)$;
- et de contrôle $P(A \mid L_t\ L_{t'})$.

Par « pertinente », nous entendons que les distributions de probabilités associées à ces trois questions soit informatives, c'est-à-dire que leur entropie soit assez loin de son maximum (atteint rappelons le pour une distribution uniforme, [Shannon48]). Bien que cette contrainte soit assez peu formelle (notez le « assez loin » dans la phrase précédente), il est tout de même naturel de particulariser de la sorte ces trois questions probabilistes. En effet, les trois capacités de prédiction, de localisation et de contrôle semblent être les capacités nécessaires, bien identifiées dans la littérature, pour s'assurer de pouvoir naviguer. Pour chaque exemple de carte bayésienne développée dans la suite, nous vérifierons que les réponses à ces trois questions sont bien pertinentes : ceci nous « garantira » la possibilité de tirer de la carte des comportements.

6.1.2 Les trois termes de localisation, prédiction, et contrôle

$P(L_t \mid P)$ est une des parties constituantes de la localisation. Ce terme peut même être suffisant pour une localisation sans ambiguïté, si la représentation de l'environnement choisie ne présente pas de *perceptual aliasing* [2]. Cette localisation « immédiate » prend alors le nom d'*inférence directe de position* [Filliat01].

En revanche, dans les environnements souffrant de *perceptual aliasing*, une seule donnée sensorielle n'est pas suffisante pour localiser le robot, et il devient nécessaire de prendre en compte l'historique de l'évolution du robot dans l'environnement, par la question $P(L_{t'} \mid A\ L_t)$. Cette solution est couramment employée dans la littérature, notamment dans les formalismes de localisation markovienne et les PDMPOs, voir Sections 3.8 et 3.9.2, où l'enchaînement des deux questions $P(L_t \mid P)$ et $P(L_{t'} \mid A\ L_t)$ pour la localisation sont décrits. $P(L_{t'} \mid A\ L_t)$ est parfois appelé *matrice de transition*, ou encore *matrice stochastique*. Dans ce cas, ce terme est instancié « dans le passé » : L_t correspond au lieu de l'instant précédent, A est l'action qui vient d'être appliquée (cela devient donc une lecture passive, A peut ici être considérée comme un capteur), et $L_{t'}$ décrit les connaissances sur le lieu que l'on vient d'atteindre.

Mais ce terme, s'il est instancié « dans le futur », peut aussi servir à prédire le lieu d'arrivée $L_{t'}$, sachant l'action A qui va être appliquée et le lieu de départ actuel L_t. L'intérêt pour un tel *anticipateur* ou *simulateur d'action* est grandissant en robotique, notamment

[1] Nous verrons en effet au Chapitre 7 que la variable A peut être une conjonction de plusieurs variables d'actions.

[2] C'est-à-dire où il n'existe pas deux lieux différents qui génèrent les mêmes perceptions.

dans le domaine de la robotique bio-inspirée. En effet, la plausibilité de telles capacités dans les êtres vivants a été soulignée par de nombreux auteurs en sciences cognitives [Berthoz97, Changeux02]. Dans notre travail, nous considérerons de plus le terme de prédiction comme primordial pour réaliser la *planification*, sans aborder plus avant ce sujet dans ce document.

La dernière question, $P(A \mid L_t \; L_{t'})$, permet, connaissant la localisation actuelle du robot et celle que le robot doit atteindre, de choisir l'action à appliquer, générant ainsi une loi de contrôle à suivre.

6.1.3 Utilisation d'une carte bayésienne

Il existe plusieurs manières d'utiliser les réponses à ces trois questions dans le cadre d'un programme de contrôle robotique.

La première manière simple consiste à enchaîner, dans le temps, les trois questions, dans un algorithme déterministe, comme par exemple l'Algorithme 6.1. Dans cet algorithme, le robot se localise tout d'abord, ensuite la question de prédiction est utilisée pour laisser à l'utilisateur le choix du lieu d'arrivée $L_{t'}$, puis, en fonction du lieu sélectionné comme but et du lieu reconnu par la localisation, le robot choisit l'action appropriée. Ce programme demande donc à l'utilisateur régulièrement un lieu *consigne* ou *objectif*, et s'efforce ensuite de l'atteindre.

Alg. 6.1 Algorithme utilisant les trois questions de localisation, prédiction et contrôle.

```
/* lecture des capteurs */
p <- lire_capteurs();
/* localisation */
lt <- draw( P(Lt | [P = p]) );
/* pour chaque action, on prédit le lieu d'arrivée */
pour tout a dans A
      proposer_objectif( draw( P(Lt+Dt | [A = a] [Lt = lt]) ) );
fin pour;
/* on demande à l'utilisateur un but */
o <- lire_but();
/* contrôle */
a <- draw( P(A | [Lt = lt] [Lt+Dt = o]) );
```

Mais il est également possible de se passer de localisation explicite, et d'utiliser à la place un contrôle sur la prédiction du pas de temps précédent. En effet, le calcul de la localisation $P(L_t \mid P)$ à partir de la carte peut être coûteux, notamment s'il faut inverser le terme $P(P \mid L_t)$. Dans ce cas, prédisons le lieu actuel par calcul au pas de temps précédent $t - \Delta t$:

$$l_t \leftarrow draw(P(L_t \mid A \; L_{t-\Delta t})).$$

Au temps t, nous lisons les capteurs (valeur p), et vérifions la probabilité du terme $P([P = p] \mid [L_t = l_t])$: quelle est la probabilité de percevoir p sur les capteurs, en supposant que nous soyons effectivement à l'endroit l_t prévu au pas de temps précédent ? Si cette valeur numérique est forte (plus haute qu'un seuil k, qui est dépendant de l'expérience en question), c'est que tout se passe en accord avec les prédictions. Nous pouvons donc, pour générer le contrôle, supposer que le robot se trouve effectivement en l_t, et nous nous passons ainsi du calcul explicite de la localisation $P(L_t \mid P)$. Nous résumons cette deuxième possibilité de programme de contrôle robotique par l'Algorithme 6.2.

Alg. 6.2 Algorithme privilégiant la prédiction et le contrôle.

```
/* supposons disposer d'une prédiction lt de lieu,
   générée par le pas de temps précédent */
/* lecture des capteurs */
p <- lire_capteurs();
/* calcul de la vraisemblance de l'observation */
v <- P([P = p] | [Lt = lt]);
/* comparaison avec le seuil k */
si (v < k) alors
      /* relocalisation */
      lt <- draw( P(Lt | [P = p]) );
fin si;
/* reste du contrôle (choix de l'action a), avec lt comme lieu courant */
[...]
/* prédiction de lt' pour le pas de temps suivant */
lt' <- draw( P(Lt+Dt | [A = a] [Lt = lt]) );
```

Une autre possibilité d'utilisation d'une carte consiste à résoudre une tâche, en posant une question probabiliste à la description qui définit la carte. Rappelons que lorsque cette question est de la forme $P(A_i \mid X)$, nous parlons de comportement élémentaire.

Nous verrons dans l'exemple qui suit Section 6.3 comment créer les comportements élémentaires $SeCacher$, $SeMontrer$, et $Longer$ par des questions probabilistes, et comment définir un comportement plus complexe $PatrouillerBord$, en *alternant* dans le temps des comportements élémentaires. Ces comportements sont nommés symboliquement, et peuvent servir de ressource à de plus hauts niveaux d'abstraction. Nous verrons ainsi, dans l'exemple d'abstraction de cartes Chapitre 9, apparaître des actions qui auront été extraites de cartes de plus bas niveau d'abstraction (définies Annexe E).

La plupart du temps, pour générer un comportement élémentaire, nous fixons une consigne l' à atteindre, et posons donc la question probabiliste $P(A \mid [L_{t'} = l'] \, [P = p])$. Cette question peut se résoudre de deux manières.

La première manière est apparentée au premier algorithme que nous avons exposé plus haut (Algorithme 6.1). Nous enchaînons les questions de localisation et de contrôle, en les

faisant apparaître explicitement dans la dérivation suivante :

$$\begin{aligned}
P(A \mid [L_{t'} = l'] \; [P = p]) &= \frac{P(A \; [L_{t'} = l'] \; [P = p])}{P([L_{t'} = l'] \; [P = p])} \\
&= \frac{1}{Z_1} P(A \; [L_{t'} = l'] \; [P = p]) \\
&= \frac{1}{Z_1} \sum_{L_t} P(A \; L_t \; [L_{t'} = l'] \; [P = p]) \\
&= \frac{1}{Z_1} \sum_{L_t} \left(\begin{array}{l} P([L_{t'} = l'] \; [P = p]) P(L_t \mid [P = p]) \\ P(A \mid [L_{t'} = l'] \; L_t) \end{array} \right) \\
P(A \mid [L_{t'} = l'] \; [P = p]) &= \frac{1}{Z_2} \sum_{L_t} P(L_t \mid [P = p]) P(A \mid [L_{t'} = l'] \; L_t).
\end{aligned}$$

Dans ce calcul, les deux termes $P(L_t \mid [P = p])$ et $P(A \mid [L_{t'} = l'] \; L_t)$ sont des questions probabilistes qui sont posées à la carte. Le choix de faire apparaître ces termes est intéressant, d'une part, car ces termes ont une sémantique évidente – ce sont les termes de localisation et de contrôle – et d'autre part, car le terme de localisation permet par exemple un affichage graphique de la connaissance du robot sur sa position. Cette inférence peut être vue comme une version probabiliste de l'Algorithme 6.1, où le tirage sur $P(L_t \mid [P = p])$ est remplacé par une sommation sur la variable L_t.

La seconde manière pour répondre à la question globale $P(A \mid [L_{t'} = l'] \; [P = p])$ ne fait pas explicitement intervenir les termes de localisation et de contrôle. Il s'agit tout simplement de poser cette question probabiliste à la carte bayésienne. Or rien ne force cette carte à faire apparaître dans sa décomposition ces termes particuliers. Ainsi l'inférence ne fera apparaître que les termes qui apparaissent dans la décomposition, et il sera peut-être alors difficile de retrouver dans ce calcul quelque chose d'apparenté aux questions de localisation, prédiction, ou contrôle. Par exemple, si la décomposition est $P(P \; L_t \; L_{t'} \; A) = P(L_t)P(P \mid L_t)P(A \mid L_t)P(L_{t'} \mid L_t)$, poser la question $P(A \mid L_{t'} \; P)$ se résout selon :

$$P(A \mid L_{t'} \; P) = \frac{1}{Z} \sum_{L_t} P(L_t)P(P \mid L_t)P(A \mid L_t)P(L_{t'} \mid L_t).$$

Cette méthode est la plus naturelle dans le cadre de la PBR – on résout une question probabiliste en se ramenant systématiquement à la décomposition choisie – mais nous perdons en explicabilité du calcul, en perdant par exemple la possibilité de visualiser la connaissance du robot sur sa localisation.

Enfin, nous verrons Section 7.5 que les trois termes de localisation, prédiction, et contrôle, ne sont pas les seuls à être potentiellement intéressants.

6.2 Graphe induit par une carte bayésienne

Traditionnellement, les cartes sont représentées par des graphes. La théorie des graphes est d'ailleurs tout naturellement liée aux problèmes de planification de chemins, les deux

problèmes canoniques dans ce domaine étant la recherche du plus court chemin et celui du voyageur de commerce. Si, dans notre formalisme, un graphe n'apparaît pas directement dans la définition, nous pouvons cependant considérer qu'une carte bayésienne est un modèle probabiliste qui représente les connaissances *subjectives* que le robot a de la carte *objective*. Une telle carte peut être formellement représentée par le graphe aux arcs étiquetés $\langle X, R, F\rangle$, où

- X est l'ensemble de sommets,
- R est une relation sur X, c'est-à-dire une partie $\mathcal{P}(X \times X)$ (les arcs),
- F est une application $F : R \to S$, qui pour chaque arc lui associe une étiquette qui est un symbole de l'ensemble S.

Se référer à [Stockmeyer00] pour plus de détails sur les graphes étiquetés en général.

Il est possible de tirer de n'importe quelle carte bayésienne le graphe objectif correspondant, simplement en posant :

- $X = L_t$: les valeurs de la variable de lieux L_t sont les nœuds du graphe ;
- $S = A \times [0, 1]$: l'arc allant de l'état l_1 à l'état l_2 est étiqueté par une action a et une valeur de probabilité p telles que $P([L_{t'} = l_2] \mid [A = a]\ [L_t = l_1]) = p$ (il faut donc poser la question de prédiction à la carte).

Par exemple, supposons que nous ayons une variable de lieu L_t ayant trois valeurs l_1, l_2, l_3 possibles, alors le graphe a trois nœuds, portant les mêmes noms l_1, l_2, l_3. Supposons de plus que $P(L_{t'} \mid [A = a_1]\ [L_t = l_1])$ soit uniforme, alors le graphe possède trois arcs, issus de l_1, qui vont vers l_1, l_2 et l_3, et qui portent chacun l'étiquette $\langle a_1, 1/3\rangle$.

Cependant, si l'on se restreint à cette définition du graphe induit, la plupart des graphes que nous tirerons de cartes seront complets : les seuls arcs que nous pourrons légitimement ne pas faire apparaître traduiront des probabilités de transition égales à 0. En pratique, comme nous souhaitons que les graphes fassent ressortir les informations les plus saillantes des distributions de probabilités, nous n'y feront apparaître que les arcs qui correspondent à des valeurs de probabilités supérieures à un seuil ϵ (pour un ϵ donné, dépendant de la carte considérée). Cette « limitation » de la traduction d'une carte bayésienne en un graphe est la conséquence de la différence de richesse d'expression entre les formalismes probabilistes et graphiques. De plus, nous ne ferons pas apparaître les valeurs de probabilités associées aux arcs dans nos figures, pour plus de lisibilité.

Une telle traduction automatique du modèle probabiliste vers un graphe aura de nombreux avantages :

- il sera souvent plus aisé de représenter graphiquement le graphe que le modèle probabiliste complet ;
- il sera possible de tirer parti de la théorie des graphes, en calculant par exemple le diamètre du graphe, en vérifiant si le graphe n'est pas fait de sous-composantes connexes indépendantes, etc.

À l'inverse, il est également possible de traduire un graphe en carte bayésienne correspondante, ce qui pourra être une aide à la programmation intéressante : au vu du graphe, le programmeur pourra identifier la variable de lieu pertinente, pourra tirer certaines valeurs de probabilités pour le terme de transition (les arcs manquant seront traduits en 0), etc.

6.3 Expérience : carte d'une pièce basée sur les capteurs de proximétrie

Dans cette expérience, nous plaçons le robot dans l'environnement que nous avons décrit Section 2.1. Rappelons qu'il s'agit d'une sorte d'enclos fermé, dont les murs sont des planches amovibles, ce qui nous permet de définir diverses configurations expérimentales. Ici, nous créons une arène simple, avec des angles aigus proches d'angles droits. Un exemple d'un environnement de ce type est présenté Figure 6.1(a) (les contours de la pièce).

Dans cet environnement, nous souhaitons que le robot puisse naviguer jusqu'à en atteindre un des coins, quel que soit son point de départ. Nous appellerons cette tâche *SeCacher*.

6.3.1 Objectifs et protocole expérimental

6.3.1.1 Choix d'une variable de lieux

Afin que le robot réponde à cette tâche dans cet environnement, nous devons lui fournir une abstraction de cet environnement, qu'il puisse construire à partir de ses capteurs et lier à ses actionneurs. Nous choisissons trois lieux à reconnaître : « libre », lorsque le robot est loin de tout mur, « mur », lorsque le robot est à proximité d'un mur, et « coin », lorsque le robot est à proximité de deux murs s'intersectant avec un angle aigu (les angles obtus sont assimilés à des murs).

La Figure 6.1(a) présente une interprétation géométrique des lieux « mur », « coin » et « libre ». Remarquons cependant que tous les coins de cette pièce sont bien indiscernables pour le robot : à chacun de ces lieux, géographiquement distincts, ne correspond qu'une valeur de la variable interne de lieux. Autrement dit, tous les coins seront représentés par la valeur *coin*, tous les murs par la valeur *mur*, et tout l'espace vide par la valeur *libre*.

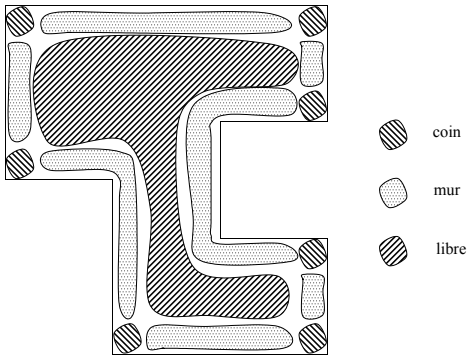
6.3.1.2 Graphe induit

On trouve Figure 6.1(b) un extrait du graphe pour la carte d'une pièce basée sur la proximétrie : tous les nœuds sont présents, mais pas tous les arcs, afin d'alléger la figure.

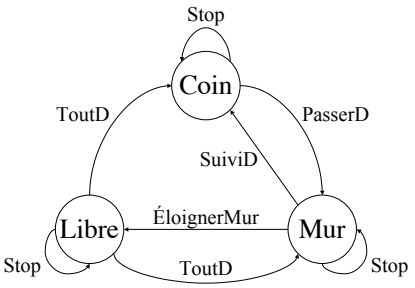
6.3.2 Description

Nous présentons maintenant la carte bayésienne c^{prox} :

1. Spécification des connaissances préalables c^{prox}
   (a) Variables : La variable interne L_t a trois cas, qui correspondent à chaque situation possible (*mur*, *libre* ou *coin*). Les capteurs utilisés sont les seize proximètres, et les actions possibles sont des comportements permettant de passer d'une situation à l'autre, ou éventuellement de rester dans une situation. Nous



(a) Interprétation des zones affectées aux valeurs « mur », « coin » et « libre ».



(b) Un extrait du graphe induit.

FIG. 6.1: Exemple de carte bayésienne d'une pièce, basée sur les informations de proximétrie.

supposons pour le moment que ces comportements sont réalisés par des programmes, que nous ne définirons pas ici [3].

P $Px_0,\ Px_1,\ \ldots,\ Px_{15}$. Chaque variable Px_i est définie par son domaine $\mathcal{D}_{Px_i} = \{0, 1, \ldots, 15\}$, et son cardinal $k_{Px_i} = 16$. Pour alléger les écritures, nous notons Px leur conjonction : $P = Px = Px_0 \wedge \ldots \wedge Px_{15}, \mathcal{D}_{Px} = \mathcal{D}_{Px_0} \times \ldots \times \mathcal{D}_{Px_{15}}, k_{Px} = 16^{16}$.

L_t $Sit_t\ : \mathcal{D}_{Sit_t} = \{mur,\ libre,\ coin\}, k_{Sit_t} = 3$,

L_t' $Sit_{t+\Delta t}\ : \mathcal{D}_{Sit_t+\Delta t} = \{mur,\ libre,\ coin\}, k_{Sit_{t+\Delta t}} = 3$,

A $A_{prox}\ :\ \mathcal{D}_{A_{prox}} = \{Stop,\ ToutD,\ SuiviD,\ \acute{E}loignerMur,\ PasserD\}$, $k_{A_{prox}} = 5$, l'ensemble des comportements pour les cas « mur », « coin » et « libre ». Dans l'ordre, nous avons :

- $Stop$, où le robot ne bouge pas ;
- $ToutD$, où le robot avance en ligne droite ;
- $SuiviD$, où le robot suit un mur en le gardant sur son côté droit ;
- $\acute{E}loignerMur$, où le robot s'éloigne du mur le plus proche jusqu'à ne plus le voir ;
- et $PasserD$, où le robot quitte un coin pour en suivre un des murs en le gardant sur son côté droit.

(b) Décomposition : Nous choisissons la décomposition suivante :

$$\begin{aligned} & P(Px\ Sit_t\ Sit_{t+\Delta t}\ A_{prox}) \\ & \quad = \ P(Px)P(Sit_t \mid Px)P(Sit_{t+\Delta t})P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t). \end{aligned}$$

[3] Nous verrons Chapitre 9 que ces comportements sont issus de cartes bayésiennes de plus bas niveau d'abstraction.

(c) Formes paramétriques :

$P(Px)$ À ce terme est associée une loi uniforme, car nous ne n'avons pas d'a priori sur la répartition des données sensorielles.

$P(Sit_t \mid Px)$ Dans ce premier exemple, nous définissons ce terme par une loi Dirac, qui vaut 1 pour la valeur de Sit_t donnée par une fonction déterministe f, qui prend en entrée les valeurs des seize capteurs de proximétrie. Cette fonction f est définie par l'Algorithme 6.3, et commentée dans la partie « identification » de cet exemple. Nous verrons Chapitre 9 comment nous passer de cette fonction dans ce terme – en realité son inverse, $P(Px \mid Sit_t)$ – par la méthode d'abstraction de cartes. Nous notons :

$$P(Sit_t \mid Px) = \boldsymbol{\delta}_{f(Px)}(Sit_t).$$

$P(Sit_{t+\Delta t})$ À ce terme est associée une loi uniforme, car nous n'avons pas d'a priori sur les consignes à atteindre.

$P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t)$ La variable A_{prox} n'ayant pas de continuité entre ses différentes valeurs, seule une loi de succession de Laplace a un sens pour ce terme. Nous notons :

$$P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t) = \mathbf{L}_{n_1,\ldots,n_{k_{A_{prox}} * k_{Sit_{t+\Delta t}} * k_{Sit_t}}}(A_{prox})$$

2. Identification : Nous allons supposer pour le moment que tous les termes sont fournis *a priori*. La Figure 6.2 donne les tables de probabilités du terme $P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t)$, et l'Algorithme 6.3 définit la fonction f, utilisée par le terme $P(Sit_t \mid Px)$. Dans cet algorithme, les primitives booléennes `AvG`, `AvD`, `ArG`, et `ArD`, décrivent les conditions que les capteurs doivent remplir pour considérer le robot dans un coin « Avant-Gauche », « Avant-Droit », « Arrière-Gauche », et « Arrière-Droit », respectivement. Par exemple, nous décidons que le robot est dans un coin `AvG` si les capteurs avant et gauche du robot sont excités au dessus de seuils λ_{av} et λ_g, et que les autres capteurs sont à 0. Expérimentalement, ces conditions correspondent effectivement à une zone dans un coin de notre environnement.

Alg. 6.3 Algorithme calculant la fonction f, utilisée par le terme $P(Sit_t \mid Px)$.

```
si max des pxi = 0
      alors Sit <- " libre "
      sinon si AvG ou AvD ou ArG ou ArD
            alors Sit <- " coin "
            sinon Sit <- " mur "
```

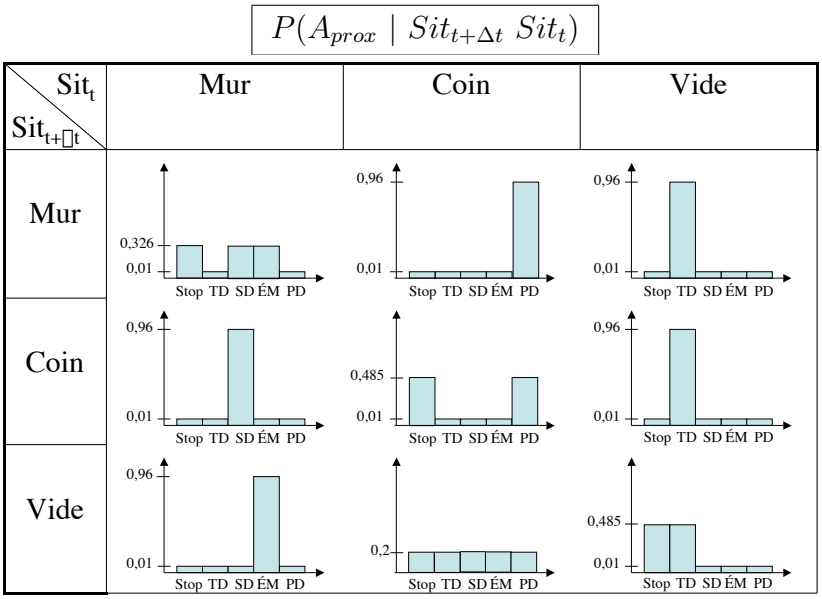


FIG. 6.2: La spécification du terme $P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t)$. Sur les graphiques, les actions sont dénotées comme suit : Stop pour $Stop$, TD pour $ToutD$, SD pour $SuiviD$, ÉM pour $ÉloignerMur$, PD pour $PasserD$. Les valeurs 0,01 ne sont pas à l'échelle.

6.3.3 Utilisation

Afin de conclure que les connaissances fournies sont bien une carte bayésienne, nous devons nous assurer que les réponses aux trois questions de localisation, contrôle et prédiction sont effectivement informatives. Nous vérifions ensuite expérimentalement que la carte permet la résolution de la tâche définie.

6.3.3.1 Programme et question probabiliste

Commençons par la question probabiliste de localisation : dans notre expérience, elle devient $P(L_t \mid P) = P(Sit_t \mid Px)$. Nous remarquons que ce terme apparaît dans la décomposition que nous avons choisi. Ainsi, comme nous l'avions noté Section 2.2.3.4, l'inférence ne fera que restituer ce terme. La présence ou non de signal pour la question de localisation sera donc facilement vérifiée, au vu de la définition même de ce terme.

Cela est également le cas pour le terme $P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t)$, qui correspond à la question de contrôle : poser cette question probabiliste revient à lire la spécification du terme correspondant, et nous vérifions facilement, Figure 6.2, la présence de signal sur les distributions de probabilités.

Enfin, nous posons la question probabiliste de prédiction $P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox})$, qui se

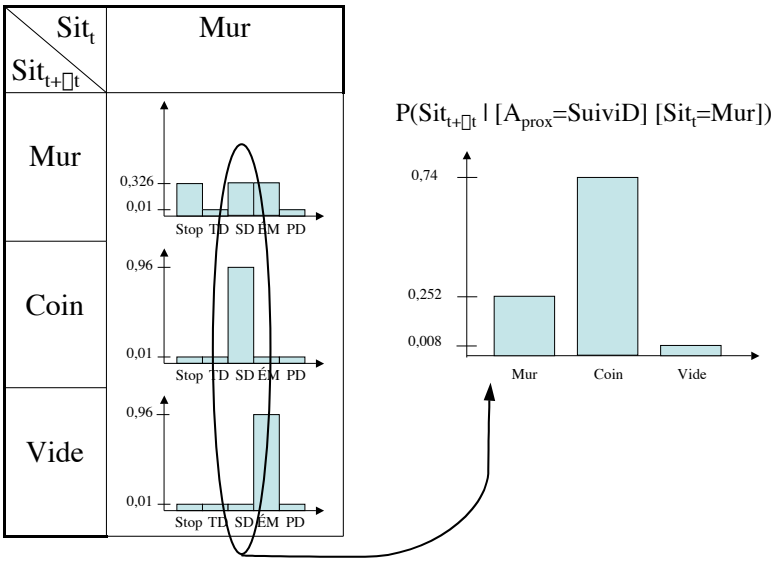


FIG. 6.3: Le résultat de l'inférence pour la question de prédiction $P(Sit_{t+\Delta t} \mid [Sit_t = mur]\ [A_{prox} = SuiviD])$. La barre pour la valeur 0,008 n'est pas à l'échelle.

résout par [4] :

$$\begin{aligned}
P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) &= \frac{1}{Z_1}\sum_{Px} P(Px)P(Sit_t \mid Px)P(Sit_{t+\Delta t})P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t) \\
&= \frac{1}{Z_1}P(Sit_{t+\Delta t})P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t)\sum_{Px} P(Px)P(Sit_t \mid Px) \\
&= \frac{1}{Z_2}P(Sit_{t+\Delta t})P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t) \\
P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) &= \frac{1}{Z_3}P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t).
\end{aligned}$$

La réponse à la question de prédiction peut donc se déduire du terme de contrôle, en le lisant en « sens inverse » , ou « transversalement » : nous montrons comment lire le résultat de la question $P(Sit_{t+\Delta t} \mid [Sit_t = mur]\ [A_{prox} = SuiviD])$ sur la Figure 6.3. Les valeurs numériques sont le résultat de la renormalisation des valeurs lues en « sens inverse » sur le terme $P([A_{prox} = SuiviD] \mid Sit_{t+\Delta t}\ [Sit_t = mur])$.

Pour les autres parties droites possibles de la question de prédiction ($[Sit_t = coin]$, $[A_{prox} = Stop]$, etc.), nous invitons le lecteur à lire transversalement la Figure 6.2 : nous vérifions ainsi aisément que le terme $P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox})$, pour toutes ses parties droites possibles, est bien informatif.

[4] Nous réferrons le lecteur à l'Annexe A pour une dérivation plus détaillée et commentée.

6.3.3.2 Résultats

Cette carte bayésienne peut donc maintenant servir à répondre à la tâche $SeCacher$. Mathématiquement, nous posons donc à la carte c^{prox} la question :

$$P(A_{prox} \mid [Sit_{t+\Delta t} = coin]\ [Px = px]).$$

Nous vérifions expérimentalement que le robot résout bien la tâche qui lui a été assignée. Notons à ce stade que la carte est une représentation suffisamment riche pour répondre à d'autres tâches. Les deux tâches évidentes sont celles qui consistent à retourner dans l'espace vide ($SeMontrer$) ou auprès d'un mur ($Longer$), quel que soit le point de départ. Ces deux tâches se traduisent respectivement par les comportements élémentaires suivants :

$$P(A_{prox} \mid [Sit_{t+\Delta t} = libre]\ [Px = px]),$$
$$P(A_{prox} \mid [Sit_{t+\Delta t} = mur]\ [Px = px]).$$

Ces trois comportements, $SeCacher$, $SeMontrer$, et $Longer$, sont les seuls comportements élémentaires que l'on peut définir en fixant uniquement une consigne de lieu à atteindre (puisque $k_{Sit_t} = 3$).

Il est également possible de générer d'autres tâches grâce à c^{prox}, par exemple en alternant des buts, au moyen d'un programme déterministe simple. Nous avons ainsi programmé le robot à explorer les bords de son environnement, en résolvant tout d'abord $SeCacher$, puis en alternant les tâches $Longer$ et $SeCacher$. Cette tâche est nommée $PatrouillerBord$.

6.4 Discussion

Avant de discuter de nos questions « fil conducteur » au vu de notre formalisme de carte bayésienne, replaçons notre définition dans le cadre de la programmation bayésienne des robots.

Dans ce chapitre, nous avons proposé de contraindre le schéma général d'un programme robotique bayésien pour le cas des cartes bayésiennes. Autrement dit, une carte bayésienne est un programme robotique bayésien dont les choix des variables et les questions qui lui sont posées sont restreintes. La Figure 6.4 donne la définition d'une carte bayésienne dans le cadre du formalisme de la programmation probabiliste.

Reprenons maintenant certaines de nos questions définies Section 1.2.

Q1. Qu'est-ce qu'une carte ? Nous avons donné ici une définition formelle de ce que nous appellerons par la suite une carte. Dans notre cadre, une carte est un ensemble de connaissances, exprimées sous la forme d'une description probabiliste.

Q2. Qu'est-ce qu'un lieu, pour le robot ? Dans la définition d'une carte bayésienne, les valeurs des variables L_t et $L_{t'}$ représentent les lieux connus par le robot. Avoir ces lieux à deux instants différents correspond au souhait de disposer :

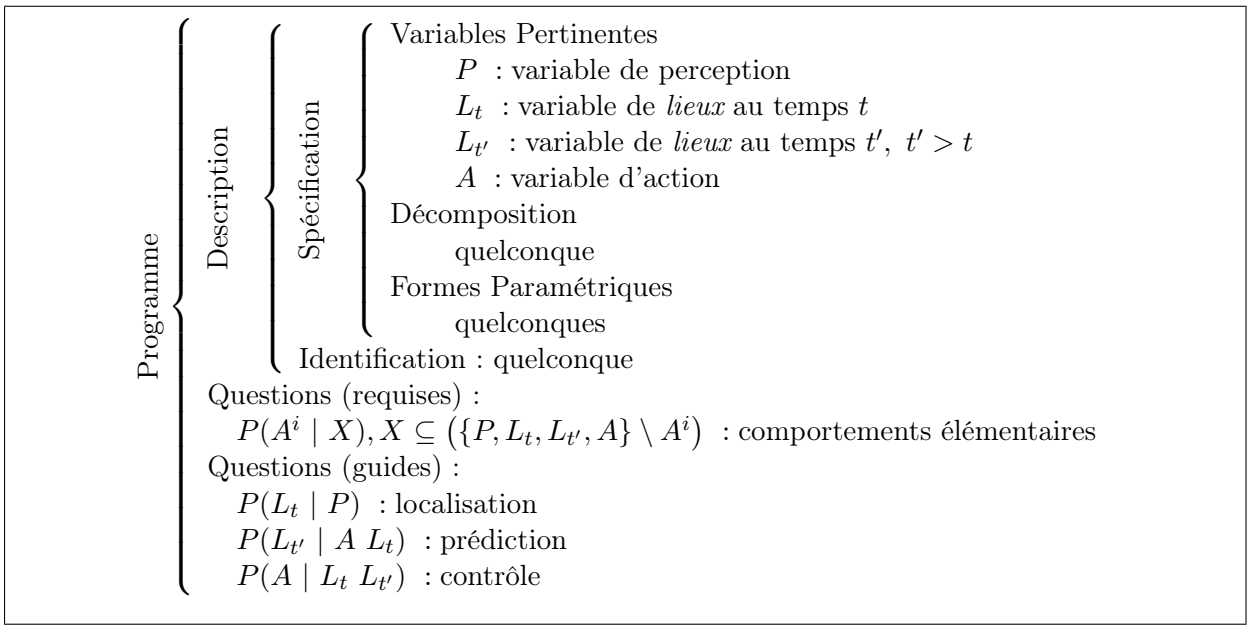


Fig. 6.4: La définition d'une carte bayésienne dans le formalisme PBR.

- de connaissances relatives à l'évolution dans le temps (la dynamique) de la position du robot ;
- et des connaissances traditionnelles de reconnaissance perceptives (statiques) des lieux.

Notre formalisme ne pose pas de contrainte sur la sémantique de la variable de lieux L_t choisie. Il est donc possible de définir des cartes bayésiennes métriques ou des cartes bayésiennes topologiques. Mais, d'une manière plus générale, une carte bayésienne peut aussi être interprétée comme un *modèle de l'interaction entre le robot et un phénomène* donné, quel que soit le type de phénomène considéré. Ainsi nous pourrons utiliser notre formalisme pour décrire l'interaction entre le robot et l'environnement statique, ce qui justifie le nom de « carte », mais nous pourrons aussi avoir des cartes d'espaces abstraits, pas forcément liés à la structure physique de l'environnement du robot :

- nous pourrions ainsi définir une « carte » d'obstacle dynamique, qui décrit comment un tel obstacle est perçu et sait prédire ses positions futures, sachant son mouvement précédent par exemple ;
- nous pourrions définir une « carte » d'un ballon, qui décrit comment un ballon est perçu et comment les actions du robot influent sur sa relation avec le robot ;
- nous pourrions définir une « carte » de la consommation énergétique du robot, qui décrit les sources pouvant alimenter le robot, et la relation entre les mouvements du robot et son niveau d'énergie (certaines actions – boire, manger – font remonter l'énergie, d'autres – se déplacer – font baisser l'énergie) ;
- etc.

Dans la suite de ce travail, nous nous concentrerons uniquement sur des cartes ayant trait à la description de l'environnement physique statique du robot, afin de ne pas diluer notre

argumentation à force de l'étayer.

Q3. Qu'est-ce que naviguer ? Nous l'avons vu, une carte est un ensemble de connaissances. C'est donc un objet que l'on peut mémoriser. En revanche, la navigation est un processus, une compétence, qui est issue d'une carte. Ainsi, dans ce travail, nous reprenons la définition de Van Zwynsvoorde : « *naviguer* c'est [...] réaliser une tâche dont le but est de se rendre à un endroit donné de l'environnement. » [Van Zwynsvoorde00].

La phrase précédente insiste bien sur le fait que naviguer, c'est *faire* quelque chose. Naviguer, c'est restituer les connaissances contenues dans une carte, en agissant dans le monde physique. Elle reste également imprécise sur la nature de l'endroit à atteindre et la nature de l'environnement. Ainsi, nous englobons, avec cette définition, les navigations à base de cartes métriques et topologiques. Enfin, cette citation ne dit rien sur la manière de résoudre cette tâche de navigation : nous avons pour notre part présenté plusieurs manières de le faire, soit par le biais des questions de localisation, prédiction, et contrôle, soit par le biais de comportements.

Q4.5.6. Qu'est-ce que se localiser, prédire, générer une loi de contrôle ? Comme pour naviguer, « se localiser », « prédire » et « contrôler » sont des compétences. Celles-ci étant primordiales dans le cadre de la navigation, nous leur avons associé des questions probabilistes particulières, et la contrainte de disposer de suffisamment d'information sur ces questions. Elles servent donc à *utiliser* une carte, à restituer les connaissances de la carte. Nous avons montré Section 6.1.3 quelques utilisations possibles de ces trois questions probabilistes.

Q3. Qu'est-ce qu'un comportement ? Les comportements sont une autre manière de restituer les connaissances d'une carte en vue de la résolution d'une tâche de navigation. Dans notre travail, un comportement est défini par une question probabiliste posée à une carte bayésienne. Cette question est seulement contrainte par le fait de posséder une variable d'action en partie gauche. La résolution de cette question peut faire intervenir des termes particuliers (qui intéressent le programmeur), ou être le résultat de l'inférence générale sur la carte.

En général, une carte sert à définir plusieurs comportements élémentaires. Le programmeur choisit quel ensemble de tâches il veut résoudre, et fournit, dans la carte, les connaissances nécessaires à leurs résolutions. Une carte est donc un modèle qui n'est pas indépendant de la tâche à résoudre (il est plus ou moins optimisé pour la résolution de certaines tâches choisies), mais n'est pas non plus complètement dépendant d'*une tâche particulière*.

Q2. Qu'est-ce qu'un lieu, pour le robot ? Nous l'avons vu, les choix faits lors de la spécification de la carte c^{prox} étaient guidés par la tâche $SeCacher$. Nous avons ainsi choisi une représentation simple, avec une variable de lieu Sit_t ne prenant que trois valeurs, dont la sémantique est étroitement reliée aux situations à reconnaître pour résoudre $SeCacher$.

Chacune de ces valeurs n'est pas associée à un point de l'environnement, mais à une zone complète de l'environnement. Cet aspect peut être rapproché de le représentation choisie par Gaussier pour résoudre une tâche de retour à la base [GJBR00], mais, au contraire de cette approche, nos zones ne sont pas nécessairement connexes dans l'environnement (voir Figure 6.1(a)).

Notre représentation est si simple, qu'elle souffre évidemment de perceptual aliasing au sens métrique : il existe bien des endroits aux coordonnées cartésiennes différentes qui sont représentées par la même valeur de Sit_t (voir Figure 6.1(a)). Cependant, au vu de la tâche à résoudre, la carte c^{prox} ne souffre d'aucun perceptual aliasing, car la tâche $SeCacher$ ne fait pas référence à rejoindre *un coin particulier*, mais à rejoindre *un coin quelconque* de l'environnement.

Q7. Comment spécifier une carte ? Ainsi la représentation choisie pour c^{prox} est adéquate pour résoudre $SeCacher$, mais pas pour des tâches plus complexes. Par exemple, si le robot doit rejoindre un coin particulier, il faut alors résoudre le problème de perceptual aliasing. Pour cela, il est possible d'enrichir les capacités perceptives du robot :

- en utilisant le capteur d'odeurs et en distinguant le coin à atteindre par une caractéristique odorante. C'est ce que nous ferons avec notre opérateur de superposition de cartes, Chapitre 8 ;
- en utilisant l'odométrie pour caractériser les murs à proximité du coin à atteindre ;
- ou encore, en utilisant de la mémoire pour désambiguïser les différents coins de l'environnement. La plupart du temps, cette solution est utilisée dans la littérature, mais elle est souvent coûteuse à mettre en œuvre, et la représentation obtenue souffre en général de global connectivity (puisqu'il sera possible d'arriver au coin objectif par plusieurs chemins possibles).

Chapitre 7

Cartes bayésiennes possibles

> « He walked up toward the Playground, planning the future. It seemed you did more planning in autumn than any other season. This had to do with dying, perhaps. You thought of death and you automatically planned. »
>
> Ray Bradbury, *The Playground*, 1953.

Dans ce chapitre, nous allons explorer l'éventail de possibilités à la disposition du programmeur, lorsqu'il définit une carte bayésienne et lorsqu'il l'utilise. Nous parcourrons ainsi les processus de spécification (choix des variables, décompositions, et formes paramétriques), d'identification, et enfin d'utilisation, puis conclurons par quelques remarques sur le rôle du programmeur face à ces choix.

7.1 Variables possibles

Rappelons ici qu'une carte est définie sur quatre variables, P, L_t, $L_{t'}$ et A, respectivement variable de perception, de lieu au temps t, de lieu au temps t', et d'action.

7.2 Décompositions possibles

Dans notre formalisme de cartes bayésiennes, nous n'avons pas contraint le programmeur quant au choix de la décomposition de la distribution conjointe $P(P\ L_t\ L_{t'}\ A)$, au contraire des approches classiques trouvées dans la littérature (voir Chapitre 3). Quelle est exactement la latitude laissée au programmeur en conséquence ?

Pour répondre à cette question, nous avons développé l'Algorithme 7.1, qui permet de générer toutes les décompositions valides portant sur un ensemble de variables $X = \{X_1, \ldots, X_n\}$.

Par exemple, pour 3 variables $X = \{X_1, X_2, X_3\}$, la première étape consiste à générer les partitions suivantes :

- $\{X_1\}, \{X_2\}, \{X_3\}$,
- $\{X_1\}, \{X_2, X_3\}$,

Alg. 7.1 Algorithme générant toutes les décompositions valides pour n variables.

```
pour i allant de 1 à n
    générer les partitions de X en i sous-ensembles
pour chacune des partitions
    générer les permutations ordonnées des sous-ensembles
/* on dispose maintenant de tous les produits de termes possibles */
pour chaque produit de terme
    générer toutes les hypothèses d'indépendances conditionnelles
enlever les doublons
```

– $\{X_2\}, \{X_1, X_3\}$,
– $\{X_3\}, \{X_1, X_2\}$,
– $\{X_1, X_2, X_3\}$.

La seconde étape produit six éléments à partir du premier ci-dessus, deux pour chacun des trois suivants, le dernier ne générant que lui-même :

– $\{X_1\}, \{X_2\}, \{X_3\}$,
– $\{X_1\}, \{X_3\}, \{X_2\}$,
– $\{X_2\}, \{X_1\}, \{X_3\}$,
– $\{X_2\}, \{X_3\}, \{X_1\}$,
– $\{X_3\}, \{X_1\}, \{X_2\}$,
– $\{X_3\}, \{X_2\}, \{X_1\}$,
– $\{X_1\}, \{X_2, X_3\}$,
– $\{X_2, X_3\}, \{X_1\}$,
– $\{X_2\}, \{X_1, X_3\}$,
– $\{X_1, X_3\}, \{X_2\}$,
– $\{X_3\}, \{X_1, X_2\}$,
– $\{X_1, X_2\}, \{X_3\}$,
– $\{X_1, X_2, X_3\}$.

À ce stade, chaque élément de la liste ci-dessus encode un produit de distributions probabilistes, avec les conventions suivantes : chaque sous-ensemble correspond à la partie gauche d'un terme, sa partie droite étant composée de tous les éléments des sous-ensembles apparus avant ce terme. Par exemple le premier élément, $\{X_1\}, \{X_2\}, \{X_3\}$, correspond implicitement à la décomposition

$$P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2\ X_1).$$

La dernière étape consiste, à partir d'un tel élément, à générer toutes les hypothèses d'indépendances conditionnelles. Dans cet exemple, à partir du seul élément

$$P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2\ X_1),$$

nous obtiendrons (noter les parties droites disparaissant, dans toutes les combinaisons possibles) :

- $P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2\ X_1)$,
- $P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2)$,
- $P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1)$,
- $P(X_1)P(X_2 \mid X_1)P(X_3)$,
- $P(X_1)P(X_2)P(X_3 \mid X_2\ X_1)$,
- $P(X_1)P(X_2)P(X_3 \mid X_2)$,
- $P(X_1)P(X_2)P(X_3 \mid X_1)$,
- $P(X_1)P(X_2)P(X_3)$.

Au total, lorsque l'ensemble X contient 2 variables, il existe 4 décompositions possibles :

- $P(X_1\ X_2)$,
- $P(X_1)P(X_2)$,
- $P(X_1)P(X_2 \mid X_1)$,
- $P(X_2)P(X_1 \mid X_2)$.

Pour 3 variables, il y a 41 décompositions :

- $P(X_1\ X_2\ X_3)$,
- $P(X_1)P(X_2\ X_3)$,
- $P(X_1)P(X_2\ X_3 \mid X_1)$,
- $P(X_1 \mid X_2)P(X_2\ X_3)$,
- $P(X_1 \mid X_3)P(X_2\ X_3)$,
- $P(X_1 \mid X_2\ X_3)P(X_2\ X_3)$,
- $P(X_1\ X_2)P(X_3)$,
- $P(X_1\ X_2)P(X_3 \mid X_1)$,
- $P(X_1\ X_2)P(X_3 \mid X_2)$,
- $P(X_1\ X_2)P(X_3 \mid X_1\ X_2)$,
- $P(X_1\ X_3)P(X_2)$,
- $P(X_1\ X_3)P(X_2 \mid X_1)$,
- $P(X_1\ X_3)P(X_2 \mid X_3)$,
- $P(X_1\ X_3)P(X_2 \mid X_1\ X_3)$,
- $P(X_1\ X_2 \mid X_3)P(X_3)$,
- $P(X_1\ X_3 \mid X_2)P(X_2)$,
- $P(X_1)P(X_2)P(X_3)$,
- $P(X_1)P(X_2)P(X_3 \mid X_1)$,
- $P(X_1)P(X_2)P(X_3 \mid X_2)$,
- $P(X_1)P(X_2)P(X_3 \mid X_1\ X_2)$,
- $P(X_1)P(X_2 \mid X_1)P(X_3)$,
- $P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1)$,
- $P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2)$,
- $P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1\ X_2)$,
- $P(X_1)P(X_2 \mid X_3)P(X_3)$,
- $P(X_1)P(X_2 \mid X_3)P(X_3 \mid X_1)$,
- $P(X_1)P(X_2 \mid X_1\ X_3)P(X_3)$,
- $P(X_1)P(X_2 \mid X_1\ X_3)P(X_3 \mid X_1)$,
- $P(X_1 \mid X_2)P(X_2)P(X_3)$,

- $P(X_1 \mid X_2)P(X_2)P(X_3 \mid X_1)$,
- $P(X_1 \mid X_2)P(X_2)P(X_3 \mid X_2)$,
- $P(X_1 \mid X_2)P(X_2)P(X_3 \mid X_1\ X_2)$,
- $P(X_1 \mid X_2)P(X_2 \mid X_3)P(X_3)$,
- $P(X_1 \mid X_3)P(X_2)P(X_3)$,
- $P(X_1 \mid X_3)P(X_2)P(X_3 \mid X_2)$,
- $P(X_1 \mid X_3)P(X_2 \mid X_1)P(X_3)$,
- $P(X_1 \mid X_3)P(X_2 \mid X_3)P(X_3)$,
- $P(X_1 \mid X_3)P(X_2 \mid X_1\ X_3)P(X_3)$,
- $P(X_1 \mid X_2\ X_3)P(X_2)P(X_3)$,
- $P(X_1 \mid X_2\ X_3)P(X_2)P(X_3 \mid X_2)$,
- $P(X_1 \mid X_2\ X_3)P(X_2 \mid X_3)P(X_3)$.

La croissance du nombre de décompositions paraît exponentielle. Malheureusement, notre algorithme de génération ne conduit pas à une formule de dénombrement, à cause des doublons générés. Nous pouvons toutefois indiquer que pour 4 variables, il y a 1015 décompositions possibles (voir Annexe B), et 59072 pour 5 variables.

Le cas « 4 variables » nous intéresse tout particulièrement, car c'est le nombre minimum de variables qu'une carte bayésienne comptera (nous avons déjà noté Section 7.6 la possibilité d'en avoir plus).

Devant une telle panoplie de décompositions pour quatre variables, on comprend que les modèles probabilistes évoqués au Chapitre 3 ne sont que quelques choix possibles parmi tant d'autres. Il est donc préférable de laisser ce choix au programmeur, en fonction des nombreuses contraintes qu'il doit prendre en compte. Ces contraintes dépendent du problème particulier qu'il traite : certains termes peuvent faire apparaître des hypothèses d'indépendance conditionnelle intuitives, d'autres seront plus facile à définir ou à apprendre expérimentalement, d'autres prendront moins d'espace mémoire, d'autre rendront les inférences traitables en temps réel (ou en temps raisonnable si le but est de tabuler hors ligne les réponses à certaines questions probabilistes), etc.

Il n'est cependant pas possible de définir une carte bayésienne qui inclurait à la fois le terme de prédiction $P(L_{t'} \mid A\ L_t)$ et le terme de contrôle $P(A \mid L_t\ L_{t'})$ dans sa décomposition, à cause d'une dépendance circulaire entre $L_{t'}$ et A. Rappelons que nous avons établit une contrainte de présence de signal sur ces deux termes. Ainsi, si le programmeur veut faire apparaître un de ces deux termes dans la décomposition, il devra s'assurer que l'inversion de ce terme, pour le calcul de l'autre, conserve du signal.

Énumérons maintenant quelques exemples de décompositions potentiellement intéressantes.

Une décomposition qui permet d'obtenir des temps de calculs courts est :

$$P(P)P(L_t \mid P)P(A\ L_{t'} \mid L_t).$$

En effet, dans ce cas :

- pour répondre à $P(L_t \mid P)$, il n'y a pas besoin de calculs, juste de la lecture du terme spécifié dans la décomposition ;

- pour répondre à $P(L_{t'} \mid A\ L_t)$, il suffit de « faire une coupe » de $P(A\ L_{t'} \mid L_t)$ sur une des dimensions de l'espace $A \wedge L_{t'}$;
- pour répondre à $P(A \mid L_{t'}\ L_t)$, il suffit également de faire une coupe du terme $P(A\ L_{t'} \mid L_t)$.

Cette décomposition permet donc de répondre aux trois questions de localisation, prédiction et contrôle de manière très rapide. On comprend cependant aisément que ce gain en temps de calcul se paye. Un des désavantages est une plus grande difficulté *a priori* pour spécifier un terme sur un espace de probabilités à deux dimensions, pour le terme $P(A\ L_{t'} \mid L_t)$. Mais si la sémantique des variables en jeu, et le problème à résoudre, permettent une telle spécification, alors le programmeur aura tout intérêt à tirer parti de cette décomposition particulière. Le second désavantage notable de cette décomposition est que la récurrence temporelle sur le calcul de L_t n'est plus possible [1]. Cette décomposition ne peut donc s'appliquer que si une telle récurrence n'est pas nécessaire, c'est-à-dire dans les modèles sans *perceptual aliasing*, où le robot n'a donc besoin que d'une donnée pour se localiser. Si le modèle ne possède pas cette propriété, alors le programmeur pourra se rabattre sur la décomposition

$$P(L_t)P(P \mid L_t)P(A\ L_{t'} \mid L_t),$$

mais dans ce cas, répondre à la question de localisation $P(L_t \mid P)$ nécessitera une inversion du terme $P(P \mid L_t)$, ce qui est bien sûr plus coûteux en temps de calcul.

Une autre décomposition intéressante est :

$$P(L_t)P(P \mid L_t)P(L_{t'} \mid L_t)P(A \mid L_t\ L_{t'}).$$

Cette décomposition est proche de celle que nous avons utilisé dans notre exemple du Chapitre précédent, Section 6.3, au terme $P(L_{t'} \mid L_t)$ près, qui remplace $P(L_{t'})$. Ce terme permet d'encoder des connaissances relatives à la structure de l'environnement. En un pas de temps, sachant la position du robot, quelles peuvent-être les positions suivantes ? De telles connaissances sont très souvent faciles à spécifier, mais souvent ignorées dans les modèles comme la localisation markovienne, où le terme le plus proche est $P(L_{t'} \mid A\ L_t)$, qui dépend *à la fois* des caractéristiques motrices du robot *et* de la structure de l'espace, et donc est plus difficile à mettre en œuvre.

Dans la localisation markovienne, nous l'avons noté, le terme $P(L_t)$ est utilisé pour mémoriser la localisation du pas de temps précédent (ce que nous noterons parfois par la forme paramétrique dite de « récurrence »). Si, dans une carte donnée, le problème de perceptual aliasing ne se pose pas, alors le terme $P(L_t)$ peut être remplacé par des termes ayant plus de sens, comme par exemple dans la décomposition suivante :

$$P(A)P(L_t \mid A)P(P \mid L_t)P(L_{t'} \mid A\ L_t).$$

Le terme $P(L_t \mid A)$ a la signification suivante : sachant l'action que le robot est en train de faire, dans quel lieu se trouve-t-il ? Ce terme peut éventuellement contenir beaucoup

[1] Pour rappel, dans la localisation markovienne, on trouve un terme $P(L_t)$ qui permet justement cette récurrence, c'est-à-dire de copier dans ce terme le résultat de la localisation du pas de temps précédent.

d'information pertinente. Par exemple, dans le niveau sensori-moteur le plus bas, tourner très vite dans un sens ou l'autre est en général indicateur d'un obstacle à éviter. À plus haut niveau, ce terme peut établir la corrélation entre l'action « boire » et le fait d'être à proximité d'un point d'eau, ou « dormir » et être au gîte, etc.

Une dernière décomposition que nous souhaitons aborder ici est :

$$P(L_t)P(A \mid L_t)P(P \mid A\ L_t)P(L_{t'} \mid A\ L_t).$$

Dans cette décomposition, le terme intéressant est $P(P \mid A\ L_t)$, qui lie les perceptions non plus simplement à la localisation, comme nous l'avons fait jusqu'à maintenant (par les termes $P(P \mid L_t)$ ou $P(L_t \mid P)$), mais aussi aux actions : sachant la position courante, et l'action appliquée, que doit-on percevoir ?

À notre connaissance, aucune des décompositions énumérées ci-dessus n'ont été utilisées dans les systèmes robotiques décrits dans la littérature.

7.3 Formes paramétriques possibles

Nous préférons, dans le cadre de ce travail, nous restreindre aux formes paramétriques simples évoquées Section 2.2.3.2. Quant à énumérer les formes paramétriques particulières possibles dans l'ensemble des cartes bayésiennes possibles, notons que cela a peu de sens, car ces formes paramétriques dépendent lourdement du problème considéré. Nous n'irons pas plus loin dans l'analyse de cette partie de la définition d'une carte bayésienne.

7.4 Identifications possibles

Nous l'avons vu, dans notre formalisme, l'apprentissage consiste à identifier les paramètres de formes apparaissant dans la décomposition, à partir d'un jeu de données expérimentales. Pour identifier un terme $P(X \mid Y)$, les données expérimentales doivent donc contenir des valeurs de X et des valeurs de Y, ce qui correspond à la notion d'apprentissage supervisé (pour un travail précédent concernant l'apprentissage en présence de données manquantes, dans le cadre d'une hiérarchie de comportements, voir [Diard99]).

Dans les cartes bayésiennes, nous laissons le choix de la décomposition au programmeur, donc les termes à apprendre peuvent être variés. Cependant, nous souhaitons donner ici quelques exemples de termes intéressants à identifier par l'expérience. Faisons donc l'hypothèse de la décomposition suivante :

$$P(L_t)P(P \mid L_t)P(A \mid L_t)P(L_{t'} \mid A\ L_t).$$

Dans cette décomposition, définissons le terme $P(L_t)$ comme une loi uniforme. Nous allons apprendre les trois autres termes, un à un.

7.4.1 Identification de $P(P \mid L_t)$

Le premier terme que nous identifions est $P(P \mid L_t)$. Pour ce faire, il faut indiquer au robot le lieu l_t où il se trouve, et collecter les données sensorielles p_t correspondantes. Expérimentalement, il s'agit de poser le robot à un endroit de l'environnement, d'enregistrer des données sensorielles, et de leur attacher une valeur de la variable l_t. Une fois ce terme appris, il peut servir de base à l'inférence pour calculer $P(L_t \mid P)$. En effet, la question $P(L_t \mid P)$ se résout par :

$$\begin{aligned}
P(L_t \mid P) &= \frac{1}{Z_1} \sum_{A\ L_{t'}} P(L_t)P(P \mid L_t)P(A \mid L_t)P(L_{t'} \mid A\ L_t) \\
&= \frac{1}{Z_1} P(L_t)P(P \mid L_t) \sum_{A\ L_{t'}} P(A \mid L_t)P(L_{t'} \mid A\ L_t) \\
&= \frac{1}{Z_1} P(L_t)P(P \mid L_t) \sum_{A} P(A \mid L_t) \sum_{L_{t'}} P(L_{t'} \mid A\ L_t) \\
&= \frac{1}{Z_1} P(L_t)P(P \mid L_t) \\
P(L_t \mid P) &= \frac{1}{Z_2} P(P \mid L_t).
\end{aligned}$$

7.4.2 Identification de $P(A \mid L_t)$

Maintenant que nous pouvons calculer $P(L_t \mid P)$, il est possible de tirer aléatoirement des valeurs pour la variable L_t. Ainsi, nous pouvons obtenir des données expérimentales sur la variable L_t sans avoir à lui donner explicitement des valeurs. Si nous pilotons le robot, par exemple au joystick, nous collectons des données sur les variables de perception P et d'action A, sous la forme de couples $\langle p, a \rangle$. Pour chacune de ces données, nous posons la question probabiliste $P(L_t \mid [P = p])$, et tirons sur la distribution obtenue. Avec les valeurs l tirées aléatoirement, nous obtenons des triplets $\langle p, a, l \rangle$, qui nous permettent d'identifier le terme $P(A \mid L_t)$.

Cet apprentissage a été exploré en détail au niveau sensori-moteur par [Lebeltel99], et nous l'avons appliqué, dans un travail précédent, à un plus haut niveau d'abstraction (niveau « tâche », voir [Diard99]).

7.4.3 Identification de $P(L_{t'} \mid A\ L_t)$

Nous disposons maintenant des termes $P(L_t)$, $P(P \mid L_t)$ et $P(A \mid L_t)$. Bien qu'il nous manque encore le terme $P(L_{t'} \mid A\ L_t)$, nous pouvons tout de même calculer $P(A \mid P)$,

grâce à la dérivation suivante :

$$\begin{aligned}
P(A \mid P) &= \frac{1}{Z_1} \sum_{L_t \; L_{t'}} P(L_t) P(P \mid L_t) P(A \mid L_t) P(L_{t'} \mid A \; L_t) \\
&= \frac{1}{Z_1} \sum_{L_t} P(L_t) P(P \mid L_t) P(A \mid L_t) \sum_{L_{t'}} P(L_{t'} \mid A \; L_t) \\
&= \frac{1}{Z_1} \sum_{L_t} P(L_t) P(P \mid L_t) P(A \mid L_t) \\
P(A \mid P) &= \frac{1}{Z_2} \sum_{L_t} P(P \mid L_t) P(A \mid L_t).
\end{aligned}$$

Répondre à cette question probabiliste, en pratique, donne au robot un comportement : le robot est capable de calculer une distribution de probabilités sur les actions, selon les données sensorielles. Afin de compléter ce modèle probabiliste et d'obtenir une carte bayésienne, il nous faut maintenant identifier le dernier terme, $P(L_{t'} \mid A \; L_t)$. Pour cela, nous faisons appliquer au robot le comportement (par le terme $P(A \mid P)$), et collectons des données $\langle l_t, a_t \rangle$. En considérant deux de ces données, l'une au temps t, l'autre au temps $t + \Delta t$, nous disposons de données $\langle l_t, a_t, l_{t+\Delta t}, a_{t+\Delta t} \rangle$. En ôtant $a_{t+\Delta t}$, nous obtenons des triplets $\langle l_t, a_t, l_{t+\Delta t} \rangle$ qui nous permettent d'identifier le terme $P(L_{t'} \mid A \; L_t)$, ce qui complète la carte bayésienne. Quand une carte peut ainsi être basée sur un comportement, nous parlerons de « comportement générateur de cartes ».

7.4.4 Expérience

Dans une expérience préliminaire récente, nous avons utilisé un comportement d'évitement d'obstacles pour apprendre le terme $P(L_{t'} \mid A \; L_t)$. Le résultat est bon, et nous obtenons dans ce terme des informations très riches. Par exemple, la structure de l'espace de la variable de lieux L_t apparaît clairement. En effet, dans l'espace de la variable L_t, en partant du point $[L_t = l_t]$, appliquer l'action $[A = a]$:

- a de bonnes chances de nous laisser au même lieu L_t (si Δt est petit) ;
- nous amène au mieux à quelques lieux du voisinage de l_t (selon l'action a).

Il est même possible d'utiliser ce type « d'analyse après expérience » pour avoir une meilleure idée des relations de voisinages entre lieux : seront voisins des lieux l_t et $l_{t'}$ tels que la probabilité de transition $P([L_{t'} = l_{t'}] \mid A \; [L_t = l_t])$ est non-nulle.

Nous pouvons aussi voir dans la forme apprise certaines caractéristiques du comportement d'évitement d'obstacle. Par exemple, si un mur est sur la gauche, le fait de tourner vers la droite amène le mur encore plus à gauche (ce qui est une manœuvre typique d'évitement).

Enfin, nous avons comparé les termes $P(L_{t'} \mid A \; L_t)$ appris sur la base de deux constantes de temps différentes ($\Delta t = 100 \; ms$ et $\Delta t = 200 \; ms$), et observé qu'un pas de temps à 200 ms faisait apparaître plus de signal pertinent dans le terme $P(L_{t'} \mid A \; L_t)$. En effet, avec

cette constante, on obtient moins de données du type $\langle l_t, a_t, l_{t'} = l_t \rangle$, qui sont identifiées comme de l'auto-maintien. On vérifie aussi que le terme appris conserve suffisamment de signal : en effet, à la limite où Δt est très grand, tout point de l'espace interne est atteignable, et la probabilité de rester dans le même état est très faible, donc l'auto-maintien est minimum. Étant donné ce critère (diminuer l'auto-maintien, tout en conservant une information pertinente sur l'espace interne), nous pouvons envisager de chercher la constante de temps Δt qui le minimise. Notons que l'espace de recherche de cette minimisation est très petit, puisque nous pouvons a priori supposer que notre comportement a une dynamique intéressante à identifier entre 1 et une dizaine de cycles (entre 100 ms et 1 s).

7.4.5 Conclusions sur l'apprentissage

Nous voyons qu'il est donc possible d'apprendre une carte en sa basant sur un comportement. Une fois une carte satisfaisante à disposition (après le choix d'un bon Δt, par exemple), nous pouvons en exploiter la richesse comme toute autre carte. Par exemple, nous pouvons en tirer le graphe induit, calculer sur ce graphe son diamètre, sa connectivité moyenne, ses cliques, etc. Nous pouvons utiliser ce graphe pour guider un algorithme de planification, ou même planifier directement dans la carte probabiliste, ce qui est impossible si l'on a que le comportement. Il est aussi possible de caractériser la qualité de la carte obtenue en analysant les entropies des formes qui y apparaissent. Par exemple, si la distribution conjointe est proche de l'uniforme, nous comprenons que la carte aura peu d'utilité. À l'autre extrême, si toutes les formes paramétriques sont des Diracs, nous pouvons tirer du modèle probabiliste un modèle logique, et raisonner avec (par exemple, trouver les points de convergence du phénomène). Dans notre expérience préliminaire, nous étions clairement situés loin de ces deux cas extrêmes : la carte obtenue en se basant sur le comportement d'évitement d'obstacles contenait beaucoup d'informations exploitables, mais restituait également les incertitudes dues à la simplicité du modèle.

Pour finir, remarquons que les exemples d'apprentissage que nous avons présenté ici ensemble peuvent être utilisés indépendamment. Par exemple, dans la carte de la situation « mur », seul le terme $P(P \mid L_t)$ est identifié expérimentalement, les autres termes étant définis *a priori*. Nous pouvons aussi apprendre au robot un comportement par le terme $P(A \mid L_t)$, même si le terme $P(P \mid L_t)$ a été spécifié à la main. Il est également possible d'essayer de transformer un comportement en carte en apprenant le terme $P(L_{t'} \mid A\ L_t)$, même si ce comportement est programmé. Enfin, les exemples d'apprentissage détaillés ici se basent sur des termes particuliers, mais la méthode s'applique également pour d'autres décompositions possibles.

7.5 Questions possibles

Pour une description donnée, il existe de nombreuses questions probabilistes possibles (voir Section 2.2.3.4 pour la définition formelle). Pour n variables, il y a $3^n - 2^n$ questions différentes : 3^n est le nombre de partitions de n variables en trois sous-ensembles, auquel on

soustrait 2^n, le nombre de partitions de n variables en deux sous-ensembles, car la partie gauche de la question ne doit pas être vide. Pour une carte bayésienne, nous n'avons que les quatre variables A, P, L_t et $L_{t'}$ à considérer, et donc $81 - 16 = 65$ questions distinctes.

Nous pouvons générer ces 65 questions avec l'Algorithme 7.2 ; le résultat pour quatre variables est présenté Table 7.1.

Alg. 7.2 Algorithme générant toutes les questions valides pour quatre variables.

```
pour i parcourant {A P Lt Lt'}
    placer i dans le sous-ensemble Search
    pour toutes les autres variables
        les placer soit dans Search, soit dans Known, soit dans Unknown,
        selon toutes les combinaisons possibles.
    fin pour
fin pour
enlever les doublons
```

$P(A\ L_t\ L_{t'}\ P)$,	$P(A\ L_t\ P \mid L_{t'})$,	$P(A\ L_{t'}\ P \mid L_t)$,	$P(L_t\ L_{t'}\ P \mid A)$,
$P(A\ L_t\ L_{t'} \mid P)$,	$P(L_t\ L_{t'}\ P)$,	$P(A\ L_{t'}\ P)$,	$P(A\ L_t\ P)$,
$P(A\ L_t\ L_{t'})$,	$P(A\ P \mid L_t\ L_{t'})$,	$P(L_t\ P \mid A\ L_{t'})$,	$P(L_{t'}\ P \mid A\ L_t)$,
$P(A\ L_t \mid L_{t'}\ P)$,	$P(A\ L_{t'} \mid L_t\ P)$,	$P(L_t\ L_{t'} \mid A\ P)$,	$P(L_t\ P \mid L_{t'})$,
$P(L_{t'}\ P \mid L_t)$,	$P(A\ P \mid L_{t'})$,	$P(A\ P \mid L_t)$,	$P(L_{t'}\ P \mid A)$,
$P(L_t\ P \mid A)$,	$P(A\ L_{t'} \mid P)$,	$P(A\ L_t \mid L_{t'})$,	$P(A\ L_t \mid P)$,
$P(A\ L_{t'} \mid L_t)$,	$P(L_t\ L_{t'} \mid P)$,	$P(L_t\ L_{t'} \mid A)$,	$P(L_{t'}\ P)$,
$P(L_t\ P)$,	$P(A\ P)$,	$P(A\ L_{t'})$,	$P(A\ L_t)$,
$P(L_t\ L_{t'})$,	$P(P \mid A\ L_t\ L_{t'})$,	$P(A \mid L_t\ L_{t'}\ P)$,	$P(L_t \mid A\ L_{t'}\ P)$,
$P(L_{t'} \mid A\ L_t\ P)$,	$P(\ P \mid L_t\ L_{t'})$,	$P(P \mid A\ L_{t'})$,	$P(P \mid A\ L_t)$,
$P(A \mid L_{t'}\ P)$,	$P(A \mid L_t\ L_{t'})$,	$P(A \mid L_t\ P)$,	$P(L_t \mid L_{t'}\ P)$,
$P(L_t \mid A\ L_{t'})$,	$P(L_t \mid A\ P)$,	$P(L_{t'} \mid L_t\ P)$,	$P(L_{t'} \mid A\ L_t)$,
$P(L_{t'} \mid A\ P)$,	$P(P \mid L_{t'})$,	$P(P \mid L_t)$,	$P(P \mid A)$,
$P(A \mid L_{t'})$,	$P(A \mid P)$,	$P(A \mid L_t)$,	$P(L_t \mid L_{t'})$,
$P(L_t \mid P)$,	$P(L_t \mid A)$,	$P(L_{t'} \mid L_t)$,	$P(L_{t'} \mid P)$,
$P(L_{t'} \mid A)$,	$P(P)$,	$P(A)$,	$P(L_t)$,
$P(L_{t'})$.			

Tab. 7.1 – Les 65 questions pour une carte bayésienne à quatre variables.

Quelques unes de ces questions ont des interprétations évidentes. Ainsi nous voyons apparaître les trois questions de localisation $P(L_t \mid P)$, de prédiction $P(L_{t'} \mid A\ L_t)$ et de contrôle $P(A \mid L_t\ L_{t'})$, qui sont partie intégrante de la définition d'une carte bayésienne. Mais d'autre termes sont également remarquables, comme par exemple :

- $P(A \mid P\ L_{t'})$ est la question que nous poserons la plupart du temps à une carte, pour définir un comportement élémentaire : ne connaissant que les entrées sensorielles et la consigne à atteindre, quelle doit être l'action à appliquer aux moteurs du robot ?
- $P(A\ L_t\ L_{t'}\ P)$ est la distribution conjointe elle-même ;
- $P(P \mid L_t)$ est parfois appelé le modèle direct d'observation [BFJ+99], voire même la carte de l'environnement [Thrun98a]. Nous l'appelons *modèle capteur direct* (par opposition aux termes de la forme $P(L_t \mid P)$, que nous nommons « question de localisation » dans le cadre de ce travail, mais que nous appelons *modèle capteur inverse* d'une manière plus générale).

Notons que pour une décomposition donnée, certaines de ces questions sont résolues par le même calcul. Par exemple, si l'on se place dans le cadre d'une décomposition de type localisation markovienne

$$P(A\ P\ L_t\ L_{t'}) = P(A)P(L_t)P(P \mid L_t)P(L_{t'} \mid A\ L_t),$$

alors les deux questions $P(L_{t'} \mid A\ L_t)$ et $P(L_{t'} \mid A\ L_t\ P)$ ont le même résultat. En effet,

$$\begin{aligned}
P(L_{t'} \mid A\ L_t\ P) &= \frac{P(A\ P\ L_t\ L_{t'})}{\sum_{L_{t'}} P(A\ P\ L_t\ L_{t'})} \\
&= \frac{P(A\ L_t)P(P \mid L_t)P(L_{t'} \mid A\ L_t)}{\sum_{L_{t'}} P(A\ L_t)P(P \mid L_t)P(L_{t'} \mid A\ L_t)} \\
&= \frac{P(A\ L_t)P(P \mid L_t)P(L_{t'} \mid A\ L_t)}{P(A\ L_t)P(P \mid L_t)\sum_{L_{t'}} P(L_{t'} \mid A\ L_t)} \\
&= \frac{P(A\ L_t)P(P \mid L_t)P(L_{t'} \mid A\ L_t)}{P(A\ L_t)P(P \mid L_t)} \\
P(L_{t'} \mid A\ L_t\ P) &= P(L_{t'} \mid A\ L_t).
\end{aligned}$$

Nous avons également déjà évoqué Section 6.1.3 le fait qu'un programmeur va en pratique utiliser plus d'une question pour exploiter une carte. Il peut en effet soit enchaîner plusieurs questions dont la sémantique l'intéresse (nous avons vu le cas des questions de localisation et de contrôle à ce sujet), soit utiliser une question quelconque à une carte comme forme paramétrique utilisée dans une autre carte (des exemples en seront donnés dans nos opérateurs d'assemblage de cartes, Chapitre 8 et 9).

7.6 Discussion

Avant de conclure ce chapitre avec nos questions « fil conducteur », nous souhaitons revenir sur le choix des variables qui apparaissent dans une carte bayésienne.

7.6.1 « Explosion » des variables

En effet, jusqu'à maintenant, nous avons considéré les quatre variables P, L_t, $L_{t'}$ et A comme atomiques. Elles le sont rarement en pratique, et sont en réalité des conjonctions

de variables. Dans le cas contraire, le robot n'aurait qu'un capteur et qu'un moteur ! Dans notre premier exemple de carte bayésienne (Section 6.3), cela n'était déjà plus le cas : nous considérions l'ensemble des seize capteurs de proximétrie comme variable de perception P.

Chaque variable P, L_t, $L_{t'}$ et A peut donc être « explosée », et le programmeur peut tirer parti de ce pouvoir d'expression supplémentaire, notamment par le biais de décompositions ou de questions particulières. Les possibilités étant quasi-infinies, nous n'aborderons ici que quelques exemples.

7.6.2 Explosion de P

Supposons pour commencer que la variable P soit une conjonction de variables, $P = P_1 \wedge \ldots \wedge P_n$. Il est alors possible de poser à la carte bayésienne des questions particulières. La question $P(P_3 \mid P_2\ P_4\ L_t)$ par exemple, permet de calculer ce que devrait répondre le capteur P_3, sachant les lectures des capteurs P_2 et P_4 et sachant la situation courante. Cette question permet donc de faire du diagnostique sur les capteurs.

Il est aussi possible de profiter du fait que P soit une conjonction de variables au moment de la décomposition. Par exemple, un terme $P(P \mid L_t)$ devient $P(P_1\ \ldots\ P_n \mid L_t)$, qui peut être décomposé plus simplement, par exemple en $P(P \mid L_t) = \prod_i P(P_i \mid L_t)$. Notons qu'une décomposition de ce type, $P(V)\prod_i P(P_i \mid V)$, où V est une variable non-mesurable directement par un capteur, est une instance du schéma de *fusion de modèles capteurs*, pour reprendre les termes de [Lebeltel99]. De plus, cette fusion est une extension possible de la localisation markovienne (voir Section 3.8), dans laquelle le modèle d'observation unique $P(P \mid L_t)$ est remplacé par une fusion de plusieurs modèles de la forme $P(P_i \mid L_t)$.

7.6.3 Explosion de A

De manière similaire, nous utiliserons très souvent, pour les couches de bas niveaux moteurs, une conjonction de variables pour la variable A. Dans nos expériences sur le robot Koala, en effet, nous contrôlons le robot par les vitesses de rotation et de translation, donc $A = Vrot \wedge Vtrans$. Ce fait sera exploité par certaines de nos cartes, où $Vrot$ et $Vtrans$ pourront être considérées indépendantes (voir les exemples Annexe E).

7.6.4 Explosion de L_t

Supposons maintenant que ce soit la variable de lieu L_t qui soit une conjonction de variables. Un exemple est l'explosion de L_t en $L_t \wedge L_{t-\Delta t}$, qui correspond à une hypothèse de markov d'ordre 2 [Aycard98]. En effet dans ce cas, on peut exprimer des termes tels que $P(P \mid L_t\ L_{t-\Delta t})$, ou encore $P(L_{t+\Delta t} \mid L_t\ A\ L_{t-\Delta t})$. Il sera possible de cette manière d'écrire des cartes faisant des hypothèses de Markov d'ordre arbitrairement élevé.

7.6.5 Explosion de $L_{t'}$

Supposons enfin que la variable $L_{t'}$ soit une conjonction de variables [2]. Par exemple, si la variable de lieu porte sur un grand espace, il est possible que le robot ne puisse se rendre de n'importe quel lieu à n'importe quel autre lieu en une période Δt. Il sera alors préférable de diviser $L_{t'}$ en $L_{t+\Delta t} \wedge L_{t_{obj}}$, où $L_{t+\Delta t}$ est la variable de lieu au pas de temps suivant, et $L_{t_{obj}}$ le lieu objectif à atteindre. Le programmeur pourra alors spécifier des connaissances sur l'évolution à court terme de la variable de lieu, avec un terme $P(L_{t+\Delta t} \mid L_{t'})$, mais il pourra aussi indiquer avec $P(L_{t_{obj}} \mid L_{t+\Delta t})$ l'évolution possible de la variable de lieu à plus long terme, sachant la direction choisie pour le pas de temps suivant. Bien sûr, beaucoup d'autres alternatives existent, comme par exemple de spécifier non pas $P(L_{t_{obj}} \mid L_{t+\Delta t})$, mais le terme inverse, $P(L_{t+\Delta t} \mid L_{t_{obj}})$: sachant le but à long terme, quel doit être le lieu suivant à atteindre ?

Nous touchons ici un problème assez courant de désaccord entre le niveau de détail de la carte et son échelle. En effet, les cartes cherchent en général à couvrir l'espace le plus grand possible, tout en conservant une granularité qui soit en relation avec les capacités motrices du robot. Typiquement, les cartes à base de grilles d'occupation ont des cases de taille 50 cm par 50 cm, et couvrent une surface de l'ordre de 60 m par 60 m. Il est facile de donner au robot un modèle local de transition $P(L_{t+\Delta t} \mid A\ L_t)$, alors que ce qui est en réalité cherché est un modèle global de contrôle $P(A \mid L_t\ L_{t+\Delta t})$: sachant le lieu courant, sachant le but à atteindre, quel est l'action à appliquer. Ce problème est souvent résolu par un mécanisme, éventuellement très sophistiqué, de *planification*. Cependant, les algorithmes comme ceux rencontrés dans les PDMPOs, ne s'appliquent que sur des espaces petits (quelques dizaines d'états au mieux). En effet dans la planification, propager les incertitudes le long de l'inférence en avant dans le temps provoque une explosion combinatoire.

Dans ce travail, pour traiter ce problème, nous nous appuierons sur le fait d'établir des *hiérarchies de cartes*. Cela rendra d'une part abordable les calculs, en considérant un ensemble de petits espaces plutôt qu'une seule grande représentation, le désavantage étant la perte d'optimalité (au sens du plus court chemin dans l'espace métrique) des plans générés. Malgré cela, cette solution gagne en popularité en planification, dans les approches type Fil d'Ariane [MAB98] ou PPP [KSLO96, SO98] [3], où l'on essaie de capturer la topologie de l'espace par un graphe superposé à l'espace de travail du robot, mais aussi dans les approches probabilistes hiérarchiques (voir Section 4.1).

De plus, grâce à la décomposition en hiérarchie de cartes, les espaces de lieux que nous utiliserons dans ce travail auront en général des cardinaux très petits. Ainsi, les variables correspondantes auront des sémantiques tellement intuitives qu'il sera plus facile de résoudre ce problème de planification directement au moment de la spécification de la

[2] En réalité les explosions des variables L_t et $L_{t'}$ ne peuvent pas être considérées indépendamment. En effet, une carte bayésienne impose qu'elles soient définies sur le même domaine de valeurs. Cependant, il est toujours possible de satisfaire cette contrainte sans remettre en cause la généralité de ce qui est présenté ici. Techniquement, il s'agira par exemple d'indiquer « l'égalité » entre des variables probabilistes par la définition de formes Diracs.

[3] PPP pour « Probabilistic Path Planning » ; [SO98] est également disponible, sous la forme d'un rapport technique du LAAS, à l'adresse http ://www.laas.fr/~jpl/book.html

carte, par la définition de termes comme $P(L_{t+\Delta t} \mid L_{t_{obj}})$, ou la définition directe de lois de contrôle $P(A \mid L_t\ L_{t_{obj}})$.

7.6.6 Questions « fil conducteur »

Q7. Comment apprendre expérimentalement les cartes ? Nous avons présenté Section 7.4 quelques exemples de *cartographie* que nous considérons dans ce travail (d'autres aspects seront traités aux Chapitres 8 et 9, dans le cadre d'assemblages de cartes). Dans ces exemples, nous avons présenté une méthode d'apprentissage de cartes « par étapes » : nous apprenons d'abord le modèle capteur, puis le modèle d'action, puis le modèle de transition.

Nous n'aborderons pas dans ce travail les constructions de cartes automatiques, ni les problèmes de construction de carte et de localisation simultanée [4]. Nous préférons en effet ici centrer notre approche sur les choix de modélisation, et le travail du programmeur. Nous pensons en effet que le choix d'une bonne représentation de l'espace *précède* la résolution des problèmes de perceptual aliasing et global connectivity éventuellement présents dans la carte apprise – surtout si l'on considère que la présence même de perceptual aliasing ou de global connectivity est un mauvais point quant à savoir si la représentation en question est « bonne » ou pas, dans le cadre de la tâche à résoudre.

Q7. Comment spécifier une carte ? Devant le grand éventail de possibilités que nous avons couvert dans ce chapitre, nous voyons notre formalisme comme un canevas de base. Le programmeur doit adapter ce canevas au problème particulier qu'il traite. Spécifier une carte va donc consister en un ensemble de choix, guidés par les nombreuses contraintes dont nous avons énuméré quelques éléments au Chapitre 5, pour trouver, face à cette complexité, une solution viable. De plus, souvenons nous que nous n'avons pas couvert la totalité des possibilités : par exemple, le nombre de questions possibles que nous avons donné ne tient compte que de quatre variables, alors que nous avons vu qu'elles étaient en pratique plus nombreuses (voir l'explosion des variables en leur conjonction, Section 7.6).

[4] En anglais, SLAM, pour *Simultaneous Localization And Mapping*.

Chapitre 8

Assemblages de cartes : superposition

> « Il nous faut remarquer d'emblée que la notion d'efficacité recouvre plusieurs significations. Il peut s'agir tout d'abord, d'une capacité de prédiction ou de rétrodiction. Une théorie mathématique est dite efficace dans un domaine des sciences si elle peut anticiper les résultats d'expérimentations ou reproduire les données obtenues précédemment. »
>
> D. Lambert, *Prédiction & probabilités dans les sciences*, E. Klein & Y. Sacquin éditeurs, Éditions Frontières, 1998 [Lambert98].

Dans les chapitres précédents, nous avons décrit un formalisme pour la définition de cartes probabilistes, nous l'avons illustré sur un exemple, et avons détaillé l'éventail de choix laissé au programmeur pour définir ces cartes bayésiennes. Dans ce chapitre et le suivant, nous définissons des outils qui permettent au programmeur d'assembler des cartes déjà existantes. Ces outils sont deux opérateurs d'assemblage, nommés *la superposition* et *l'abstraction* de cartes.

Intuitivement, le premier opérateur est un opérateur binaire qui assemble deux cartes en superposant les lieux des cartes sous-jacentes : ainsi le robot se localise dans les deux cartes simultanément, ce qui enrichit son vocabulaire de description de son interaction avec l'environnement. Le second opérateur, l'abstraction de cartes, sera détaillé Chapitre 9.

Puisque nous allons manipuler des variables de perception, d'action, et de lieu appartenant à des cartes différentes, précisons les notations que nous utiliserons. Lors de l'assemblage des deux cartes c^1, c^2 pour obtenir la carte c, les variables provenant de la carte c^1 (resp. c^2) auront un 1 (resp 2) en exposant : P^1, L^1_t, $L^1_{t'}$, A^1 (resp. P^2, L^2_t, $L^2_{t'}$, A^2). Les variables P, L_t, $L_{t'}$, et A, sans exposant, seront systématiquement les variables de la carte c obtenue par assemblage.

8.1 Superposition de cartes

Soient deux cartes bayésiennes c^1 et c^2, portant respectivement sur les variables P^1, L^1_t, $L^1_{t'}$, A^1 et P^2, L^2_t, $L^2_{t'}$, A^2, et répondant de façon pertinente aux trois questions de localisa-

tion $P(L_t^1 \mid P^1)$, de prédiction $P(L_{t'}^1 \mid A^1\ L_t^1)$, et de contrôle $P(A^1 \mid L_t^1\ L_{t'}^1)$ (respectivement $P(L_t^2 \mid P^2)$, $P(L_{t'}^2 \mid A^2\ L_t^2)$ et $P(A^2 \mid L_t^2\ L_{t'}^2)$).

Supposons que les cartes c^1 et c^2 couvrent (approximativement) le *même espace physique*. Les deux cartes savent donc décrire, mais en des termes différents (L_t^1 et L_t^2), cet espace. Elles savent aussi sélectionner des actions, dans les domaines des variables A^1 et A^2, pour accomplir des tâches.

Nous allons envisager deux variantes d'écriture de l'opérateur de superposition, qui diffèrent par le choix de la variable d'action A de la carte superposée. Dans le premier cas, nous supposons que nous ne savons pas assembler les actions qui proviennent des deux cartes : le robot peut appliquer un comportement qui provient soit de la première carte, soit de la seconde, mais ne sait pas comment les mélanger. Dans le second cas, nous supposerons le contraire, c'est-à-dire que le robot sait exécuter simultanément deux actions provenant des variables A^1 et A^2.

8.2 $A = A^1 \oplus A^2$

Dans cette première variante, nous supposons qu'il est impossible d'exécuter simultanément les actions des domaines de A^1 et A^2. En effet, dans le cas général, les actions sont des symboles, décrivant des tâches de plus bas niveau d'abstraction, et qui n'apportent aucune garantie, aucune propriété (de convergence, de visiter les différents états, etc.). Supposons par exemple que la variable A^1 contienne des actions de remontée et de descente d'un gradient lumineux, ou d'arrêt : $A^1 = \{Phototaxie, Photophobie, Stop\}$. Supposons que la variable A^2 contienne des actions liées à la présence d'un mur à proximité : $A^2 = \{SuiviGauche, SuiviDroit, Stop\}$. Dans ce cas, essayer de combiner la remontée du gradient et le suivi de mur n'a pas forcément de sens : ces deux comportements peuvent même être antagonistes, nous n'avons pas de garantie sur leur exécution simultanée. Il pourrait en résulter des comportements absurdes, comme tourner en rond. Nous choisissons donc de n'appliquer à un instant donné qu'une seule action, A^1 ou A^2. Dans ce cas, la variable d'action A est une nouvelle variable dont le domaine est l'union des domaines des variables A^1 et A^2 : $\mathcal{D}_A = \mathcal{D}_{A^1} \cup \mathcal{D}_{A^2}$. Nous notons $A = A^1 \oplus A^2$. Dans notre exemple, nous aurions : $A = A^1 \oplus A^2, \mathcal{D}_A = \{Phototaxie, Photophobie, Stop, SuiviGauche, SuiviDroit\}$.

8.2.1 $A = A^1 \oplus A^2$, sans ajout d'information

Nous choisissons ici de superposer les deux cartes c^1 et c^2 sans ajouter d'informations. Nous allons créer une nouvelle carte c ne contenant que des formes paramétriques issues de c^1, de c^2, et des distributions uniformes. Cela a l'avantage d'être simple à présenter, systématique d'écriture, et laisse la possibilité ensuite au programmeur d'enrichir la superposée c en ajoutant de l'information, possibilité que nous explorerons Section 8.2.2.

Nous commençons par décrire formellement la carte bayésienne superposée dans le cas général, puis présentons un exemple d'application. Enfin, nous discutons de cette méthode d'assemblage et des questions introduites Section 1.2.

8.2.1.1 Description

1. Spécification des connaissances préalables c
   (a) Variables :
   $\boldsymbol{P}$ $P = P^1 \wedge P^2, \mathcal{D}_P = \mathcal{D}_{P^1} \times \mathcal{D}_{P^2}, k_P = k_{P^1} k_{P^2}$,
   $\boldsymbol{L_t}$ $L_t = L_t^1 \wedge L_t^2, \mathcal{D}_{L_t} = \mathcal{D}_{L_t^1} \times \mathcal{D}_{L_t^2}, k_{L_t} = k_{L_t^1} k_{L_t^2}$,
   $\boldsymbol{L_{t'}}$ $L_{t'} = L_{t'}^1 \wedge L_{t'}^2, \mathcal{D}_{L_{t'}} = \mathcal{D}_{L_{t'}^1} \times \mathcal{D}_{L_{t'}^2}, k_{L_{t'}} = k_{L_{t'}^1} k_{L_{t'}^2}$,
   $\boldsymbol{A}$ $A = A^1 \oplus A^2$: $\mathcal{D}_A = \mathcal{D}_{A^1} \cup \mathcal{D}_{A^2}, k_A = k_{A^1} + k_{A^2} - k_{\mathcal{D}_{A^1} \cap \mathcal{D}_{A^2}}$ [1].
   (b) Décomposition : Nous définissons la décomposition :

$$\begin{aligned} & P(P\ L_t\ L_{t'}\ A) \\ = \ & P(P^1\ P^2\ L_t^1\ L_t^2\ L_{t'}^1\ L_{t'}^2\ A) \\ = \ & P(L_t^1) P(L_t^2) P(P^1 \mid L_t^1) P(P^2 \mid L_t^2) P(A) P(L_{t'}^1 \mid A\ L_t^1) P(L_{t'}^2 \mid A\ L_t^2). \end{aligned}$$

   À la manière d'une modélisation de type localisation markovienne, nous choisissons ici de particulariser les modèles capteurs directs ($P(P^1 \mid L_t^1)$ et $P(P^2 \mid L_t^2)$), et les modèles de transitions ($P(L_{t'}^1 \mid A\ L_t^1)$ et $P(L_{t'}^2 \mid A\ L_t^2)$).
   (c) Formes paramétriques :
   $P(L_t^1)$, $P(L_t^2)$ À ces deux termes sont associées des lois uniformes, car nous n'avons pas d'a priori sur la position du robot.
   $P(P^1 \mid L_t^1)$, $P(P^2 \mid L_t^2)$ Ces deux termes sont obtenus par des questions probabilistes aux cartes c^1 et c^2, respectivement.
   $P(A)$ À ce terme est associée une loi uniforme, car nous n'avons pas d'a priori sur le choix de l'action que doit faire le robot.
   $P(L_{t'}^1 \mid A\ L_t^1)$, $P(L_{t'}^2 \mid A\ L_t^2)$ Ces deux termes ne peuvent être directement tirés des cartes c^1 et c^2 sous-jacentes, en raison du domaine de valeur de A qui est plus grand que les domaines de A^1 et A^2. En revanche, pour les valeurs de A tirées du domaine de A^1, le premier terme pourra être tiré de la carte c^1 par la question probabiliste correspondante ; pour les valeurs de A hors du domaine de A^1, on définira ce terme par une uniforme (et vice-versa pour le second terme). Nous résumons ceci par la notation suivante :

$$P(L_{t'}^1 \mid [A = a]\ L_t^1) = \begin{cases} P(L_{t'}^1 \mid [A^1 = a]\ L_t^1\ c^1) \text{ si } a \in \mathcal{D}_{A^1} \\ \mathbf{U}_{k_{L_{t'}^1}}(L_{t'}^1) \text{ sinon} \end{cases}$$

$$P(L_{t'}^2 \mid [A = a]\ L_t^2) = \begin{cases} P(L_{t'}^2 \mid [A^2 = a]\ L_t^2\ c^2) \text{ si } a \in \mathcal{D}_{A^2} \\ \mathbf{U}_{k_{L_{t'}^2}}(L_{t'}^2) \text{ sinon} \end{cases}$$

2. Identification : Toutes les formes paramétriques sont déjà définies, il n'y a pas de phase d'identification.

[1] Notons ici que les deux variables A^1 et A^2 peuvent être totalement différentes ($\mathcal{D}_{A^1} \cap \mathcal{D}_{A^2} = \emptyset$), ou au contraire identiques ($\mathcal{D}_{A^1} = \mathcal{D}_{A^2}$) ; nous étudierons ces deux cas particuliers par la suite, Section 8.2.1.4.

8.2.1.2 Résolution des questions de localisation, prédiction et contrôle

Nous vérifions ici que la carte superposée c choisie répond bien de manière pertinente aux trois questions de localisation, prédiction et contrôle.

Commençons par la résolution de la question de localisation :

$$\begin{aligned}
P(L_t \mid P) &= P(L_t^1\ L_t^2 \mid P^1\ P^2) \\
&= \frac{1}{Z_1} \sum_{A\ L_{t'}^1\ L_{t'}^2} \left(\begin{array}{l} P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \\ P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2) \end{array} \right) \\
&= \frac{1}{Z_1} \left(\begin{array}{l} P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \\ \sum_{A\ L_{t'}^1\ L_{t'}^2} P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2) \end{array} \right) \\
&= \frac{1}{Z_1} \left(\begin{array}{l} P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \\ \sum_A \left(P(A) \sum_{L_{t'}^1} \left(P(L_{t'}^1 \mid A\ L_t^1) \sum_{L_{t'}^2} P(L_{t'}^2 \mid A\ L_t^2) \right) \right) \end{array} \right) \\
&= \frac{1}{Z_1} P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \\
P(L_t \mid P) &= \frac{1}{Z_1'} P(P^1 \mid L_t^1)P(P^2 \mid L_t^2).
\end{aligned}$$

Nous remarquons que la question de localisation de la carte superposée ne fait intervenir (à une renormalisation près) que les résolutions des questions de localisation dans les deux cartes c^1 et c^2. Passons à la question de prédiction :

$$\begin{aligned}
P(L_{t'} \mid A\ L_t) &= P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2) \\
&= \frac{1}{Z_2} \sum_{P^1\ P^2} \left(\begin{array}{l} P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \\ P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2) \end{array} \right) \\
&= \frac{1}{Z_2} \left(\begin{array}{l} P(L_t^1)P(L_t^2)P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2) \\ \sum_{P^1\ P^2} P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \end{array} \right) \\
&= \frac{1}{Z_2} P(L_t^1)P(L_t^2)P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2) \\
P(L_{t'} \mid A\ L_t) &= \frac{1}{Z_2'} P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2).
\end{aligned}$$

Une fois de plus, cette question se résout simplement en posant les questions de prédiction aux deux cartes sous-jacentes, puis en renormalisant. Terminons par la question de

contrôle :

$$\begin{aligned}
P(A \mid L_t\ L_{t'}) &= P(A \mid L_t^1\ L_t^2\ L_{t'}^1\ L_{t'}^2) \\
&= \frac{1}{Z_3} \sum_{P^1\ P^2} \begin{pmatrix} P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \\ P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2) \end{pmatrix} \\
&= \frac{1}{Z_3} \begin{pmatrix} P(L_t^1)P(L_t^2)P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2) \\ \sum_{P^1\ P^2} P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \end{pmatrix} \\
&= \frac{1}{Z_3} P(L_t^1)P(L_t^2)P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2) \\
P(A \mid L_t\ L_{t'}) &= \frac{1}{Z_3'} P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2).
\end{aligned}$$

Notons que, bien que la résolution symbolique soit identique à la question de prédiction, les formes résultantes ne seront pas les mêmes, car nous utilisons ici un raccourci d'écriture. En effet, dans la question $P(A \mid L_t\ L_{t'})$, il faut se souvenir que les variables en partie droite ont des valeurs connues. Cette question et sa résolution s'écrivent donc :

$$\begin{aligned}
&P(A \mid [L_t = l_t]\ [L_{t'} = l_{t'}]) \\
&\quad = P(A \mid [L_t^1 = l_t^1]\ [L_t^2 = l_t^2]\ [L_{t'}^1 = l_{t'}^1]\ [L_{t'}^2 = l_{t'}^2]) \\
&\quad = \frac{1}{Z_3'} P([L_{t'}^1 = l_{t'}^1] \mid A\ [L_t^1 = l_t^1]) P([L_{t'}^2 = l_{t'}^2] \mid A\ [L_t^2 = l_t^2]).
\end{aligned}$$

Comparons avec la question de prédiction :

$$P(L_{t'} \mid [A = a]\ [L_t = l_t]) = \frac{1}{Z_2'} P(L_{t'}^1 \mid [A = a]\ [L_t^1 = l_t^1]) P(L_{t'}^2 \mid [A = a]\ [L_t^2 = l_t^2]).$$

Ces deux questions sont en réalité des « coupes » différentes dans la distribution conjointe $P(A\ L_t\ L_{t'})$. En effet le terme $P(L_{t'} \mid A\ L_t)$ calcule cette conjointe, puis la « coupe » en fixant les valeur de A et L_t pour considérer une distribution sur la variable $L_{t'}$. Le résultat sera donc mathématiquement différent du calcul pour le terme $P(A \mid L_t\ L_{t'})$ qui lui, fixe les variables L_t et $L_{t'}$ pour obtenir une distribution sur A.

Nous avions fait l'hypothèse jusqu'ici de cartes sous-jacentes apportant effectivement suffisamment d'information dans leurs questions de localisation, prédiction et contrôle. Les résolutions de ces mêmes questions dans la carte superposée se réduisent toutes à questionner les cartes sous-jacentes (grâce à notre choix de décomposition). La carte superposée disposera donc elle aussi de suffisamment de signal sur ces questions [2], et répondra à nos critères de définition d'une carte bayésienne. Nous pouvons résumer la méthode de superposition de cartes, dans le cas $A = A^1 \oplus A^2$, sans ajout d'information, Figure 8.1.

[2] Pas formellement en réalité. Des contre-exemples existent où la résolution de ces questions sur la carte superposée peuvent avoir une entropie arbitrairement proche du maximum. Mais ces cas, en pratique, ne se rencontrent jamais.

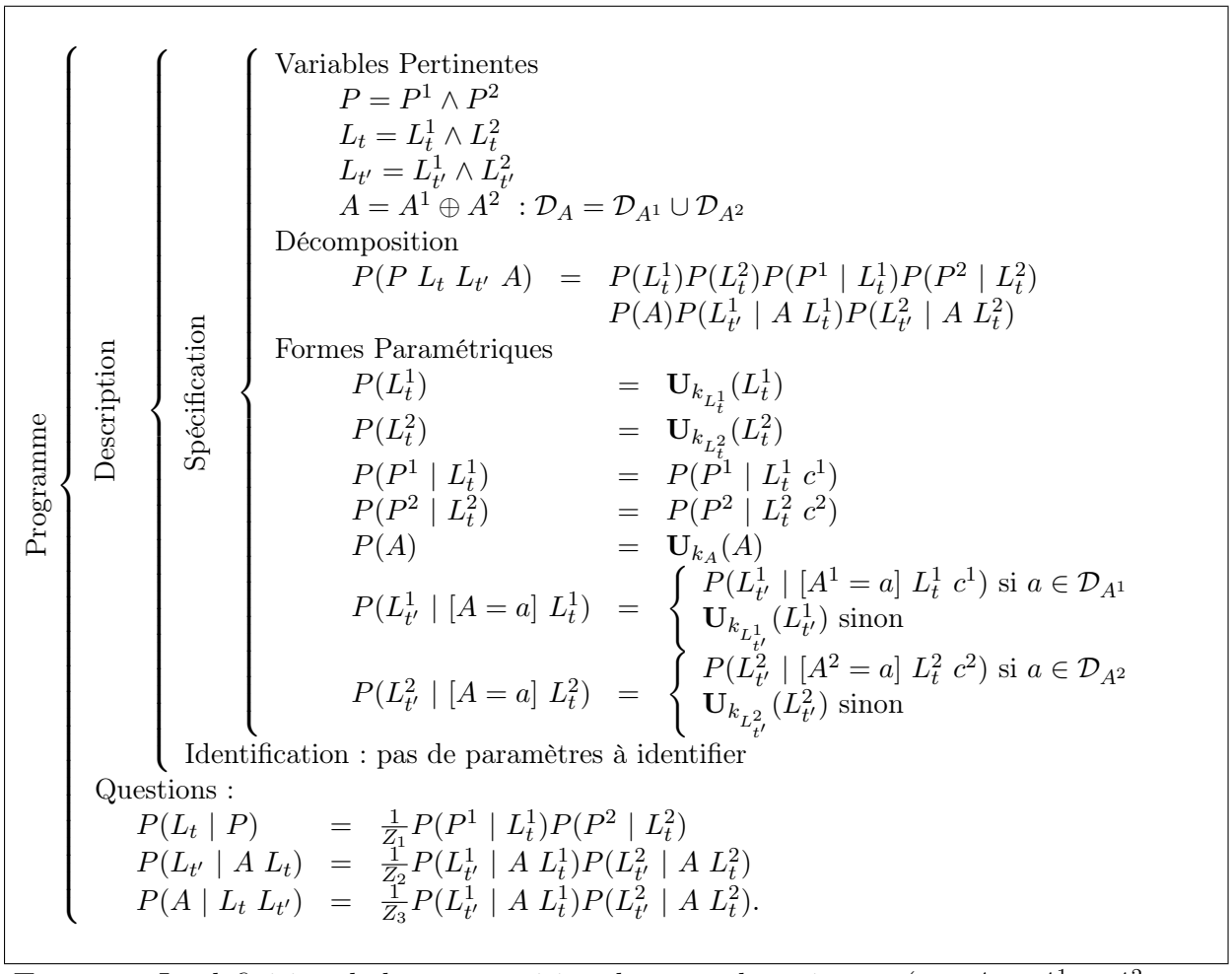


Fig. 8.1: La définition de la superposition de cartes bayésiennes (cas $A = A^1 \oplus A^2$, sans ajout d'information).

8.2.1.3 Expérience de superposition : cartes d'odeurs et de proximétrie

Nous développons maintenant un exemple d'application de l'opérateur de superposition de cartes. Dans cette expérience, nous disposons de deux cartes, l'une basée sur un gradient d'odeurs, l'autre basée sur les informations de proximétrie, qui décrit l'environnement en termes de « murs », de « coins », et « d'espace libre ». Avant de présenter la carte superposée obtenue, nous résumons brièvement la carte basée sur les odeurs, référons le lecteur à l'Annexe C pour la définition complète de cette carte, et rappelons que la définition de la carte de proximétrie se trouve Section 6.3.

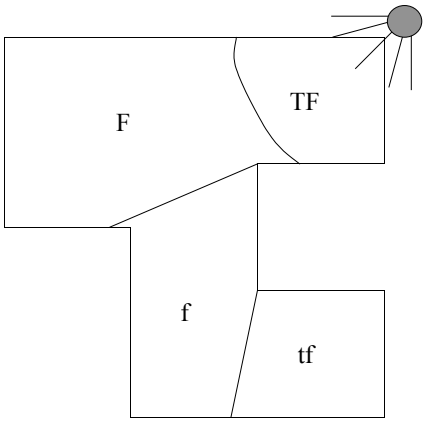
Résumé de la carte d'odeurs c^{odeur} Dans cet exemple, le robot utilise son capteur d'odeur pour discrétiser l'environnement en zones grossières d'intensités odorantes égales.

Il s'agit, dans le « simulateur de nez », de fournir au robot un capteur supplémentaire, lui permettant de détecter la position de sources odorantes virtuelles. Pour ce faire, le simu-

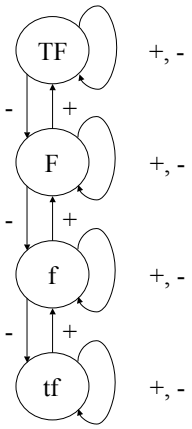
lateur dispose d'un plan de l'environnement du robot, fourni par le programmeur. Celui-ci place ensuite dans l'environnement simulé des sources « odorantes », et le simulateur gère la propagation des odeurs virtuelles. Finalement, le programmeur place le robot dans l'environnement virtuel, et le suivi de la position du robot se fait par calcul odométrique. Étant donné la position calculée du robot et les positions des sources odorantes, le simulateur fournit au robot le gisement α de la source la plus proche, et l'intensité d'odeur *Odeur* correspondante.

La variable de lieu que nous choisissons est une discrétisation grossière de l'intensité d'odeur perçue, en quatre valeurs « TF », « F », « f » et « tf » (pour Très Fort, Fort, faible et très faible, respectivement). Nous montrons Figure 8.2(a) une interprétation géométrique de ces lieux. Contrairement à la carte de proximétrie, pour l'environnement de la Figure 8.2(a), chaque lieu de cette carte a une « unité géographique », autrement dit chaque lieu correspond à un ensemble de points connexes.

Les actions pertinentes dans ce cadre seront soit de remonter vers une zone de plus forte intensité (*chimiotropisme* positif, qui sera noté « + » par la suite), soit au contraire de s'éloigner de la source d'odeur (chimiotropisme négatif, noté comportement « - »), soit enfin de rester dans une zone d'intensité donnée (« = », patrouille aléatoire dans la zone, ou même plus simplement, arrêt). Ces valeurs constituent la variable d'action A_{odeur}.



(a) Interprétation des zones affectées aux valeurs « TF », « F », « f » et « tf ».



(b) Le graphe pour la carte d'une pièce basée sur une odeur.

FIG. 8.2: Résumé de la carte c^{odeur}.

Nous montrons Figure 8.2(b) le graphe induit par la carte d'une pièce basée sur l'odeur. Nous ne faisons pas apparaître tous les arcs, notamment pas ceux étiquetés par le comportement de patrouille dans une zone, afin de ne pas alourdir la figure.

La définition complète de la carte c^{odeur} se trouve Annexe C.

Objectifs et protocole expérimental

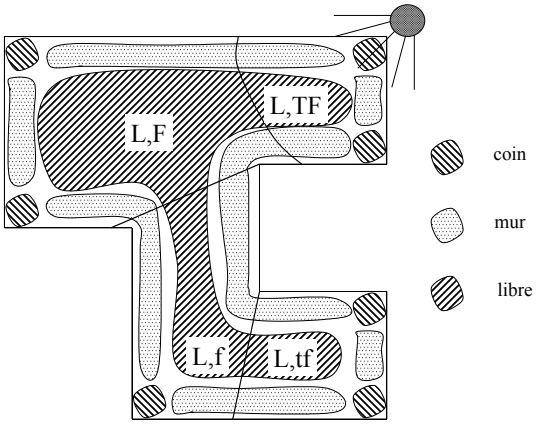


FIG. 8.3: Interprétation géographique de la superposition des cartes d'odeurs et de proximétrie.

Choix d'une variable de lieux Nous disposons d'une carte de la pièce basée sur les proximètres, et d'une autre basée sur le capteur d'odeurs. La première, c^{prox}, décrit le monde en des termes liés à la présence des murs, par la variable Sit_t, l'autre, c^{odeur}, en des termes liés à la distance de la source d'odeur, par la variable Od_t. La carte de superposition c a donc comme variable de lieu L_t, qui est la *conjonction* des variables Sit_t et Od_t. Il s'agit donc de se localiser à la fois sur les deux cartes sous-jacentes, ce qui nous fournira une richesse de description du monde plus grande (L_t a pour cardinal le produit des cardinaux de Sit_t et Od_t). Nous montrons Figure 8.3 une interprétation géographique des lieux représentés par cette nouvelle variable $L_t = Sit_t \wedge Od_t$.

Graphe induit Le graphe induit par la carte superposée, traduit graphiquement le modèle de transition $P(L_{t'} \mid A\ L_t)$, donc dans notre exemple, $P(Sit_{t+\Delta t}\ Od_{t+\Delta t} \mid A\ Sit_t\ Od_t)$. Puisque nous supposons l'indépendance des cartes dans la carte superposée, ce terme n'apparaîtra pas tel quel, mais devra être calculé à partir des termes $P(Sit_{t+\Delta t} \mid A\ Sit_t\ c^{prox})$ et $P(Od_{t+\Delta t} \mid A\ Od_t\ c^{odeur})$. La résolution est (voir plus loin, page 99) :

$$P(Sit_{t+\Delta t}\ Od_{t+\Delta t} \mid A\ Sit_t\ Od_t) = \frac{1}{Z} P(Sit_{t+\Delta t} \mid A\ Sit_t) P(Od_{t+\Delta t} \mid A\ Od_t).$$

Prenons l'action de remontée de gradient d'odeur (chimiotropie positive) pour exemple : $[A = +]$. Le graphe traduisant le terme $P(Od_{t+\Delta t} \mid [A = +]\ Od_t)$ se trouve Figure C.2, Annexe C. La loi de probabilités associée est définie Table 8.1. Pour l'action $[A = +]$ en revanche, le robot ne saura rien prédire sur les variations de la variable Sit_t : le terme $P(Sit_{t+\Delta t} \mid [A = +]\ Sit_t)$ est donc une uniforme, présentée Table 8.1.

Multiplions maintenant les deux termes $P(Od_{t+\Delta t} \mid [A = +]\ Od_t)$ et $P(Sit_{t+\Delta t} \mid [A = +]\ [Sit_t = s])$: le résultat est présenté Table 8.2. Les régularités observées résultent de

$P(Sit_{t+\Delta t} \mid [A = +]\ Sit_t\ c^{prox})$			
$Sit_t \rightarrow$ $\downarrow\ Sit_{t+\Delta t}$	C	M	L
C	1/3	1/3	1/3
M	1/3	1/3	1/3
L	1/3	1/3	1/3

$P(Od_{t+\Delta t} \mid [A = +]\ Od_t\ c^{odeur})$				
$Od_t \rightarrow$ $\downarrow\ Od_{t+\Delta t}$	TF	F	f	tf
TF	1	1/2	0	0
F	0	1/2	1/2	0
f	0	0	1/2	1/2
tf	0	0	0	1/2

TAB. 8.1 – Les termes $P(Sit_{t+\Delta t} \mid [A = +]\ Sit_t\ c^{prox})$ et $P(Od_{t+\Delta t} \mid [A = +]\ Od_t\ c^{odeur})$. Une colonne représente une loi de probabilité sur la variable en partie gauche, et donc somme à 1. Pour les variables Sit_t et $Sit_{t+\Delta t}$, C, M et L dénotent les valeurs *coin*, *mur* et *libre*, respectivement.

$P(Sit_{t+\Delta t}\ Od_{t+\Delta t} \mid [A = +]\ Sit_t\ Od_t)$												
$\langle Sit_t, Od_t\rangle \rightarrow$ $\downarrow\ \langle Sit_{t+\Delta t}, Od_{t+\Delta t}\rangle$	$\langle C, TF\rangle$	$\langle C, F\rangle$	$\langle C, f\rangle$	$\langle C, tf\rangle$	$\langle M, TF\rangle$	$\langle M, F\rangle$	$\langle M, f\rangle$	$\langle M, tf\rangle$	$\langle L, TF\rangle$	$\langle L, F\rangle$	$\langle L, f\rangle$	$\langle L, tf\rangle$
$\langle C, TF\rangle$	1/3	1/6	0	0	1/3	1/6	0	0	1/3	1/6	0	0
$\langle C, F\rangle$	0	1/6	1/6	0	0	1/6	1/6	0	0	1/6	1/6	0
$\langle C, f\rangle$	0	0	1/6	1/6	0	0	1/6	1/6	0	0	1/6	1/6
$\langle C, tf\rangle$	0	0	0	1/6	0	0	0	1/6	0	0	0	1/6
$\langle M, TF\rangle$	1/3	1/6	0	0	1/3	1/6	0	0	1/3	1/6	0	0
$\langle M, F\rangle$	0	1/6	1/6	0	0	1/6	1/6	0	0	1/6	1/6	0
$\langle M, f\rangle$	0	0	1/6	1/6	0	0	1/6	1/6	0	0	1/6	1/6
$\langle M, tf\rangle$	0	0	0	1/6	0	0	0	1/6	0	0	0	1/6
$\langle L, TF\rangle$	1/3	1/6	0	0	1/3	1/6	0	0	1/3	1/6	0	0
$\langle L, F\rangle$	0	1/6	1/6	0	0	1/6	1/6	0	0	1/6	1/6	0
$\langle L, f\rangle$	0	0	1/6	1/6	0	0	1/6	1/6	0	0	1/6	1/6
$\langle L, tf\rangle$	0	0	0	1/6	0	0	0	1/6	0	0	0	1/6

TAB. 8.2 – Le terme $P(Sit_{t+\Delta t}\ Od_{t+\Delta t} \mid [A = +]\ Sit_t\ Od_t)$. Une colonne représente une loi de probabilité sur la variable conjointe $Sit_{t+\Delta t} \wedge Od_{t+\Delta t}$, et donc somme à 1. Pour les variables Sit_t et $Sit_{t+\Delta t}$, C, M et L dénotent les valeurs *coin*, *mur* et *libre*, respectivement.

l'hypothèse d'indépendance conditionnelle entre $Od_{t+\Delta t}$ et $Sit_{t+\Delta t}$. De ce tableau, nous pouvons tirer le graphe induit par la carte superposée des cartes de proximétrie et d'odeur, en établissant les douze nœuds correspondant aux douze valeurs possibles pour $Sit_{t+\Delta t} \wedge Od_{t+\Delta t}$, et en dessinant un arc entre deux nœuds lorsque la Table 8.2 indique une probabilité différente de 0 ; un extrait du graphe résultat est montré Figure 8.4.

Description

1. Spécification des connaissances préalables c
   (a) Variables :

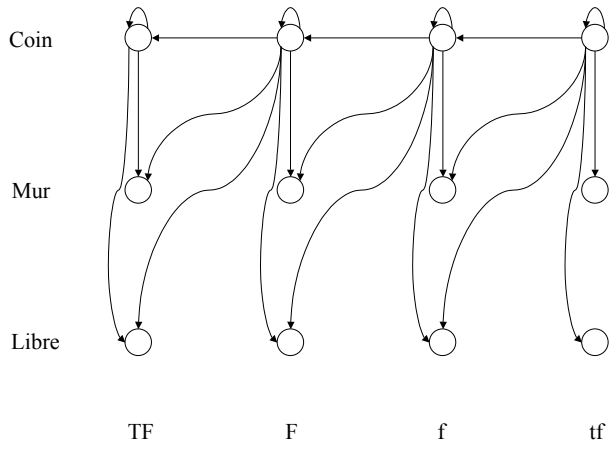


FIG. 8.4: Le graphe pour la carte issue de la superposition des cartes de proximétrie et d'odeur d'une pièce, dans le cas $A = A_{prox} \oplus A_{odeur}$, sans ajout d'informations. Sur cette figure n'apparaissent que les arcs issus des quatre nœuds $\langle coin, TF\rangle$, $\langle coin, F\rangle$, $\langle coin, f\rangle$, et $\langle coin, tf\rangle$.

P Les variables de perceptions sont celles issues des deux cartes sous-jacentes : les seize proximètres, $Px \ : \mathcal{D}_{Px} = \{0, 1, \ldots, 15\}^{16}, k_{Px} = 16^{16}$, et les indications d'odeurs, $\alpha \ : \mathcal{D}_{\alpha} = \{-180, -170, \ldots, 170\}, k_{\alpha} = 36$, $Odeur \ : \mathcal{D}_{Odeur} = \{0, 1, \ldots, 15\}, k_{Odeur} = 16$. On a donc $P = Px,\ \alpha,\ Odeur$;

L_t $L_t = Sit_t \wedge Od_t$:

$$\begin{aligned} \mathcal{D}_{L_t} &= \mathcal{D}_{Sit_t} \times \mathcal{D}_{Od_t} = \{\langle coin, TF\rangle, \langle coin, F\rangle, \ldots, \langle libre, tf\rangle\}, \\ k_{L_t} &= k_{Sit_t} * k_{Od_t} = 12, \end{aligned}$$

L_t' $L_{t+\Delta t} = Sit_{t+\Delta t} \wedge Od_{t+\Delta t}$:

$$\begin{aligned} \mathcal{D}_{L_{t+\Delta t}} &= \mathcal{D}_{Sit_{t+\Delta t}} \times \mathcal{D}_{Od_{t+\Delta t}} = \{\langle coin, TF\rangle, \langle coin, F\rangle, \ldots, \langle libre, tf\rangle\}, \\ k_{L_{t+\Delta t}} &= k_{Sit_{t+\Delta t}} * k_{Od_{t+\Delta t}} = 12, \end{aligned}$$

A $A = A_{prox} \oplus A_{odeur}$:

$$\begin{aligned} \mathcal{D}_A &= \mathcal{D}_{A_{prox}} \cup \mathcal{D}_{A_{odeur}} \\ &= \{Stop,\ ToutD,\ SuiviD,\ \acute{E}loignerMur,\ PasserD\} \cup \{+, -, =\}, \\ k_A &= k_{A_{prox}} + k_{A_{odeur}} = 9, \end{aligned}$$

puisque les domaines de A_{prox} et A_{odeur} sont disjoints.

(b) Décomposition : Nous instancions ici la définition formelle, qui se trouve Sec-

tion 8.2.1.1, en remplaçant P^1 par Px, P^2 par α, $Odeur$, etc. :

$$\begin{aligned}
&P(P\ L_t\ L_{t'}\ A)\\
&\quad = P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2)P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2)\\
&\quad = P(Sit_t)P(Od_t)P(Px \mid Sit_t)P(\alpha\ Odeur \mid Od_t)\\
&\qquad P(A)P(Sit_{t+\Delta t} \mid A\ Sit_t)P(Od_{t+\Delta t} \mid A\ Od_t).
\end{aligned}$$

(c) Formes paramétriques : Conformément à la définition formelle :

$P(Sit_t)$, $P(Od_t)$ À ces deux termes sont associées des lois uniformes.

$P(Px \mid Sit_t)$, $P(\alpha\ Odeur \mid Od_t)$ Ces deux termes sont obtenus par des questions probabilistes aux cartes c^{prox} et c^{odeur}, respectivement.

$P(A)$ À ce terme est associée une loi uniforme.

$P(Sit_{t+\Delta t} \mid A\ Sit_t)$, $P(Od_{t+\Delta t} \mid A\ Od_t)$

$$\begin{aligned}
&P(Sit_{t+\Delta t} \mid [A = a]\ Sit_t)\\
&\quad = \begin{cases} P(Sit_{t+\Delta t} \mid [A_{prox} = a]\ Sit_t\ c^{prox}) \text{ si } a \in \mathcal{D}_{A_{prox}} \\ \mathbf{U}_{k_{Sit_{t+\Delta t}}}(Sit_{t+\Delta t}) \text{ sinon} \end{cases}\\
&P(Od_{t+\Delta t} \mid [A = a]\ Od_t)\\
&\quad = \begin{cases} P(Od_{t+\Delta t} \mid [A_{odeur} = a]\ Od_t\ c^{odeur}) \text{ si } a \in \mathcal{D}_{A_{odeur}} \\ \mathbf{U}_{k_{Od_{t+\Delta t}}}(Od_{t+\Delta t}) \text{ sinon} \end{cases}
\end{aligned}$$

2. Identification : Toutes les formes paramétriques sont définies, pas de phase d'identification.

Utilisation

Programme et question probabiliste Rappelons ici que les trois questions de localisation, prédiction, et contrôle se résolvent par :

$$\begin{aligned}
P(L_t \mid P) &= \frac{1}{Z_1}P(P^1 \mid L_t^1)P(P^2 \mid L_t^2)\\
&= \frac{1}{Z_1}P(Px \mid Sit_t)P(\alpha\ Odeur \mid Od_t),\\
P(L_{t'} \mid A\ L_t) &= \frac{1}{Z_2}P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2)\\
&= \frac{1}{Z_2}P(Sit_{t+\Delta t} \mid A\ Sit_t)P(Od_{t+\Delta t} \mid A\ Od_t),\\
P(A \mid L_t\ L_{t'}) &= \frac{1}{Z_3}P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2)\\
&= \frac{1}{Z_3}P(Sit_{t+\Delta t} \mid A\ Sit_t)P(Od_{t+\Delta t} \mid A\ Od_t).
\end{aligned}$$

Résultats Par manque de temps, cette expérience n'a pas encore été implantée telle quelle sur le robot. Nous avons tout de même souhaité la présenter car elle apporte beaucoup sur la compréhension intuitive de l'opérateur de superposition. Le choix même de ce terme de « superposition » devrait maintenant apparaître plus clair au lecteur, au vu de la Figure 8.3.

Rappelons que cette version de superposition, sans ajout d'information, ne fait que rassembler dans une même connaissance c deux cartes sous-jacentes, et qu'une telle manipulation, qui n'est que symbolique, est toujours possible. Elle est tout de même nécessaire, et servira de base par la suite, lorsque nous décrirons Section 8.2.2 les manières de complexifier la carte c que nous obtiendrions par apprentissage.

8.2.1.4 Discussion

Commençons par quelques remarques, avant d'analyser ce que ce schéma de superposition nous apprend sur les réponses possibles aux questions énoncées Section 1.2.

Nous avons vu dans cette section la création d'une nouvelle variable par l'opérateur $\oplus$, qui fait l'union des domaines de deux variables A^1 et A^2. Formellement, il existe deux cas extrêmes d'application de cet opérateur : les cas où A^1 et A^2 sont totalement différentes ($\mathcal{D}_{A^1} \cap \mathcal{D}_{A^2} = \emptyset$), ou au contraire identiques ($\mathcal{D}_{A^1} = \mathcal{D}_{A^2}$).

$\mathcal{D}_{A^1} \cap \mathcal{D}_{A^2} = \emptyset$ Le cas où les domaines des variables sont disjointes a été développé par l'exemple Section 8.2.1.3. Nous avons déjà noté que l'assemblage de deux cartes de cette manière était toujours possible, grâce à un opérateur purement syntaxique.

$\mathcal{D}_{A^1} = \mathcal{D}_{A^2}$ Le cas où les deux variables d'actions sont identiques nous ramène en réalité à une extension de la localisation markovienne, en vertu de notre choix de décomposition. En effet, dans ce cas, $A = A^1 \oplus A^2 = A^1$, et la décomposition devient :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(P^1\ P^2\ L_t^1\ L_t^2\ L_{t'}^1\ L_{t'}^2\ A^1) \\ &= P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \\ &\quad P(A^1)P(L_{t'}^1 \mid A^1\ L_t^1)P(L_{t'}^2 \mid A^1\ L_t^2). \end{aligned}$$

Nous nous retrouvons donc dans un cas de mise « côte à côte » des modèles de transitions, la variable commune étant A^1. Nous verrons un autre exemple de cette forme particulière de combinaison de termes probabilistes Section 8.2.2.4, où les variables communes seront L_t et $L_{t'}$.

Cependant, le cas général est celui où l'intersection n'est ni vide, ni égale à l'une des variables. En effet en pratique, les variables d'actions ont souvent au moins une action commune, comme par exemple le comportement *Stop*, qui consiste simplement à arrêter le robot.

Dans notre exemple Section 8.2.1.3, nous avons supposé les domaines de A_{prox} et A_{odeur} disjoints. Ainsi, les deux termes $P(Sit_{t+\Delta t} \mid [A = a]\ Sit_t)$ et $P(Od_{t+\Delta t} \mid [A = a]\ Od_t)$ n'étaient jamais simultanément des questions probabilistes. Au moins l'un des deux était une uniforme. Supposons maintenant que le comportement « = » de la carte c^{odeur} s'implémente par un arrêt du robot. Dans ce cas, ce comportement peut se renommer *Stop* :

A_{prox} et A_{odeur} ont maintenant cette action en commun, et les deux cartes c^1 et c^2 peuvent être questionnées simultanément :

$$\begin{aligned} P(Sit_{t+\Delta t} \mid [A = Stop]\ Sit_t) &= P(Sit_{t+\Delta t} \mid [A_{prox} = Stop]\ Sit_t\ c^{prox}) \\ P(Od_{t+\Delta t} \mid [A = Stop]\ Od_t) &= P(Od_{t+\Delta t} \mid [A_{odeur} = Stop]\ Od_t\ c^{odeur}) \end{aligned}$$

Nous comprenons que, dans ce cas, cet opérateur de superposition, même simple, permet d'ajouter de la richesse d'utilisation aux cartes. En effet dans ce cas, poser une question qui fait intervenir $[A = Stop]$ à la carte superposée c engendre une réponse différente que ce que donne chacune des cartes isolément.

Examinons maintenant certaines des questions définies au début de ce document.

Q1. Qu'est-ce qu'une carte ? Nous avons défini dans cette Section un opérateur pour créer une nouvelle carte, à partir de deux cartes sous-jacentes, et nous avons vérifié que la carte obtenue répond à notre définition d'une carte bayésienne. Les cartes peuvent donc s'obtenir de manière progressive, au fur et à mesure que le programmeur combine des cartes de base entre elles.

Q7. Comment assembler les cartes ? Nous avons vu dans cette première méthode de superposition un opérateur systématique, car purement syntaxique, d'assemblage de cartes. Bien que cet opérateur n'ajoute aucune connaissance à la carte superposée, du point de vue des formes mathématiques, le fait même de regrouper des informations venant de deux cartes sous-jacentes est riche. En effet, une question probabiliste comme la localisation $P(L_t^1\ L_t^2 \mid P^1\ P^2)$ ne prend un sens que dans le cadre de la carte superposée c, qui définit la conjointe $P(P^1\ P^2\ L_t^1\ L_t^2\ L_{t'}^1\ L_{t'}^2\ A)$. Les questions que l'on peut poser à c sont autant de ressources utilisables par ailleurs (par d'autres programmes probabilistes, ou même pour le cas de la question de localisation ci-dessus, par un programme d'affichage graphique).

Q2. Qu'est-ce qu'un lieu, pour le robot ? Dans la carte superposée, la variable de lieu L_t est la conjonction des variables de lieux L_t^1 et L_t^2 des deux cartes c^1 et c^2. Le cardinal de L_t est donc plus grand que les cardinaux de L_t^1 et L_t^2 pris séparément. La superposition est donc un moyen d'enrichir les lieux que connaît le robot. Son vocabulaire de description interne des phénomènes s'élargit, ce qui permet au robot d'être plus précis (d'être moins soumis au perceptual aliasing). Par exemple, dans notre superposition de cartes de proximétrie et d'odeur, nous voyons Figure 8.3 que la zone nommée $\langle L, TF\rangle$ couvre un espace plus petit que la zone L de la carte de proximétrie (Figure 6.1(a)), et que la zone TF de la carte d'odeur (Figure 8.2(a)).

Ce principe de superposition n'est pas nouveau, et est bien connu des navigateurs. Lors d'un repérage d'amers terrestre, plusieurs amers (en général trois) sont nécessaires avant d'obtenir une localisation suffisamment précise. En effet, chaque amer – chaque carte – définit un secteur angulaire dans lequel peut se trouver le bateau (en réalité relever une demi-droite suffit, mais nous supposons ici que nous visualisons aussi les incertitudes sur l'angle de cette demi-droite). Le recoupement de ces zones par superposition permet de

réduire la zone de l'espace possible pour le bateau. L'analogie avec notre opérateur se poursuit lorsqu'on considère que le nombre d'amers nécessaire pour obtenir une précision donnée dépend de l'orthogonalité des zones angulaires d'une part, et de leurs incertitudes d'autre part. Dans notre cas de superposition de cartes, « l'orthogonalité » de deux cartes ne prend pas un sens physique (géométrique), mais un sens informationnel. Deux cartes sont « orthogonales » lorsque leur superposition amène réellement des informations supplémentaires sur l'environnement. Comme contre-exemple, imaginons la superposition d'une carte avec elle-même : bien que formellement, le nombre de lieux se trouve augmenté ($k_{L_t} = k^2_{L^1_t}$), en pratique le gain d'information est nul. Nous explorons dans la Section suivante des méthodes d'apprentissage qui permettent de reconnaître ces deux cas extrêmes de gain d'information minimum ou maximum.

8.2.2 $A = A^1 \oplus A^2$, avec ajout d'information

Dans cette Section, nous examinons les manières dont la superposée de deux cartes peut s'enrichir par l'expérimentation. Nous commençons d'ailleurs par le cas simple d'identification de paramètres de formes présentes dans la décomposition choisie, puis présentons une décomposition plus complexe laissant une plus large place à l'apprentissage.

8.2.2.1 Description

Dans cette section, nous présentons progressivement les enrichissements que nous apportons à la décomposition, en vue de l'ajout d'apprentissage. Rappelons la décomposition choisie Section 8.2.1.1 :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(L^1_t)P(L^2_t)P(P^1 \mid L^1_t)P(P^2 \mid L^2_t) \\ &\quad P(A)P(L^1_{t'} \mid A\ L^1_t)P(L^2_{t'} \mid A\ L^2_t). \end{aligned}$$

Première hypothèse levée Dans le cadre de cette description, deux termes peuvent en fait être identifiés expérimentalement. Il s'agit des termes $P(L^1_{t'} \mid A\ L^1_t)$ et $P(L^2_{t'} \mid A\ L^2_t)$. $P(L^1_{t'} \mid A\ L^1_t)$ est en effet défini par une question à c^1 lorsque A est une action du domaine de A^1. Lorsque A est une action du domaine de A^2, nous avons jusqu'à maintenant supposé une ignorance complète, traduite par une uniforme.

Or, il est en pratique peu probable que les variations de la variable de lieu de la carte c^1 soient indépendantes du choix d'une action dans la carte c^2. Dans notre exemple de superposition des cartes de proximétrie et d'odeurs d'une pièce, l'application du comportement de chimiotropie positive (remontée vers la source) va certainement amener le robot dans un coin, car la source odorante s'y trouve. Le terme $P(Sit_{t+\Delta t} \mid [A = +]\ Sit_t\ c^1)$ n'est donc pas uniforme, et peut être appris expérimentalement, sur la base d'une loi de succession de Laplace. Cette forme paramétrique est en effet adéquate si nous supposons que les variables de lieux des deux cartes sont discrètes et sans relation de distance entre leurs valeurs, ce qui est le cas dans nos exemples.

Nous redéfinissons donc formellement les deux formes paramétriques, en utilisant notre notation **L** pour les lois de succession de Laplace (voir Section 2.2.3.2) :

$$P(L^1_{t'} \mid [A = a]\ L^1_t) = \begin{cases} P(L^1_{t'} \mid [A^1 = a]\ L^1_t\ c^1) \text{ si } a \in \mathcal{D}_{A^1} \\ \mathbf{L}_{n_1,\ldots,n_k}{}_{L^1_{t'} *_k L^1_t}(L^1_{t'}) \text{ sinon} \end{cases}$$

$$P(L^2_{t'} \mid [A = a]\ L^2_t) = \begin{cases} P(L^2_{t'} \mid [A^2 = a]\ L^2_t\ c^2) \text{ si } a \in \mathcal{D}_{A^2} \\ \mathbf{L}_{n_1,\ldots,n_k}{}_{L^2_{t'} *_k L^2_t}(L^2_{t'}) \text{ sinon} \end{cases}$$

Expérimentalement, pour identifier la première loi de succession de Laplace ci-dessus, il s'agira d'appliquer les différents comportements issus de la carte c^2 tout en observant les variations de L^1_t. Nous accumulerons ainsi des données expérimentales de la forme $\{\langle l^1_{t'}, a, l^1_t\rangle\}_{i=1}^n$, avec $a \notin \mathcal{D}_{A^1}$, qui permettent d'identifier la loi de succession de Laplace $P(L^1_{t'} \mid [A = a]\ [L^1_t = l^1_t])$. Bien sûr, la démarche est symétrique pour l'identification de $P(L^2_{t'} \mid [A = a]\ L^2_t), a \notin \mathcal{D}_{A^2}$.

Seconde hypothèse levée À la lumière de ce qui vient d'être présenté, nous comprenons que de nombreuses hypothèses d'indépendance conditionnelles que nous avions faites ont de fortes chances de se révéler fausses en pratique. Il devient donc intéressant de lever certaines de ces hypothèses, et de vérifier expérimentalement, par apprentissage, les nouvelles dépendances.

Par exemple, levons l'hypothèse d'indépendance entre P^1 et L^2_t (conditionnellement à la connaissance de L^1_t), et celle entre P^2 et L^1_t (conditionnellement à la connaissance de L^2_t). Cela revient à remplacer dans la décomposition les termes $P(P^1 \mid L^1_t)$ et $P(P^2 \mid L^2_t)$ respectivement par $P(P^1 \mid L^1_t\ L^2_t)$ et $P(P^2 \mid L^1_t\ L^2_t)$. Nous obtenons la décomposition suivante :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(P^1\ P^2\ L^1_t\ L^2_t\ L^1_{t'}\ L^2_{t'}\ A) \\ &= P(L^1_t)P(L^2_t)P(P^1 \mid L^1_t\ L^2_t)P(P^2 \mid L^1_t\ L^2_t) \\ &\quad P(A)P(L^1_{t'} \mid A\ L^1_t)P(L^2_{t'} \mid A\ L^2_t). \end{aligned}$$

Nous n'avons ici lié les deux cartes que par leurs modèles direct d'observation : $P(P^1 \mid L^1_t)$ est devenu $P(P^1 \mid L^1_t\ L^2_t)$, et $P(P^2 \mid L^2_t)$ est devenu $P(P^2 \mid L^1_t\ L^2_t)$. Ce choix est déjà important, car il permet d'enrichir considérablement ces modèles : ils pourront devenir plus fins, représentant plus finement l'environnement, et, lorsqu'on les inversera pour répondre à la question de localisation, ces informations supplémentaires seront restituées (voir Section 8.2.2.2 pour les dérivations). Le désavantage évident est qu'il faudra fournir ces informations supplémentaires pour définir ces deux termes $P(P^1 \mid L^1_t\ L^2_t)$ et $P(P^2 \mid L^1_t\ L^2_t)$, soit a priori, soit expérimentalement.

Troisième hypothèse levée Ces deux termes conservent tout de même l'avantage de porter sur des espaces mono-dimensionnels, car ils n'ont qu'une variable en partie gauche. Ils sont en réalité deux projections particulières, de l'espace conjoint $P^1 \wedge P^2$. Se ramener

à cet espace plus grand permettra de lever l'hypothèse d'indépendance entre P^1 et P^2, conditionnellement à la connaissance des variables L_t^1 et L_t^2. Cela se traduit par le choix de la décomposition suivante :

$$\begin{aligned} & P(P\ L_t\ L_{t'}\ A) \\ & \quad = \ P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2)P(A)P(L_{t'}^1 \mid A\ L_t^1)P(L_{t'}^2 \mid A\ L_t^2). \end{aligned}$$

Quatrième hypothèse levée De manière similaire, il est possible symétriquement de lever l'indépendance entre les modèles d'actions. Une première façon est de remplacer les termes $P(L_{t'}^1 \mid A\ L_t^1)$ et $P(L_{t'}^2 \mid A\ L_t^2)$ respectivement par $P(L_{t'}^1 \mid A\ L_t^1\ L_t^2)$ et $P(L_{t'}^2 \mid A\ L_t^1\ L_t^2)$. Mais ici encore, il est possible d'aller plus loin et de lever l'indépendance entre $L_{t'}^1$ et $L_{t'}^2$, conditionnellement à la connaissance de A, L_t^1 et L_t^2, en choisissant la décomposition :

$$P(P\ L_t\ L_{t'}\ A) = P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2)P(A)P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2).$$

Décomposition retenue et formes paramétriques C'est cette dernière décomposition que nous avons choisi pour le reste de ce document :

$$P(P\ L_t\ L_{t'}\ A) = P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2)P(A)P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2).$$

Elle s'accompagne bien évidemment de la définition des formes paramétriques suivantes :

$P(L_t^1)$, $P(L_t^2)$ À ces deux termes sont associées des lois uniformes.

$P(P^1\ P^2 \mid L_t^1\ L_t^2)$ Les variables en partie gauche étant des variables perceptives, elles proviennent généralement, du moins pour les couches de bas niveau d'abstraction, de capteurs physiques. Ceux-ci, pour la plupart, sont représentés par des variables probabilistes qui possèdent beaucoup de valeurs, et qui ont une notion de proximité entre leurs valeurs (c'est notamment le cas des proximètres considérés jusqu'ici). Dans ce cas, le terme $P(P^1\ P^2 \mid L_t^1\ L_t^2)$ pourra être associé à l'approximation discrète d'une loi gaussienne multi-dimensionnelle. Cependant, à de plus hauts niveaux d'abstraction, les variables perceptives pourront être bien plus symboliques. Dans ce cas, seule un ensemble de lois de succession de Laplace aura un sens. Nous notons donc :

$$P(P^1\ P^2 \mid L_t^1\ L_t^2) = \mathbf{L}_{n_1,\ldots,n_{k_{P^1 *_k P^2 *_k L_t^1 *_k L_t^2}}}(P^1, P^2).$$

$P(A)$ À ce terme est associée une loi uniforme.

$P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2)$ Les variables de partie gauche de ce terme, étant des variables de lieux, sont souvent symboliques. Si tel est le cas, nous associons à ce terme un ensemble de lois de succession de Laplace :

$$P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2) = \mathbf{L}_{n_1,\ldots,n_{k_{L_{t'}^1 *_k L_{t'}^2 *_k A *_k L_t^1 *_k L_t^2}}}(L_{t'}^1, L_{t'}^2).$$

Identification des paramètres Pour finir la définition de notre description, il faut indiquer une méthode pour obtenir les valeurs des paramètres des termes sélectionnés.

Dans notre cas, il s'agit d'identifier les termes $P(P^1\ P^2 \mid L_t^1\ L_t^2)$ et $P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2)$, définis ci-dessus comme étant des lois de succession de Laplace. Ces termes peuvent être identifiés expérimentalement, en utilisant la carte superposée sans ajout d'information comme base pour récolter des données. En effet, dans cette carte, il est possible de sélectionner des comportements définis dans c^1 ou c^2. Agir dans l'environnement en appliquant un comportement défini dans c^1 permet d'obtenir des données sur les variables P^1, L_t^1 et $L_{t'}^1$. Mais il est possible, simultanément, de lire les capteurs P^2 et les valeurs des variables de lieux L_t^2 et $L_{t'}^2$ de la carte c^2.

Résumons la méthode pour aboutir à une carte superposée avec ajout d'information :

1. construire deux cartes bayésiennes c^1 et c^2 ;
2. superposer c^1 et c^2 pour obtenir c_{sans}, la carte superposée sans ajout d'information ;
3. choisir des lois uniformes à remplacer par des lois de succession de Laplace, ou des hypothèses d'indépendances conditionnelles à supprimer, dans c_{sans} ;
4. collecter des données en naviguant dans l'environnement grâce à c_{sans} ;
5. utiliser les données expérimentales pour identifier les paramètres des termes de la carte avec ajout d'information, c_{avec}.

8.2.2.2 Résolution des questions de localisation, prédiction et contrôle

Rappelons que la décomposition retenue est :

$$P(P\ L_t\ L_{t'}\ A) = P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2)P(A)P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2).$$

Nous vérifions ici que la carte superposée c choisie répond bien de manière pertinente aux trois questions de localisation, prédiction et contrôle.

Commençons par la résolution de la question de localisation :

$$\begin{aligned}
P(L_t \mid P) &= P(L_t^1\ L_t^2 \mid P^1\ P^2) \\
&= \frac{1}{Z} \sum_{A\ L_{t'}^1\ L_{t'}^2} P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2)P(A)P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2) \\
&= \frac{1}{Z} P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2) \sum_{A\ L_{t'}^1\ L_{t'}^2} P(A)P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2) \\
&= \frac{1}{Z} P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2) \\
P(L_t \mid P) &= \frac{1}{Z'} P(P^1\ P^2 \mid L_t^1\ L_t^2).
\end{aligned}$$

Passons à la question de prédiction :

$$\begin{aligned}
P(L_{t'} \mid A\ L_t) &= P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t) \\
&= \frac{1}{Z}\sum_{P^1\ P^2} P(L^1_t)P(L^2_t)P(P^1\ P^2 \mid L^1_t\ L^2_t)P(A)P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t) \\
&= \frac{1}{Z}P(L^1_t)P(L^2_t)P(A)P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t)\sum_{P^1\ P^2} P(P^1\ P^2 \mid L^1_t\ L^2_t) \\
&= \frac{1}{Z}P(L^1_t)P(L^2_t)P(A)P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t) \\
P(L_{t'} \mid A\ L_t) &= \frac{1}{Z'}P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t).
\end{aligned}$$

Terminons par la question de contrôle :

$$\begin{aligned}
P(A \mid L_t\ L_{t'}) &= P(A \mid L^1_t\ L^2_t\ L^1_{t'}\ L^2_{t'}) \\
&= \frac{1}{Z}\sum_{P^1\ P^2} P(L^1_t)P(L^2_t)P(P^1\ P^2 \mid L^1_t\ L^2_t)P(A)P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t) \\
&= \frac{1}{Z}P(L^1_t)P(L^2_t)P(A)P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t)\sum_{P^1\ P^2} P(P^1\ P^2 \mid L^1_t\ L^2_t) \\
&= \frac{1}{Z}P(L^1_t)P(L^2_t)P(A)P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t) \\
P(A \mid L_t\ L_{t'}) &= \frac{1}{Z'}P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t).
\end{aligned}$$

Notons une fois encore que, bien que les résolutions symboliques pour les questions de prédiction et de contrôle soient identiques, elles amènent des réponses numériques différentes, car elles correspondent en réalité à des coupes transversales différentes dans la conjointe $P(A\ L_t\ L_{t'})$.

Les trois questions se résolvent simplement, en se basant sur les termes $P(P^1\ P^2 \mid L^1_t\ L^2_t)$, pour la question de localisation, et $P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t)$ pour les questions de prédiction et de contrôle. Ces deux termes étant identifiés expérimentalement, il nous est impossible ici de prouver qu'ils contiennent du signal : il reviendra donc à la charge du programmeur de le vérifier expérimentalement.

Nous résumons cette forme finale de superposition, pour le cas $A = A^1 \oplus A^2$, avec ajout d'information, Figure 8.5.

8.2.2.3 Exemple

Au lieu de reprendre notre expérience de pensée de superposition d'une carte de proximétrie et d'une carte d'odeurs, nous souhaitons développer un exemple formel, simple, où l'on pourra vérifier numériquement les effets de l'ajout d'apprentissage dans la superposée. Il s'agira de superposer deux cartes basées sur des gradients aux iso-potentielles rectilignes (voir Figure 8.6), et nous étudierons l'influence de l'angle que forment ces iso-potentielles

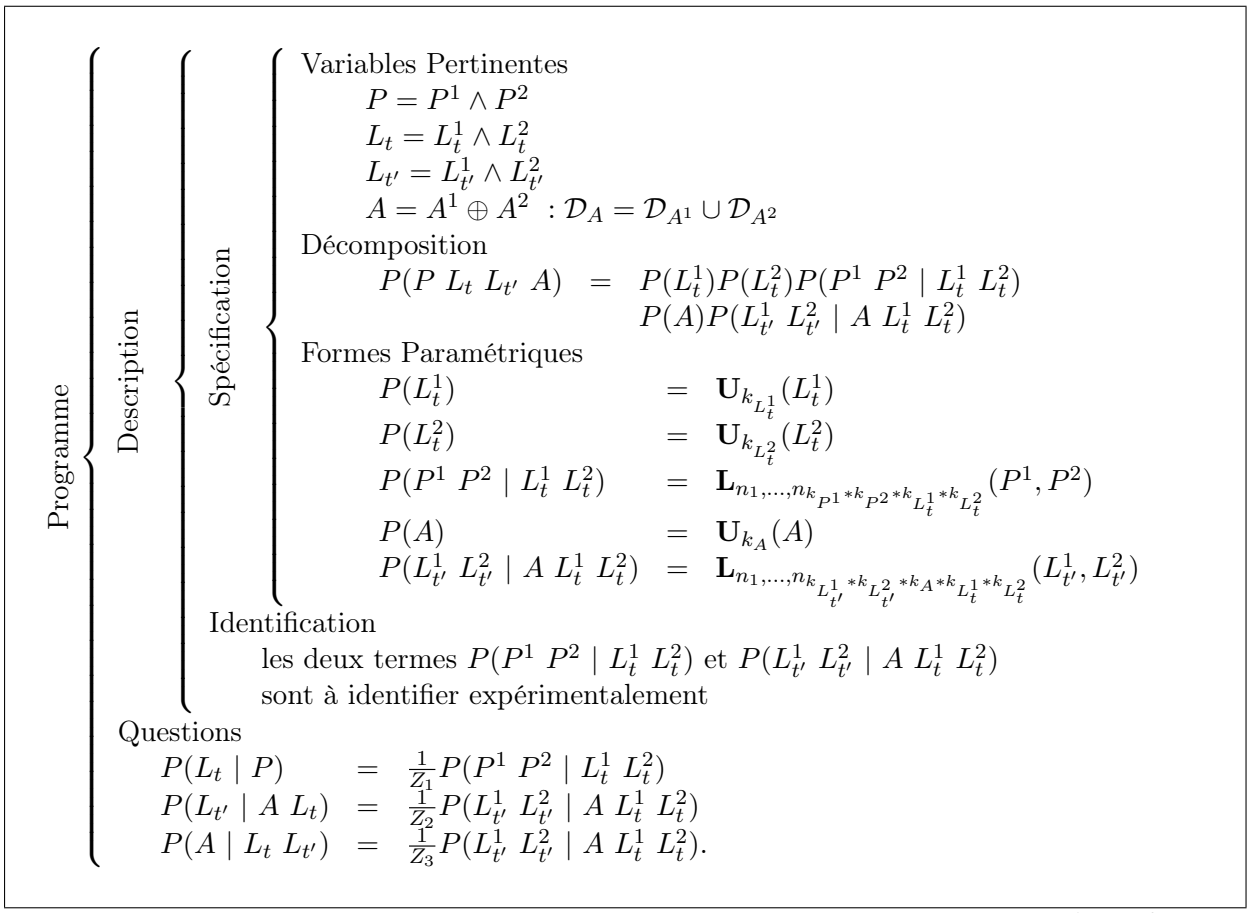


Fig. 8.5: La définition de la superposition de cartes bayésiennes (cas $A = A^1 \oplus A^2$, avec ajout d'information).

sur l'entropie des termes obtenus par apprentissage, en développant les deux cas extrêmes des gradients identiques et orthogonaux.

Afin de simplifier cette partie, nous n'allons dans la suite pas nous préoccuper des variables perceptives, ni des termes de localisation. De plus, dans la carte superposée, et pour des raisons de symétrie, nous ne présenterons que l'identification « expérimentale » de l'un des deux modèles de transition. Nous ne traiterons également qu'une des actions possibles, la remontée de gradient, la situation étant symétrique pour le comportement de descente de gradient.

Définissons maintenant les deux cartes utilisées. La première carte c^1 se base sur un gradient qui divise l'espace en trois zones, « 1 », « 2 » et « 3 », qui sont les valeurs possibles pour les variables de lieux L_t^1 et $L_{t'}^1$. Les actions possibles dans c^1 sont de remonter ce gradient, ou de le descendre, la variable A^1 a donc pour valeurs $\{remonter^1, descendre^1\}$. De la décomposition choisie pour cette carte, ne retenons que le modèle de transition $P(L_{t'}^1 \mid A^1\ L_t^1)$, défini par une loi de succession de Laplace, et les autres termes ne sont que des uniformes. Dans cette carte, l'application de l'action $remonter^1$ a des effets garantis : de la zone 3 on passe à la zone 2 obligatoirement, de la zone 2 on passe systématiquement

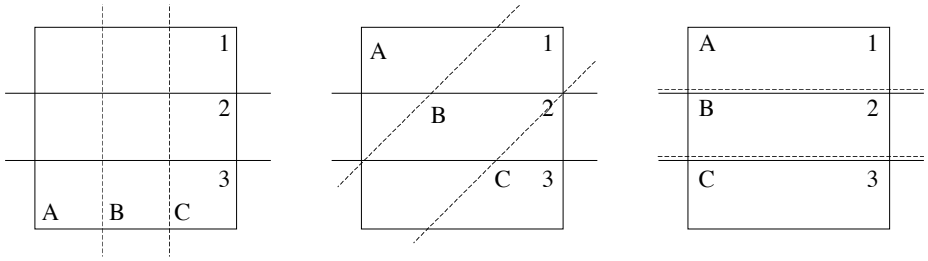


FIG. 8.6: Cas de figure pour la superposition de gradients rectilignes : il peuvent être orthogonaux (à gauche), obliques (milieu) ou colinéaires et identiques (à droite). Les noms des zones (« 1 », « 2 », « 3 » et « A », « B », « C ») font référence aux deux cartes c^1 et c^2.

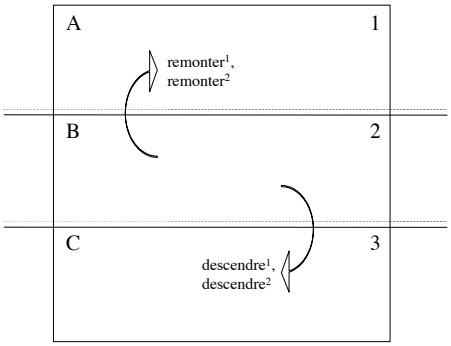
à la zone 1, et de la zone 1 on reste toujours en zone 1. Les formes paramétriques associées à cette actions sont donc des Diracs, centrés sur la zone où le robot est certain d'arriver. De plus, ce comportement se réalise en déplaçant le robot perpendiculairement aux isopotentielles : le robot naviguera donc en allant droit vers la zone à atteindre.

La deuxième carte c^2 suit la même structure que la précédente : les valeurs des variables L^2_t et $L^2_{t'}$ sont « A », « B » et « C », les actions possibles sont $\{remonter^2, descendre^2\}$, et la décomposition singularise le modèle de transition, laissant des uniformes sur les autres termes. Dans c^2, les résultats des actions sont également garantis : en appliquant le comportement $remonter^2$, on passe de la zone C à la zone B, de la B à la A (toujours en suivant des trajectoires directes), et lorsqu'on est déjà en A, on y reste. Les définitions complètes de ces cartes se trouvent Annexe D.

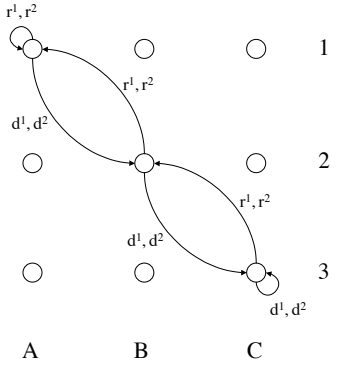
Gradients identiques Nous appliquons la méthode de superposition que nous avons présenté, et obtenons une première carte superposée où la variable d'action est $A = \{remonter^1, descendre^1, remonter^2, descendre^2\}$, et où il y a deux termes de prédiction, $P(L^1_{t'} \mid A\ L^1_t)$ et $P(L^2_{t'} \mid A\ L^2_t)$ (notons également que l'espace $L^1_t \wedge L^2_t$ dispose de neuf cas possibles). Ces deux termes sont des questions aux cartes c^1 et c^2 lorsque les valeurs de A correspondent, sinon ce sont des uniformes.

Identifions maintenant par apprentissage le terme $P(L^1_{t'} \mid A\ L^1_t)$, pour remplacer les cas uniformes, c'est-à-dire lorsque le robot applique un comportement de la carte c^2. Il s'agit donc, expérimentalement, de laisser le robot naviguer dans l'environnement, en appliquant le comportement $remonter^2$, et de mesurer les variations de L^1_t. Nous collectons des données de la forme $\{\langle l^1_{t+\Delta t}, remonter^2, l^1_t\rangle\}_{t=0}^{T}$, et les utilisons pour identifier une loi de succession de Laplace sur la variable $L^1_{t'}$.

Étant donné que les zones des gradients c^1 et c^2 sont identiques, l'application du comportement $remonter^2$, par exemple à partir de la zone 2, amène systématiquement le robot en zone 1, car celles-ci sont respectivement identiques aux zones B et A (voir Figure 8.7(a)). Ainsi, chaque fois que L^1_t vaut 2, $L^1_{t'}$ vaut 1, et on a la donnée expérimentale $\langle 1, remonter^2, 2\rangle$. La forme paramétrique $P(L^1_{t'} \mid [A = remonter^2]\ [L^1_t = 2])$, après collecte de k données de ce type, devient (voir Section 2.2.3.2 pour la formule d'une loi de succession



(a) Superposition de gradients identiques : les zones 1, 2 et 3 sont superposées aux zones A, B et C.



(b) Graphe induit dans la carte superposée, après apprentissage.

FIG. 8.7: Superposition de gradients identiques.

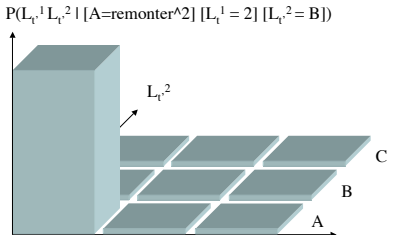
de Laplace) :

$$\begin{aligned} P([L^1_{t'} = 1] \mid [A = remonter^2]\ [L^1_t = 2]) &= \frac{k+1}{k+3} \\ P([L^1_{t'} = 2] \mid [A = remonter^2]\ [L^1_t = 2]) &= \frac{1}{k+3} \\ P([L^1_{t'} = 3] \mid [A = remonter^2]\ [L^1_t = 2]) &= \frac{1}{k+3}. \end{aligned}$$

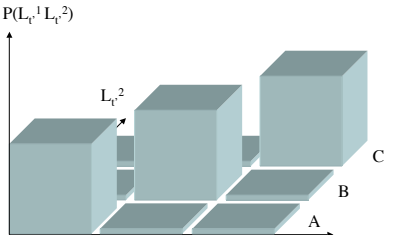
Sachant que le robot se trouve zone 2 et qu'il applique le comportement $remonter^2$, la probabilité qu'il arrive en zone 1 est très grande (tend vers 1 lorsque $k \to \infty$), et les probabilités qu'il reste en zone 2 ou arrive en zone 3 est très petite (tend vers 0 lorsque $k \to \infty$). Cette distribution de probabilité est d'ailleurs très proche du Dirac spécifié pour le terme $P(L^1_{t'} \mid [A = remonter^1]\ [L^1_t = 2])$: il serait donc possible de détecter automatiquement la similitude des résultats des actions $remonter^1$ et $remonter^2$, par une mesure de distance entre les distributions de probabilités. Cette redondance, si elle était détectée, pourrait remettre en cause la pertinence de la superposition des cartes c^1 et c^2 (ce qui est effectivement inutile dans notre exemple de gradients identiques).

Nous avons donc appris expérimentalement une partie du terme $P(L^1_{t'} \mid A\ L^1_t)$. Levons maintenant l'hypothèse d'indépendance conditionnelle entre c^1 et c^2, et identifions le terme $P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t)$. Prenons encore une fois l'exemple du robot en zone 2 (donc zone B), et qui remonte le gradient avec le comportement $remonter^2$. Le résultat de l'apprentissage sera encore une fois très proche d'un Dirac, avec une probabilité très forte pour la valeur $P([L^1_{t'} = 1]\ [L^2_{t'} = A] \mid [A = remonter^2]\ [L^1_t = 2]\ [L^2_t = B])$, et très faible pour toutes les autres valeurs (voir Figure 8.8(a)). Nous montrons Figure 8.7(b) le graphe induit dans la carte superposée, après apprentissage. Sur cette Figure, nous ne montrons que les arcs

issus de nœuds ayant été rencontrée expérimentalement (les autres sont restés sur leurs formes *a priori*, c'est-à-dire des uniformes, donc des arcs menant potentiellement à tous les autres nœuds), pour les actions $remonter^1$, $remonter^2$, $descendre^1$ et $descendre^2$.



(a) Identification expérimentale du terme $P(L^1_{t'}\ L^2_{t'} \mid [A = remonter^2]\ [L^1_t = 2]\ [L^2_t = B])$.



(b) Calcul de $P(L^1_{t'}\ L^2_{t'})$.

FIG. 8.8: Apprentissages et inférence dans la superposition avec ajout d'information.

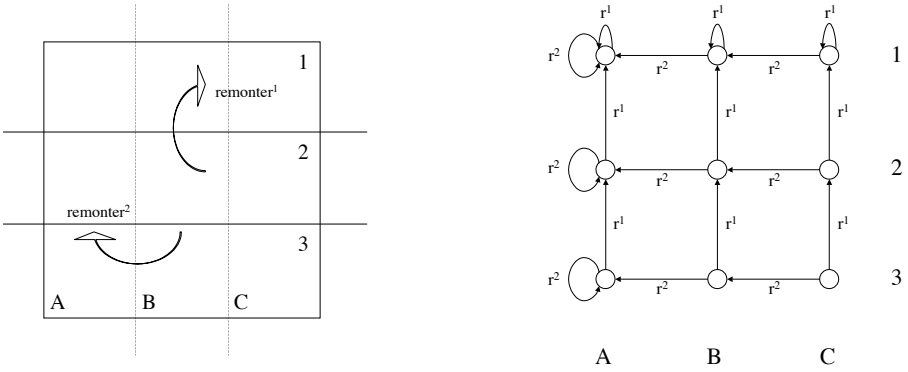
Cependant, nous aurions pu prévoir ce résultat expérimental, car ce n'est rien d'autre que ce que nous obtenons, si nous supposons l'indépendance, et utilisons la forme paramétrique identifiée un peu plus haut. En effet, dans ce cas (et en remplaçant les formes apprises qui sont arbitrairement proches de Diracs, par des Diracs, pour simplifier) :

$$\begin{aligned} & P(L^1_{t'}\ L^2_{t'} \mid [A = remonter^2]\ [L^1_t = 2]\ [L^2_t = B]) \\ = \quad & P(L^1_{t'} \mid [A = remonter^2]\ [L^1_t = 2]).P(L^2_{t'} \mid [A = remonter^2]\ [L^2_t = B]) \\ \overset{k\to\infty}{=} \quad & \boldsymbol{\delta}_1(L^1_{t'}).\delta_A(L^2_{t'}) \\ \overset{k\to\infty}{=} \quad & \boldsymbol{\delta}_{\langle 1,A\rangle}(L^1_{t'}, L^2_{t'}). \end{aligned}$$

Il est également possible de détecter la corrélation entre L^1_t et L^2_t d'une part, et $L^1_{t'}$ et $L^2_{t'}$ d'autre part. Pour ce dernier cas, cela se fait en posant la question $P(L^1_{t'}\ L^2_{t'})$ à la carte superposée. Avec le modèle de transition que nous venons d'identifier, cela s'infère en sommant sur les variables A, L^1_t et L^2_t : quelle est la distribution de probabilités sur les états suivants, ne sachant ni les points de départs, ni l'action appliquée ? Nous ne détaillerons pas ici la dérivation, mais le résultat est bien celui attendu intuitivement : les états sur la « diagonale » sont équiprobables, et toutes les autres combinaisons ($[L^1_{t'} = 3]$ et $[L^2_{t'} = B]$, par exemple), ont des probabilités très faibles, et mêmes nulles si l'on considère les formes apprises comme des Diracs (voir Figure 8.8(b)).

En conclusion, nous avons montré comment identifier expérimentalement les dépendances entre variables dans le cadre de la carte superposée. Nous avons vu que toute l'information que nous identifions nous permet surtout de détecter des redondances. Redondances entre les variables d'actions, mais également entre les variables de lieux. Il est donc possible, automatiquement, de reconnaître que les deux cartes que nous superposons sont en réalité identiques (et dans ce cas, il est inutile de les superposer).

Gradients orthogonaux Examinons maintenant le cas des gradients orthogonaux, en procédant de la même manière que dans le cas précédent : identifions tout d'abord la forme paramétrique $P(L^1_{t'} \mid [A = remonter^2]\ L^1_t)$, puis levons l'hypothèse d'indépendance entre $P(L^1_{t'} \mid A\ L^1_t)$ et $P(L^2_{t'} \mid A\ L^2_t)$.



(a) Les zones 1, 2 et 3, A, B et C créent 9 zones d'intérêt.

(b) Graphe induit dans la carte superposée, après apprentissage.

FIG. 8.9: Superposition de gradients orthogonaux.

Plaçons de nouveau le robot en zone B, faisons lui appliquer le comportement $remonter^2$, et étudions les variations de la variable L^1_t : nous remarquons Figure 8.9(a) que, quelle que soit la zone du gradient c^1 de départ, elle sera inchangée par application du comportement $remonter^2$, car il déplace le robot parallèlement aux iso-potentielles de c^1. Par exemple, si le robot commence en case $\langle B, 1\rangle$, remonter le gradient c^2 l'amènera en case $\langle A, 1\rangle$, systématiquement. Nous comprenons donc que les données expérimentales que nous collecterons seront systématiquement de la forme $\langle l, remonter^2, l\rangle$, pour tout valeur $l \in \mathcal{D}_{L^1_t}$. L'identification des paramètres des formes $P(L^1_{t'} \mid [A = remonter^2]\ L^1_t)$, pour les différentes valeurs de L^1_t, fera apparaître, comme précédemment, des lois de succession de Laplace qui tendent vers des Diracs :

$$\begin{aligned} P([L^1_{t'} = l] \mid [A = remonter^2]\ [L^1_t = l]) &= \frac{k+1}{k+3} \\ \forall j \neq l, P([L^1_{t'} = j] \mid [A = remonter^2]\ [L^1_t = l]) &= \frac{1}{k+3}. \end{aligned}$$

Il suit :

$$\begin{aligned} \lim_{k\to\infty} P(L^1_{t'} \mid [A = remonter^2]\ [L^1_t = 1]) &= \boldsymbol{\delta}_1(L^1_{t'}) \\ \lim_{k\to\infty} P(L^1_{t'} \mid [A = remonter^2]\ [L^1_t = 2]) &= \boldsymbol{\delta}_2(L^1_{t'}) \\ \lim_{k\to\infty} P(L^1_{t'} \mid [A = remonter^2]\ [L^1_t = 3]) &= \boldsymbol{\delta}_3(L^1_{t'}). \end{aligned}$$

Il est donc possible d'identifier beaucoup de signal sur ces formes paramétriques : la version « sans ajout d'information » de la superposition indiquait des uniformes, nous avons maintenant des quasi-Diracs ! Et, au contraire du cas des gradients identiques, ces informations apprises ont un sens exploitable, ceci apparaîtra encore plus clair maintenant, quand nous allons lever l'hypothèse d'indépendance entre $L^1_{t'}$ et $L^2_{t'}$.

Il s'agit donc d'identifier le terme $P(L^1_{t'}\ L^2_{t'} \mid A\ L^1_t\ L^2_t)$. Le résultat sera un ensemble de lois de succession de Laplace très proches de Diracs, et que nous représentons Figure 8.9(b), par le graphe induit de la carte superposée apprise : un nœud pour une valeur des variables de lieux $L^1_t \wedge L^2_t$, et un arc dès qu'une action amène avec une probabilité non-nulle d'un nœud à l'autre. Nous ne faisons apparaître dans cette figure que les arcs associés aux actions $remonter^1$ et $remonter^2$ (dénotées par r^1 et r^2). Nous remarquons donc que non seulement l'identification a permis de retirer beaucoup de signal de l'expérience, mais les formes apprises ont aussi beaucoup de sens : le robot a acquis une réelle structuration de l'espace $L^1_t \wedge L^2_t$, sur laquelle le raisonnement (planification) devient maintenant possible, grâce aux actions $remonter^1, remonter^2, \ldots$. Résoudre la tâche $rejoindre\ la\ case\ \langle C, 1\rangle$, par exemple, prend maintenant un sens, qui n'était inclus ni dans la carte c^1, ni dans la carte c^2, mais qui peut être identifié, nous l'avons montré, dans le cadre de la carte superposée de c^1 et c^2.

Gradients obliques Sans trop détailler cette section, notons juste ici que le cas des gradients obliques sera bien sûr intermédiaire entre les deux précédents. Selon l'angle entre les gradients, ils seront plus ou moins semblables, et il sera ainsi possible d'identifier plus ou moins d'informations nouvelles par apprentissage. Grâce au formalisme probabiliste, nous pourrons identifier les incertitudes nées de l'ignorance de cet angle qui existe entre les gradients.

8.2.2.4 Discussion

Commençons par quelques remarques, avant d'analyser ce que ce schéma de superposition avec ajout d'informations nous apprend sur les réponses possibles aux questions énoncées Section 1.2.

Dans la plupart des cas, nous pensons qu'une phase d'apprentissage permet d'identifier des nouvelles dépendances, car en pratique il serait étonnant d'avoir deux cartes dont les modalités sont réellement indépendantes. Nous l'avons vu sur l'exemple des gradients identiques, il est également possible de vérifier après coup la pertinence de ces dépendances, en calculant les entropies des distributions obtenues expérimentalement, ou la similitude entre des distributions apprises.

Par exemple, on peut détecter que les deux gradients sont identiques. Dans ce cas particulier, cela revient à poser $L^1_t = L^2_t$. Dans ce cas, superposer comme nous l'avons présenté devient inutile, car on se place dans un espace redondant ($L_t = L^1_t \wedge L^1_t$).

Cependant, dans nos exemples sur les gradients, nous ne considérions pas les variables de perception. Même si les variables internes sont les mêmes, il peut être intéressant d'assembler deux cartes $P(P^1\ L^1_t\ L^1_{t'}\ A^1 \mid c^1)$ et $P(P^2\ L^1_t\ L^1_{t'}\ A^2 \mid c^2)$. Nous fusionnons tout d'abord

les modèles capteurs (au sens de [LEBELTEL99]), et écrivons $P(P^1 \mid L_t^1)$ et $P(P^2 \mid L_t^1)$ dans la carte c. Les modèles de transition peuvent aussi s'assembler, toujours grâce à une nouvelle variable $A = A^1 \oplus A^2$, avec le terme $P(L_{t'}^1 \mid A\ L_t^1)$, défini de la manière suivante :

$$P(L_{t'}^1 \mid [A = a]\ L_t^1) \quad = \quad \begin{cases} P(L_{t'}^1 \mid [A^1 = a]\ L_t^1\ c^1) \text{ si } a \in \mathcal{D}_{A^1} \\ P(L_{t'}^1 \mid [A^2 = a]\ L_t^1\ c^2) \text{ si } a \in \mathcal{D}_{A^2} \end{cases}$$

Cette forme d'assemblage de cartes, que nous n'explorerons pas plus avant ici, peut être vue comme une extension de la localisation markovienne, pour le cas où l'on dispose de plusieurs modèles capteurs, et de plusieurs modèles de transitions (nous l'avons montré pour le cas d'assemblage de deux cartes, cela s'étend trivialement pour n cartes). Cela sera particulièrement utile lorsque le programmeur souhaitera développer indépendamment deux cartes du même espace, utilisant la même variable de lieu, mais basées sur des capteurs différents. Il sera ainsi possible de les superposer, par exemple si l'un des capteurs est moins discriminant / performant sur un certain sous-domaine de L_t, mais que l'autre capteur peut pallier cette faiblesse, etc. La mise « côte à côte » des modèles de transitions est aussi un choix intuitif, si par exemple le robot dispose de commande référencées par chacun des capteurs disponibles.

Examinons maintenant certaines des questions définies au début de ce document.

Q1. Qu'est-ce qu'une carte ? Les phases expérimentales détaillées plus haut peuvent s'effectuer alors même que la carte superposée est utilisée pour résoudre des tâches (du moins une version de cette carte supposant encore l'indépendance de c^1 et c^2). Dans ce cas, la carte « sans ajout d'information » peut être vue comme un canevas de base, qui sert à réaliser des tâches initialement. Pendant que le robot les réalise, il navigue dans l'environnement, et peut donc collecter des données servant à l'identification des termes $P(P^1\ P^2 \mid L_t^1\ L_t^2)$ et $P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2)$ de la carte superposée (phase de cartographie).

Les cartes peuvent donc non seulement s'obtenir de manière incrémentale, au fur et à mesure de la programmation, mais elles peuvent aussi s'affiner, s'adapter, au fur et à mesure de la vie du robot dans son environnement.

Q7. Comment apprendre expérimentalement les cartes ? Nous avons vu, dans la Section précédente, une méthode automatique, car syntaxique, de génération de carte superposée. Nous avons présenté dans cette Section une extension de cette méthode, dans laquelle l'apprentissage vient complémenter la structure générée automatiquement (cartographie). Nous avons également présenté, plus ou moins formellement, des outils rendant possible la détection de certaines propriétés de la carte superposée. Par exemple, il est possible de détecter la similitude entre deux cartes que l'on superpose (indice de redondance), ou au contraire, leurs différences (indice d'orthogonalité des cartes). Ces outils de détection de cas extrêmes peuvent être mis à profit par le programmeur, pour le guider dans sa conception. Par exemple, s'il dispose de n cartes probabilistes et de ressources informatiques limitées, quelles sont les deux cartes les plus intéressantes à superposer ? Dans notre approche, cela se résout en vérifiant à quel degré les cartes sont orthogonales. Mais cette

analyse ne nécessite pas l'intervention du programmeur. Nous pouvons imaginer en effet que le robot génère toutes les superpositions possibles des n cartes (grâce à notre opérateur syntaxique), collecte des données, identifie les formes grâce à ces données, et ne retienne que les cartes superposées les plus riches en information ajoutée. Un tel mécanisme, qui génère des hypothèses, sait les évaluer et rejeter les plus mauvaises, n'est pas sans rappeler les « générateur de diversité », dont l'existence à été supposée et soulignée par de nombreux auteurs en sciences cognitives [Berthoz97, Damasio95, DC91].

8.3 $A = A^1 \wedge A^2$

Dans cette seconde variante, la variable d'action A de la carte superposée c est la conjonction des variables d'actions A^1 et A^2. Contrairement au cas précédent, on sait agir en appliquant *en même temps* un comportement A^1 du domaine de A^1, et un comportement A^2 du domaine de A^2. Deux hypothèses sont possibles dans ce cas.

Ces comportements peuvent par exemple agir sur différentes parties du robot : supposons que nous ayons d'une part une carte c^1 qui permette de réaliser une tâche de suivi visuel, à l'arrêt, et d'autre part, une carte c^2 réalisant une tâche pour le corps du robot. Superposer ces deux cartes permettra d'identifier l'influence entre les réalisations de ces deux tâches qui, bien que s'appliquant sur des parties du robot séparées, ont des dépendances fortes (le mouvement du robot induit par la carte c^2 influe sur la tâche de suivi visuel dans c^1).

Les comportements peuvent également s'appliquer simultanément par exemple :

- si l'on suppose connaître des propriétés sur les actions, qui permettent de les assembler ;
- ou si l'on suppose l'existence d'un programme « arbitre » à un plus bas niveau d'abstraction, qui fera cet assemblage.

C'est le cas par exemple lorsque les actions sont des remontées de gradients, s'exprimant à plus bas niveau par des vecteurs, que l'on peut assembler par sommation vectorielle (cas en particulier dans les approches à bases de champs de potentiels).

8.3.1 $A = A^1 \wedge A^2$, sans ajout d'information

Nous suivrons ici une présentation calquée sur la Section 8.2.1 : nous commençons par superposer les deux cartes c^1 et c^2 sans ajouter d'informations, puis explorerons Section 8.3.2 la possibilité d'enrichir cette première carte.

8.3.1.1 Description

1. Spécification des connaissances préalables c
   (a) Variables :
   $\boldsymbol{P}$ $P = P^1 \wedge P^2$,
   $\boldsymbol{L_t}$ $L_t = L_t^1 \wedge L_t^2$,

$\boldsymbol{L_{t'}}$ $L_{t'} = L^1_{t'} \wedge L^2_{t'}$,
$\boldsymbol{A}$ $A = A^1 \wedge A^2$.

(b) Décomposition : Nous choisissons la décomposition suivante :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(P^1\ P^2\ L^1_t\ L^2_t\ L^1_{t'}\ L^2_{t'}\ A^1\ A^2) \\ &= P(P^1\ L^1_t\ L^1_{t'}\ A^1)P(P^2\ L^2_t\ L^2_{t'}\ A^2). \end{aligned}$$

Cette décomposition revient à supposer une indépendance totale entre les deux cartes c^1 et c^2.

(c) Formes paramétriques : Les deux termes s'obtiennent par des questions aux cartes sous-jacentes :

$$\begin{aligned} P(P^1\ L^1_t\ L^1_{t'}\ A^1) &= P(P^1\ L^1_t\ L^1_{t'}\ A^1 \mid c^1), \\ P(P^2\ L^2_t\ L^2_{t'}\ A^2) &= P(P^2\ L^2_t\ L^2_{t'}\ A^2 \mid c^2). \end{aligned}$$

2. Identification : Toutes les formes paramétriques sont définies, il n'y a pas de phase d'identification.

8.3.1.2 Résolution des questions de localisation, prédiction et contrôle

Nous vérifions ici que la carte superposée c choisie répond bien de manière pertinente aux trois questions de localisation, prédiction et contrôle.

Commençons par la résolution de la question de localisation :

$$\begin{aligned} P(L_t \mid P) &= P(L^1_t\ L^2_t \mid P^1\ P^2) \\ &= \frac{\sum_{A^1\ A^2\ L^1_{t'}\ L^2_{t'}} P(P^1\ L^1_t\ L^1_{t'}\ A^1)P(P^2\ L^2_t\ L^2_{t'}\ A^2)}{\sum_{A^1\ A^2\ L^1_t\ L^2_t\ L^1_{t'}\ L^2_{t'}} P(P^1\ L^1_t\ L^1_{t'}\ A^1)P(P^2\ L^2_t\ L^2_{t'}\ A^2)} \\ &= \frac{\sum_{A^1\ L^1_{t'}} P(P^1\ L^1_t\ L^1_{t'}\ A^1)}{\sum_{A^1\ L^1_t\ L^1_{t'}} P(P^1\ L^1_t\ L^1_{t'}\ A^1)} \cdot \frac{\sum_{A^2\ L^2_{t'}} P(P^2\ L^2_t\ L^2_{t'}\ A^2)}{\sum_{A^2\ L^2_t\ L^2_{t'}} P(P^2\ L^2_t\ L^2_{t'}\ A^2)} \\ &= \frac{P(P^1\ L^1_t)}{P(P^1)} \cdot \frac{P(P^2\ L^2_t)}{P(P^2)} \\ P(L_t \mid P) &= P(L^1_t \mid P^1)P(L^2_t \mid P^2). \end{aligned}$$

Passons à la question de prédiction :

$$\begin{aligned} P(L_{t'} \mid A\ L_t) &= P(L^1_{t'}\ L^2_{t'} \mid A^1\ A^2\ L^1_t\ L^2_t) \\ &= \frac{\sum_{P^1\ P^2} P(P^1\ L^1_t\ L^1_{t'}\ A^1)P(P^2\ L^2_t\ L^2_{t'}\ A^2)}{\sum_{P^1\ P^2\ L^1_{t'}\ L^2_{t'}} P(P^1\ L^1_t\ L^1_{t'}\ A^1)P(P^2\ L^2_t\ L^2_{t'}\ A^2)} \\ &= \frac{\sum_{P^1} P(P^1\ L^1_t\ L^1_{t'}\ A^1)}{\sum_{P^1\ L^1_{t'}} P(P^1\ L^1_t\ L^1_{t'}\ A^1)} \cdot \frac{\sum_{P^2} P(P^2\ L^2_t\ L^2_{t'}\ A^2)}{\sum_{P^2\ L^2_{t'}} P(P^2\ L^2_t\ L^2_{t'}\ A^2)} \\ &= \frac{P(L^1_t\ L^1_{t'}\ A^1)}{P(L^1_t\ A^1)} \cdot \frac{P(L^2_t\ L^2_{t'}\ A^2)}{P(L^2_t\ A^2)} \\ P(L_{t'} \mid A\ L_t) &= P(L^1_{t'} \mid A^1\ L^1_t)P(L^2_{t'} \mid A^2\ L^2_t). \end{aligned}$$

Une fois de plus, cette question se résout indépendamment sur les deux cartes sous-jacentes. Terminons par la question de contrôle :

$$\begin{aligned}
P(A^1\ A^2 \mid L_t\ L_{t'}) &= P(A^1\ A^2 \mid L_t^1\ L_t^2\ L_{t'}^1\ L_{t'}^2) \\
&= \frac{\sum_{P^1\ P^2} P(P^1\ L_t^1\ L_{t'}^1\ A^1)P(P^2\ L_t^2\ L_{t'}^2\ A^2)}{\sum_{P^1\ P^2\ A^1\ A^2} P(P^1\ L_t^1\ L_{t'}^1\ A^1)P(P^2\ L_t^2\ L_{t'}^2\ A^2)} \\
&= \frac{\sum_{P^1} P(P^1\ L_t^1\ L_{t'}^1\ A^1)}{\sum_{P^1\ A^1} P(P^1\ L_t^1\ L_{t'}^1\ A^1)} \cdot \frac{\sum_{P^2} P(P^2\ L_t^2\ L_{t'}^2\ A^2)}{\sum_{P^2\ A^2} P(P^2\ L_t^2\ L_{t'}^2\ A^2)} \\
&= \frac{P(L_t^1\ L_{t'}^1\ A^1)}{P(L_t^1\ L_{t'}^1)} \cdot \frac{P(L_t^2\ L_{t'}^2\ A^2)}{P(L_t^2\ L_{t'}^2)} \\
P(A^1\ A^2 \mid L_t\ L_{t'}) &= P(A^1 \mid L_t^1\ L_{t'}^1)P(A^2 \mid L_t^2\ L_{t'}^2).
\end{aligned}$$

Nous remarquons que ces trois questions se résolvent indépendamment dans les cartes sous-jacentes, ce qui est bien évidemment dû à notre choix d'indépendance totale entre les deux cartes.

Programme
- Description
  - Spécification
    - Variables Pertinentes
      - $P = P^1 \wedge P^2$
      - $L_t = L_t^1 \wedge L_t^2$
      - $L_{t'} = L_{t'}^1 \wedge L_{t'}^2$
      - $A = A^1 \wedge A^2$
    - Décomposition
      - $P(P\ L_t\ L_{t'}\ A) = P(P^1\ L_t^1\ L_{t'}^1\ A^1)P(P^2\ L_t^2\ L_{t'}^2\ A^2)$
    - Formes Paramétriques
      - $P(P^1\ L_t^1\ L_{t'}^1\ A^1) = P(P^1\ L_t^1\ L_{t'}^1\ A^1 \mid c^1)$
      - $P(P^2\ L_t^2\ L_{t'}^2\ A^2) = P(P^2\ L_t^2\ L_{t'}^2\ A^2 \mid c^2)$
  - Identification : pas de paramètres à identifier
- Questions :
  - $P(L_t \mid P) = P(L_t^1 \mid P^1)P(L_t^2 \mid P^2)$
  - $P(L_{t'} \mid A\ L_t) = P(L_{t'}^1 \mid A^1\ L_t^1)P(L_{t'}^2 \mid A^2\ L_t^2)$
  - $P(A \mid L_t\ L_{t'}) = P(A^1 \mid L_t^1\ L_{t'}^1)P(A^2 \mid L_t^2\ L_{t'}^2)$

Fig. 8.10: La définition de la superposition de cartes bayésiennes (cas $A = A^1 \wedge A^2$, sans ajout d'information).

Selon notre hypothèse, les cartes sous-jacentes apportent effectivement suffisamment d'information dans leurs questions de localisation, prédiction et contrôle. Puisque les résolutions de ces mêmes questions dans la carte superposée se réduisent tous à questionner les cartes sous-jacentes (grâce à notre choix de décomposition), alors la carte superposée disposera elle aussi de suffisamment de signal sur ces questions, et répondra donc à nos critères de définition d'une carte bayésienne. Nous pouvons donc résumer la méthode de superposition de cartes, dans le cas $A = A^1 \wedge A^2$, sans ajout d'information, Figure 8.10.

Comme dans la variante $A = A^1 \oplus A^2$, cette version sans ajout d'information à l'intérêt de regrouper, dans une description, deux cartes différentes. Si l'hypothèse d'indépendance

entre les cartes est valide (par exemple, contrôle de deux parties indépendantes du robot), alors la superposition de cartes que nous avons présenté permet de les exploiter simultanément, par des questions comme la localisation, la prédiction ou le contrôle.

Étant donnée la simplicité de cette variante de superposition, nous ne la discuterons pas plus avant, et passons tout de suite aux ajouts d'information dans cette carte, par programmation *a priori* ou par apprentissage.

8.3.2 $A = A^1 \wedge A^2$, avec ajout d'information

Contrairement à la variante de superposition $A = A^1 \oplus A^2$, nous ne disposons ici que d'une méthode pour complexifier la carte superposée. En effet, il n'y a, dans cette version où $A = A^1 \wedge A^2$, qu'un seul type d'indépendance conditionnelle à lever, par le biais de la décomposition (puisque toutes les formes paramétriques sont différentes de l'uniforme). Pour appuyer notre présentation, nous allons supposer que les cartes sous-jacentes sont décomposées une fois encore selon une localisation markovienne.

Nous présenterons tout d'abord la définition formelle de cet opérateur de superposition, vérifierons ensuite que la carte superposée répond correctement aux questions de localisation, prédiction, et contrôle, présenterons un exemple formel, puis conclurons par une discussion, notamment en mentionnant les réponses que cet opérateur suggère à nos questions « fil conducteur ».

8.3.2.1 Description

Si les cartes c^1 et c^2 ont une structure de type localisation markovienne, leur superposition selon le schéma $A = A^1 \wedge A^2$, sans ajout d'information, s'écrit :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(P^1\ L_t^1\ L_{t'}^1\ A^1)P(P^2\ L_t^2\ L_{t'}^2\ A^2) \\ &= P(L_t^1)P(P^1 \mid L_t^1)P(A^1)P(L_{t'}^1 \mid A^1\ L_t^1) \\ &\quad P(L_t^2)P(P^2 \mid L_t^2)P(A^2)P(L_{t'}^2 \mid A^2\ L_t^2) \\ &= P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1)P(P^2 \mid L_t^2) \\ &\quad P(A^1)P(A^2)P(L_{t'}^1 \mid A^1\ L_t^1)P(L_{t'}^2 \mid A^2\ L_t^2). \end{aligned}$$

La dernière égalité ci-dessus n'étant qu'un réarrangement des termes.

De manière identique à la Section 8.2.2.1, nous pouvons lever tout d'abord l'hypothèse d'indépendance entre les modèles capteurs $P(P^1 \mid L_t^1)$ et $P(P^2 \mid L_t^2)$:

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(L_t^1)P(L_t^2)P(P^1 \mid L_t^1\ L_t^2)P(P^2 \mid L_t^1\ L_t^2) \\ &\quad P(A^1)P(A^2)P(L_{t'}^1 \mid A^1\ L_t^1)P(L_{t'}^2 \mid A^2\ L_t^2), \end{aligned}$$

puis passer de ces termes mono-dimensionnels à des termes portant sur l'espace conjoint $P^1 \wedge P^2$:

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2) \\ &\quad P(A^1)P(A^2)P(L_{t'}^1 \mid A^1\ L_t^1)P(L_{t'}^2 \mid A^2\ L_t^2), \end{aligned}$$

Nous nous éloignons maintenant de la Section 8.2.2.1, car nous allons à présent considérer les termes liés aux actions.

Notons tout de suite qu'il serait peu intéressant de trop imiter ce que nous avions fait dans le cas $A = A^1 \oplus A^2$. En effet, nous passerions par les étapes suivantes :

1. levons l'hypothèse d'indépendance conditionnelle entre $L^1_{t'}$ et L^2_t, sachant A^1 et L^1_t (et le symétrique pour l'autre terme) :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(L^1_t)P(L^2_t)P(P^1\ P^2 \mid L^1_t\ L^2_t) \\ &\quad P(A^1)P(A^2)P(L^1_{t'} \mid A^1\ L^1_t\ L^2_t)P(L^2_{t'} \mid A^2\ L^1_t\ L^2_t), \end{aligned}$$

2. levons l'hypothèse d'indépendance conditionnelle entre $L^1_{t'}$ et A^2, sachant A^1, L^1_t et L^2_t (et le symétrique pour l'autre terme) :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(L^1_t)P(L^2_t)P(P^1\ P^2 \mid L^1_t\ L^2_t) \\ &\quad P(A^1)P(A^2)P(L^1_{t'} \mid A^1\ A^2\ L^1_t\ L^2_t)P(L^2_{t'} \mid A^1\ A^2\ L^1_t\ L^2_t), \end{aligned}$$

3. finalement, passons à l'espace conjoint en levant l'hypothèse d'indépendance conditionnelle entre $L^1_{t'}$ et $L^2_{t'}$, sachant A^1, A^2, L^1_t et L^2_t :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(L^1_t)P(L^2_t)P(P^1\ P^2 \mid L^1_t\ L^2_t) \\ &\quad P(A^1)P(A^2)P(L^1_{t'}\ L^2_{t'} \mid A^1\ A^2\ L^1_t\ L^2_t). \end{aligned}$$

Malheureusement, ce dernier résultat n'est rien d'autre qu'une carte bayésienne, sur l'espace $L^1_t \wedge L^2_t$, avec les variables de perception $P^1 \wedge P^2$ et d'action $A^1 \wedge A^2$, structurée comme une localisation markovienne. Cette carte peut être intéressante en soi, mais si nous supposons maintenant obtenir les modèles capteurs et de transition par apprentissage, alors ce modèle ne contient plus aucune information des cartes sous-jacentes c^1 et c^2. Certes, dans le cadre du cycle de programmation, rien n'empêche un concepteur d'écrire tout d'abord ces deux cartes, puis, s'il en est satisfait (par exemple si elles confirment que les variables de lieu L^1_t et L^2_t choisies sont bien pertinentes en pratique), d'essayer d'écrire une carte directement dans l'espace conjoint. Mais cet opération ne relève plus de la superposition de cartes.

L'option que nous souhaitons conserver dans la définition de notre opérateur de superposition la suivante : il s'agit d'observer les résultats des actions A^1 et A^2 sur les variations des variables de lieu L^1_t d'une part et L^2_t d'autre part, puisqu'on peut les appliquer simultanément par hypothèse. Ceci correspond à la décomposition suivante :

$$\begin{aligned} P(P\ L_t\ L_{t'}\ A) &= P(L^1_t)P(L^2_t)P(P^1\ P^2 \mid L^1_t\ L^2_t) \\ &\quad P(A^1)P(A^2)P(L^1_{t'} \mid A^1\ A^2\ L^1_t)P(L^2_{t'} \mid A^1\ A^2\ L^2_t). \end{aligned}$$

En effet, ce choix permet d'identifier, au sein de la carte superposée, les effets qu'ont l'application simultanée des actions provenant des deux cartes sous-jacentes, information qui n'est évidemment contenue dans aucune d'entre elles.

Les formes paramétriques liées à cette décomposition sont soit des uniformes, soit des lois de succession de Laplace, comme Section 8.2.2.1 :

$P(L_t^1)$, $P(L_t^2)$ À ces deux termes sont associées des lois uniformes.

$P(P^1\ P^2 \mid L_t^1\ L_t^2)$ Dans le cas où les variables de partie gauche ont peu de cas, ou pas d'hypothèse de distance entre leurs valeurs, seule une loi de succession de Laplace aura un sens. Nous notons :

$$P(P^1\ P^2 \mid L_t^1\ L_t^2) = \mathbf{L}_{n_1,\ldots,n_{k_{P^1}{}^{k_{P^2}{}^{k_{L_t^1}{}^{k_{L_t^2}}}}}}(P^1, P^2).$$

Dans le cas contraire il sera possible de tirer parti de représentations plus compactes, comme par exemple des gaussiennes multi-dimensionnelles.

$P(A^1)$, $P(A^2)$ À ces deux termes sont associées des lois uniformes.

$P(L_{t'}^1 \mid A^1\ A^2\ L_t^1)$, $P(L_{t'}^2 \mid A^1\ A^2\ L_t^2)$ Nous associons à ces termes des lois de succession de Laplace :

$$\begin{aligned} P(L_{t'}^1 \mid A^1\ A^2\ L_t^1) &= \mathbf{L}_{n_1,\ldots,n_{k_{L_{t'}^1}{}^{k_{A^1}{}^{k_{A^2}{}^{k_{L_t^1}}}}}}(L_{t'}^1), \\ P(L_{t'}^2 \mid A^1\ A^2\ L_t^1) &= \mathbf{L}_{n_1,\ldots,n_{k_{L_{t'}^2}{}^{k_{A^1}{}^{k_{A^2}{}^{k_{L_t^1}}}}}}(L_{t'}^2). \end{aligned}$$

8.3.2.2 Résolution des questions de localisation, prédiction et contrôle

Nous vérifions ici que la carte superposée c choisie répond bien de manière pertinente aux trois questions de localisation, prédiction et contrôle.

Commençons par la résolution de la question de localisation :

$$\begin{aligned} & P(L_t \mid P) \\ &= P(L_t^1\ L_t^2 \mid P^1\ P^2) \\ &= \frac{1}{Z} \sum_{A^1\ A^2\ L_{t'}^1\ L_{t'}^2} \left(\begin{array}{l} P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2)P(A^1)P(A^2) \\ P(L_{t'}^1 \mid A^1\ A^2\ L_t^1)P(L_{t'}^2 \mid A^1\ A^2\ L_t^2) \end{array} \right) \\ &= \frac{1}{Z} \left(\begin{array}{l} P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2) \\ \sum_{A^1\ A^2\ L_{t'}^1\ L_{t'}^2} P(A^1)P(A^2)P(L_{t'}^1 \mid A^1\ A^2\ L_t^1)P(L_{t'}^2 \mid A^1\ A^2\ L_t^2) \end{array} \right) \\ &= \frac{1}{Z} P(L_t^1)P(L_t^2)P(P^1\ P^2 \mid L_t^1\ L_t^2) \\ &= \frac{1}{Z'} P(P^1\ P^2 \mid L_t^1\ L_t^2). \end{aligned}$$

Passons à la question de prédiction :

$$\begin{aligned}
P(L_{t'} \mid A\ L_t) &= P(L^1_{t'}\ L^2_{t'} \mid A^1\ A^2\ L^1_t\ L^2_t) \\
&= \frac{1}{Z} \sum_{P^1\ P^2} \begin{pmatrix} P(L^1_t)P(L^2_t)P(P^1\ P^2 \mid L^1_t\ L^2_t)P(A^1)P(A^2) \\ P(L^1_{t'} \mid A^1\ A^2\ L^1_t)P(L^2_{t'} \mid A^1\ A^2\ L^2_t) \end{pmatrix} \\
&= \frac{1}{Z} \begin{pmatrix} P(L^1_t)P(L^2_t)P(A^1)P(A^2)P(L^1_{t'} \mid A^1\ A^2\ L^1_t) \\ P(L^2_{t'} \mid A^1\ A^2\ L^2_t) \sum_{P^1\ P^2} P(P^1\ P^2 \mid L^1_t\ L^2_t) \end{pmatrix} \\
&= \frac{1}{Z} P(L^1_t)P(L^2_t)P(A^1)P(A^2)P(L^1_{t'} \mid A^1\ A^2\ L^1_t)P(L^2_{t'} \mid A^1\ A^2\ L^2_t) \\
P(L_{t'} \mid A\ L_t) &= \frac{1}{Z'} P(L^1_{t'} \mid A^1\ A^2\ L^1_t)P(L^2_{t'} \mid A^1\ A^2\ L^2_t).
\end{aligned}$$

Une fois de plus, la question de contrôle aura la même dérivation symbolique, mais avec un résultat numérique différent.

Les trois questions se résolvent simplement, en se basant sur les termes $P(P^1\ P^2 \mid L^1_t\ L^2_t)$, pour la question de localisation, et $P(L^1_{t'} \mid A^1\ A^2\ L^1_t)$ et $P(L^2_{t'} \mid A^1\ A^2\ L^2_t)$ pour les questions de prédiction et de contrôle. Ces trois termes étant identifiés expérimentalement, il nous est impossible ici de prouver qu'ils contiennent du signal : il reviendra donc à la charge du programmeur de le vérifier expérimentalement.

Nous résumons cette forme finale de superposition, pour le cas $A = A^1 \wedge A^2$, avec ajout d'information, Figure 8.11.

8.3.2.3 Exemple

Développons un exemple proche de celui Section 8.2.2.3, qui permet d'illustrer l'opérateur de superposition dans le cas $A = A^1 \wedge A^2$, et ses limites. Reprenons nos gradients à trois valeurs, mais donnons leur une géométrie circulaire, autour d'un point d'attraction p_1 pour le gradient de la carte c^1, et p_2 pour le gradient de la carte c^2. Cette situation est représentée Figure 8.12.

Comme précédemment, ces gradients créent trois zones chacun, 1, 2 et 3 pour c^1, et A, B et C pour c^2. Les variables d'actions associées sont $remonter^1$ et $remonter^2$ (nous laisserons une fois de plus de côté les comportements de descente de gradients), et nous porterons encore une fois notre attention sur les modèles de transitions, $P(L^1_{t'} \mid A^1\ L^1_t)$ et $P(L^2_{t'} \mid A^2\ L^2_t)$. Dans les cartes c^1 et c^2, ces derniers amènent toujours systématiquement au but (Diracs).

Il faut également nous soucier de quelques caractéristiques du modèle qui combine les actions $remonter^1$ et $remonter^2$. Une manière simple d'écrire ce modèle est de se placer dans le cas de sommation vectorielle dans un espace 2D. Il faut supposer que l'action $remonter^1$ (resp. $remonter^2$) est traduite à plus bas niveau par une consigne à un modèle qui calcule un vecteur pointant vers la source p_1 (resp. p_2), et dont la norme est fonction de la distance à p_1 (resp. p_2). Typiquement, ce vecteur est la dérivée d'une fonction potentielle E_1 (resp E_2). Une façon simple d'appliquer simultanément les deux actions est de sommer leurs vecteurs correspondants. L'effet, en terme de bassin d'attraction de la carte

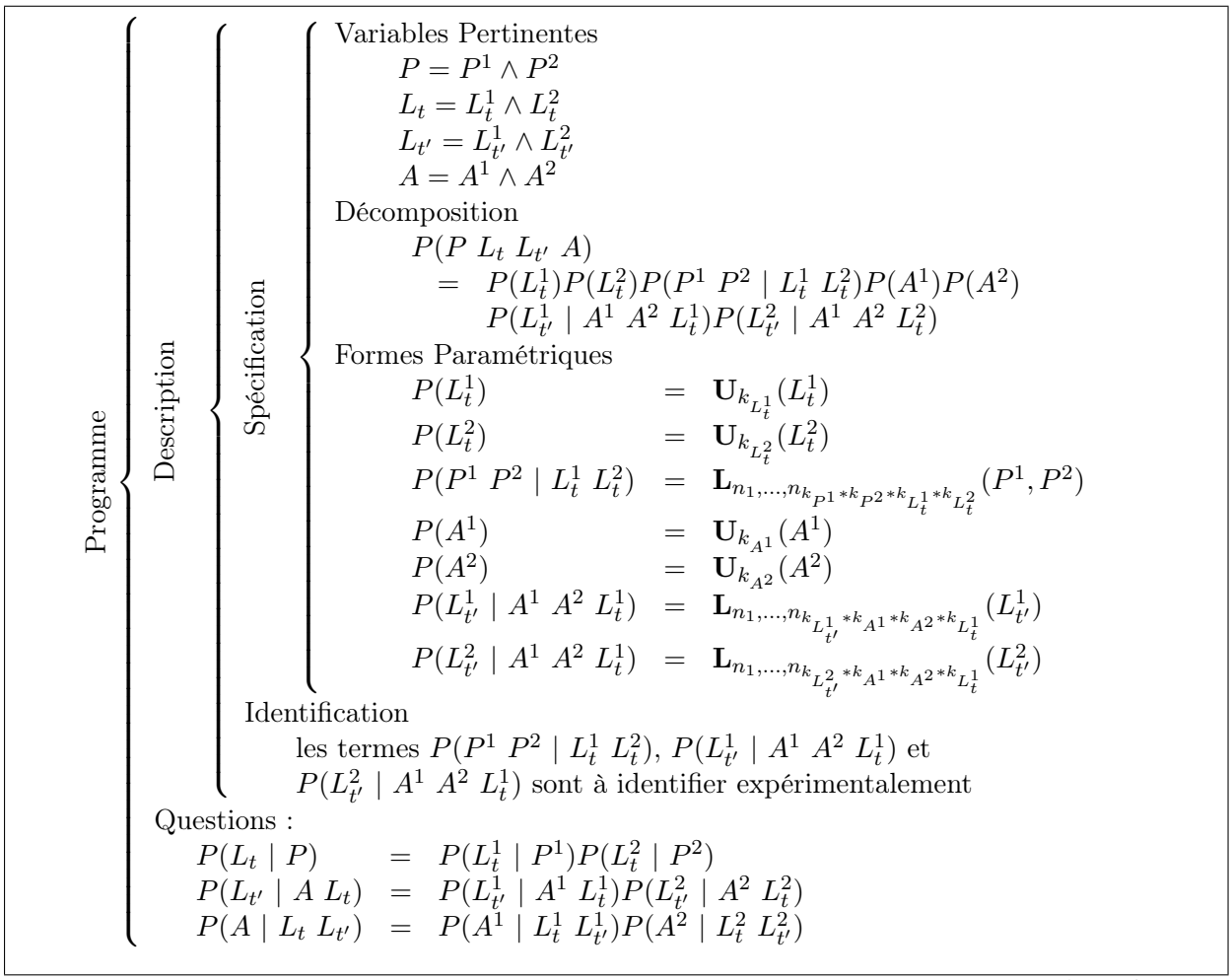


Fig. 8.11: La définition de la superposition de cartes bayésiennes (cas $A = A^1 \wedge A^2$, avec ajout d'information).

superposée, de la sommation des deux puits de potentiels de c^1 et c^2, est dépendante des potentiels E_1 et E_2. Prenons comme exemple deux potentiels quadratiques en fonction de la distance à la source attirante. Plaçons nous dans le repère centré en p_1, notons $(k, 0)$ les coordonnées de p_2 dans ce repère (voir Figure 8.12), et écrivons : $E_1(x, y) = x^2 + y^2$ et $E_2(x, y) = (x - k)^2 + y^2$. Si nous sommons ces deux fonctions potentielles, nous trouvons le potentiel qui sera utilisé par la carte superposée :

$$E(x, y) = E_1 + E_2 = 2x^2 - 2kx + k^2 + 2y^2,$$

et dérivons le pour trouver le champ de gradient résultant :

$$\frac{\partial E(x, y)}{\partial x} = 4x - 2k \; ; \frac{\partial E(x, y)}{\partial y} = 4y.$$

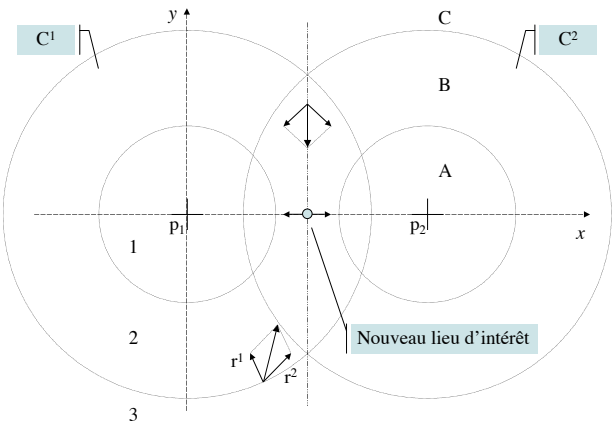


FIG. 8.12: Création d'un nouveau lieu d'intérêt par superposition de deux cartes basées sur des gradients circulaires.

Le point d'attraction sera donc aux coordonnées :

$$\frac{\partial E(x,y)}{\partial x} = 0 \Leftrightarrow x = k/2 \; ; \frac{\partial E(x,y)}{\partial y} = 0 \Leftrightarrow y = 0.$$

Nous comprenons donc qu'appliquer simultanément $remonter^1$ et $remonter^2$ fera converger vers ce point de coordonnées $(k/2, 0)$.

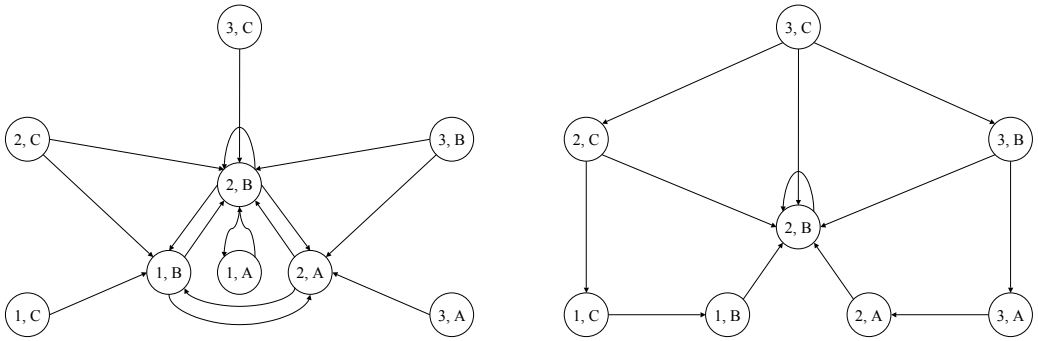
L'existence de ce nouveau point d'intérêt de l'espace, qui n'a de sens ni dans c^1, ni dans c^2, pourra être identifiée grâce aux termes $P(L^1_{t'} \mid A^1\ A^2\ L^1_t)$ et $P(L^2_{t'} \mid A^1\ A^2\ L^2_t)$. Le robot pourra en effet apprendre que, par l'application conjointe de $remonter^1$ et $remonter^2$, la zone 1 n'est plus un point d'arrivée, alors qu'elle l'était dans c^1. En effet, si le robot passe par la zone 1, il continue ensuite à se déplacer vers le nouveau point d'intérêt, ce qui crée des données expérimentales telles que

$$P(L^1_{t'} \mid [A^1 = remonter^1]\ [A^2 = remonter^2]\ [L^1_t = 1])$$

n'est plus un Dirac centré sur $L^1_{t'} = 1$. Sachant qu'on se trouve en zone 1, sachant qu'on remonte les deux gradients, on ne reste plus systématiquement en zone 1.

Cependant, pris individuellement, les termes $P(L^1_{t'} \mid A^1\ A^2\ L^1_t)$ et $P(L^2_{t'} \mid A^1\ A^2\ L^2_t)$ ne vont pas pouvoir capturer la nouvelle structure de l'espace. Ceci est normal, L^1_t ou L^2_t ne sont pas des représentations assez riches pour ce nouveau phénomène. Dans la carte superposée, le produit de ces deux termes induira le graphe présenté Figure 8.13(a). Certains éléments présents dans ce graphe n'ont aucun sens par rapport à la situation présentée Figure 8.12, comme par exemple :

- le nœud $\langle 1, A\rangle$ n'existe pas physiquement, aucun arc ne devrait y arriver ;
- aucun arc ne devrait sortir de $\langle 2, B\rangle$, puisque ce lieu est le nouveau point d'attraction ;
- etc.



(a) Graphe identifié expérimentalement dans la carte superposée.

(b) Le graphe identifié dans une carte bayésienne portant sur l'espace conjoint $L_t^1 \wedge L_t^2$.

FIG. 8.13: Les limites de l'opérateur de superposition : le graphe de gauche contient des erreurs par rapport au graphe « réel », à droite.

Finalement, si l'on calcule le graphe induit par le terme $P(L_{t'}^1\ L_{t'}^2 \mid A^1\ A^2\ L_t^1\ L_t^2)$ appris expérimentalement, plutôt que le produit de $P(L_{t'}^1 \mid A^1\ A^2\ L_t^1)$ et $P(L_{t'}^2 \mid A^1\ A^2\ L_t^2)$, nous obtenons le graphe Figure 8.13(b). Celui ci est plus conforme à la topologie réelle de la situation courante. Cet exemple illustre donc bien les limites qu'apportent les hypothèses simplificatrices faites dans le cadre de l'opérateur de superposition.

Avant d'aborder la discussion, nous souhaitons revenir un instant sur l'exemple des gradients rectilignes orthogonaux que nous avions défini précédemment (Section 8.2.2.3).

Supposons que le programmeur ait superposé les cartes de gradients rectilignes, avec $A = A^1 \oplus A^2$, puis qu'il ait appris les dépendances $P(P^1\ P^2 \mid L_t^1\ L_t^2)$ et $P(L_{t'}^1\ L_{t'}^2 \mid A\ L_t^1\ L_t^2)$ avec succès. Constatant que cette dernière forme ne présente pas de redondance, et qu'au contraire elle permet une description intéressante de l'environnement pour le robot, il teste sur celui-ci la réalisation de tâches qui était auparavant impossibles dans c^1 ou c^2. Par exemple, il demande au robot de rejoindre la case $\langle 1, A\rangle$. Grâce à la carte superposée obtenue par apprentissage, le robot pourra résoudre cette tâche, mais seulement en alternant les consignes $remonter^1$ et $remonter^2$ car A^1 et A^2 ne peuvent être appliquées simultanément. Si le programmeur trouve ce comportement insatisfaisant, il peut alors chercher à re-superposer les cartes c^1 et c^2, dans la variante $A = A^1 \wedge A^2$. Dans ce cas, il sera possible d'apprendre les termes $P(L_{t'}^1 \mid A^1\ A^2\ L_t^1)$ et $P(L_{t'}^2 \mid A^1\ A^2\ L_t^2)$ avec succès, ce que nous ne développerons pas ici. Une fois ces termes appris, le robot pourra appliquer simultanément les actions $remonter^1$ et $remonter^2$, ce qui créera une trajectoire « diagonale », plus plaisante à l'œil. Le coût de cette amélioration étant bien évidemment le problème de définir, pour le programmeur, à un niveau d'abstraction inférieur, un modèle permettant de combiner (au sens de [LEBELTEL99]) les actions $remonter^1$ et $remonter^2$.

8.3.2.4 Discussion

Nous avons déjà discuté dans les Sections précédentes de points généraux relatifs à la superposition de cartes. Nous souhaitons maintenant apporter un élément de réponse à la question suivante :

Q7. Comment spécifier une carte ? Nous avons vu dans la première section de ce chapitre que le cas le plus simple de superposition était la variante $A = A^1 \oplus A^2$. Nous avons développé un exemple de gradients orthogonaux dans lequel, après expérimentation, le programmeur pouvait se rendre compte que la carte obtenue avait de bonnes propriétés. En l'occurrence, la carte apprise, qui contenait de nouvelles informations, permettait de résoudre des tâches nouvelles, par rapport aux deux cartes sous-jacentes. Grâce à ce succès, nous avons noté que le programmeur pouvait tenter d'améliorer le résultat, en passant à la variante $A = A^1 \wedge A^2$. Le prix à payer était l'écriture d'une description qui permette de combiner, à plus bas niveau, les actions a^1 et a^2. Nous avons ensuite vu que le programmeur, par l'expérience, avait un moyen de constater les limites de la carte superposée obtenue. Si la tâche ne peut se résoudre à cause de ces limites, il faut passer à une carte dans l'espace conjoint $L_t^1 \wedge L_t^2$. Notons que dans ce cas, les étapes précédentes ont fourni des outils pour écrire cette carte bayésienne sur cet espace plus complexe. Notamment, tout ce processus de développement incrémental aura permis au programmeur d'identifier la pertinence de certaines hypothèses, mais lui aura aussi fait écrire des éléments qu'il pourra réutiliser pour obtenir la carte finale (le module de combinaison d'actions, certaines termes de localisation, etc.)

L'opérateur de superposition peut donc être vu non seulement comme un moyen d'obtenir des cartes plus riches, mais aussi comme un moyen de vérifier si la carte superposée obtenue est satisfaisante. Cet opérateur s'intègre donc de manière intéressante, comme un guide dans le cycle de développement d'un programme robotique : à la fois comme outil pour obtenir des cartes, mais aussi comme outil pour évaluer leur validité, leur intérêt, leur richesse.

8.4 Conclusion sur la superposition de cartes

Pour conclure, nous présentons quelques extensions possibles de l'opérateur de superposition.

Dans ce chapitre, nous avons présenté des superpositions pour *deux* cartes sous-jacentes. Mathématiquement, l'extension des superpositions sans apprentissage pour le cas de n cartes est trivial. Cependant, il pourrait s'avérer difficile, dans les phases d'apprentissage, de tester la suppression de plusieurs hypothèses d'indépendances conditionnelles en même temps. Cependant, un cycle de développement par étapes, avec superpositions de cartes deux à deux, est envisageable.

Une autre extension possible concerne le choix des décompositions intervenant dans les cartes superposées. Dans nos exemples, les cartes sous-jacentes et les cartes obtenues par

superposition suivaient des décompositions de type localisation markovienne. Ce choix a l'avantage d'être simple à présenter, mais il ne doit pas faire oublier le fait que, *in fine*, notre formalisme ne contraint pas le choix de la décomposition. En effet, dans certaines expériences, certaines dépendances, autres que celles de la localisation markovienne, peuvent se révéler cruciales à identifier. Il peut donc être intéressant de choisir dans la carte superposée, une décomposition différente de celle de la localisation markovienne. Le fait de coller – ou non – à la lettre à notre opérateur de superposition devient un critère supplémentaire à prendre en compte par le programmeur : si d'autres critères l'y poussent, il peut s'écarter de notre choix d'écriture, et réaliser une superposition de cartes différemment, pour peu qu'il ait bien compris ce que l'on nommait ici « superposition ».

Finalement, notons que, bien qu'ayant présenté un opérateur d'assemblage de cartes, nous ne pouvons encore parler de *hiérarchies* de cartes. En effet, bien que la carte superposée sans ajout d'information utilise les cartes sous-jacentes en tant que ressources, cela n'est plus tout à fait le cas si l'on inclut de l'apprentissage. Dans ce cas, certaines connaissances identifiées expérimentalement peuvent remplacer les cartes sous-jacentes. Nous aborderons donc au chapitre suivant un opérateur d'assemblage de cartes qui répondra à ce point.

Chapitre 9

Assemblages de cartes : abstraction

> « I would often ask Miss R. what she was thinking about.
> 'Nothing, just nothing', she would say.
> 'But how can you possibly be thinking of nothing ?'
> 'It's dead easy, once you know how.'
> '*How* exactly do you think about nothing ?'
> 'One way is to think about the same thing again and again. [...]
> ... And then there are maps.'
> 'Maps ? What do you mean ?'
> 'Everything I do is a map of itself, everything I do is a part of itself. [...] I think of a map ; then a map of that map ; then a map of that map of that map, and each map perfect, though smaller and smaller... Worlds within worlds within worlds within worlds... ' »
>
> Dialogue (réel) entre le Docteur Sacks et Miss R., patiente atteinte d'*encephalitis lethargica* ; Oliver Sacks, *Awakenings*, First Vintage Books Ed., New-York, 1999, pp. 75–76 [Sacks99].

Au chapitre précédent, nous avons montré différentes manières d'assembler des cartes décrivant le même espace géographique, qui se superposaient sur cet espace. Nous allons, dans ce chapitre, expliquer comment assembler des cartes qui décrivent des espaces géographiques *différents*. Nous appellerons cet opérateur d'assemblage l'*abstraction* de carte, car, intuitivement, il s'agira de créer une carte de plus haut niveau d'abstraction, dont chaque lieu représentera une carte de plus bas niveau d'abstraction, et dont chaque action sera un comportement défini dans une carte sous-jacente.

9.1 Description

Nous supposons disposer de n cartes bayésiennes, $c^1 \dots c^n$, qui portent sur les variables P^i, L^i_t, $L^i_{t'}$ et A^i, pour i allant de 1 à n, et qui répondent de manière pertinente aux trois questions de localisation $P(L^i_t \mid P^i)$, de prédiction $P(L^i_{t'} \mid A^i\ L^i_t)$, et de contrôle $P(A^i \mid L^i_t\ L^i_{t'})$. La carte abstraite c a pour variable interne L_t, une variable possédant n cas $c^1 \dots c^n$, représentant chacun une carte sous-jacente. Sa variable de perception P est la conjonction de toutes les variables qui apparaissent dans les cartes sous-jacentes. Quant

à la variable d'action, A, elle rassemble l'ensemble des k comportements $\{a_1, a_2, \ldots, a_k\}$ issus des cartes sous-jacentes.

La carte c décrit l'environnement à un niveau de détail plus petit que les cartes sous-jacentes. Nous disposons dans c d'autant de lieux qu'il y a de cartes sous-jacentes, donc nous avons un petit nombre de lieux. Pour relier ces lieux entre eux, nous ne retenons que les comportements définis dans les cartes sous-jacentes, qui sont également en petit nombre. Ainsi, la carte c est l'abstraction à la fois des cartes et des comportements qui y sont définis. Elle sera donc d'une part facile à définir, car portant sur des petits espaces, mais d'autre part elle sera très riche (l'application d'un mouvement à ce niveau d'abstraction se traduira par l'exécution d'un comportement, éventuellement complexe, à plus bas niveau d'abstraction).

Nous définissons maintenant le schéma général de la description c.

1. Spécification des connaissances préalables c

   (a) Variables :

   P $P = \bigwedge_{i=1}^{n} (P^i \wedge L_t^i \wedge L_{t'}^i \wedge A^i), \mathcal{D}_P = \prod_{i=1}^{n} \mathcal{D}_{P^i} \times \mathcal{D}_{L_t^i} \times \mathcal{D}_{L_{t'}^i} \times \mathcal{D}_{A^i}, k_P = \prod_{i=1}^{n} k_{P^i} k_{L_t^i} k_{L_{t'}^i} k_{A^i}$. Notons que certains des P^i ou A^i peuvent être identiques, si par exemple deux cartes sous-jacentes utilisent les mêmes capteurs, ce qui est courant en pratique. En revanche, cela ne sera généralement pas le cas pour les variables internes L_t^i et $L_{t'}^i$, car des cartes différentes décrivent des phénomènes différents avec des « vocabulaires » internes différents. S'il arrive donc que certaines variables apparaissent plusieurs fois dans les variables P^i ou A^i, nous les considérerons tout de même distinctes (elles seront indicées par la carte dont elles proviennent).

   L_t $L_t : \mathcal{D}_{L_t} = \{c^1, c^2, \ldots, c^n\}, k_{L_t} = n,$

   $L_{t'}$ $L_{t'} : \mathcal{D}_{L_{t'}} = \{c^1, c^2, \ldots, c^n\}, k_{L_{t'}} = n,$

   A $A : \mathcal{D}_A = \{a_1, a_2, \ldots, a_k\}, k_A = k.$

   (b) Décomposition : Nous choisissons la décomposition suivante :

$$
\begin{aligned}
& P(P\ L_t\ L_{t'}\ A) \\
&\quad = \ P(P^1\ L_t^1\ L_{t'}^1\ A^1\ \ldots\ P^n\ L_t^n\ L_{t'}^n\ A^n\ L_t\ L_{t'}\ A) \\
&\quad = \ P(L_t) \left(\prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \right) P(L_{t'}) P(A \mid L_t\ L_{t'})
\end{aligned}
$$

   Ici, nous avons choisi d'exprimer une hypothèse d'indépendance conditionnelle

forte. En effet, le produit des termes $P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t)$ provient de :

$$\begin{aligned}
&\prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \\
&\quad = \quad P(P^1\ L_t^1\ L_{t'}^1\ A^1 \mid L_t) \\
&\qquad\quad P(P^2\ L_t^2\ L_{t'}^2\ A^2 \mid L_t\ P^1\ L_t^1\ L_{t'}^1\ A^1) \\
&\qquad\quad \ldots \\
&\qquad\quad P(P^n\ L_t^n\ L_{t'}^n\ A^n \mid L_t\ P^{n-1}\ L_t^{n-1}\ L_{t'}^{n-1}\ A^{n-1}\ \ldots\ P^1\ L_t^1\ L_{t'}^1\ A^1).
\end{aligned}$$

Nous faisons l'hypothèse que, si l'on connaît le lieu L_t dans la carte c, alors les variables $P^1\ L_t^1\ L_{t'}^1\ A^1$ n'apportent aucune connaissance supplémentaire sur les variables $P^2\ L_t^2\ L_{t'}^2\ A^2$ (et ainsi de suite pour les autres termes).

(c) Formes paramétriques :

$P(L_t)$ À ce terme est associée une récurrence.

$P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t)$ De manière similaire à la définition du terme $P(L_{t'}^1 \mid A\ L_t^1)$ de la Section 8.2.1, ce terme est soit une question probabiliste à la carte sous-jacente, soit une uniforme. Nous notons :

$$\begin{aligned}
&P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid [L_t = c]) \\
&\quad = \quad \begin{cases} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid c^i) \text{ si } c = c^i \\ \mathbf{U}_{k_{P^i} k_{L_t^i} k_{L_{t'}^i} k_{A^i}}(P^i, L_t^i, L_{t'}^i, A^i) \text{ sinon} \end{cases}
\end{aligned}$$

$P(L_{t'})$ À ce terme est associée une loi uniforme, car nous n'avons pas d'a priori sur la consigne à atteindre.

$P(A \mid L_t\ L_{t'})$ Comme ce terme porte sur un espace discret de faible cardinalité, nous lui associons un ensemble de lois de succession de Laplace. Nous notons :

$$P(A \mid L_t\ L_{t'}) = \mathbf{L}_{n_1,\ldots,n_{k_A k_{L_t} k_{L_{t'}}}}(A).$$

2. Identification : Il reste à définir l'ensemble des lois de succession de Laplace pour le terme $P(A \mid L_t\ L_{t'})$. Grâce au mécanisme de l'abstraction, on ne retient des cartes sous-jacentes que leur nom, et les noms des comportements qui y sont définis, ce qui représente une réduction d'information très importante. Donc le terme $P(A \mid L_t\ L_{t'})$ est de taille très réduite.

   Il sera donc en pratique assez aisé de spécifier ce terme à la main (spécification *a priori*), car le programmeur a en général une idée des zones couvertes par les cartes qu'il assemble. Il est également capable d'estimer leur agencement relatif, donc il sait approximativement le résultat d'application d'un comportement d'une carte, notamment pour juger si tel comportement maintient le robot dans la zone d'une carte, ou, s'il l'en fait sortir, vers quelle autre zone le robot va se diriger.

Alternativement, la petite taille de ce terme nous permet de l'apprendre expérimentalement : en effet, il sera possible, dans une phase exploratoire, de laisser le robot tester toutes les transitions possibles. Selon la carte courante, seul un petit nombre de comportements est applicable. On en choisit un, $A = a$, on l'applique, et on collecte des données expérimentales sur les variables L_t et $L_{t'}$, ce qui permet d'identifier $P(A \mid L_t\ L_{t'})$. Le nombre de cartes assemblées étant également réduit, le nombre de combinaisons possibles en partie droite est petit, ce qui permet de les explorer expérimentalement de manière exhaustive.

9.2 Résolution des questions de localisation, prédiction et contrôle

Nous vérifions ici que la carte c choisie répond bien de manière pertinente aux trois questions de localisation, prédiction et contrôle.

Commençons par la résolution de la question de localisation :

$$\begin{aligned}
P(L_t \mid P) &= P(L_t \mid P^1\ L_t^1\ L_{t'}^1\ A^1\ \ldots\ P^n\ L_t^n\ L_{t'}^n\ A^n) \\
&= \frac{1}{Z_1} \sum_{A\ L_{t'}} P(L_t) \left(\prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \right) P(L_{t'}) P(A \mid L_t\ L_{t'}) \\
&= \frac{1}{Z_1} P(L_t) \left(\prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \right) \sum_{A\ L_{t'}} P(L_{t'}) P(A \mid L_t\ L_{t'}) \\
&= \frac{1}{Z_2} P(L_t) \prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \\
P(L_t \mid P) &= \frac{1}{Z_3} \prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t)
\end{aligned}$$

Nous voyons que cette inférence inverse les termes $P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t)$. La présence ou non de signal dans le résultat est surtout dépendante de la pertinence des cartes par rapport à l'endroit courant où se trouve le robot. Si aucune carte ne décrit correctement le phénomène que vit le robot, chacun des termes $P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t)$ a une probabilité égale, et le robot est incapable de choisir un lieu plutôt qu'un autre ($P(L_t \mid P)$ proche de l'uniforme). En revanche, si l'une des cartes est pertinente, le modèle bayésien associé a une forte probabilité, et le lieu le plus probable est celui qui correspond à cette carte. En ce sens, cette inférence met donc en compétition les cartes, ce qui peut également s'interpréter comme de la *reconnaissance de modèle* probabiliste : pour une situation donnée, chacune des cartes essaie de « l'expliquer », en ses termes, et la carte qui explique le mieux cette situation est considérée comme la carte la plus pertinente. Nous renvoyons le lecteur intéressé à [Diard99] pour nos travaux précédents qui incluent aussi cette notion de reconnaissance de modèle, mais dans une expérience d'apprentissage hiérarchique.

Pour le programmeur, nous comprenons donc qu'il doive fournir des cartes pertinentes, qui couvrent tout l'espace où le robot peut se trouver. Rappelons tout de même que les cartes en question ne sont pas juste de « modèles d'endroits », mais des *modèles de l'interaction entre le robot et l'environnement*. Il faut donc que les cartes décrivent les endroits où va le robot, mais en tenant compte de la *manière dont il y va*. Le robot reconnaît les endroits non seulement par ce qu'il y perçoit, mais aussi par ce qu'il y fait. Si nous insistons sur ce point, c'est notamment car il est toujours oublié dans la littérature, où les modèles de perception de l'environnement (cartes, modules de localisation) sont systématiquement développés *indépendamment* des modules de contrôle (planification, navigation). Si l'on oublie ce point, une solution, très utilisée dans la littérature, est de se rabattre sur des modèles de l'environnement *très précis*, donc métriques, mais qui sont très lourds à mettre en œuvre et à maintenir.

Le terme de contrôle est déjà présent dans la décomposition, donc il n'y a pas d'inférence à faire, et la présence ou non de signal sur ce terme dépendra de sa définition (spécification a priori ou phase d'apprentissage).

De même, le calcul du terme de prédiction dépend de la définition du terme de contrôle, ce que nous constatons au vu de l'inférence suivante :

$$
\begin{aligned}
& P(L_{t'} \mid A\ L_t) \\
&= \frac{1}{Z_1} \sum_{P^i\ L_t^i\ L_{t'}^i\ A^i} P(L_t) \left(\prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \right) P(L_{t'}) P(A \mid L_t\ L_{t'}) \\
&= \frac{1}{Z_1} P(L_t) P(L_{t'}) P(A \mid L_t\ L_{t'}) \sum_{P^i\ L_t^i\ L_{t'}^i\ A^i} \left(\prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \right) \\
&= \frac{1}{Z_2} P(L_t) P(L_{t'}) P(A \mid L_t\ L_{t'}) \\
&= \frac{1}{Z_3} P(A \mid L_t\ L_{t'})
\end{aligned}
$$

Nous résumons l'opérateur d'abstraction de cartes Figure 9.1.

9.3 Expérience

Dans cette expérience, nous reprenons une variante du premier exemple de carte bayésienne définie dans la Section 6.3. Il s'agit de décrire l'environnement du robot, perçu par les proximètres, en utilisant des termes tels que « mur », « coin », ou « libre ». Or, dans cette expérience, nous avions donné un algorithme déterministe pour déterminer la situation courante du robot (voir l'Algorithme 6.3). Nous présentons ici une méthode pour réaliser cette détection de situation, par abstraction de cartes.

Pour ce faire, nous écrivons tout d'abord une carte bayésienne pour chacune de ces situations. Ces trois cartes, c^{mur}, c^{coin} et c^{libre}, que nous allons brièvement résumer dans les sections suivantes, sont définies complètement Annexe E.

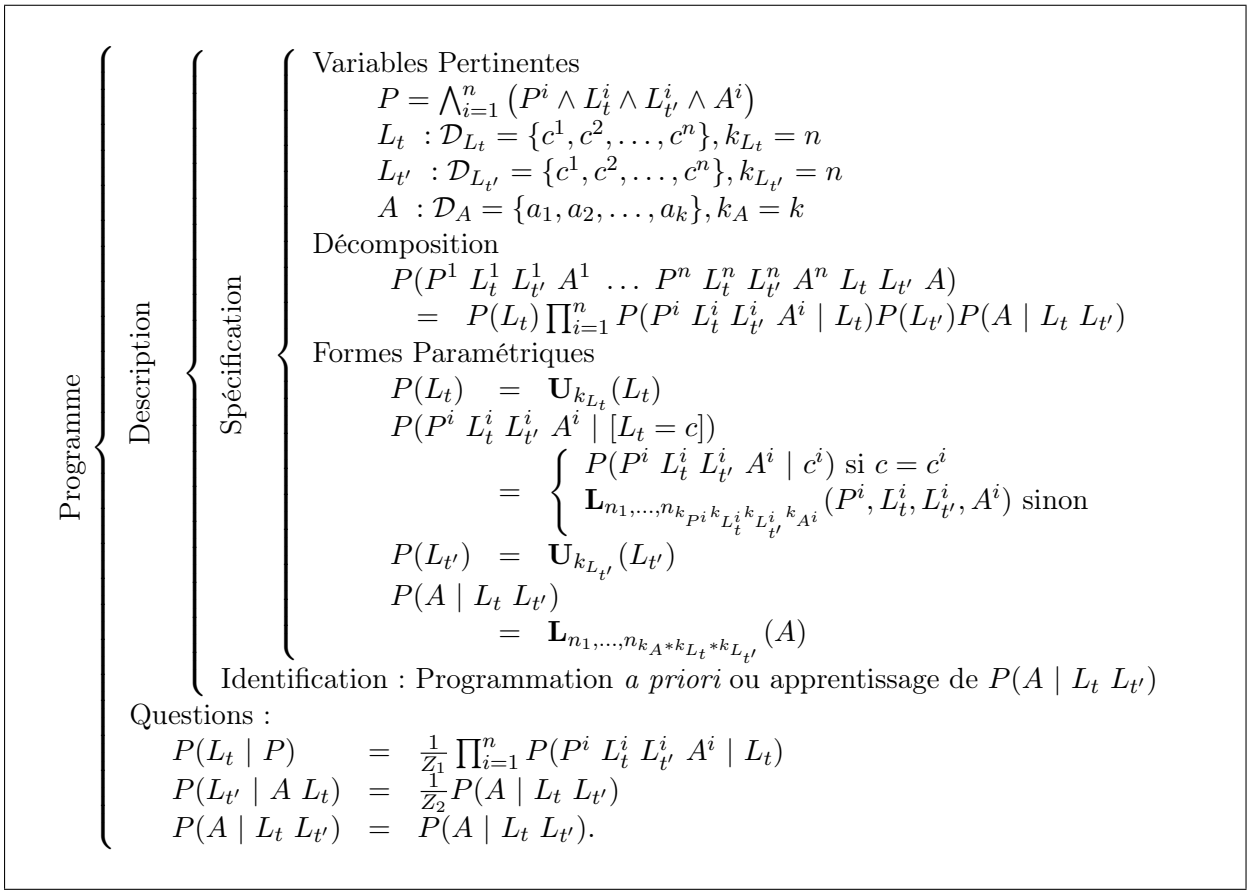


Fig. 9.1: La définition de l'abstraction de cartes bayésiennes.

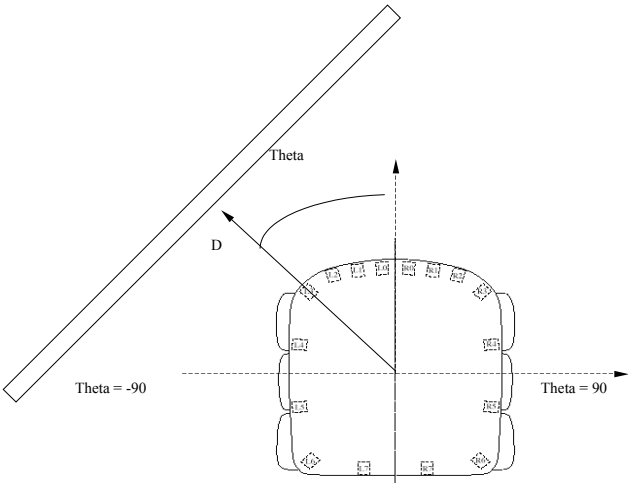
9.3.1 Les trois cartes c^{mur}, c^{coin} et c^{libre}

9.3.1.1 Résumé de la carte du mur c^{mur}

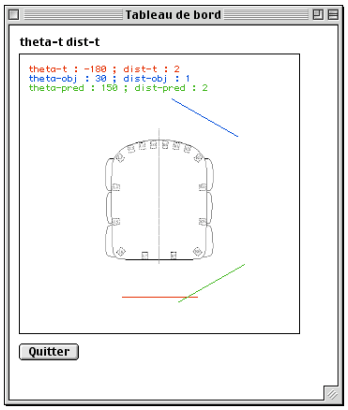
Dans cette expérience, nous plaçons le robot en présence d'un mur, que nous supposons suffisamment long par rapport au robot pour qu'il n'en voit pas les extrémités. Les comportements que nous souhaitons réaliser grâce à la carte c^{mur} sont de suivre le mur, sur la gauche ou la droite, s'en écarter, ou atteindre quelques positions particulières par rapport au mur (par exemple, très proche et face au mur, ou très proche et dos au mur, etc.).

Nous choisissons de décrire le mur par deux variables internes, un angle Θ_t et une distance $Dist_t$ (voir Figure 9.2(a)). Bien que ces variables aient une forte connotation « géométrique », nous décidons de nous passer d'une forte résolution sur ces variables. En effet, nous n'avons pas besoin d'une représentation très fine pour les tâches à accomplir dans cette situation.

Grâce à c^{mur}, nous pouvons générer différents programmes de contrôle robotique, en fixant des valeurs consignes pour les variables $\Theta_{t+\Delta t}$ et $Dist_{t+\Delta t}$. Par exemple, pour obtenir un suivi de mur où le robot garde le mur sur sa droite (comportement $SuiviD$), nous fixons comme consigne $\Theta_{t+\Delta t} = 90$ et $Dist_{t+\Delta t} = 1$, afin que le robot cherche à atteindre cette



(a) Variables retenues pour la description d'un mur.



(b) Copie d'écran du module de visualisation pour la carte c^{mur}.

FIG. 9.2: Carte bayésienne c^{mur}.

situation, où l'angle du mur perçu est de 90°, et la distance est modérée. Cela correspond à poser et résoudre la question probabiliste suivante :

$$P(Vrot\ Vtrans \mid [\Theta_{t+\Delta t} = 90]\ [Dist_{t+\Delta t} = 1]\ Px).$$

De manière similaire, nous définissons un comportement $SuiviG$ de suivi du mur sur la gauche (consigne $\Theta_{t+\Delta t} = -90$, $Dist_{t+\Delta t} = 1$), un comportement $\acute{E}loignerMur$ (consigne $\Theta_{t+\Delta t} = -180$, $Dist_{t+\Delta t} = 2$), un comportement $FaceAuMur$ (consigne $\Theta_{t+\Delta t} = 0$, $Dist_{t+\Delta t} = 0$), un comportement $DosAuMur$ (consigne $\Theta_{t+\Delta t} = -180$, $Dist_{t+\Delta t} = 0$), et enfin, un comportement $SuiviGD$ de suivi du mur dans un sens puis dans l'autre, en changeant de sens toutes les 20 secondes (alternance des consignes qui définissent $SuiviD$ et $SuiviG$).

Nous rapportons ici des résultats expérimentaux très satisfaisants : cette carte contient suffisamment d'informations pour réaliser les comportements cités ci-dessus. Nous montrons Figure 9.2(b) une copie d'écran du module de visualisation des variables internes Θ_t, $Dist_t$ (angle et distance au mur à l'instant t), Θ_{obj}, et $Dist_{obj}$ (angle et distance au mur donnés en consigne au terme de contrôle), et Θ_{pred}, $Dist_{pred}$, (angle et distance au mur prédit pour le pas de temps suivant).

9.3.1.2 Résumé de la carte du coin c^{coin}

Nous choisissons ici une représentation grossière des coins, en n'utilisant qu'une variable symbolique, $Coin_t$, qui représentera quatre situations caractéristiques : AvG lorsque le coin est devant et à gauche du robot, AvD lorsque le coin est devant et à droite du robot, ArG

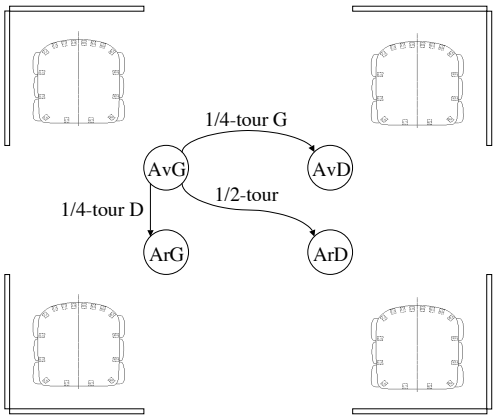


FIG. 9.3: La variable retenue pour la description d'un coin, ses valeurs, et un extrait du graphe induit.

lorsque le coin est derrière et à gauche du robot, et ArD lorsque le coin est derrière et à droite du robot. Ces différentes situations sont présentées Figure 9.3. Cette figure présente également un extrait du graphe induit, où les labels « 1/4-tour G », « 1/2-tour » résument en réalité des commandes sur $Vrot$ et $Vtrans$.

La carte c^{coin} permet d'obtenir, par les comportements élémentaires qui ne consistent qu'en une consigne à atteindre pour la variable $Coin_t$ (il y en a 4 possibles), les comportements suivants :

- nous obtenons un comportement qui permet de « passer » le coin et longer le mur gauche en posant la consigne $[Coin_{t+\Delta t} = ArG]$ au robot (comportement $PasserG$) ;
- de manière similaire nous obtenons un comportement $PasserD$:
- enfin, nous obtenons deux comportements où le robot s'arrête dans le coin, en posant les consignes $[Coin_{t+\Delta t} = AvG]$ et $[Coin_{t+\Delta t} = AvD]$.

9.3.1.3 Résumé de la carte de l'espace libre c^{libre}

Dans cette carte, il s'agit d'identifier les cas où le robot est en espace libre, c'est-à-dire où aucun de ses capteurs de proximétrie n'est excité. Malheureusement, ce phénomène est tellement simple qu'il ne varie jamais : nous ne pouvons pas *différencier* plusieurs aspects de ce phénomène « espace libre » (alors que c'est effectivement le rôle des variables internes des cartes bayésienne en général : les variables Θ_t et $Dist_t$ permettent de caractériser différentes facettes du même phénomène « mur »). Nous choisissons donc ici de représenter l'espace libre, par opposition à l'espace encombré. Nous nous donnons une variable $Espace_t$, qui prend les valeurs $\{vide, encombré\}$.

Dans cette carte, nous résolvons les tâches $Stop$ et $ToutDroit$, qui consistent respectivement à s'arrêter et aller tout droit.

9.3.2 Objectifs et protocole expérimental

9.3.2.1 Choix d'une variable de lieux

Nous disposons maintenant des trois cartes c^{mur}, c^{coin} et c^{libre} : définissons la carte abstraite c^{prox}, qui se base sur ces trois cartes.

Rappelons à ce stade du document que nous allons abstraire les trois cartes, qui modélisent trois situations particulières : soit le robot ne perçoit rien sur aucun de ses capteurs (cas *vide* de la carte c^{libre}), soit il perçoit un phénomène qu'il reconnaît comme étant un mur (carte c^{mur}), soit il perçoit un phénomène qu'il reconnaît comme étant un coin (carte c^{coin}). La variable de lieux L_t aura donc trois cas possibles, $\{c^{mur}, c^{coin}, c^{libre}\}$. De ces trois cartes sous-jacentes, nous tirons un petit nombre de comportements : de c^{mur} nous retenons les comportements $Stop$, $SuiviD$ et $\acute{E}loignerMur$, de c^{coin} nous gardons $PasserD$, enfin, de la carte c^{libre}, nous conservons $ToutDroit$ et $Stop$. Ces comportements constituent les cas de la variable d'action A_{prox} de la carte abstraite.

9.3.2.2 Graphe induit

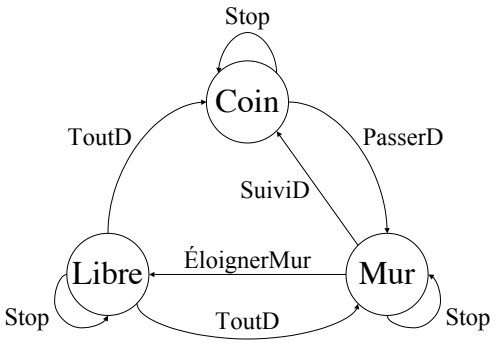


FIG. 9.4: Un extrait du graphe induit par la carte abstraite c^{prox}.

Rappelons qu'un extrait du graphe induit par la carte c^{prox} est montré Figure 6.1(b) (nous le reproduisons ici, Figure 9.4).

9.3.3 Description

Nous appliquons notre opérateur d'abstraction pour définir la carte c^{prox}.

1. Spécification des connaissances préalables c^{prox}
   (a) Variables :
   P Cette variable regroupe l'ensemble des variables qui apparaissent dans les

cartes c^{mur}, c^{coin} et c^{libre} :

$$\begin{aligned} P &= Px^{mur} \wedge Px^{coin} \wedge Px^{libre} \wedge \Theta_t \wedge Dist_t \wedge Coin_t \wedge Espace_t \\ &\quad \wedge \Theta_{t+\Delta t} \wedge Dist_{t+\Delta t} \wedge Coin_{t+\Delta t} \wedge Espace_{t+\Delta t} \wedge Vrot^{mur} \\ &\quad \wedge Vtrans^{mur} \wedge Vrot^{coin} \wedge Vtrans^{coin} \wedge Vrot^{libre} \wedge Vtrans^{libre}, \\ \mathcal{D}_P &= \mathcal{D}_{Px^{mur}} \times \mathcal{D}_{Px^{coin}} \times \mathcal{D}_{Px^{libre}} \times \mathcal{D}_{\Theta_t} \times \mathcal{D}_{Dist_t} \times \mathcal{D}_{Coin_t} \times \mathcal{D}_{Espace_t} \\ &\quad \times \mathcal{D}_{\Theta_{t+\Delta t}} \times \mathcal{D}_{Dist_{t+\Delta t}} \times \mathcal{D}_{Coin_{t+\Delta t}} \times \mathcal{D}_{Espace_{t+\Delta t}} \times \mathcal{D}_{Vrot^{mur}} \\ &\quad \times \mathcal{D}_{Vtrans^{mur}} \times \mathcal{D}_{Vrot^{coin}} \times \mathcal{D}_{Vtrans^{coin}} \times \mathcal{D}_{Vrot^{libre}} \times \mathcal{D}_{Vtrans^{libre}}, \\ k_P &= 16^{16*3} * (12 * 3 * 4 * 2)^2 * 101^6. \end{aligned}$$

$\boldsymbol{L_t}$ Sit_t : $\mathcal{D}_{Sit_t} = \{c^{mur}, c^{coin}, c^{libre}\}, k_{Sit_t} = 3$,

$\boldsymbol{L_{t'}}$ $Sit_{t+\Delta t}$: $\mathcal{D}_{Sit_t+\Delta t} = \{c^{mur}, c^{coin}, c^{libre}\}, k_{Sit_{t+\Delta t}} = 3$,

$\boldsymbol{A}$ A_{prox} : $\mathcal{D}_{A_{prox}} = \{Stop,\ ToutDroit,\ SuiviD,\ \acute{E}loignerMur,\ PasserD\}$, $k_{A_{prox}} = 5$.

(b) Décomposition : Nous appliquons la décomposition choisie :

$$\begin{aligned} &P(P\ L_t\ L_{t'}\ A) \\ &= P\left(\begin{array}{l} Px^{mur}\ Px^{coin}\ Px^{libre}\ \Theta_t\ Dist_t\ Coin_t\ Espace_t \\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Coin_{t+\Delta t}\ Espace_{t+\Delta t}\ Vrot^{mur}\ Vtrans^{mur} \\ Vrot^{coin}\ Vtrans^{coin}\ Vrot^{libre}\ Vtrans^{libre} Sit_t\ Sit_{t+\Delta t}\ A_{prox} \end{array}\right) \\ &= P(Sit_t) \\ &\quad P(Px^{mur}\ \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Vrot^{mur}\ Vtrans^{mur} \mid Sit_t) \\ &\quad P(Px^{coin}\ Coin_t\ Coin_{t+\Delta t}\ Vrot^{coin}\ Vtrans^{coin} \mid Sit_t) \\ &\quad P(Px^{libre}\ Espace_t\ Espace_{t+\Delta t}\ Vrot^{libre}\ Vtrans^{libre} \mid Sit_t) \\ &\quad P(Sit_{t+\Delta t}) P(A_{prox} \mid Sit_t\ Sit_{t+\Delta t}) \end{aligned}$$

(c) Formes paramétriques :

$P(Sit_t)$ À ce terme est associée une récurrence.

$P(Px^{mur}\ \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Vrot^{mur}\ Vtrans^{mur} \mid Sit_t)$ Ce terme est soit une question à la carte c^{mur}, soit une loi uniforme, selon la valeur de la partie droite Sit_t :

$$\begin{aligned} &P(Px^{mur}\ \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Vrot^{mur}\ Vtrans^{mur} \mid [Sit_t = c]) \\ &= \left\{\begin{array}{l} P(Px\ \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Vrot\ Vtrans \mid c^{mur}) \text{ si } c = c^{mur} \\ \mathbf{U}_{16^{16}*36^2*101^2}(Px, \Theta_t, Dist_t, \Theta_{t+\Delta t}, Dist_{t+\Delta t}, Vrot, Vtrans) \text{ sinon} \end{array}\right. \end{aligned}$$

$P(Px^{coin}\ Coin_t\ Coin_{t+\Delta t}\ Vrot^{coin}\ Vtrans^{coin} \mid Sit_t)$ De manière similaire au terme précédent :

$$\begin{aligned} &P(Px^{coin}\ Coin_t\ Coin_{t+\Delta t}\ Vrot^{coin}\ Vtrans^{coin} \mid [Sit_t = c]) \\ &= \left\{\begin{array}{l} P(Px\ Coin_t\ Coin_{t+\Delta t}\ Vrot\ Vtrans \mid c^{mur}) \text{ si } c = c^{coin} \\ \mathbf{U}_{16^{16}*4^2*101^2}(Px, Coin_t, Coin_{t+\Delta t}, Vrot, Vtrans) \text{ sinon} \end{array}\right. \end{aligned}$$

$P(Px^{libre}\ Espace_t\ Espace_{t+\Delta t}\ Vrot^{libre}\ Vtrans^{libre} \mid Sit_t)$ Ce troisième terme fait le lien avec la carte de l'espace libre :

$$\begin{aligned} & Px^{libre}\ Espace_t\ Espace_{t+\Delta t}\ Vrot^{libre}\ Vtrans^{libre} \mid [Sit_t = c]) \\ = & \begin{cases} P(Px\ Espace_t\ Espace_{t+\Delta t}\ Vrot\ Vtrans \mid c^{mur}) \text{ si } c = c^{libre} \\ \mathbf{U}_{16^{16}*2^2*101^2}(Px_0, Espace_t, Espace_{t+\Delta t}, Vrot, Vtrans) \text{ sinon} \end{cases} \end{aligned}$$

$P(Sit_{t+\Delta t})$ À ce terme est associée une loi uniforme.

$P(A_{prox} \mid Sit_t\ Sit_{t+\Delta t})$ Ce terme est défini par un ensemble de lois de succession de Laplace. Nous notons :

$$P(A_{prox} \mid Sit_t\ Sit_{t+\Delta t}) = \mathbf{L}_{n_1,\ldots,n_{6*3*3}}(A).$$

2. Identification : Nous trouvons Figure 6.2 les lois de succession de Laplace que nous avons spécifié *a priori* pour le terme $P(A_{prox} \mid Sit_t\ Sit_{t+\Delta t})$.

9.3.4 Utilisation

9.3.4.1 Programme et question probabiliste

Rappelons tout d'abord que les trois questions de localisation, prédiction et contrôle se résolvent par :

$$\begin{aligned} P(L_t \mid P) &= \frac{1}{Z_1}\prod_{i=1}^{n} P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid L_t) \\ &= \frac{1}{Z_1}\begin{pmatrix} P(Px^{mur}\ \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Vrot^{mur}\ Vtrans^{mur} \mid Sit_t) \\ P(Px^{coin}\ Coin_t\ Coin_{t+\Delta t}\ Vrot^{coin}\ Vtrans^{coin} \mid Sit_t) \\ P(Px^{libre}\ Espace_t\ Espace_{t+\Delta t}\ Vrot^{libre}\ Vtrans^{libre} \mid Sit_t) \end{pmatrix} \\ P(L_{t'} \mid A\ L_t) &= \frac{1}{Z_2}P(A \mid L_t\ L_{t'}) \\ &= \frac{1}{Z_2}P(A_{prox} \mid Sit_t\ Sit_{t+\Delta t}) \\ P(A \mid L_t\ L_{t'}) &= P(A_{prox} \mid Sit_t\ Sit_{t+\Delta t}). \end{aligned}$$

Afin d'illustrer la première inférence sur la *reconnaissance de modèle*, supposons que le robot se trouve à l'instant considéré en espace libre : tous ses proximètres sont ainsi à 0. Dans cette situation, supposons que le robot applique comportement $ToutDroit$, qui fixe $Vrot$ proche de 0 et $Vtrans$ proche de 40 (par des gaussiennes très piquées [1]). La question de localisation dans la carte abstraite met en compétition les différentes cartes. Dans la carte c^{mur}, cette situation sensorielle a peu de sens : en effet, dans c^{mur}, nous faisons l'hypothèse qu'il y a un mur à proximité du robot. Ce mur doit donc vraisemblablement

[1] C'est-à-dire des gaussiennes à écart-types très faibles.

exciter au moins l'un des capteurs. Le contraire, que tous les proximètres soient à 0, n'est pas impossible, mais fortement improbable. Bien qu'improbables, la carte décide tout de même d'un angle et d'une distance pour le mur dans cette situation. Les buts courant, venant de plus haut niveaux d'abstraction, sont alors traduits en un angle et une distance consigne à atteindre, afin de choisir une valeur pour les variables motrices $Vrot$ et $Vtrans$. Le robot se base sur une hypothèse Θ_t, $Dist_t$ qui a peu de sens, il choisit donc un ordre moteur improbable. Au final, la probabilité conjointe :

$$P(Px^{mur}\ \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Vrot^{mur}\ Vtrans^{mur} \mid Sit_t),$$

dans cette situation sensori-motrice, est une valeur très faible, notons ϵ_1. Le raisonnement est similaire en ce qui concerne la carte c^{coin}. En revanche, la carte c^{libre} verra toutes ses hypothèses vérifiées : rappelons que dans sa définition (voir Section 9.3.1.3 et Annexe E), ses termes $P(Px_i \mid Espace_t)$ sont tous des Diracs sur 0 (lorsque $[Espace_t = vide]$), le terme $P(Vrot \mid Espace_t\ Espace_{t+\Delta t})$ peut être une gaussienne très piquée sur 0, et le terme $P(Vtrans)$ une gaussienne très piquée sur 40. Le produit

$$\begin{array}{l}\left(\prod_{i=1}^{15} P([Px_i = 0] \mid [Espace_t = libre])\right)\\ P([Vrot = 0] \mid [Espace_t = libre]\ [Espace_{t+\Delta t} = encombré])\\ P([Vtrans = 40] \mid [Espace_t = libre]\ [Espace_{t+\Delta t} = encombré])\end{array}$$

vaut donc une valeur G très grande. L'inférence se poursuit en multipliant les trois valeurs G, ϵ_1 et ϵ_2 par les uniformes, dont l'influence est égale sur les trois termes, et que nous négligeons donc dans la suite. Après normalisation (constante Z_1), la question de localisation vaut G/Z_1 pour $[Sit_t = c^{libre}]$, ϵ_1/Z_1 pour $[Sit_t = c^{mur}]$ et ϵ_2/Z_1 pour $[Sit_t = c^{coin}]$, avec $G \gg \epsilon_1$ et $G \gg \epsilon_2$. Nous voyons ainsi qu'un tirage sur cette distribution rend quasi-systématiquement la valeur $[Sit_t = c^{libre}]$: le robot reconnaît que dans la situation sensori-motrice courante, la carte c^{libre} est, de loin, la plus pertinente.

9.3.4.2 Résultats

Grâce à la question de localisation, nous pouvons estimer expérimentalement les zones couvertes par les cartes c^{mur}, c^{coin} et c^{libre}. Nous déplaçons le robot dans l'environnement, et pour chaque position du robot, nous posons la question $P(Sit_t \mid P)$ que nous avons détaillé ci-dessus, puis tirons aléatoirement sur cette distribution une valeur pour Sit_t. Le résultat est montré graphiquement Figure 9.5 (cette figure est identique à Figure 6.1(a))

Notons que les zones qui apparaissent ont deux interprétations : ce sont à la fois les zones de validité des cartes sous-jacentes, mais aussi les zones correspondantes aux valeurs c^{mur}, c^{coin}, et c^{libre} de la variable Sit_t. Rappelons de plus que les zones « blanches » sur cette figure ne servent qu'à la rendre plus lisible. En réalité, la variable Sit_t prend une valeur en tout point de l'espace, tirée sur une distribution de probabilités. Dans une grande majorité de l'espace, une seule carte sous-jacente est clairement reconnue, ce qui permet de dessiner des zones simples sur la figure. En revanche, les « frontières » correspondent à des distributions de probabilités sur Sit_t qui ne sont plus aussi tranchées. Le tirage aléatoire devient dans ces zones moins catégorique, ce qui rend les frontières imprécises, « floues ».

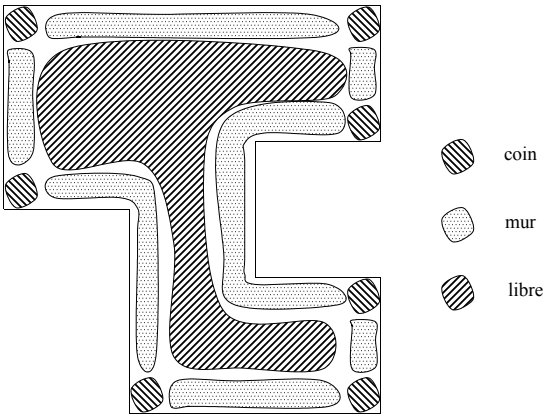


FIG. 9.5: Projection en deux dimensions des zones de validité estimées des modèles c^{mur}, c^{coin} et c^{libre}.

Nous n'utilisons pas la carte c^{prox} que pour répondre aux trois questions de localisation, prédiction et contrôle : comme Section 6.3, nous générons aussi les comportements $SeCacher$, $SeMontrer$, et $Longer$ (en fixant des consignes pour $Sit_{t+\Delta t}$), ainsi que $PatrouillerBord$ (en alternant les comportements $SeCacher$ et $Longer$).

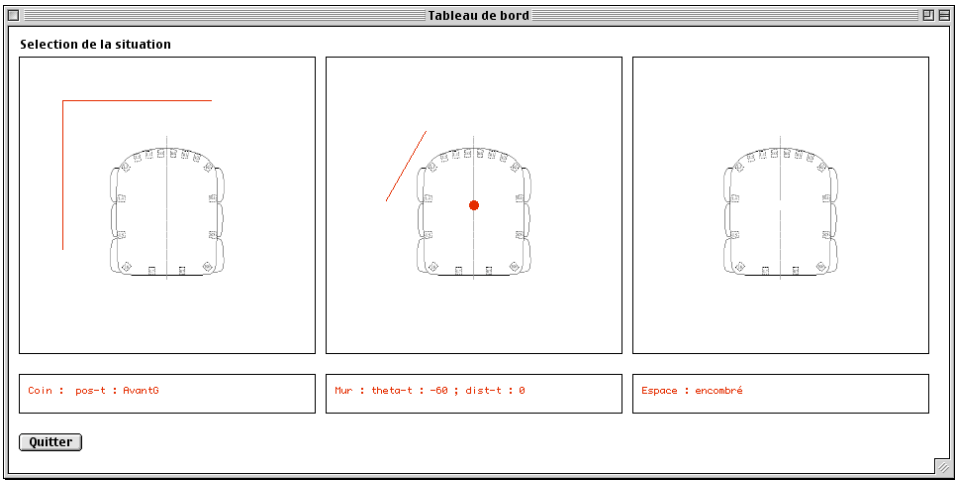


FIG. 9.6: Copie d'écran du module de visualisation de la variable Sit_t (le gros point n'apparaît que sur la carte correspondante à la valeur de Sit_t). On peut également y voir les valeurs des variables internes de chaque sous-carte.

Nous rapportons ici des résultats expérimentaux satisfaisants : le robot résout la tâche $PatrouillerBord$ avec succès. Nous montrons Figure 9.6 une copie d'écran du module de visualisation que nous avons développé pour cette tâche. Cette figure est constituée de trois cadres, et nous y visualisons ce que chaque carte évalue pour ses variables de lieux. Nous voyons par exemple que le modèle c^{coin}, à gauche, indique que, dans l'hypothèse où le robot serait dans un coin, ce coin serait en position AvG (avant-gauche). Au centre, le

modèle c^{mur} indique un mur sur l'avant et la gauche du robot. À droite, la carte c^{libre} pense être en espace encombré. Le point au centre de ce cadre central indique que la variable Sit_t vaut la valeur c^{mur} : c'est la carte du mur la plus pertinente, selon le robot. En effet, expérimentalement, le robot est bien en présence d'un mur, et l'angle et la distance indiqués par c^{mur} sont corrects.

9.4 Discussion

Nous commençons cette partie par quelques remarques concernant les aspects techniques de notre opérateur d'abstraction, avant de discuter de son interprétation par le biais de nos questions « fil conducteur ».

9.4.1 Prédiction et contrôle dans une carte abstraite

Commençons par noter que cet opérateur concerne principalement la définition des termes liés à la localisation dans la carte abstraite. Mis à part le choix de la variable d'action A, nous n'avons quasiment rien dit sur les aspects de contrôle et de prédiction. En conséquence, il est tout à fait envisageable, dans la décomposition donnée, de remplacer les termes $P(L_{t'})P(A \mid L_t\ L_{t'})$ par toute autre structure de dépendance concernant ces termes (par exemple, $P(A)P(L_{t'} \mid A\ L_t\ )$).

9.4.2 Flot d'information global

Une deuxième remarque que nous souhaitons développer ici est liée au flot d'information global s'écoulant entre les différentes cartes, et notamment entre les différents niveaux hiérarchiques. En effet, au Chapitre 6, nous n'avions qu'une carte, et faisions l'hypothèse que la consigne à atteindre était soit donnée par l'utilisateur, soit fonction du comportement, et l'utilisateur choisissait alors le comportement à réaliser.

Maintenant que nous disposons de plusieurs niveaux d'abstraction, il apparaît, grâce au choix fait pour la variable A, que la carte abstraite est le « pilote » des cartes sous-jacentes. Dans la carte abstraite, la réalisation de tâches se fait comme précédemment, en interaction avec l'utilisateur. En revanche, la résolution de cette tâche se traduit par l'application de comportements dans les cartes sous-jacentes.

Par exemple, résoudre la tâche $SeCacher$ dans c^{prox} se fait en posant la question probabiliste $P(A_{prox} \mid P\ [Sit_{t+\Delta t} = coin])$. Si le robot se trouve près d'un mur à l'instant considéré, l'inférence sélectionnera donc le comportement $SuiviD$ pour rejoindre le coin le plus proche. $SuiviD$ est donc transmis à la carte c^{mur}, qui résout ce comportement, notamment en fixant comme consigne $[\Theta_{t+\Delta t} = 90]$ et $[Dist_{t+\Delta t} = 1]$.

Le paragraphe précédent pose un problème : en effet, le terme $P(A_{prox} \mid P\ [Sit_{t+\Delta t} = coin])$ contient la variable P en partie droite (donc de valeur connue). Cette variable est la conjonction de toutes les variables qui proviennent des cartes sous-jacentes. En particulier,

P inclut $\Theta_{t+\Delta t}$ et $Dist_{t+\Delta t}$. Leur valeur est donc encore inconnue, puisqu'elle sera choisie en fonction du résultat de cette inférence !

La solution que nous avons choisi est de baser l'inférence $P(A_{prox} \mid P\ [Sit_{t+\Delta t} = coin])$ non pas sur les valeurs de P les plus récentes, mais sur celles du pas de temps précédent. Si nous explicitons les indices de temps, nous écrivons donc $P(A_{t_{prox}} \mid P_{t-\Delta t}\ [Sit_{t+\Delta t} = coin])$. Cette solution résout effectivement le problème cité. En pratique, elle a aussi une seconde conséquence, car elle induit un léger retard dans la carte abstraite, puisque celle-ci base ses calculs sur des valeurs sensorielles vieilles d'un cycle. Mais cela n'a en pratique que peu d'importance, car la dynamique à ce niveau d'abstraction est plus lente, ce qui est naturel en réalité. En effet, nous nous intéressons ici à la navigation entre des zones de l'environnement dont les tailles correspondent aux zones couvertes par les cartes sous-jacentes. Ces zones sont en pratique grande, et le robot ne change pas tous les cycles de carte sous-jacente ! Il est donc peu gênant d'évaluer avec un cycle de retard le terme le comportement de haut niveau à appliquer.

Expérimentalement, nous pouvons abaisser la fréquence de calcul dans c^{prox} à 1 Hz ($\Delta t = 1$ s), sans perte de performance pour le robot, si ce n'est de parfois traverser les coins sans les reconnaître (ce qui n'est pas critique dans notre tâche $PatrouillerBord$). Malheureusement, nous n'avons que cette observation empirique unique comme donnée, au sujet de la dépendance entre les tailles des zones représentées par les lieux d'une carte et le choix d'un Δt.

9.4.3 Chevauchement des zones

Nous abordons ici un point concernant les zones couvertes par les cartes sous-jacentes. Chaque lieu de la carte abstraite correspond à une zone couverte par une carte sous-jacente. Que se passe-t-il si ces zones se chevauchent ? Supposons que deux cartes c^i et c^j aient ainsi une zone en commun. Si l'une des deux cartes décrit *mieux* cette zone, alors le chevauchement aura peu d'influence, car, dès que le robot entrera dans cette zone, la meilleure carte prendra le dessus, et elle sera le lieu reconnu dans la carte abstraite. C'est ce qui se passe par exemple dans la carte c^{prox} : bien que la carte c^{mur} s'applique dans les coins, la carte c^{coin} décrit bien mieux ces zones. La variable L_t prend donc pour valeur c^{coin} dès que le robot passe dans un coin. De manière similaire, la carte c^{libre} s'applique partout, mais est une description des espaces encombrés très inadaptée, et est donc systématiquement perdante face aux modèles plus fins que sont c^{mur} et c^{coin}.

En revanche, que se passe-t-il si deux cartes décrivent une zone de chevauchement aussi bien l'une que l'autre ? Dans ce cas, lorsque le robot passe dans la zone ambiguë, les valeurs de la variable L_t risque d'osciller entre les deux valeurs c^i et c^j. Si les comportements dictés par ces deux cartes sont (suffisamment) en accord, alors le robot pourra sortir, tant bien que mal, de cette zone d'incertitude. Malheureusement, il est possible que les actions indiquées par ces deux cartes soient antagonistes. Ainsi le robot pourra se mettre à avoir des comportements absurdes, comme par exemple de tourner sur place, ce qui présente le risque de ne jamais le faire sortir de la zone posant problème.

Expérimentalement, il est possible d'abaisser la fréquence de calcul dans la carte abs-

traite pour résoudre le problème. Le robot passe les zones de chevauchements sans remettre en question trop souvent la carte qu'il est en train d'utiliser pour se guider. Ainsi, la dynamique est plus « lisse », et, si les zones de chevauchements ne sont pas trop grandes (ce qui était notre cas), alors le robot ne s'y trouve plus piégé. Malheureusement, de nouveau, nous n'avons que cette observation empirique unique comme donnée, au sujet de la dépendance entre les tailles des zones de chevauchements entre carte sous-jacentes et le choix d'un Δt.

9.4.4 Agencement des zones

Enfin, nous abordons l'indépendance entre les cartes sous-jacentes. Il serait tentant de penser que celles-ci peuvent être développées indépendamment, puis ensuite assemblées facilement, mais ce n'est pas le cas. En pratique, il faut s'assurer que les comportements définis dans une carte amènent bien le robot dans la zone voulue. Par exemple, dans notre carte c^{prox}, le robot arrive la plupart du temps dans un coin en ayant longé le mur qui arrive dans ce coin. Il faut donc que le comportement de suivi de mur $SuiviD$, à l'approche d'un coin, positionne le robot à un endroit effectivement reconnaissable comme un coin. Dans nos expérimentations, nous avions ainsi fait l'erreur de développer une carte du coin dont la zone était trop petite, par rapport au suivi de mur, qui se faisait à une distance trop grande : le robot « ratait » les coins. Nous avons résolu ce problème en comprenant que la carte du coin ne pouvait être développé complètement indépendamment de celle du mur, qui définit le comportement $SuiviD$.

Cette contrainte n'est pas propre à notre approche. Elle apparaît également dans les travaux de Kuipers, où un comportement de suivi de trajectoire doit *assurer* d'arriver au voisinage du lieu suivant. Dans les travaux de Kuipers, cette contrainte est forte, car au niveau topologique, il utilise la logique du premier ordre comme formalisme : il doit donc traduire en *prédicats* des propriétés du niveau d'abstraction inférieur. Dans notre cas, grâce à l'utilisation des probabilités, nous pouvons être plus souples, car nous pourrons représenter numériquement la probabilité que le comportement de suivi de mur nous amène à reconnaître le coin. Dans notre approche, cette probabilité n'a pas besoin d'être 1 (et n'a même pas besoin d'être connue *a priori*, car elle peut être identifiée expérimentalement).

9.4.5 Questions « fil conducteur »

Reprenons maintenant les questions que nous avons défini au début de ce document.

Q1. Qu'est-ce qu'une carte ? Nous avons vu dans ce Chapitre un opérateur d'assemblage de cartes qui, au contraire du précédent, inclut réellement les cartes sous-jacentes, par le biais de questions probabilistes. Elles sont donc, pour la carte abstraite, des *ressources*.

Q7. Comment apprendre expérimentalement les cartes ? L'opérateur d'abstraction concerne principalement le choix des variables, et la manière d'obtenir des valeurs de la variable de lieu, à partir des variables des cartes sous-jacentes. Nous n'avons pratiquement pas évoqué la relation entre les actions de la carte abstraite, et sa variable de lieu, si

ce n'est pour mentionner les faibles cardinalité des variables en jeu, et donc la possibilité d'explorer exhaustivement l'espace pour l'apprentissage. Si ces variables ont une si peu de valeurs, c'est justement grâce à la réduction d'information associée à l'opérateur d'abstraction. Abstraire, c'est oublier les détails, résumer une carte sous-jacente en n'utilisant qu'un symbole (le nom de la carte), et qu'une méthode pour estimer la pertinence de cette carte (la reconnaissance de cartes).

Q7. Comment assembler les cartes ? Dans notre opérateur d'abstraction, nous avons défini des questions probabilistes qui lient la carte abstraite aux cartes sous-jacentes. Ces questions portent sur les distributions conjointes $P(P^i\ L_t^i\ L_{t'}^i\ A^i \mid c^i)$ des cartes sous-jacentes. Une interprétation que nous pouvons donner à ce calcul est qu'il s'agit de reconnaissance de modèles : la carte c^i dont la probabilité (conjointe) est la plus haute est reconnue comme étant le lieu courant. En ce sens, nous utilisons pleinement le pouvoir de représentation de connaissances du formalisme probabiliste, en utilisant aussi les valeurs numériques des probabilité comme indice. Cette valeur numérique peut aussi être interprétée comme une confirmation ou une infirmation des *hypothèses faites par chaque carte sur le monde*. En effet, un modèle est reconnu comme le lieu courant s'il décrit adéquatement la situation sensori-motrice vécue à l'instant donné par le robot. Si ce modèle fait des hypothèses qui ne sont pas pertinentes, la valeur de probabilité de la conjointe sera faible. Nous avons illustré ce point dans le cas où le robot se trouvait en espace libre : le modèle du mur donnait bien une valeur d'angle et de distance, mais qui se trouvait être très peu probable. En réalité, et à cause de la contrainte de normalisation, nous nous intéressons en général peu aux valeurs numériques de probabilités. Mais nous voyons ici qu'elles prennent tout leur sens lorsqu'on les compare les unes aux autres. C'est ce que nous exploitons dans notre opérateur d'abstraction.

Nous comprenons aussi que les zones couvertes par les cartes ne prennent également sens que grâce à la comparaison de cartes entre elles. Nous l'avons évoqué, la carte de l'espace libre par exemple est « valide » dans tout l'environnement, mais est un moins bon modèle des situations « mur » et « coin » que les cartes c^{mur} et c^{coin}. La zone couverte par la carte c^{libre} que nous avons dessiné Figure 9.5 n'est donc pertinente que dans le cadre de la carte abstraite c^{prox}. Ceci ce comprend aisément si nous considérons une carte comme un « modèle de l'interaction entre le robot et l'environnement ». En tant que modèle, une carte traduit donc un ensemble d'hypothèses sur le monde. Et une hypothèse unique n'est jamais « fausse », elle peut au mieux être reconnue comme « moins bonne » qu'une autre (c'est pourquoi nous avons mis le mot « valide » plus haut entre guillemets).

Q7. Comment assembler les cartes ? Dans des travaux précédents [DIARD99], nous avions, dans le cadre de l'apprentissage hiérarchique, interprété une de nos inférences comme de la *reconnaissance de comportements*. Dans notre opérateur d'abstraction, nous avons mentionné la *reconnaissance de cartes*. Est-il possible de combiner ces deux concepts ?

Dans notre exemple d'abstraction, nous avons utilisé trois cartes, c^{mur}, c^{coin}, et c^{libre}. Or cette dernière est si simple, qu'elle est à la limite de notre formalisme. Que se passerait-

il si cette carte était encore simplifiée, et remplacée par un simple comportement (un modèle probabiliste sans la variable $L_{t'}$: peut-on abstraire ensemble deux cartes et un comportement ?

La réponse est « oui ». En effet, rien dans notre écriture probabiliste ne force les modèles à être des cartes. Les modèles que nous assemblons peuvent être de natures variées. Dans le cas où l'un des modèles est un comportement, défini sur des variables P^{cpt}, L_t^{cpt} et A^{cpt}, il suffit d'inclure la définition du terme $P(P^{cpt}\ L_t^{cpt}\ A^{cpt} \mid c)$ dans la carte abstraite, comme une question au comportement en question lorsque $[c = cpt]$, et comme une uniforme sinon.

Ainsi nous pouvons abstraire des comportements entre eux, des cartes entre elles, ou des cartes et des comportements mélangés. Cette dernière combinaison n'est pas sans rappeler les travaux de Kuipers [KUIPERS00], où les représentations de bas niveau se classent en deux : les remontées de gradients (qui couvrent des zones en deux dimensions de l'environnement), et les suivis de trajectoires (qui sont représentés par des lignes). Dans les travaux de Kuipers, l'hétérogénéité de ces modèles induit des contraintes, par exemple qu'ils doivent s'alterner. Ceci permet, lors du passage à l'abstraction topologique, de considérer les zones de gradients comme des nœuds, et les suivis de trajectoires comme des arcs. Cette restriction n'apparaît pas dans notre approche, où la nature des modèles (carte ou comportement) n'influe pas dans la manière de les abstraire, si ce n'est au niveau syntaxique.

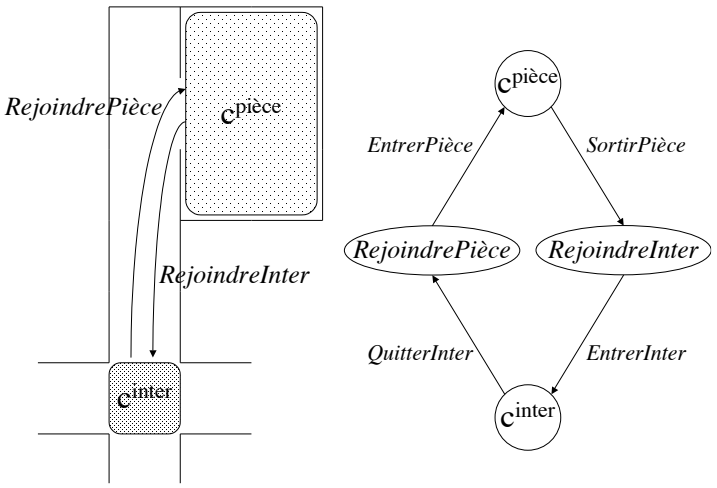


FIG. 9.7: Exemple de carte abstraite qui inclut des cartes et des comportements.

Q1. Qu'est-ce qu'une carte ? Supposons que nous incluons des comportements dans une carte abstraite. Par exemple, décrivons deux lieux de l'environnement (une intersection de couloirs et une pièce) par deux cartes c^{inter} et $c^{pièce}$, mais supposons qu'on ne sache relier l'un à l'autre que par deux comportements, $RejoindreIntersection$ et $RejoindrePièce$. Ces deux comportements ne sont définis ni dans c^{inter}, ni dans $c^{pièce}$, et sont applicables sur toute la largeur du couloir qui relie la pièce à l'intersection. Ainsi, le même espace

géographique sera décrit par deux modèles, qui seront deux nœuds différents dans la carte abstraite. Cette situation est présentée Figure 9.7. Nous comprenons donc ici un intérêt de l'utilisation de cartes, au lieu de simples comportements. En effet, une carte est un moyen de décrire une classe d'interaction entre le robot et son environnement, qui soit relativement *indépendante de la résolution d'une tâche particulière*. De tels modèles ne sont pas toujours nécessaires, comme nous l'avons vu dans l'exemple précédent, où des comportements pouvaient suffire, et se mélanger à des cartes. Mais la création de cartes sera souvent avantageuse, notamment car elles permettront une meilleure interaction avec l'utilisateur. Dans notre exemple, il aurait été en effet plus intuitif de n'avoir qu'un nœud dans la carte abstraite pour le couloir, indépendamment du comportement utilisé pour le traverser. Cependant, les cartes elles-mêmes ne peuvent être *complètement* indépendantes de leur utilisation (d'où l'utilisation ci-dessus de la formule « classe d'interaction »). Nous invitons le lecteur à s'imaginer traversant son appartement les yeux fermés pour se convaincre de cette dernière affirmation.

En résumé, une *carte* est un modèle (probabiliste) qui décrit l'interaction entre le robot et l'environnement, et permet de résoudre un ensemble de tâches. En revanche, un *comportement* est un modèle orienté vers la résolution d'une tâche unique (dans ce travail, nous utilisons des cartes pour définir des comportements, mais il est bien évidemment aussi possible de les définir directement, en se passant de cartes).

9.5 Conclusion sur l'abstraction de cartes

Pour conclure, nous présentons quelques extensions possibles de l'opérateur d'abstraction.

Dans ce chapitre, contrairement au chapitre précédent, nous n'avons présenté que le cas où les cartes sont supposées indépendantes : la levée d'hypothèses et l'identification expérimentale de nouvelles dépendances est cependant possible. Ainsi, dans notre expérience, nous avons levé l'hypothèse d'indépendance entre les cartes, et identifié la dépendance entre les variables $\Theta, Dist$ de la carte c^{mur} et $Coin$ de la carte c^{coin}. En effet, lorsque le robot se trouve dans un coin, le modèle de la situation « mur » ne répond pas n'importe quoi. Par exemple, c^{mur} donne systématiquement $[\Theta_t = -60]$ ou $[\Theta_t = -30]$ si le robot se trouve dans un coin avant-gauche, $[Coin_t = AvG]$. C'est ce type de dépendance que nous pouvons identifier expérimentalement.

Dans notre expérience, nous avons obtenu une carte abstraite, c^{prox}, qui se base sur trois cartes sous-jacentes, c^{mur}, c^{coin} et c^{libre}. Nous avons donc construit une hiérarchie de cartes, à deux niveaux. Il est bien sûr possible d'envisager des hiérarchies de cartes possédant plus de niveaux, car une carte abstraite peut devenir carte sous-jacente du niveau suivant. Par exemple, assembler c^{prox}, qui décrit une zone couvrant une pièce entière, avec une carte décrivant un couloir, et avec une carte décrivant un hall, nous permet d'obtenir une carte de plus haut niveau d'abstraction décrivant un étage de bâtiment. Cette même carte, assemblée avec des cartes des autres étages, et des cartes décrivant les escaliers ou ascenseurs, permet d'obtenir une carte d'un bâtiment entier. Et ainsi de suite...

Troisième partie

Conclusion

Chapitre 10

Perspectives et Conclusion

> « Je considère donc qu'un organisme possède un fonctionnement mental à partir du moment où il élabore des représentations neurales [...], lesquelles peuvent subir un traitement dans le cadre d'un processus appelé pensée, et finalement influencer le comportement, dans la mesure où elles peuvent permettre de faire des prédictions sur l'avenir, de former des plans en fonction de ces dernières et de choisir la prochaine des actions. »
>
> Antonio R. Damasio, *L'Erreur de Descartes*, 1995, [Damasio95].

10.1 Perspectives

Nous souhaitons rappeler certaines des pistes de recherches futures évoquées tout au long de ce document en les organisant par thématique.

10.1.1 Développements applicatifs

Nous avons illustré nos concepts sur des environnements expérimentaux simples. Cependant, la généralité et l'homogénéité théorique de notre approche nous rend confiant sur la possibilité de leur application à plus grande échelle. Deux axes nous semblent prometteurs.

1. Le premier consiste en une implémentation à caractère « technologique », où il s'agirait d'implanter notre formalisme sur un robot doté de nombreux capteurs et confronté à un environnement de plus grande taille. Notre approche subirait ainsi une mise à l'échelle, dont nous avons déjà noté quelques aspects :
   - la superposition d'un grand nombre de cartes ;
   - l'abstraction de plusieurs cartes ;
   - et la construction d'une hiérarchie à n niveaux, $n > 2$.

   Une telle implantation permettrait de valider notre formalisme en tant qu'outil de développement. Elle montrerait sa facilité d'utilisation par un programmeur roboticien. Nous souhaiterions enfin considérer l'étude de la construction de cartes sur

des robots dotés de capteurs et d'actionneurs de natures variées : par exemple, notre concept de cartes s'applique-t-il à des bras manipulateurs ? Nous pensons que oui.

2. Le second axe concerne les sciences cognitives. Il consiste en une étude sur la plausibilité biologique de notre formalisme. Pour le moment, le monde du vivant constitue pour nous une source d'inspiration. Notre modèle peut-il devenir source d'analogies, voire même source de compréhension et de modélisation pour l'éthologie ou la neurophysiologie ?

Certains de ces aspects applicatifs font l'objet de notre projet de recherche post-doctoral, dans le cadre d'une collaboration entre roboticiens (Laboratoire GRAVIR et Université NUS de Singapour) et neurophysiologistes (Laboratoire LPPA, Collège de France, Paris).

10.1.2 Développements théoriques

D'un point de vue théorique, certains aspects de notre formalisme peuvent être approfondis, notamment – nous l'espérons – dans le cadre des implantations considérées ci-dessus.

Le premier domaine est celui de la planification. En effet, disposer de hiérarchies de représentations est un point clé pour de nombreuses techniques de planification : que ce soit dans les formalismes à base de processus de décision markoviens [HMK+98, LK02, PT02] ou dans les approches dites de « roadmap », comme le Fil d'Ariane [MAB98] ou PPP [SO98]. À notre connaissance, toutes ces approches décrivent l'environnement en n'utilisant que deux niveaux hiérarchiques.

Au contraire, notre formalisme permet la spécification de hiérarchies plus profondes. Dans notre approche, la dérivation formelle d'un algorithme de planification peut éventuellement donner naissance à des extensions ou des généralisations de ces travaux.

Un second domaine théorique prometteur est celui de l'apprentissage. Pour l'instant, nous nous sommes restreints à des méthodes d'apprentissage supervisé. Ces approches nécessitent l'intervention d'un opérateur pour améliorer les performances du robot. Il est également possible de considérer l'inclusion de méthodes d'apprentissage semi-supervisé ou non-supervisé à notre formalisme.

L'utilisation de telles méthodes d'apprentissage permettrait d'envisager de franchir un pas supplémentaire vers l'autonomie en robotique mobile.

10.2 Conclusion

Nous avons proposé un formalisme original de modélisation probabiliste de l'interaction entre un robot et son environnement : **la carte bayésienne**. Nous avons illustré cette approche par des exemples implantés sur un robot Koala.

Dans ce formalisme, nous avons pu étudier et définir certaines notions de navigation pour les robots mobiles. Une carte est représentée par une distribution de probabilités : c'est notre façon de décrire une interaction. La définition de cette carte correspond à la phase déclarative de notre approche. Naviguer, se localiser, prédire, et générer une loi de contrôle sont en revanche des compétences, donc du domaine procédural. Il s'agit alors de restituer les connaissances contenues dans la carte, par le biais de questions probabilistes.

Nous avons détaillé l'éventail des choix de modélisation laissés au concepteur lors du cycle de développement, à la fois pour la programmation et l'apprentissage.

Notre formalisme s'enrichit de la définition de deux opérateurs permettant d'assembler des cartes pour en créer de nouvelles : l'opérateur de superposition, et celui d'abstraction.

Le premier permet d'enrichir le vocabulaire pour décrire l'environnement du robot, en fusionnant des modèles disponibles. La carte obtenue dispose de plus de lieux que les cartes initiales. Cela donne au robot une plus grande finesse de représentation de l'environnement.

Le second opérateur permet de construire des hiérarchies de cartes. Grâce à cet opérateur, nous obtenons une carte abstraite qui résume de manière considérable les cartes sous-jacentes. Cette réduction d'information permet d'obtenir une carte portant sur un petit espace, donc facilement manipulable (spécification, apprentissage, planification), mais néanmoins fortement ancré dans le niveau d'abstraction inférieur (reconnaissance de modèles).

Ce dernier point souligne la sémantique originale que nous associons à nos cartes : ce sont des *modèles de la dynamique de l'interaction entre un robot et son environnement.*

Bibliographie

[AHC02] Orlando Avila-Garciá, Elena Hafner, and Lola Cañamero. Relating behavior selection architectures to environmental complexity. In *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior (SAB)*, pages 127–128, 2002.

[ALC98] Olivier Aycard, Pierre Laroche, and François Charpillet. Mobile robot localization in dynamic environments using place recognition. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation (ICRA'98)*, pages 3135–3140, 1998.

[AMS01] Olivier Aycard, Jean-François Mari, and René Schott. *Probabilistic and Statistical Methods in Computer Science*, chapter Application of Markov models in robotics, pages 177–205. Kluwer Academic Publishers, 2001.

[Aycard98] Olivier Aycard. *Architecture de contrôle pour robot mobile en environnement intérieur structuré*. Thèse de doctorat, Université Henri Poincaré - Nancy 1, Nancy, France, Juin 1998.

[Bachelard72] Gaston Bachelard. *L'engagement rationaliste*. Presses Universitaires de France, Paris, 1972.

[BBM94] Allen L. Barker, Donald E. Brown, and Worthy N. Martin. Bayesian estimation and the kalman filter. Technical Report IPC-94-02, Institute for Parallel Computation, University of Virginia, August 5 1994.

[BCF+99] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schultz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1–2) :3–55, 1999.

[BDH99] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision theoretic planning : Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 10 :1–94, 1999.

[BDL+98a] Pierre Bessière, Eric Dedieu, Olivier Lebeltel, Emmanuel Mazer, and Kamel Mekhnacha. Interprétation vs. description I : Proposition pour une théorie probabiliste des systèmes cognitifs sensori-moteurs. *Intellectica*, 26-27 :257–311, 1998.

[BDL+98b] Pierre Bessière, Eric Dedieu, Olivier Lebeltel, Emmanuel Mazer, and Kamel Mekhnacha. Interprétation vs. description II : Fondements mathématiques. *Intellectica*, 26-27 :313–336, 1998.

[BEF95] Johann Borenstein, H. R. Everett, and Liqiang Feng. *Navigating Mobile Robots : Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1995.

[BEF96] Johann Borenstein, H. R. Everett, and Liqiang Feng. Where am i ? sensors and methods for mobile robot positioning. Technical report, University of Michigan, april 1996.

[Bellot02] David Bellot. *Fusion de données avec des réseaux bayésiens pour la modélisation des systèmes dynamiques et son application en télémédecine*. Thèse de doctorat, Université Henri Poincaré - Nancy 1, Nancy, France, Novembre 2002.

[Berthoz97] Alain Berthoz. *Le sens du mouvement*. Odile Jacob, 1997.

[BF95] Y. Bengio and P. Frasconi. An input/output HMM architecture. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 427–434. MIT Press, Cambridge, MA, 1995.

[BFHS96] Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 896–901, Menlo Park, August 4–8 1996. AAAI Press / MIT Press.

[BFJ+99] Wolfram Burgard, Dieter Fox, Hauke Jans, Christian Matenar, and Sebastian Thrun. Sonar-based mapping with mobile robots using EM. In *Proc. 16th International Conf. on Machine Learning*, pages 67–76. Morgan Kaufmann, San Francisco, CA, 1999.

[BMR97] John Binder, Kevin Murphy, and Stuart Russell. Space-efficient inference in dynamic probabilistic networks. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1292–1296, San Francisco, August 23–29 1997. Morgan Kaufmann Publishers.

[BP02] Josh BONGARD and Rolf PFEIFER. A method for isolating morphological effects on evolved behaviour. In *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior (SAB)*, pages 305–311, 2002.

[CHANGEUX02] Jean-Pierre CHANGEUX. *L'Homme de vérité*. Odile Jacob, Paris, 2002.

[CKK96] Anthony R. CASSANDRA, Leslie Pack KAELBLING, and James A. KURIEN. Acting under uncertainty : Discrete bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems*, 1996.

[CN94] Timothy W. CACCIATORE and Steven J. NOWLAN. Mixtures of controllers for jump linear and non-linear plants. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 719–726. Morgan Kaufmann Publishers, Inc., 1994.

[COUÉ00] Christophe COUÉ. Programmation bayésienne des robots : poursuite d'une cible fuyante par un robot mobile. Rapport de dea « imagerie, vision, robotique », Institut National Polytechnique de Grenoble, Grenoble, France, Juin 2000.

[CROWLEY89] James CROWLEY. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation (ICRA)*, pages 674–680, Scottsdale, AZ, 1989.

[DAMASIO95] Antonio R. DAMASIO. *L'Erreur de Descartes*. Odile Jacob, Paris, 1995.

[DBZ99] Jacques DROULEZ, Alain BERTHOZ, and René ZAPATA. Prédiction et programmation des mouvements. In Vincent Bloch, editor, *Cerveaux et Machines*, pages 181–193. Hermès Science Publications, Paris, France, 1999.

[DC91] Stanislas DEHAENE and Jean-Pierre CHANGEUX. The wisconsin card sorting test : theoretical analysis and simulation of a reasoning task in a model neuronal network. *Cereb. Cortex*, 1 :62–79, 1991.

[DIARD99] Julien DIARD. Apprentissage hiérarchique bayésien. Rapport de dea « informatique, systèmes et communications », Université Joseph Fourier, Grenoble, France, Juin 1999.

[DK89] Thomas DEAN and Keiji KANAZAWA. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3) :142–150, 1989.

[ECO97] Umberto ECO. *Comment voyager avec un saumon. Nouveaux pastiches et postiches.* Grasset, Paris, 1997.

[FBDT99] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization : Efficient position estimation for mobile robots. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Orlando, FL, 1999.

[FBKT00] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3) :255–266, 2000.

[FBT98] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3–4) :195–207, 1998.

[FBT99] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11 :391–427, 1999.

[Filliat01] David Filliat. *Cartographie et estimation globale de la position pour un robot mobile autonome*. Thèse de doctorat, Université Paris 6, Paris, France, Décembre 2001.

[FM00] Matthias Franz and Hanspeter Mallot. Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30 :133–153, 2000.

[Ghahramani98] Zoubin Ghahramani. Learning dynamic Bayesian networks. In C. Lee Giles and Marco Gori, editors, *Adaptive Processing of Sequences and Data Structures*, number 1387 in Lecture Notes in Artificial Intelligence, LNAI, pages 168–197. Springer-Verlag, 1998.

[Ghahramani01] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *Journal of Pattern Recognition and Artificial Intelligence*, 15(1) :9–42, 2001.

[GJBR00] Philippe Gaussier, Cédric Joulain, Jean-Paul Banquet, and Sacha Leprêtre and Arnaud Revel. The visual homing problem : an example of robotics/biology cross fertilization. *Robotics and Autonomous Systems*, 30 :155–180, 2000.

[Heckerman96] David Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1996.

[HMK+98] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 220–229, San Francisco, July 24–26 1998. Morgan Kaufmann.

[Jaynes95] Edwin T. Jaynes. Probability theory - the logic of science. Manuscript non publié, disponible à http ://bayes.wustl.edu, 1995.

[KH02] Dae Eun Kim and John Hallam. An evolutionary approach to quantify internal states needed for the Woods problem. In *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior (SAB)*, pages 312–322, 2002.

[KKR95] Keiji Kanazawa, Daphne Koller, and Stuart Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, pages 346–351, San Francisco, CA, USA, August 1995. Morgan Kaufmann Publishers.

[KLC98] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2) :99–134, 1998.

[KS96] Sven Koenig and Reid Simmons. Probabilistic robot navigation in partially observable environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2301–2308, 1996.

[KSLO96] Lydia Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12(4) :566–580, 1996.

[Kuipers96] Benjamin J. Kuipers. A hierarchy of qualitative representations for space. In *Working Papers of the Tenth International Workshop on Qualitative Reasoning (QR-96)*, 1996.

[Kuipers00] Benjamin J. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1–2) :191–233, 2000.

[Lambert98] Dominique Lambert. La théorie des probabilités et le problème de l'efficacité (dé)raisonnable des mathématiques. In E. Klein & Y. Sacquin, editor, *Prédiction & probabilités dans les sciences*, page 65. Editions Frontières, 1998.

[LBDM03] Olivier Lebeltel, Pierre Bessière, Julien Diard, and Emmanuel Mazer. Bayesian robot programming. *Autonomous Robots (à paraître)*, 2003.

[LD91] John J. Leonard and Hugh F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3) :376–382, June 1991.

[LDC92] John Leonard, Hugh Durrant-Whyte, and Ingemar Cox. Dynamic map-building for an autonomous mobile robot. *The International Journal of Robotics Research*, 11(4) :286–298, 1992.

[Lebeltel99] Olivier Lebeltel. *Programmation Bayésienne des Robots*. Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, France, Septembre 1999.

[LK01a] Terran Lane and Leslie Pack Kaelbling. Approaches to macro decompositions of large markov decision process planning problems. In *Proceedings of the 2001 SPIE Conference on Mobile Robotics*, Newton, MA, 2001. SPIE.

[LK01b] Terran Lane and Leslie Pack Kaelbling. Toward hierarchical decomposition for planning in uncertain environments. In *Proceedings of the 2001 IJCAI Workshop on Planning under Uncertainty and Incomplete Information*, Seattle, WA, August 2001. AAAI Press.

[LK02] Terran Lane and Leslie Pack Kaelbling. Nearly deterministic abstractions of markov decision processes. In *Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, 2002.

[LL90] Tod S. Levitt and Daryl T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44(3) :305–360, 1990.

[LS88] S. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society series B*, 50 :157–224, 1988.

[MAB98] Emmanuel Mazer, Juan-Manuel Ahuactzin, and Pierre Bessière. The ariadne's clew algorithm. *Journal of Artificial Intelligence Research (JAIR)*, 9 :295–31, 1998.

[Mitchell97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MJ96] Marina Meila and Michael I. Jordan. Learning fine motion by markov mixtures of experts. In D. Touretzky, M. C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages –. Morgan Kaufmann Publishers, Inc., 1996.

[Murphy01] Kevin Murphy. An introduction to graphical models. Technical report, University of California, Berkeley, May 2001.

[Murphy02] Kevin Murphy. *Dynamic Bayesian Networks : Representation, Inference and Learning*. Ph. D. thesis, University of California, Berkeley, Berkeley, CA, July 2002.

[Pearl91] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, Ca, 1988 (seconde édition 1991).

[PK97] David Pierce and Benjamin Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92 :169–229, 1997.

[PT02] Joelle Pineau and Sebastian Thrun. An integrated approach to hierarchy and abstraction for POMDPs. Technical Report CMU-RI-TR-02-21, Carnegie Mellon University, August 2002.

[Rabiner89] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE Trans. on ASSP*, 77(2) :257–285, February 1989.

[Raspail02] Frédéric Raspail. Apprentissage bayésien par imitation. Rapport de dea « informatique, systèmes et communications », Université Joseph Fourier, Grenoble, France, Juin 2002.

[RBFT99] Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation : Robot navigation under uncertainty in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[Real91] L. A. Real. Animal choice behavior and the evolution of cognitive architecture. *Science*, 253 :980–986, 1991.

[RG99] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11(2) :305–345, February 1999.

[RJ93] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*, chapter Theory and implementation of Hidden Markov Models, pages 321–389. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[Russell98] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *COLT : Proceedings of the Workshop on Computational Learning Theory,*, Madison, Wisconsin, 1998. ACM Press.

[Sacks99] Oliver Sacks. *Awakenings*. First Vintage Books Ed., New-York, 1999.

[SBC99] Lawrence Stone, Carl Barlow, and Thomas Corwin. *Bayesian Multiple Target Tracking*. Artech House, 1999.

[Shannon48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 :379–423, 623–656, 1948.

[Shatkay98] Hagit Shatkay. Learning models for robot navigation. Technical Report CS-98-11, Brown University - Department of Computer Science, December 1998.

[SHJ97] Padhraic Smyth, David Heckerman, and Michael I. Jordan. Probabilistic independence networks for hidden markov probability models. *Neural Computation*, 9(2) :227–269, 1997.

[SK95] Reid SIMMONS and Sven KOENIG. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1080–1087, 1995.

[SK97] Hagit SHATKAY and Leslie Pack KAELBLING. Learning topological maps with weak local odometric information. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 920–929, San Francisco, August 23–29 1997. Morgan Kaufmann Publishers.

[SMYTH98] Padhraic SMYTH. Belief networks, hidden markov models, and markov random fields : a unifying view. *Pattern Recognition Letters*, 1998.

[SO98] Petr SVESTKA and Mark OVERMARS. Probabilistic path planning. In Jean-Paul Laumond, editor, *Lecture Notes in Control and Information Sciences 229*. Springer, 1998.

[STOCKMEYER00] Paul STOCKMEYER. Enumerating graphs. In Kenneth H. Rosen, editor, *Handbook of Discrete and combinatorial mathematics*, pages 580–586. CRC Press, Boca Raton, Florida, 2000.

[TBB+98] Sebastian THRUN, Arno BÜCKEN, Wolfram BURGARD, Dieter FOX, Thorsten FRÖHLINGHAUS, Daniel HENNIG, Thomas HOFMANN, Michael KRELL, and Timo SCHMIDT. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R Murphy, editors, *AI-based Mobile Robots : Case Studies of Successful Robot Systems*, pages 580–586. MIT Press, 1998.

[TBB+99a] Sebastian THRUN, Maren BENNEWITZ, Wolfram BURGARD, Armin B. CREMERS, Frank DELLAERT, Dieter FOX, Dirk HÄHNEL, Charles ROSENBERG, Nicholas ROY, Jamieson SCHULTE, and Dirk SCHULZ. MINERVA : A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[TBB+99b] Sebastian THRUN, Maren BENNEWITZ, Wolfram BURGARD, Armin B. CREMERS, Frank DELLAERT, Dieter FOX, Dirk HÄHNEL, Charles ROSENBERG, Nicholas ROY, Jamieson SCHULTE, and Dirk SCHULZ. MINERVA : A tour-guide robot that learns. In Wolfram Burgard, Thomas Christaller, and Armin B. Cremers, editors, *Proceedings of the 23rd Annual German Conference on Advances in Artificial Intelligence (KI-99)*, volume 1701 of *LNAI*, pages 14–26, Berlin, September 13–15 1999. Springer.

[TBF98] Sebastian THRUN, Wolfram BURGARD, and Dieter FOX. A probabilistic approach to concurrent mapping and localization for

mobile robots. *Machine Learning and Autonomous Robots (joint issue)*, 31/5 :1–25, 1998.

[TBF00] Sebastian THRUN, Wolfram BURGARD, and Dieter FOX. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, 2000.

[TFB00] Sebastian THRUN, Dieter FOX, and Wolfram BURGARD. Monte carlo localization with mixture proposal distribution. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Austin, TX, 2000.

[THRUN98a] Sebastian THRUN. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 33(1) :41–76, 1998.

[THRUN98b] Sebastian THRUN. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1) :21–71, 1998.

[THRUN00] Sebastian THRUN. Probabilistic algorithms in robotics. *AI Magazine*, 21(4) :93–109, 2000.

[THRUN02] Sebastian THRUN. Robotic mapping : A survey. Technical Report CMU-CS-02-111, Carnegie Mellon University, February 2002.

[TNAS01] Nicola TOMATIS, Illah NOURBAKHSH, Kai ARRAS, and Roland SIEGWART. A hybrid approach for robust and precise mobile robot navigation with compact environment modeling. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA'01)*, pages 1111–1116, 2001.

[TWBM97] Olivier TRULLIER, Sidney WIENER, Alain BERTHOZ, and Jean-Arcady MEYER. Biologically-based artificial navigation systems : Review and prospects. *Progress in Neurobiology*, 51 :483–544, 1997.

[VAN ZWYNSVOORDE00] Dominique VAN ZWYNSVOORDE. *Construction incrémentale de modèles topologiques pour la navigation d'un robot mobile.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France, Décembre 2000.

[WB95] Greg WELCH and Gary BISHOP. An introduction to the kalman filter. Technical Report TR95-041, Computer Science Department, Univerity of North Carolina, Chapel Hill, 1995.

Annexe A

Un exemple détaillé d'inférence

Dans cette annexe, nous développons les calculs correspondant à l'inférence pour le terme $P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox})$, dans le cadre de la carte c^{prox} (voir Section 6.3.3.1). Les calculs présentés ici sont généraux, et s'appliquent presque toujours de cette manière. En effet, le terme $P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox})$ possède des variables en partie droite et une variable inconnue, Px.

Nous commençons par appliquer le théorème de Bayes (voir Section 2.2.2.1).

$$P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) = \frac{P(Sit_{t+\Delta t}\ Sit_t\ A_{prox})}{P(Sit_t\ A_{prox})}.$$

Si nous utilisons la notation qui fait apparaître les propositions en partie droite, nous voyons sur

$$P(Sit_{t+\Delta t} \mid [Sit_t = s]\ [A_{prox} = a]) = \frac{P(Sit_{t+\Delta t}\ [Sit_t = s]\ [A_{prox} = a]}{P([Sit_t = s]\ [A_{prox} = a])}$$

que le dénominateur est un constante, indépendante de la variable $Sit_{t+\Delta t}$ en partie gauche. Nous renommons cette constante de normalisation Z_1 :

$$P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) = \frac{1}{Z_1} P(Sit_{t+\Delta t}\ Sit_t\ A_{prox}).$$

Appliquons ensuite la règle de marginalisation (Section 2.2.2.3), afin de faire apparaître la distribution conjointe :

$$P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) = \frac{1}{Z_1} \sum_{Px} P(Px\ Sit_{t+\Delta t}\ Sit_t\ A_{prox}).$$

Nous comprenons donc maintenant que savoir calculer la distribution conjointe est la clé de l'inférence, car on y ramène systématiquement. Remplaçons la conjointe par la décomposition employée :

$$P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) \quad = \quad \frac{1}{Z_1} \sum_{Px} P(Px) P(Sit_t \mid Px) P(Sit_{t+\Delta t}) P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t).$$

Sortons de la somme les termes qui sont indépendants des indices de sommation :

$$P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) \quad = \quad \frac{1}{Z_1} P(Sit_{t+\Delta t}) P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t) \sum_{Px} P(Px) P(Sit_t \mid Px).$$

Le terme $P(Px)$ étant défini comme suivant une loi uniforme, sa valeur numérique est $1/(k_{Px})$, quelle que soit la valeur de l'indice de sommation courante. Ce terme peut donc être sorti de la somme et intégrée dans la constante de normalisation, qui devient Z_2. Nous obtenons :

$$P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) \quad = \quad \frac{1}{Z_2} P(Sit_{t+\Delta t}) P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t) \sum_{Px} P(Sit_t \mid Px).$$

Le terme $P(Sit_t \mid Px)$, quant à lui, a une valeur fixée en partie gauche, donc la somme sur Px de $P([Sit_t = s] \mid Px)$ est également indépendante de $Sit_{t+\Delta t}$. Cette somme peut donc être intégrée à la constante de normalisation :

$$P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) = \frac{1}{Z_3} P(Sit_{t+\Delta t}) P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t).$$

Enfin, comme le terme $P(Sit_{t+\Delta t})$ est défini dans c^{prox} comme étant une uniforme, il peut également être absorbé dans la constante de normalisation. Finalement, nous obtenons :

$$P(Sit_{t+\Delta t} \mid Sit_t\ A_{prox}) = \frac{1}{Z_4} P(A_{prox} \mid Sit_{t+\Delta t}\ Sit_t),$$

ou, avec la notation indiquant les propositions en partie droite :

$$P(Sit_{t+\Delta t} \mid [Sit_t = s]\ [A_{prox} = a]) = \frac{1}{Z_4} P([A_{prox} = a] \mid Sit_{t+\Delta t}\ [Sit_t = s]).$$

Annexe B

1015 décompositions...

Nous présentons dans les pages suivantes les 1015 décompositions possibles pour une description portant sur quatre variables. La décomposition

$$P(A)P(L_t)P(P \mid L_t)P(L_{t'} \mid A\ L_t),$$

qui correspond à la localisation markovienne (voir Section 3.8) se trouve page 167.

$P(APL_tL_{t'})$,
$P(A|P)P(PL_tL_{t'})$,
$P(A|PL_t)P(PL_tL_{t'})$,
$P(A|PL_tL_{t'})P(PL_tL_{t'})$,
$P(AP)P(L_tL_{t'}|P)$,
$P(AL_t)P(PL_{t'}|A)$,
$P(AL_{t'})P(PL_t)$,
$P(AL_{t'})P(PL_t|AL_{t'})$,
$P(AL_t|P)P(PL_{t'})$,
$P(AL_{t'}|L_t)P(PL_t)$,
$P(AL_{t'}|PL_t)P(PL_t)$,
$P(APL_t)P(L_{t'}|P)$,
$P(APL_t)P(L_{t'}|AL_t)$,
$P(APL_{t'})P(L_t)$,
$P(APL_{t'})P(L_t|L_{t'})$,
$P(APL_{t'})P(L_t|PL_{t'})$,
$P(AL_tL_{t'})P(P|A)$,
$P(AL_tL_{t'})P(P|AL_t)$,
$P(AL_tL_{t'})P(P|AL_tL_{t'})$,
$P(AL_tL_{t'}|P)P(P)$,
$P(A)P(P)P(L_tL_{t'}|P)$,
$P(A)P(P|A)P(L_tL_{t'}|A)$,
$P(A)P(P|L_t)P(L_tL_{t'})$,
$P(A)P(P|L_{t'})P(L_tL_{t'}|A)$,
$P(A)P(P|AL_{t'})P(L_tL_{t'})$,
$P(A)P(P|L_tL_{t'})P(L_tL_{t'}|A)$,
$P(A)P(PL_t)P(L_{t'})$,
$P(A)P(PL_t)P(L_{t'}|L_t)$,
$P(A)P(PL_t)P(L_{t'}|PL_t)$,
$P(A)P(PL_{t'})P(L_t|A)$,
$P(A)P(PL_{t'})P(L_t|AP)$,
$P(A)P(PL_{t'})P(L_t|APL_{t'})$,
$P(A)P(PL_t|A)P(L_{t'}|P)$,
$P(A)P(PL_t|A)P(L_{t'}|AL_t)$,
$P(A)P(PL_t|L_{t'})P(L_{t'})$,
$P(A)P(PL_{t'}|A)P(L_t|A)$,
$P(A)P(PL_{t'}|A)P(L_t|AP)$,
$P(A)P(PL_{t'}|A)P(L_t|APL_{t'})$,
$P(A)P(PL_t|AL_{t'})P(L_{t'})$,
$P(A)P(PL_{t'}|AL_t)P(L_t|A)$,
$P(A|P)P(P)P(L_tL_{t'}|P)$,
$P(A|P)P(P|L_{t'})P(L_tL_{t'})$,
$P(A|P)P(PL_t)P(L_{t'}|A)$,
$P(A|P)P(PL_t)P(L_{t'}|AP)$,
$P(A|P)P(PL_t)P(L_{t'}|APL_t)$,
$P(A|P)P(PL_{t'})P(L_t|P)$,
$P(A|P)P(PL_{t'})P(L_t|AL_{t'})$,
$P(A|P)P(PL_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P)P(L_tL_{t'}|P)$,
$P(A)P(PL_tL_{t'})$,
$P(A|L_t)P(PL_tL_{t'})$,
$P(A|PL_{t'})P(PL_tL_{t'})$,
$P(AP)P(L_tL_{t'})$,
$P(AP)P(L_tL_{t'}|AP)$,
$P(AL_t)P(PL_{t'}|L_t)$,
$P(AL_{t'})P(PL_t|A)$,
$P(AP|L_t)P(L_tL_{t'})$,
$P(AL_t|L_{t'})P(PL_{t'})$,
$P(AP|L_tL_{t'})P(L_tL_{t'})$,
$P(APL_t)P(L_{t'})$,
$P(APL_t)P(L_{t'}|L_t)$,
$P(APL_t)P(L_{t'}|PL_t)$,
$P(APL_{t'})P(L_t|A)$,
$P(APL_{t'})P(L_t|AP)$,
$P(APL_{t'})P(L_t|APL_{t'})$,
$P(AL_tL_{t'})P(P|L_t)$,
$P(AL_tL_{t'})P(P|AL_{t'})$,
$P(APL_t|L_{t'})P(L_{t'})$,
$P(A)P(P)P(L_tL_{t'})$,
$P(A)P(P)P(L_tL_{t'}|AP)$,
$P(A)P(P|A)P(L_tL_{t'}|P)$,
$P(A)P(P|L_t)P(L_tL_{t'}|A)$,
$P(A)P(P|AL_t)P(L_tL_{t'})$,
$P(A)P(P|AL_{t'})P(L_tL_{t'}|A)$,
$P(A)P(P|AL_tL_{t'})P(L_tL_{t'})$,
$P(A)P(PL_t)P(L_{t'}|A)$,
$P(A)P(PL_t)P(L_{t'}|AP)$,
$P(A)P(PL_t)P(L_{t'}|APL_t)$,
$P(A)P(PL_{t'})P(L_t|P)$,
$P(A)P(PL_{t'})P(L_t|AL_{t'})$,
$P(A)P(PL_t|A)P(L_{t'})$,
$P(A)P(PL_t|A)P(L_{t'}|L_t)$,
$P(A)P(PL_t|A)P(L_{t'}|PL_t)$,
$P(A)P(PL_t|L_{t'})P(L_{t'}|A)$,
$P(A)P(PL_{t'}|A)P(L_t|P)$,
$P(A)P(PL_{t'}|A)P(L_t|AL_{t'})$,
$P(A)P(PL_{t'}|L_t)P(L_t)$,
$P(A)P(PL_t|AL_{t'})P(L_{t'}|A)$,
$P(A|P)P(P)P(L_tL_{t'})$,
$P(A|P)P(P)P(L_tL_{t'}|AP)$,
$P(A|P)P(P|L_tL_{t'})P(L_tL_{t'})$,
$P(A|P)P(PL_t)P(L_{t'}|P)$,
$P(A|P)P(PL_t)P(L_{t'}|AL_t)$,
$P(A|P)P(PL_{t'})P(L_t)$,
$P(A|P)P(PL_{t'})P(L_t|L_{t'})$,
$P(A|P)P(PL_{t'})P(L_t|PL_{t'})$,
$P(A|P)P(PL_{t'}|L_t)P(L_t)$,
$P(A|L_t)P(P|A)P(L_tL_{t'})$,
$P(A)P(PL_tL_{t'}|A)$,
$P(A|L_{t'})P(PL_tL_{t'})$,
$P(A|L_tL_{t'})P(PL_tL_{t'})$,
$P(AP)P(L_tL_{t'}|A)$,
$P(AL_t)P(PL_{t'})$,
$P(AL_t)P(PL_{t'}|AL_t)$,
$P(AL_{t'})P(PL_t|L_{t'})$,
$P(AP|L_{t'})P(L_tL_{t'})$,
$P(AL_{t'}|P)P(PL_t)$,
$P(AL_t|PL_{t'})P(PL_{t'})$,
$P(APL_t)P(L_{t'}|A)$,
$P(APL_t)P(L_{t'}|AP)$,
$P(APL_t)P(L_{t'}|APL_t)$,
$P(APL_{t'})P(L_t|P)$,
$P(APL_{t'})P(L_t|AL_{t'})$,
$P(AL_tL_{t'})P(P)$,
$P(AL_tL_{t'})P(P|L_{t'})$,
$P(AL_tL_{t'})P(P|L_tL_{t'})$,
$P(APL_{t'}|L_t)P(L_t)$,
$P(A)P(P)P(L_tL_{t'}|A)$,
$P(A)P(P|A)P(L_tL_{t'})$,
$P(A)P(P|A)P(L_tL_{t'}|AP)$,
$P(A)P(P|L_{t'})P(L_tL_{t'})$,
$P(A)P(P|AL_t)P(L_tL_{t'}|A)$,
$P(A)P(P|L_tL_{t'})P(L_tL_{t'})$,
$P(A)P(P|AL_tL_{t'})P(L_tL_{t'}|A)$,
$P(A)P(PL_t)P(L_{t'}|P)$,
$P(A)P(PL_t)P(L_{t'}|AL_t)$,
$P(A)P(PL_{t'})P(L_t)$,
$P(A)P(PL_{t'})P(L_t|L_{t'})$,
$P(A)P(PL_{t'})P(L_t|PL_{t'})$,
$P(A)P(PL_t|A)P(L_{t'}|A)$,
$P(A)P(PL_t|A)P(L_{t'}|AP)$,
$P(A)P(PL_t|A)P(L_{t'}|APL_t)$,
$P(A)P(PL_{t'}|A)P(L_t)$,
$P(A)P(PL_{t'}|A)P(L_t|L_{t'})$,
$P(A)P(PL_{t'}|A)P(L_t|PL_{t'})$,
$P(A)P(PL_{t'}|L_t)P(L_t|A)$,
$P(A)P(PL_{t'}|AL_t)P(L_t)$,
$P(A|P)P(P)P(L_tL_{t'}|A)$,
$P(A|P)P(P|L_t)P(L_tL_{t'})$,
$P(A|P)P(PL_t)P(L_{t'})$,
$P(A|P)P(PL_t)P(L_{t'}|L_t)$,
$P(A|P)P(PL_t)P(L_{t'}|PL_t)$,
$P(A|P)P(PL_{t'})P(L_t|A)$,
$P(A|P)P(PL_{t'})P(L_t|AP)$,
$P(A|P)P(PL_{t'})P(L_t|APL_{t'})$,
$P(A|L_t)P(P)P(L_tL_{t'})$,
$P(A|L_t)P(P|L_t)P(L_tL_{t'})$,

$P(A|L_t)P(P|L_{t'})P(L_tL_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_tL_{t'})$,
$P(A|L_t)P(PL_t)P(L_{t'}|A)$,
$P(A|L_t)P(PL_t)P(L_{t'}|AP)$,
$P(A|L_t)P(PL_t)P(L_{t'}|APL_t)$,
$P(A|L_t)P(PL_{t'})P(L_t|L_{t'})$,
$P(A|L_t)P(PL_{t'}|A)P(L_t)$,
$P(A|L_{t'})P(P)P(L_tL_{t'})$,
$P(A|L_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|L_{t'})P(P|AL_{t'})P(L_tL_{t'})$,
$P(A|L_{t'})P(PL_t)P(L_{t'})$,
$P(A|L_{t'})P(PL_t)P(L_{t'}|PL_t)$,
$P(A|L_{t'})P(PL_{t'})P(L_t|P)$,
$P(A|L_{t'})P(PL_{t'})P(L_t|AL_{t'})$,
$P(A|L_{t'})P(PL_t|A)P(L_{t'})$,
$P(A|L_{t'})P(PL_t|AL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P|L_t)P(L_tL_{t'})$,
$P(A|PL_t)P(PL_t)P(L_{t'})$,
$P(A|PL_t)P(PL_t)P(L_{t'}|L_t)$,
$P(A|PL_t)P(PL_t)P(L_{t'}|PL_t)$,
$P(A|PL_t)P(PL_{t'})P(L_t|P)$,
$P(A|PL_t)P(PL_t|L_{t'})P(L_{t'})$,
$P(A|PL_{t'})P(P)P(L_tL_{t'}|P)$,
$P(A|PL_{t'})P(P|L_tL_{t'})P(L_tL_{t'})$,
$P(A|PL_{t'})P(PL_t)P(L_{t'}|L_t)$,
$P(A|PL_{t'})P(PL_{t'})P(L_t|A)$,
$P(A|PL_{t'})P(PL_{t'})P(L_t|AP)$,
$P(A|PL_{t'})P(PL_{t'})P(L_t|AL_{t'})$,
$P(A|L_tL_{t'})P(P)P(L_tL_{t'})$,
$P(A|L_tL_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|L_tL_{t'})P(P|AL_{t'})P(L_tL_{t'})$,
$P(A|L_tL_{t'})P(PL_t)P(L_{t'})$,
$P(A|L_tL_{t'})P(PL_t)P(L_{t'}|PL_t)$,
$P(A|L_tL_{t'})P(PL_{t'})P(L_t|L_{t'})$,
$P(A|L_tL_{t'})P(PL_{t'}|L_t)P(L_t)$,
$P(A|PL_tL_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|PL_tL_{t'})P(PL_t)P(L_{t'})$,
$P(A|PL_tL_{t'})P(PL_t)P(L_{t'}|PL_t)$,
$P(A|PL_tL_{t'})P(PL_{t'})P(L_t|L_{t'})$,
$P(A|PL_tL_{t'})P(PL_{t'}|L_t)P(L_t)$,
$P(AP)P(L_t)P(L_{t'}|P)$,
$P(AP)P(L_t)P(L_{t'}|AL_t)$,
$P(AP)P(L_t)P(L_{t'}|A)$,
$P(AP)P(L_t)P(L_{t'}|L_t)$,
$P(AP)P(L_t)P(L_{t'}|PL_t)$,
$P(AP)P(L_t|P)P(L_{t'}|A)$,
$P(AP)P(L_t|P)P(L_{t'}|AP)$,
$P(AP)P(L_t|P)P(L_{t'}|APL_t)$,
$P(AP)P(L_t|L_{t'})P(L_{t'}|P)$,
$P(AP)P(L_t|AP)P(L_{t'}|A)$,
$P(AP)P(L_t|AP)P(L_{t'}|AP)$,
$P(AP)P(L_t|AP)P(L_{t'}|APL_t)$,
$P(AP)P(L_t|AL_{t'})P(L_{t'}|P)$,
$P(AP)P(L_t|PL_{t'})P(L_{t'}|A)$,
$P(AP)P(L_t|APL_{t'})P(L_{t'})$,
$P(AP)P(L_t|APL_{t'})P(L_{t'}|AP)$,
$P(AL_t)P(P)P(L_{t'}|P)$,
$P(AL_t)P(P)P(L_{t'}|AL_t)$,
$P(AL_t)P(P|A)P(L_{t'})$,
$P(AL_t)P(P|A)P(L_{t'}|L_t)$,
$P(AL_t)P(P|A)P(L_{t'}|PL_t)$,
$P(AL_t)P(P|L_t)P(L_{t'}|A)$,
$P(AL_t)P(P|L_t)P(L_{t'}|AP)$,
$P(AL_t)P(P|L_t)P(L_{t'}|APL_t)$,
$P(AL_t)P(P|L_{t'})P(L_{t'}|L_t)$,
$P(AL_t)P(P|AL_t)P(L_{t'}|A)$,
$P(AL_t)P(P|AL_t)P(L_{t'}|AP)$,
$P(AL_t)P(P|AL_t)P(L_{t'}|APL_t)$,
$P(AL_t)P(P|AL_{t'})P(L_{t'}|L_t)$,
$P(AL_t)P(P|L_tL_{t'})P(L_{t'}|A)$,
$P(AL_t)P(P|AL_tL_{t'})P(L_{t'})$,
$P(AL_t)P(P|AL_tL_{t'})P(L_{t'}|AL_t)$,
$P(AL_{t'})P(P)P(L_t|P)$,
$P(AL_{t'})P(P)P(L_t|AL_{t'})$,
$P(AL_{t'})P(P|A)P(L_t)$,
$P(AL_{t'})P(P|A)P(L_t|L_{t'})$,
$P(AL_{t'})P(P|A)P(L_t|PL_{t'})$,
$P(AL_{t'})P(P|L_t)P(L_t|A)$,
$P(AL_{t'})P(P|L_{t'})P(L_t)$,
$P(AL_{t'})P(P|L_{t'})P(L_t|L_{t'})$,
$P(AL_{t'})P(P|L_{t'})P(L_t|PL_{t'})$,
$P(AL_{t'})P(P|AL_t)P(L_t|A)$,
$P(AL_{t'})P(P|AL_{t'})P(L_t)$,
$P(AL_{t'})P(P|AL_{t'})P(L_t|L_{t'})$,
$P(AL_{t'})P(P|AL_{t'})P(L_t|PL_{t'})$,
$P(AL_{t'})P(P|L_tL_{t'})P(L_t|A)$,
$P(A|L_t)P(P|AL_t)P(L_tL_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_tL_{t'})$,
$P(A|L_t)P(PL_t)P(L_{t'}|P)$,
$P(A|L_t)P(PL_t)P(L_{t'}|AL_t)$,
$P(A|L_t)P(PL_{t'})P(L_t)$,
$P(A|L_t)P(PL_{t'})P(L_t|PL_{t'})$,
$P(A|L_t)P(PL_{t'}|L_t)P(L_t)$,
$P(A|L_{t'})P(P)P(L_tL_{t'}|P)$,
$P(A|L_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|L_{t'})P(PL_t)P(L_{t'})$,
$P(A|L_{t'})P(PL_t)P(L_{t'}|PL_t)$,
$P(A|L_{t'})P(PL_{t'})P(L_t|P)$,
$P(A|L_{t'})P(PL_{t'})P(L_t|AL_{t'})$,
$P(A|L_{t'})P(PL_t|A)P(L_{t'})$,
$P(A|L_{t'})P(PL_t|L_{t'})P(L_{t'})$,
$P(A|PL_t)P(P)P(L_tL_{t'})$,
$P(A|PL_t)P(P|L_{t'})P(L_tL_{t'})$,
$P(A|PL_t)P(PL_t)P(L_{t'}|A)$,
$P(A|PL_t)P(PL_t)P(L_{t'}|AP)$,
$P(A|PL_t)P(PL_t)P(L_{t'}|APL_t)$,
$P(A|PL_t)P(PL_{t'})P(L_t|L_{t'})$,
$P(A|PL_t)P(PL_{t'}|L_t)P(L_t)$,
$P(A|PL_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|PL_{t'})P(PL_t)P(L_{t'})$,
$P(A|PL_{t'})P(PL_t)P(L_{t'}|PL_t)$,
$P(A|PL_{t'})P(PL_{t'})P(L_t|P)$,
$P(A|PL_{t'})P(PL_{t'})P(L_t|AL_{t'})$,
$P(A|PL_{t'})P(PL_t|L_{t'})P(L_{t'})$,
$P(A|L_tL_{t'})P(P)P(L_tL_{t'}|P)$,
$P(A|L_tL_{t'})P(P|L_{t'})P(L_tL_{t'})$,
$P(A|L_tL_{t'})P(PL_t)P(L_{t'}|P)$,
$P(A|L_tL_{t'})P(PL_{t'})P(L_t)$,
$P(A|L_tL_{t'})P(PL_{t'}|L_t)P(L_{t'})$,
$P(A|PL_tL_{t'})P(P)P(L_tL_{t'}|P)$,
$P(A|PL_tL_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|PL_tL_{t'})P(PL_t)P(L_{t'}|P)$,
$P(A|PL_tL_{t'})P(PL_{t'})P(L_t|P)$,
$P(A|PL_tL_{t'})P(PL_t|L_{t'})P(L_{t'})$,
$P(AP)P(L_t)P(L_{t'}|A)$,
$P(AP)P(L_t)P(L_{t'}|AP)$,
$P(AP)P(L_t)P(L_{t'}|APL_t)$,
$P(AP)P(L_t)P(L_{t'}|P)$,
$P(AP)P(L_t)P(L_{t'}|AL_t)$,
$P(AP)P(L_t)P(L_{t'}|L_t)$,
$P(AP)P(L_t)P(L_{t'}|PL_t)$,
$P(AP)P(L_t)P(L_{t'}|A)$,
$P(AP)P(L_t)P(L_{t'}|AP)$,
$P(AP)P(L_t)P(L_{t'}|APL_t)$,
$P(AP)P(L_t|P)P(L_{t'}|P)$,
$P(AP)P(L_t|P)P(L_{t'}|AL_t)$,
$P(AP)P(L_t|L_{t'})P(L_{t'})$,
$P(AP)P(L_t|L_{t'})P(L_{t'}|A)$,
$P(AP)P(L_t|L_{t'})P(L_{t'}|AP)$,
$P(AP)P(L_t|AP)P(L_{t'}|P)$,
$P(AP)P(L_t|AP)P(L_{t'}|AL_t)$,
$P(AP)P(L_t|AL_{t'})P(L_{t'})$,
$P(AP)P(L_t|AL_{t'})P(L_{t'}|A)$,
$P(AP)P(L_t|PL_{t'})P(L_{t'}|P)$,
$P(AP)P(L_t|APL_{t'})P(L_{t'}|A)$,
$P(AL_t)P(P)P(L_{t'})$,
$P(AL_t)P(P)P(L_{t'}|L_t)$,
$P(AL_t)P(P)P(L_{t'}|PL_t)$,
$P(AL_t)P(P|A)P(L_{t'}|A)$,
$P(AL_t)P(P|A)P(L_{t'}|AP)$,
$P(AL_t)P(P|A)P(L_{t'}|APL_t)$,
$P(AL_t)P(P|L_t)P(L_{t'}|P)$,
$P(AL_t)P(P|L_t)P(L_{t'}|AL_t)$,
$P(AL_t)P(P|L_{t'})P(L_{t'})$,
$P(AL_t)P(P|L_{t'})P(L_{t'}|AL_t)$,
$P(AL_t)P(P|AL_t)P(L_{t'}|P)$,
$P(AL_t)P(P|AL_t)P(L_{t'}|AL_t)$,
$P(AL_t)P(P|AL_{t'})P(L_{t'})$,
$P(AL_t)P(P|AL_{t'})P(L_{t'}|AL_t)$,
$P(AL_t)P(P|L_tL_{t'})P(L_{t'}|L_t)$,
$P(AL_t)P(P|AL_tL_{t'})P(L_{t'}|A)$,
$P(AL_{t'})P(P)P(L_t)$,
$P(AL_{t'})P(P)P(L_t|L_{t'})$,
$P(AL_{t'})P(P)P(L_t|PL_{t'})$,
$P(AL_{t'})P(P|A)P(L_t|A)$,
$P(AL_{t'})P(P|A)P(L_t|AP)$,
$P(AL_{t'})P(P|A)P(L_t|APL_{t'})$,
$P(AL_{t'})P(P|L_t)P(L_t|L_{t'})$,
$P(AL_{t'})P(P|L_{t'})P(L_t|A)$,
$P(AL_{t'})P(P|L_{t'})P(L_t|AP)$,
$P(AL_{t'})P(P|L_{t'})P(L_t|APL_{t'})$,
$P(AL_{t'})P(P|AL_t)P(L_t|L_{t'})$,
$P(AL_{t'})P(P|AL_{t'})P(L_t|A)$,
$P(AL_{t'})P(P|AL_{t'})P(L_t|AP)$,
$P(AL_{t'})P(P|AL_{t'})P(L_t|APL_{t'})$,
$P(AL_{t'})P(P|L_tL_{t'})P(L_t|L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_tL_{t'})$,
$P(A|L_t)P(PL_t)P(L_{t'})$,
$P(A|L_t)P(PL_t)P(L_{t'}|L_t)$,
$P(A|L_t)P(PL_t)P(L_{t'}|PL_t)$,
$P(A|L_t)P(PL_{t'})P(L_t|P)$,
$P(A|L_t)P(PL_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(PL_{t'})P(L_t|A)$,
$P(A|L_t)P(PL_{t'}|L_t)P(L_t)$,
$P(A|L_{t'})P(P)P(L_tL_{t'}|P)$,
$P(A|L_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|L_{t'})P(PL_t)P(L_{t'})$,
$P(A|L_{t'})P(PL_t)P(L_{t'}|PL_t)$,
$P(A|L_{t'})P(PL_{t'})P(L_t|P)$,
$P(A|L_{t'})P(PL_{t'}|L_t)P(L_t)$,
$P(A|PL_t)P(P)P(L_tL_{t'})$,
$P(A|PL_t)P(P|L_{t'})P(L_tL_{t'})$,
$P(A|PL_t)P(PL_t)P(L_{t'}|A)$,
$P(A|PL_t)P(PL_t)P(L_{t'}|AP)$,
$P(A|PL_t)P(PL_t)P(L_{t'}|APL_t)$,
$P(A|PL_t)P(PL_{t'})P(L_t|L_{t'})$,
$P(A|PL_t)P(PL_{t'}|L_t)P(L_t)$,
$P(A|PL_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|PL_{t'})P(PL_t)P(L_{t'})$,
$P(A|PL_{t'})P(PL_t)P(L_{t'}|PL_t)$,
$P(A|PL_{t'})P(PL_{t'})P(L_t|P)$,
$P(A|PL_{t'})P(PL_{t'})P(L_t|AL_{t'})$,
$P(A|PL_{t'})P(PL_t|L_{t'})P(L_{t'})$,
$P(A|L_tL_{t'})P(P)P(L_tL_{t'}|P)$,
$P(A|L_tL_{t'})P(P|L_{t'})P(L_tL_{t'})$,
$P(A|L_tL_{t'})P(PL_t)P(L_{t'}|P)$,
$P(A|L_tL_{t'})P(PL_{t'})P(L_t)$,
$P(A|L_tL_{t'})P(PL_{t'}|L_t)P(L_{t'})$,
$P(A|PL_tL_{t'})P(P)P(L_tL_{t'}|P)$,
$P(A|PL_tL_{t'})P(P|L_t)P(L_tL_{t'})$,
$P(A|PL_tL_{t'})P(PL_t)P(L_{t'}|P)$,
$P(A|PL_tL_{t'})P(PL_{t'})P(L_t|P)$,
$P(A|PL_tL_{t'})P(PL_t|L_{t'})P(L_{t'})$,
$P(AP)P(L_t)P(L_{t'}|A)$,
$P(AP)P(L_t)P(L_{t'}|AP)$,
$P(AP)P(L_t)P(L_{t'}|APL_t)$,
$P(AP)P(L_t)P(L_{t'}|P)$,
$P(AP)P(L_t)P(L_{t'}|AL_t)$,
$P(AP)P(L_t)P(L_{t'}|L_t)$,
$P(AP)P(L_t)P(L_{t'}|PL_t)$,
$P(AP)P(L_t|L_{t'})P(L_{t'}|A)$,
$P(AP)P(L_t|AP)P(L_{t'})$,
$P(AP)P(L_t|AP)P(L_{t'}|L_t)$,
$P(AP)P(L_t|AP)P(L_{t'}|PL_t)$,
$P(AP)P(L_t|AL_{t'})P(L_{t'}|A)$,
$P(AP)P(L_t|PL_{t'})P(L_{t'})$,
$P(AP)P(L_t|PL_{t'})P(L_{t'}|AP)$,
$P(AP)P(L_t|APL_{t'})P(L_{t'}|P)$,
$P(AL_t)P(P)P(L_{t'}|A)$,
$P(AL_t)P(P)P(L_{t'}|AP)$,
$P(AL_t)P(P)P(L_{t'}|APL_t)$,
$P(AL_t)P(P|A)P(L_{t'}|P)$,
$P(AL_t)P(P|A)P(L_{t'}|AL_t)$,
$P(AL_t)P(P|L_t)P(L_{t'})$,
$P(AL_t)P(P|L_t)P(L_{t'}|L_t)$,
$P(AL_t)P(P|L_t)P(L_{t'}|PL_t)$,
$P(AL_t)P(P|L_{t'})P(L_{t'}|A)$,
$P(AL_t)P(P|AL_t)P(L_{t'})$,
$P(AL_t)P(P|AL_t)P(L_{t'}|L_t)$,
$P(AL_t)P(P|AL_t)P(L_{t'}|PL_t)$,
$P(AL_t)P(P|AL_{t'})P(L_{t'}|A)$,
$P(AL_t)P(P|L_tL_{t'})P(L_{t'})$,
$P(AL_t)P(P|L_tL_{t'})P(L_{t'}|AL_t)$,
$P(AL_t)P(P|AL_tL_{t'})P(L_{t'}|L_t)$,
$P(AL_{t'})P(P)P(L_t|A)$,
$P(AL_{t'})P(P)P(L_t|AP)$,
$P(AL_{t'})P(P)P(L_t|APL_{t'})$,
$P(AL_{t'})P(P|A)P(L_t|P)$,
$P(AL_{t'})P(P|A)P(L_t|AL_{t'})$,
$P(AL_{t'})P(P|L_t)P(L_t)$,
$P(AL_{t'})P(P|L_t)P(L_t|AL_{t'})$,
$P(AL_{t'})P(P|L_{t'})P(L_t|P)$,
$P(AL_{t'})P(P|L_{t'})P(L_t|AL_{t'})$,
$P(AL_{t'})P(P|AL_t)P(L_t)$,
$P(AL_{t'})P(P|AL_t)P(L_t|AL_{t'})$,
$P(AL_{t'})P(P|AL_{t'})P(L_t|P)$,
$P(AL_{t'})P(P|AL_{t'})P(L_t|AL_{t'})$,
$P(AL_{t'})P(P|L_tL_{t'})P(L_t)$,
$P(AL_{t'})P(P|L_tL_{t'})P(L_t|AL_{t'})$,

$P(AL_{t'})P(P|AL_tL_{t'})P(L_t),$
$P(AL_{t'})P(P|AL_tL_{t'})P(L_t|AL_{t'}),$
$P(AP|L_t)P(L_t)P(L_{t'}|P),$
$P(AP|L_t)P(L_t)P(L_{t'}|AL_t),$
$P(AP|L_t)P(L_t|L_{t'})P(L_{t'}),$
$P(AP|L_{t'})P(L_t|A)P(L_{t'}),$
$P(AP|L_{t'})P(L_t|AP)P(L_{t'}),$
$P(AP|L_{t'})P(L_t|APL_{t'})P(L_{t'}),$
$P(AL_t|P)P(P)P(L_{t'}|P),$
$P(AL_t|P)P(P)P(L_{t'}|AL_t),$
$P(AL_t|P)P(P|L_{t'})P(L_{t'}),$
$P(AL_t|L_{t'})P(P|A)P(L_{t'}),$
$P(AL_t|L_{t'})P(P|AL_t)P(L_{t'}),$
$P(AL_t|L_{t'})P(P|AL_tL_{t'})P(L_{t'}),$
$P(AL_{t'}|P)P(P)P(L_t|P),$
$P(AL_{t'}|P)P(P)P(L_t|AL_{t'}),$
$P(AL_{t'}|P)P(P|L_t)P(L_t),$
$P(AL_{t'}|L_t)P(P|A)P(L_t),$
$P(AL_{t'}|L_t)P(P|AL_t)P(L_t),$
$P(AL_{t'}|L_t)P(P|AL_tL_{t'})P(L_t),$
$P(AP|L_tL_{t'})P(L_t|L_{t'})P(L_{t'}),$
$P(AL_t|PL_{t'})P(P|L_{t'})P(L_{t'}),$
$P(AL_{t'}|PL_t)P(P|L_t)P(L_t),$
$P(A)P(P)P(L_t)P(L_{t'}|P),$
$P(A)P(P)P(L_t)P(L_{t'}|AL_t),$
$P(A)P(P)P(L_t|A)P(L_{t'}),$
$P(A)P(P)P(L_t|A)P(L_{t'}|L_t),$
$P(A)P(P)P(L_t|A)P(L_{t'}|PL_t),$
$P(A)P(P)P(L_t|P)P(L_{t'}|A),$
$P(A)P(P)P(L_t|P)P(L_{t'}|AP),$
$P(A)P(P)P(L_t|P)P(L_{t'}|APL_t),$
$P(A)P(P)P(L_t|L_{t'})P(L_{t'}|P),$
$P(A)P(P)P(L_t|AP)P(L_{t'}|A),$
$P(A)P(P)P(L_t|AP)P(L_{t'}|AP),$
$P(A)P(P)P(L_t|AP)P(L_{t'}|APL_t),$
$P(A)P(P)P(L_t|AL_{t'})P(L_{t'}|P),$
$P(A)P(P)P(L_t|PL_{t'})P(L_{t'}|A),$
$P(A)P(P)P(L_t|APL_{t'})P(L_{t'}),$
$P(A)P(P)P(L_t|APL_{t'})P(L_{t'}|AP),$
$P(A)P(P|A)P(L_t)P(L_{t'}|P),$
$P(A)P(P|A)P(L_t)P(L_{t'}|AL_t),$
$P(A)P(P|A)P(L_t|A)P(L_{t'}),$
$P(A)P(P|A)P(L_t|A)P(L_{t'}|L_t),$
$P(A)P(P|A)P(L_t|A)P(L_{t'}|PL_t),$
$P(A)P(P|A)P(L_t|P)P(L_{t'}|A),$
$P(A)P(P|A)P(L_t|P)P(L_{t'}|AP),$
$P(A)P(P|A)P(L_t|P)P(L_{t'}|APL_t),$
$P(A)P(P|A)P(L_t|L_{t'})P(L_{t'}|P),$
$P(A)P(P|A)P(L_t|AP)P(L_{t'}|A),$
$P(A)P(P|A)P(L_t|AP)P(L_{t'}|AP),$
$P(A)P(P|A)P(L_t|AP)P(L_{t'}|APL_t),$
$P(A)P(P|A)P(L_t|AL_{t'})P(L_{t'}|P),$
$P(A)P(P|A)P(L_t|PL_{t'})P(L_{t'}|A),$
$P(A)P(P|A)P(L_t|APL_{t'})P(L_{t'}),$
$P(A)P(P|A)P(L_t|APL_{t'})P(L_{t'}|AP),$
$P(A)P(P|L_t)P(L_t)P(L_{t'}|P),$
$P(A)P(P|L_t)P(L_t)P(L_{t'}|AL_t),$
$P(A)P(P|L_t)P(L_t|A)P(L_{t'}),$
$P(A)P(P|L_t)P(L_t|A)P(L_{t'}|L_t),$
$P(A)P(P|L_t)P(L_t|A)P(L_{t'}|PL_t),$
$P(A)P(P|L_t)P(L_t|L_{t'})P(L_{t'}|A),$
$P(A)P(P|L_{t'})P(L_t)P(L_{t'}),$
$P(A)P(P|L_{t'})P(L_t)P(L_{t'}|AL_t),$
$P(A)P(P|L_{t'})P(L_t|A)P(L_{t'}|L_t),$
$P(A)P(P|L_{t'})P(L_t|P)P(L_{t'}|A),$
$P(A)P(P|L_{t'})P(L_t|AP)P(L_{t'}),$
$P(A)P(P|L_{t'})P(L_t|AL_{t'})P(L_{t'}|A),$
$P(A)P(P|L_{t'})P(L_t|APL_{t'})P(L_{t'}),$
$P(A)P(P|AL_t)P(L_t)P(L_{t'}|A),$
$P(A)P(P|AL_t)P(L_t)P(L_{t'}|AP),$
$P(A)P(P|AL_t)P(L_t)P(L_{t'}|APL_t),$
$P(A)P(P|AL_t)P(L_t|A)P(L_{t'}|P),$
$P(A)P(P|AL_t)P(L_t|A)P(L_{t'}|AL_t),$
$P(A)P(P|AL_t)P(L_t|L_{t'})P(L_{t'}),$
$P(A)P(P|AL_t)P(L_t|AL_{t'})P(L_{t'}|A),$
$P(A)P(P|AL_{t'})P(L_t)P(L_{t'}|L_t),$
$P(A)P(P|AL_{t'})P(L_t|A)P(L_{t'}|A),$
$P(A)P(P|AL_{t'})P(L_t|P)P(L_{t'}),$
$P(A)P(P|AL_{t'})P(L_t|L_{t'})P(L_{t'}|A),$
$P(A)P(P|AL_{t'})P(L_t|AL_{t'})P(L_{t'}),$
$P(A)P(P|AL_{t'})P(L_t|PL_{t'})P(L_{t'}|A),$
$P(A)P(P|L_tL_{t'})P(L_t)P(L_{t'}),$
$P(A)P(P|L_tL_{t'})P(L_t)P(L_{t'}|AL_t),$
$P(A)P(P|L_tL_{t'})P(L_t|A)P(L_{t'}|L_t),$
$P(A)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'}|A),$
$P(A)P(P|AL_tL_{t'})P(L_t)P(L_{t'}),$
$P(AL_{t'})P(P|AL_tL_{t'})P(L_t|A),$
$P(AP|L_t)P(L_t)P(L_{t'}),$
$P(AP|L_t)P(L_t)P(L_{t'}|L_t),$
$P(AP|L_t)P(L_t)P(L_{t'}|PL_t),$
$P(AP|L_{t'})P(L_t)P(L_{t'}),$
$P(AP|L_{t'})P(L_t|P)P(L_{t'}),$
$P(AP|L_{t'})P(L_t|AL_{t'})P(L_{t'}),$
$P(AL_t|P)P(P)P(L_{t'}),$
$P(AL_t|P)P(P)P(L_{t'}|L_t),$
$P(AL_t|P)P(P)P(L_{t'}|PL_t),$
$P(AL_t|L_{t'})P(P)P(L_{t'}),$
$P(AL_t|L_{t'})P(P|L_t)P(L_{t'}),$
$P(AL_t|L_{t'})P(P|AL_{t'})P(L_{t'}),$
$P(AL_{t'}|P)P(P)P(L_t),$
$P(AL_{t'}|P)P(P)P(L_t|L_{t'}),$
$P(AL_{t'}|P)P(P)P(L_t|PL_{t'}),$
$P(AL_{t'}|L_t)P(P)P(L_t),$
$P(AL_{t'}|L_t)P(P|L_t)P(L_t),$
$P(AL_{t'}|L_t)P(P|AL_{t'})P(L_t),$
$P(AP|L_tL_{t'})P(L_t)P(L_{t'}),$
$P(AL_t|PL_{t'})P(P)P(L_{t'}),$
$P(AL_{t'}|PL_t)P(P)P(L_t),$
$P(A)P(P)P(L_t)P(L_{t'}),$
$P(A)P(P)P(L_t)P(L_{t'}|L_t),$
$P(A)P(P)P(L_t)P(L_{t'}|PL_t),$
$P(A)P(P)P(L_t|A)P(L_{t'}|A),$
$P(A)P(P)P(L_t|A)P(L_{t'}|AP),$
$P(A)P(P)P(L_t|A)P(L_{t'}|APL_t),$
$P(A)P(P)P(L_t|P)P(L_{t'}|P),$
$P(A)P(P)P(L_t|P)P(L_{t'}|AL_t),$
$P(A)P(P)P(L_t|L_{t'})P(L_{t'}),$
$P(A)P(P)P(L_t|L_{t'})P(L_{t'}|AP),$
$P(A)P(P)P(L_t|AP)P(L_{t'}|P),$
$P(A)P(P)P(L_t|AP)P(L_{t'}|AL_t),$
$P(A)P(P)P(L_t|AL_{t'})P(L_{t'}),$
$P(A)P(P)P(L_t|AL_{t'})P(L_{t'}|AP),$
$P(A)P(P)P(L_t|PL_{t'})P(L_{t'}|P),$
$P(A)P(P)P(L_t|APL_{t'})P(L_{t'}|A),$
$P(A)P(P|A)P(L_t)P(L_{t'}),$
$P(A)P(P|A)P(L_t)P(L_{t'}|L_t),$
$P(A)P(P|A)P(L_t)P(L_{t'}|PL_t),$
$P(A)P(P|A)P(L_t|A)P(L_{t'}|A),$
$P(A)P(P|A)P(L_t|A)P(L_{t'}|AP),$
$P(A)P(P|A)P(L_t|A)P(L_{t'}|APL_t),$
$P(A)P(P|A)P(L_t|P)P(L_{t'}|P),$
$P(A)P(P|A)P(L_t|P)P(L_{t'}|AL_t),$
$P(A)P(P|A)P(L_t|L_{t'})P(L_{t'}),$
$P(A)P(P|A)P(L_t|L_{t'})P(L_{t'}|AP),$
$P(A)P(P|A)P(L_t|AP)P(L_{t'}|P),$
$P(A)P(P|A)P(L_t|AP)P(L_{t'}|AL_t),$
$P(A)P(P|A)P(L_t|AL_{t'})P(L_{t'}),$
$P(A)P(P|A)P(L_t|AL_{t'})P(L_{t'}|AP),$
$P(A)P(P|A)P(L_t|PL_{t'})P(L_{t'}|P),$
$P(A)P(P|A)P(L_t|APL_{t'})P(L_{t'}|A),$
$P(A)P(P|L_t)P(L_t)P(L_{t'}),$
$P(A)P(P|L_t)P(L_t)P(L_{t'}|L_t),$
$P(A)P(P|L_t)P(L_t)P(L_{t'}|PL_t),$
$P(A)P(P|L_t)P(L_t|A)P(L_{t'}|A),$
$P(A)P(P|L_t)P(L_t|A)P(L_{t'}|AP),$
$P(A)P(P|L_t)P(L_t|A)P(L_{t'}|APL_t),$
$P(A)P(P|L_t)P(L_t|AL_{t'})P(L_{t'}),$
$P(A)P(P|L_{t'})P(L_t)P(L_{t'}|A),$
$P(A)P(P|L_{t'})P(L_t|A)P(L_{t'}),$
$P(A)P(P|L_{t'})P(L_t|A)P(L_{t'}|AL_t),$
$P(A)P(P|L_{t'})P(L_t|L_{t'})P(L_{t'}),$
$P(A)P(P|L_{t'})P(L_t|AP)P(L_{t'}|A),$
$P(A)P(P|L_{t'})P(L_t|PL_{t'})P(L_{t'}),$
$P(A)P(P|L_{t'})P(L_t|APL_{t'})P(L_{t'}|A),$
$P(A)P(P|AL_t)P(L_t)P(L_{t'}|P),$
$P(A)P(P|AL_t)P(L_t)P(L_{t'}|AL_t),$
$P(A)P(P|AL_t)P(L_t|A)P(L_{t'}),$
$P(A)P(P|AL_t)P(L_t|A)P(L_{t'}|L_t),$
$P(A)P(P|AL_t)P(L_t|A)P(L_{t'}|PL_t),$
$P(A)P(P|AL_t)P(L_t|L_{t'})P(L_{t'}|A),$
$P(A)P(P|AL_{t'})P(L_t)P(L_{t'}),$
$P(A)P(P|AL_{t'})P(L_t)P(L_{t'}|AL_t),$
$P(A)P(P|AL_{t'})P(L_t|A)P(L_{t'}|L_t),$
$P(A)P(P|AL_{t'})P(L_t|P)P(L_{t'}|A),$
$P(A)P(P|AL_{t'})P(L_t|AP)P(L_{t'}),$
$P(A)P(P|AL_{t'})P(L_t|AL_{t'})P(L_{t'}|A),$
$P(A)P(P|AL_{t'})P(L_t|APL_{t'})P(L_{t'}),$
$P(A)P(P|L_tL_{t'})P(L_t)P(L_{t'}|A),$
$P(A)P(P|L_tL_{t'})P(L_t|A)P(L_{t'}),$
$P(A)P(P|L_tL_{t'})P(L_t|A)P(L_{t'}|AL_t),$
$P(A)P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'}),$
$P(A)P(P|AL_tL_{t'})P(L_t)P(L_{t'}|A),$
$P(AL_{t'})P(P|AL_tL_{t'})P(L_t|L_{t'}),$
$P(AP|L_t)P(L_t)P(L_{t'}|A),$
$P(AP|L_t)P(L_t)P(L_{t'}|AP),$
$P(AP|L_t)P(L_t)P(L_{t'}|APL_t),$
$P(AP|L_{t'})P(L_t)P(L_{t'}|L_t),$
$P(AP|L_{t'})P(L_t|L_{t'})P(L_{t'}),$
$P(AP|L_{t'})P(L_t|PL_{t'})P(L_{t'}),$
$P(AL_t|P)P(P)P(L_{t'}|A),$
$P(AL_t|P)P(P)P(L_{t'}|AP),$
$P(AL_t|P)P(P)P(L_{t'}|APL_t),$
$P(AL_t|L_{t'})P(P)P(L_{t'}|P),$
$P(AL_t|L_{t'})P(P|L_{t'})P(L_{t'}),$
$P(AL_t|L_{t'})P(P|L_tL_{t'})P(L_{t'}),$
$P(AL_{t'}|P)P(P)P(L_t|A),$
$P(AL_{t'}|P)P(P)P(L_t|AP),$
$P(AL_{t'}|P)P(P)P(L_t|APL_{t'}),$
$P(AL_{t'}|L_t)P(P)P(L_t|P),$
$P(AL_{t'}|L_t)P(P|L_{t'})P(L_t),$
$P(AL_{t'}|L_t)P(P|L_tL_{t'})P(L_t),$
$P(AP|L_tL_{t'})P(L_t)P(L_{t'}|L_t),$
$P(AL_t|PL_{t'})P(P)P(L_{t'}|P),$
$P(AL_{t'}|PL_t)P(P)P(L_t|P),$
$P(A)P(P)P(L_t)P(L_{t'}|A),$
$P(A)P(P)P(L_t)P(L_{t'}|AP),$
$P(A)P(P)P(L_t)P(L_{t'}|APL_t),$
$P(A)P(P)P(L_t|A)P(L_{t'}|P),$
$P(A)P(P)P(L_t|A)P(L_{t'}|AL_t),$
$P(A)P(P)P(L_t|P)P(L_{t'}),$
$P(A)P(P)P(L_t|P)P(L_{t'}|L_t),$
$P(A)P(P)P(L_t|P)P(L_{t'}|PL_t),$
$P(A)P(P)P(L_t|L_{t'})P(L_{t'}|A),$
$P(A)P(P)P(L_t|AP)P(L_{t'}),$
$P(A)P(P)P(L_t|AP)P(L_{t'}|L_t),$
$P(A)P(P)P(L_t|AP)P(L_{t'}|PL_t),$
$P(A)P(P)P(L_t|AL_{t'})P(L_{t'}|A),$
$P(A)P(P)P(L_t|PL_{t'})P(L_{t'}),$
$P(A)P(P)P(L_t|PL_{t'})P(L_{t'}|AP),$
$P(A)P(P)P(L_t|APL_{t'})P(L_{t'}|P),$
$P(A)P(P|A)P(L_t)P(L_{t'}|A),$
$P(A)P(P|A)P(L_t)P(L_{t'}|AP),$
$P(A)P(P|A)P(L_t)P(L_{t'}|APL_t),$
$P(A)P(P|A)P(L_t|A)P(L_{t'}|P),$
$P(A)P(P|A)P(L_t|A)P(L_{t'}|AL_t),$
$P(A)P(P|A)P(L_t|P)P(L_{t'}),$
$P(A)P(P|A)P(L_t|P)P(L_{t'}|L_t),$
$P(A)P(P|A)P(L_t|P)P(L_{t'}|PL_t),$
$P(A)P(P|A)P(L_t|L_{t'})P(L_{t'}|A),$
$P(A)P(P|A)P(L_t|AP)P(L_{t'}),$
$P(A)P(P|A)P(L_t|AP)P(L_{t'}|L_t),$
$P(A)P(P|A)P(L_t|AP)P(L_{t'}|PL_t),$
$P(A)P(P|A)P(L_t|AL_{t'})P(L_{t'}|A),$
$P(A)P(P|A)P(L_t|PL_{t'})P(L_{t'}),$
$P(A)P(P|A)P(L_t|PL_{t'})P(L_{t'}|AP),$
$P(A)P(P|A)P(L_t|APL_{t'})P(L_{t'}|P),$
$P(A)P(P|L_t)P(L_t)P(L_{t'}|A),$
$P(A)P(P|L_t)P(L_t)P(L_{t'}|AP),$
$P(A)P(P|L_t)P(L_t)P(L_{t'}|APL_t),$
$P(A)P(P|L_t)P(L_t|A)P(L_{t'}|P),$
$P(A)P(P|L_t)P(L_t|A)P(L_{t'}|AL_t),$
$P(A)P(P|L_t)P(L_t|L_{t'})P(L_{t'}),$
$P(A)P(P|L_t)P(L_t|AL_{t'})P(L_{t'}|A),$
$P(A)P(P|L_{t'})P(L_t)P(L_{t'}|L_t),$
$P(A)P(P|L_{t'})P(L_t|A)P(L_{t'}|A),$
$P(A)P(P|L_{t'})P(L_t|P)P(L_{t'}),$
$P(A)P(P|L_{t'})P(L_t|L_{t'})P(L_{t'}|A),$
$P(A)P(P|L_{t'})P(L_t|AL_{t'})P(L_{t'}),$
$P(A)P(P|L_{t'})P(L_t|PL_{t'})P(L_{t'}|A),$
$P(A)P(P|AL_t)P(L_t)P(L_{t'}),$
$P(A)P(P|AL_t)P(L_t)P(L_{t'}|L_t),$
$P(A)P(P|AL_t)P(L_t)P(L_{t'}|PL_t),$
$P(A)P(P|AL_t)P(L_t|A)P(L_{t'}|A),$
$P(A)P(P|AL_t)P(L_t|A)P(L_{t'}|AP),$
$P(A)P(P|AL_t)P(L_t|A)P(L_{t'}|APL_t),$
$P(A)P(P|AL_t)P(L_t|AL_{t'})P(L_{t'}),$
$P(A)P(P|AL_{t'})P(L_t)P(L_{t'}|A),$
$P(A)P(P|AL_{t'})P(L_t|A)P(L_{t'}),$
$P(A)P(P|AL_{t'})P(L_t|A)P(L_{t'}|AL_t),$
$P(A)P(P|AL_{t'})P(L_t|L_{t'})P(L_{t'}),$
$P(A)P(P|AL_{t'})P(L_t|AP)P(L_{t'}|A),$
$P(A)P(P|AL_{t'})P(L_t|PL_{t'})P(L_{t'}),$
$P(A)P(P|AL_{t'})P(L_t|APL_{t'})P(L_{t'}|A),$
$P(A)P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t),$
$P(A)P(P|L_tL_{t'})P(L_t|A)P(L_{t'}|A),$
$P(A)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'}),$
$P(A)P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'}|A),$
$P(A)P(P|AL_tL_{t'})P(L_t)P(L_{t'}|L_t),$

$P(A)P(P|AL_tL_{t'})P(L_t)P(L_{t'}|AL_t)$,
$P(A)P(P|AL_tL_{t'})P(L_t|A)P(L_{t'}|L_t)$,
$P(A)P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t)P(L_{t'})$,
$P(A|P)P(P)P(L_t)P(L_{t'}|L_t)$,
$P(A|P)P(P)P(L_t)P(L_{t'}|PL_t)$,
$P(A|P)P(P)P(L_t|A)P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|A)P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t|A)P(L_{t'}|APL_t)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|AL_t)$,
$P(A|P)P(P)P(L_t|L_{t'})P(L_{t'})$,
$P(A|P)P(P)P(L_t|L_{t'})P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|AL_t)$,
$P(A|P)P(P)P(L_t|AL_{t'})P(L_{t'})$,
$P(A|P)P(P)P(L_t|AL_{t'})P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t|PL_{t'})P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t|APL_{t'})P(L_{t'}|A)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'})$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|L_t)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|PL_t)$,
$P(A|P)P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|P)P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|P)P(P|L_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|P)P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P)P(L_t)P(L_{t'}|L_t)$,
$P(A|L_t)P(P)P(L_t)P(L_{t'}|PL_t)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|A)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|AP)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|APL_t)$,
$P(A|L_t)P(P)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t)P(L_{t'}|A)$,
$P(A|L_t)P(P|A)P(L_t)P(L_{t'}|AP)$,
$P(A|L_t)P(P|A)P(L_t)P(L_{t'}|APL_t)$,
$P(A|L_t)P(P|L_t)P(L_t)P(L_{t'}|A)$,
$P(A|L_t)P(P|L_t)P(L_t)P(L_{t'}|AP)$,
$P(A|L_t)P(P|L_t)P(L_t)P(L_{t'}|APL_t)$,
$P(A|L_t)P(P|L_{t'})P(L_t)P(L_{t'}|A)$,
$P(A|L_t)P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t)P(L_{t'}|L_t)$,
$P(A|L_t)P(P|AL_t)P(L_t)P(L_{t'}|PL_t)$,
$P(A|L_t)P(P|AL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t)P(L_{t'}|AL_t)$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t)P(L_{t'}|A)$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|L_{t'})P(P)P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t)P(L_{t'}|PL_t)$,
$P(A|L_{t'})P(P)P(L_t|P)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|P)P(L_{t'}|PL_t)$,
$P(A|L_{t'})P(P)P(L_t|AP)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|AL_{t'})P(L_{t'}|P)$,
$P(A|L_{t'})P(P)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t)P(L_{t'}|L_t)$,
$P(A|L_{t'})P(P|A)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_t)P(L_t)P(L_{t'}|P)$,
$P(A|L_{t'})P(P|L_t)P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_t)P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|AL_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|L_{t'})P(P|AL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_{t'})P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|L_t)$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|PL_t)$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'}|A)$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'}|AP)$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'}|APL_t)$,
$P(A|PL_t)P(P)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P|L_t)P(L_t)P(L_{t'}|A)$,
$P(A|PL_t)P(P|L_t)P(L_t)P(L_{t'}|AP)$,
$P(A|PL_t)P(P|L_t)P(L_t|P)P(L_{t'}|L_t)$,
$P(A|PL_t)P(P|L_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|PL_t)P(P|L_{t'})P(L_t|PL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|PL_{t'})P(P)P(L_t)P(L_{t'})$,
$P(A|PL_{t'})P(P)P(L_t)P(L_{t'}|L_t)$,
$P(A)P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A)P(P|AL_tL_{t'})P(L_t|A)P(L_{t'}|AL_t)$,
$P(A)P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|P)P(P)P(L_t)P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t)P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t)P(L_{t'}|APL_t)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'})$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|AL_t)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|PL_t)$,
$P(A|P)P(P)P(L_t|L_{t'})P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'})$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|L_t)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|PL_t)$,
$P(A|P)P(P)P(L_t|AL_{t'})P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|P)P(P)P(L_t|PL_{t'})P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t|APL_{t'})P(L_{t'}|P)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|A)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|AP)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|APL_t)$,
$P(A|P)P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|P)P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|P)P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|L_t)P(P)P(L_t)P(L_{t'}|A)$,
$P(A|L_t)P(P)P(L_t)P(L_{t'}|AP)$,
$P(A|L_t)P(P)P(L_t)P(L_{t'}|APL_t)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|P)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|AL_t)$,
$P(A|L_t)P(P)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P)P(L_t|PL_{t'})P(L_{t'}|P)$,
$P(A|L_t)P(P|A)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t)P(L_{t'}|P)$,
$P(A|L_{t'})P(P)P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|P)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|AP)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|AP)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_t)P(L_t)P(L_{t'}|L_t)$,
$P(A|L_{t'})P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|A)$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|AP)$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|APL_t)$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'}|P)$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'}|AL_t)$,
$P(A|PL_t)P(P)P(L_t|L_{t'})P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t|PL_{t'})P(L_{t'}|P)$,
$P(A|PL_t)P(P|L_t)P(L_t)P(L_{t'}|A)$,
$P(A|PL_t)P(P|L_t)P(L_t)P(L_{t'}|AP)$,
$P(A|PL_t)P(P|L_t)P(L_t|P)P(L_{t'}|L_t)$,
$P(A|PL_t)P(P|L_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|PL_t)P(P|L_{t'})P(L_t|PL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|PL_{t'})P(P)P(L_t)P(L_{t'})$,
$P(A|PL_{t'})P(P)P(L_t)P(L_{t'}|PL_t)$,
$P(A)P(P|AL_tL_{t'})P(L_t|A)P(L_{t'}|A)$,
$P(A)P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A)P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'}|AL_t)$,
$P(A|P)P(P)P(L_t)P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t)P(L_{t'}|AL_t)$,
$P(A|P)P(P)P(L_t|A)P(L_{t'}|L_t)$,
$P(A|P)P(P)P(L_t|A)P(L_{t'}|PL_t)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|APL_t)$,
$P(A|P)P(P)P(L_t|L_{t'})P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|APL_t)$,
$P(A|P)P(P)P(L_t|AL_{t'})P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t|PL_{t'})P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|P)P(P)P(L_t|APL_{t'})P(L_{t'}|AP)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|P)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|AL_t)$,
$P(A|P)P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|P)P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|P)P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|L_t)P(P)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P)P(L_t)P(L_{t'}|AL_t)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|L_t)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|PL_t)$,
$P(A|L_t)P(P|A)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t)P(L_{t'}|P)$,
$P(A|L_{t'})P(P)P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|P)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|AP)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|AP)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_t)P(L_t)P(L_{t'}|L_t)$,
$P(A|L_{t'})P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|A)$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|AP)$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|APL_t)$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'}|AL_t)$,
$P(A|PL_t)P(P)P(L_t|L_{t'})P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t|PL_{t'})P(L_{t'}|P)$,
$P(A|PL_t)P(P|L_t)P(L_t)P(L_{t'})$,
$P(A|PL_t)P(P|L_t)P(L_t|P)P(L_{t'}|L_t)$,
$P(A|PL_t)P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|PL_t)P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|PL_t)P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|PL_t)P(P|L_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|PL_t)P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|PL_{t'})P(P)P(L_t)P(L_{t'})$,
$P(A|PL_{t'})P(P)P(L_t)P(L_{t'}|PL_t)$,
$P(A)P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A)P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A)P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t)P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t)P(L_{t'}|AL_t)$,
$P(A|P)P(P)P(L_t|A)P(L_{t'}|L_t)$,
$P(A|P)P(P)P(L_t|A)P(L_{t'}|PL_t)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t|P)P(L_{t'}|APL_t)$,
$P(A|P)P(P)P(L_t|L_{t'})P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|AP)$,
$P(A|P)P(P)P(L_t|AP)P(L_{t'}|APL_t)$,
$P(A|P)P(P)P(L_t|AL_{t'})P(L_{t'}|P)$,
$P(A|P)P(P)P(L_t|PL_{t'})P(L_{t'}|A)$,
$P(A|P)P(P)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|P)P(P)P(L_t|APL_{t'})P(L_{t'}|AP)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|P)$,
$P(A|P)P(P|L_t)P(L_t)P(L_{t'}|AL_t)$,
$P(A|P)P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|P)P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|P)P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|L_t)P(P)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P)P(L_t)P(L_{t'}|AL_t)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|L_t)$,
$P(A|L_t)P(P)P(L_t|P)P(L_{t'}|PL_t)$,
$P(A|L_t)P(P|A)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|A)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_{t'})P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_t)P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t)P(L_{t'}|P)$,
$P(A|L_{t'})P(P)P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|P)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|AP)P(L_{t'})$,
$P(A|L_{t'})P(P)P(L_t|AL_{t'})P(L_{t'}|P)$,
$P(A|L_{t'})P(P)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|AP)P(L_{t'})$,
$P(A|L_{t'})P(P|A)P(L_t|APL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_t)P(L_t)P(L_{t'}|L_t)$,
$P(A|L_{t'})P(P|L_t)P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_t)P(L_t|PL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|P)P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|L_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|A)P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'})$,
$P(A|L_{t'})P(P|AL_tL_{t'})P(L_t|AL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|A)$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|AP)$,
$P(A|PL_t)P(P)P(L_t)P(L_{t'}|APL_t)$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'}|P)$,
$P(A|PL_t)P(P)P(L_t|P)P(L_{t'}|AL_t)$,
$P(A|PL_t)P(P)P(L_t|L_{t'})P(L_{t'})$,
$P(A|PL_t)P(P)P(L_t|PL_{t'})P(L_{t'}|P)$,
$P(A|PL_t)P(P|L_t)P(L_t)P(L_{t'}|A)$,
$P(A|PL_t)P(P|L_t)P(L_t)P(L_{t'}|AP)$,
$P(A|PL_t)P(P|L_t)P(L_t|P)P(L_{t'}|L_t)$,
$P(A|PL_t)P(P|L_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|PL_t)P(P|L_{t'})P(L_t|PL_{t'})P(L_{t'})$,
$P(A|PL_t)P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t)$,
$P(A|PL_{t'})P(P)P(L_t)P(L_{t'})$,
$P(A|PL_{t'})P(P)P(L_t)P(L_{t'}|PL_t)$,
$P(A|PL_{t'})P(P)P(L_t|A)P(L_{t'})$,

$$
\begin{array}{lll}
P(A|PL_{t'})P(P)P(L_t|A)P(L_{t'}|P), & P(A|PL_{t'})P(P)P(L_t|P)P(L_{t'}), & P(A|PL_{t'})P(P)P(L_t|P)P(L_{t'}|P), \\
P(A|PL_{t'})P(P)P(L_t|P)P(L_{t'}|L_t), & P(A|PL_{t'})P(P)P(L_t|P)P(L_{t'}|PL_t), & P(A|PL_{t'})P(P)P(L_t|L_{t'})P(L_{t'}), \\
P(A|PL_{t'})P(P)P(L_t|L_{t'})P(L_{t'}|P), & P(A|PL_{t'})P(P)P(L_t|AP)P(L_{t'}), & P(A|PL_{t'})P(P)P(L_t|AP)P(L_{t'}|P), \\
P(A|PL_{t'})P(P)P(L_t|AL_{t'})P(L_{t'}), & P(A|PL_{t'})P(P)P(L_t|AL_{t'})P(L_{t'}|P), & P(A|PL_{t'})P(P)P(L_t|PL_{t'})P(L_{t'}), \\
P(A|PL_{t'})P(P)P(L_t|PL_{t'})P(L_{t'}|P), & P(A|PL_{t'})P(P)P(L_t|APL_{t'})P(L_{t'}), & P(A|PL_{t'})P(P)P(L_t|APL_{t'})P(L_{t'}|P), \\
P(A|PL_{t'})P(P|L_t)P(L_t)P(L_{t'}), & P(A|PL_{t'})P(P|L_t)P(L_t)P(L_{t'}|P), & P(A|PL_{t'})P(P|L_t)P(L_t)P(L_{t'}|L_t), \\
P(A|PL_{t'})P(P|L_t)P(L_t)P(L_{t'}|PL_t), & P(A|PL_{t'})P(P|L_t)P(L_t|L_{t'})P(L_{t'}), & P(A|PL_{t'})P(P|L_{t'})P(L_t)P(L_{t'}), \\
P(A|PL_{t'})P(P|L_{t'})P(L_t)P(L_{t'}|L_t), & P(A|PL_{t'})P(P|L_{t'})P(L_t|A)P(L_{t'}), & P(A|PL_{t'})P(P|L_{t'})P(L_t|P)P(L_{t'}), \\
P(A|PL_{t'})P(P|L_{t'})P(L_t|L_{t'})P(L_{t'}), & P(A|PL_{t'})P(P|L_{t'})P(L_t|AP)P(L_{t'}), & P(A|PL_{t'})P(P|L_{t'})P(L_t|AL_{t'})P(L_{t'}), \\
P(A|PL_{t'})P(P|L_{t'})P(L_t|PL_{t'})P(L_{t'}), & P(A|PL_{t'})P(P|L_{t'})P(L_t|APL_{t'})P(L_{t'}), & P(A|PL_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'}), \\
P(A|PL_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t), & P(A|PL_{t'})P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'}), & P(A|L_tL_{t'})P(P)P(L_t)P(L_{t'}), \\
P(A|L_tL_{t'})P(P)P(L_t)P(L_{t'}|P), & P(A|L_tL_{t'})P(P)P(L_t)P(L_{t'}|L_t), & P(A|L_tL_{t'})P(P)P(L_t)P(L_{t'}|PL_t), \\
P(A|L_tL_{t'})P(P)P(L_t|P)P(L_{t'}), & P(A|L_tL_{t'})P(P)P(L_t|P)P(L_{t'}|P), & P(A|L_tL_{t'})P(P)P(L_t|P)P(L_{t'}|L_t), \\
P(A|L_tL_{t'})P(P)P(L_t|P)P(L_{t'}|PL_t), & P(A|L_tL_{t'})P(P)P(L_t|L_{t'})P(L_{t'}), & P(A|L_tL_{t'})P(P)P(L_t|L_{t'})P(L_{t'}|P), \\
P(A|L_tL_{t'})P(P)P(L_t|PL_{t'})P(L_{t'}), & P(A|L_tL_{t'})P(P)P(L_t|PL_{t'})P(L_{t'}|P), & P(A|L_tL_{t'})P(P|A)P(L_t)P(L_{t'}), \\
P(A|L_tL_{t'})P(P|A)P(L_t)P(L_{t'}|L_t), & P(A|L_tL_{t'})P(P|A)P(L_t|L_{t'})P(L_{t'}), & P(A|L_tL_{t'})P(P|L_t)P(L_t)P(L_{t'}), \\
P(A|L_tL_{t'})P(P|L_t)P(L_t)P(L_{t'}|P), & P(A|L_tL_{t'})P(P|L_t)P(L_t)P(L_{t'}|L_t), & P(A|L_tL_{t'})P(P|L_t)P(L_t)P(L_{t'}|PL_t), \\
P(A|L_tL_{t'})P(P|L_t)P(L_t|L_{t'})P(L_{t'}), & P(A|L_tL_{t'})P(P|L_{t'})P(L_t)P(L_{t'}), & P(A|L_tL_{t'})P(P|L_{t'})P(L_t)P(L_{t'}|L_t), \\
P(A|L_tL_{t'})P(P|L_{t'})P(L_t|P)P(L_{t'}), & P(A|L_tL_{t'})P(P|L_{t'})P(L_t|L_{t'})P(L_{t'}), & P(A|L_tL_{t'})P(P|L_{t'})P(L_t|PL_{t'})P(L_{t'}), \\
P(A|L_tL_{t'})P(P|AL_t)P(L_t)P(L_{t'}), & P(A|L_tL_{t'})P(P|AL_t)P(L_t)P(L_{t'}|L_t), & P(A|L_tL_{t'})P(P|AL_t)P(L_t|L_{t'})P(L_{t'}), \\
P(A|L_tL_{t'})P(P|AL_{t'})P(L_t)P(L_{t'}), & P(A|L_tL_{t'})P(P|AL_{t'})P(L_t)P(L_{t'}|L_t), & P(A|L_tL_{t'})P(P|AL_{t'})P(L_t|L_{t'})P(L_{t'}), \\
P(A|L_tL_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'}), & P(A|L_tL_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t), & P(A|L_tL_{t'})P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'}), \\
P(A|L_tL_{t'})P(P|AL_tL_{t'})P(L_t)P(L_{t'}), & P(A|L_tL_{t'})P(P|AL_tL_{t'})P(L_t)P(L_{t'}|L_t), & P(A|L_tL_{t'})P(P|AL_tL_{t'})P(L_t|L_{t'})P(L_{t'}), \\
P(A|PL_tL_{t'})P(P)P(L_t)P(L_{t'}), & P(A|PL_tL_{t'})P(P)P(L_t)P(L_{t'}|P), & P(A|PL_tL_{t'})P(P)P(L_t)P(L_{t'}|L_t), \\
P(A|PL_tL_{t'})P(P)P(L_t)P(L_{t'}|PL_t), & P(A|PL_tL_{t'})P(P)P(L_t|P)P(L_{t'}), & P(A|PL_tL_{t'})P(P)P(L_t|P)P(L_{t'}|P), \\
P(A|PL_tL_{t'})P(P)P(L_t|P)P(L_{t'}|L_t), & P(A|PL_tL_{t'})P(P)P(L_t|P)P(L_{t'}|PL_t), & P(A|PL_tL_{t'})P(P)P(L_t|L_{t'})P(L_{t'}), \\
P(A|PL_tL_{t'})P(P)P(L_t|L_{t'})P(L_{t'}|P), & P(A|PL_tL_{t'})P(P)P(L_t|PL_{t'})P(L_{t'}), & P(A|PL_tL_{t'})P(P)P(L_t|PL_{t'})P(L_{t'}|P), \\
P(A|PL_tL_{t'})P(P|L_t)P(L_t)P(L_{t'}), & P(A|PL_tL_{t'})P(P|L_t)P(L_t)P(L_{t'}|P), & P(A|PL_tL_{t'})P(P|L_t)P(L_t)P(L_{t'}|L_t), \\
P(A|PL_tL_{t'})P(P|L_t)P(L_t)P(L_{t'}|PL_t), & P(A|PL_tL_{t'})P(P|L_t)P(L_t|L_{t'})P(L_{t'}), & P(A|PL_tL_{t'})P(P|L_{t'})P(L_t)P(L_{t'}), \\
P(A|PL_tL_{t'})P(P|L_{t'})P(L_t)P(L_{t'}|L_t), & P(A|PL_tL_{t'})P(P|L_{t'})P(L_t|P)P(L_{t'}), & P(A|PL_tL_{t'})P(P|L_{t'})P(L_t|L_{t'})P(L_{t'}), \\
P(A|PL_tL_{t'})P(P|L_{t'})P(L_t|PL_{t'})P(L_{t'}), & P(A|PL_tL_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'}), & P(A|PL_tL_{t'})P(P|L_tL_{t'})P(L_t)P(L_{t'}|L_t), \\
P(A|PL_tL_{t'})P(P|L_tL_{t'})P(L_t|L_{t'})P(L_{t'}). & &
\end{array}
$$

Annexe C

Expérience : carte d'une pièce basée sur les odeurs

Dans cette expérience, le robot utilise son capteur d'odeur pour discrétiser l'environnement en zones grossières d'intensités odorantes égales. Les comportements pertinents dans ce cadre seront soit de remonter vers une zone de plus forte intensité (*chimiotropisme* positif, qui sera noté « + » par la suite), soit au contraire de s'éloigner de la source d'odeur (chimiotropisme négatif, noté comportement « - »), soit enfin de rester dans une zone d'intensité donnée (« = », patrouille aléatoire dans la zone, ou même plus simplement, arrêt).

C.1 Objectifs et protocole expérimental

C.1.1 Choix d'une variable de lieux

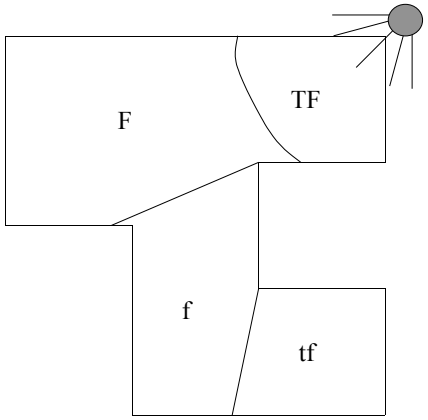


FIG. C.1: Interprétation des zones affectées aux valeurs « TF », « F », « f » et « tf ».

L'abstraction que nous choisissons ici est une discrétisation grossière de l'intensité d'odeur perçue, en quatre valeurs « TF », « F », « f » et « tf » (pour Très Fort, Fort, faible et très faible, respectivement). Nous montrons Figure C.1 une interprétation géométrique de ces lieux. Remarquons qu'il n'y a pas, pour cet environnement particulier dans cet exemple d'endroits différents de l'environnement ayant la même valeur, au contraire de l'exemple de la carte de proximétrie.

C.1.2 Graphe induit

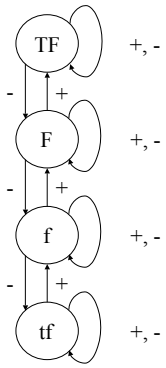


FIG. C.2: Le graphe pour la carte d'une pièce basée sur une odeur.

Nous montrons Figure C.2 le graphe induit par la carte d'une pièce basée sur l'odeur. Nous ne faisons pas apparaître tous les arcs, notamment pas ceux étiquetés par le comportement de patrouille dans une zone, afin de ne pas alourdir la figure.

C.2 Description

1. Spécification des connaissances préalables c^{odeur}
   (a) Variables :
   $\boldsymbol{P}$ α : $\mathcal{D}_\alpha = \{-180, -170, \ldots, 170\}, k_\alpha = 36$, $Odeur$: $\mathcal{D}_{Odeur} = \{0, 1, \ldots, 15\}$, $k_{Odeur} = 16$,
   $\boldsymbol{L_t}$ Od_t : $\mathcal{D}_{Od_t} = \{TF, F, f, tf\}, k_{Od_t} = 4$,
   $\boldsymbol{L_{t'}}$ $Od_{t+\Delta t}$: $\mathcal{D}_{Od_t+\Delta t} = \{TF, F, f, tf\}, k_{Od_{t+\Delta t}} = 4$,
   $\boldsymbol{A}$ A_{odeur} : $\mathcal{D}_{A_{odeur}} = \{+, -, = \ldots\}, k_{A_{odeur}} = 3$.
   (b) Décomposition :

$$\begin{aligned} & P(\alpha\ Odeur\ Od_t\ Od_{t+\Delta t}\ A_{odeur}) \\ & \quad = \ P(Od_t)P(\alpha\ Odeur \mid Od_t)P(A_{odeur})P(Od_{t+\Delta t} \mid Od_t\ A_{odeur}) \end{aligned}$$

(c) Formes paramétriques :

$P(Od_t)$ Il s'agit du résultat de la question de localisation posée à ce même modèle, mais au pas de temps précédent.

$P(A_{odeur})$ À ce terme est associée une loi uniforme.

$P(\alpha\ Odeur \mid Od_t)$ Supposons, dans le cadre de cet exemple, que nous disposons d'un tel modèle capteur direct.

$P(Od_{t+\Delta t} \mid Od_t\ A_{odeur})$ La variable $Od_{t+\Delta t}$ ayant peu de valeurs possibles, seule une loi de succession de Laplace a un sens pour ce terme.

2. Identification : Le seul terme qui reste à définir est $P(Od_{t+\Delta t} \mid Od_t\ A_{odeur})$, que nous pouvons apprendre expérimentalement.

Annexe D

Cartes de gradients

Nous trouvons ici les définitions complètes des cartes probabilistes se rapportant aux gradients rectilignes et circulaires utilisés respectivement Section 8.2.2.3 et Section 8.3.2.3.

D.1 Description du gradient rectiligne c^1

1. Spécification des connaissances préalables c^1
   (a) Variables :
   $\boldsymbol{P}$ P^1
   $\boldsymbol{L_t}$ L_t^1 $: \mathcal{D}_{L_t^1} = \{1, 2, 3\}, k_{L_t^1} = 3$
   $\boldsymbol{L_{t'}}$ $L_{t'}^1$ $: \mathcal{D}_{L_{t'}^1} = \{1, 2, 3\}, k_{L_{t'}^1} = 3$
   $\boldsymbol{A}$ A^1 $: \mathcal{D}_{A^1} = \{remonter^1, descendre^1\}, k_{A^1} = 2$
   (b) Décomposition : $P(P^1\ L_t^1\ L_{t'}^1\ A^1) = P(P^1\ L_t^1\ A^1)P(L_{t'}^1 \mid A^1\ L_t^1)$
   (c) Formes paramétriques :
   $P(P^1\ L_t^1\ A^1)$ À ce terme est associée une loi uniforme.
   $P(L_{t'}^1 \mid A^1\ L_t^1)$ Ce terme est défini par un ensemble de Diracs, traduisant les transitions suivantes : et appliquant $remonter^1$, de la zone 1 on est sûrs d'arriver en zone 1, de la zone 2 on est sûrs d'arriver en 1, de la zone 3 on est sûrs d'arriver en 2 (et inversement pour le comportement $descendre^1$). Nous notons donc :

$$\begin{aligned}
P(L_{t'}^1 \mid [A^1 = remonter^1]\ [L_t^1 = 1]) &= \boldsymbol{\delta}_1(V),\\
P(L_{t'}^1 \mid [A^1 = remonter^1]\ [L_t^1 = 2]) &= \boldsymbol{\delta}_1(V),\\
P(L_{t'}^1 \mid [A^1 = remonter^1]\ [L_t^1 = 3]) &= \boldsymbol{\delta}_2(V),\\
P(L_{t'}^1 \mid [A^1 = descendre^1]\ [L_t^1 = 1]) &= \boldsymbol{\delta}_2(V),\\
P(L_{t'}^1 \mid [A^1 = descendre^1]\ [L_t^1 = 2]) &= \boldsymbol{\delta}_3(V),\\
P(L_{t'}^1 \mid [A^1 = descendre^1]\ [L_t^1 = 3]) &= \boldsymbol{\delta}_3(V).
\end{aligned}$$

2. Identification : Toutes les formes paramétriques sont définies, pas de phase d'identification.

D.2 Description du gradient rectiligne c^2

1. Spécification des connaissances préalables c^2
   (a) Variables :
   $\boldsymbol{P}$ P^2
   $\boldsymbol{L_t}$ L_t^2 $: \mathcal{D}_{L_t^2} = \{A, B, C\}, k_{L_t^2} = 3$
   $\boldsymbol{L_{t'}}$ $L_{t'}^2$ $: \mathcal{D}_{L_{t'}^2} = \{A, B, C\}, k_{L_{t'}^2} = 3$
   $\boldsymbol{A}$ A^2 $: \mathcal{D}_{A^2} = \{remonter^2, descendre^2\}, k_{A^2} = 2$
   (b) Décomposition : $P(P^2\ L_t^2\ L_{t'}^2\ A^2) = P(P^2\ L_t^2\ A^2)P(L_{t'}^2 \mid A^2\ L_t^2)$
   (c) Formes paramétriques :
   $P(P^2\ L_t^2\ A^2)$ À ce terme est associée une loi uniforme.
   $P(L_{t'}^2 \mid A^2\ L_t^2)$ Ce terme est défini par un ensemble de Diracs, traduisant les transitions suivantes : et appliquant $remonter^2$, de la zone A on est sûrs d'arriver en zone A, de la zone B on est sûrs d'arriver en A, de la zone C on est sûrs d'arriver en B (et inversement pour le comportement $descendre^2$). Nous notons donc :

$$\begin{aligned}
P(L_{t'}^2 \mid [A^2 = remonter^2]\ [L_t^2 = A]) &= \boldsymbol{\delta}_A(V),\\
P(L_{t'}^2 \mid [A^2 = remonter^2]\ [L_t^2 = B]) &= \boldsymbol{\delta}_B(V),\\
P(L_{t'}^2 \mid [A^2 = remonter^2]\ [L_t^2 = C]) &= \boldsymbol{\delta}_C(V),\\
P(L_{t'}^2 \mid [A^2 = descendre^2]\ [L_t^2 = A]) &= \boldsymbol{\delta}_B(V),\\
P(L_{t'}^2 \mid [A^2 = descendre^2]\ [L_t^2 = B]) &= \boldsymbol{\delta}_C(V),\\
P(L_{t'}^2 \mid [A^2 = descendre^2]\ [L_t^2 = C]) &= \boldsymbol{\delta}_C(V).
\end{aligned}$$

2. Identification : Toutes les formes paramétriques sont définies, pas de phase d'identification.

D.3 Description du gradient circulaire c^1

La description associée à cette carte est strictement identique à celle du cas rectiligne, Section D.1.

D.4 Description du gradient circulaire c^2

La description associée à cette carte est strictement identique à celle du cas rectiligne, Section D.2.

Annexe E

Expériences : cartes c^{mur}, c^{coin} et c^{libre}

E.1 Exemple 1 : carte du mur

Dans cette expérience, nous plaçons le robot en présence d'un mur, que nous supposons suffisamment long par rapport au robot pour qu'il n'en voit pas les extrémités. Le modèle que nous allons construire de ce mur pourra donc faire référence à un mur « infini ». Les comportements que nous souhaitons réaliser grâce à la carte c^{mur} sont de suivre le mur, sur la gauche ou la droite, s'en écarter, ou atteindre quelques positions particulières par rapport au mur (par exemple, très proche et face au mur, ou très proche et dos au mur, etc.).

E.1.1 Objectifs et protocole expérimental

E.1.1.1 Choix d'une variable de lieux

Nous choisissons de décrire ce mur « infini » par deux variables internes, un angle Θ_t et une distance $Dist_t$ (voir Figure E.1). Bien que ces variables aient une forte connotation « géométrique », nous décidons de nous passer d'une forte résolution sur ces variables, car, d'une part, les proximètres du robot ne nous permettraient pas d'obtenir une grande précision, et d'autre part, nous n'avons pas besoin d'une représentation très fine pour les tâches à accomplir dans cette situation.

Notons qu'il est difficile de représenter graphiquement une projection en deux dimensions des valeurs des variables Θ_t et $Dist_t$ de cette carte : en effet, pour une même valeur de $Dist_t$, les lieux correspondants à différentes valeurs de Θ_t se trouvent au même endroit géométrique (mais correspondent à différentes orientations du robot).

E.1.1.2 Graphe induit

L'espace abstrait que le robot va utiliser est représenté Figure E.2 ; nous ne faisons apparaître que peu d'arcs, encore une fois pour plus de lisibilité. Les labels sont des résumés des actions choisies pour passer d'un état à un autre : par exemple, « avancer » correspond

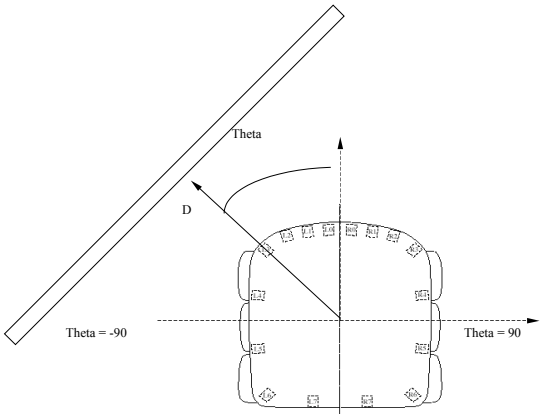


FIG. E.1: Variables retenues pour la description d'un mur.

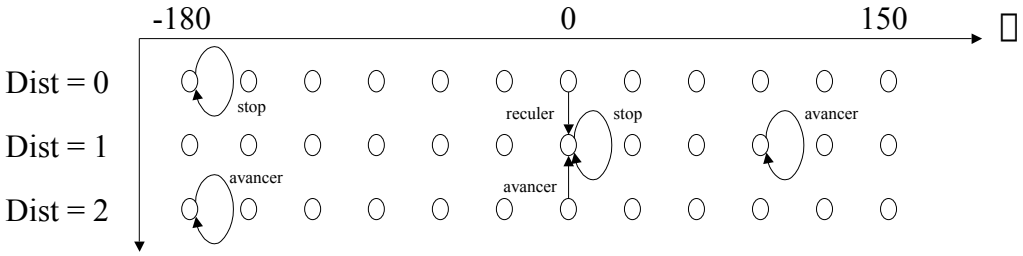


FIG. E.2: Le graphe pour la carte « mur ».

à des gaussiennes sur $Vrot$ et $Vtrans$ centrée sur 0 et 40, respectivement. Les boucles sur les états $\Theta_t = 90$, $Dist_t = 1$ et $\Theta_t = -180$, $Dist_t = 2$, qui portent les labels « avancer », permettent de réaliser les comportements $SuiviD$ et $\acute{E}loignerMur$, respectivement. Le label « stop » sur la boucle de l'état $\Theta_t = -180$, $Dist_t = 0$, permet d'arrêter le robot lorsqu'il résout la tâche $DosAuMur$.

E.1.2 Description

Nous illustrons sur cet exemple la possibilité de définir séparément les modules de localisation et de contrôle du robot. Nous présentons donc tout d'abord ces deux modules, puis la manière de les intégrer dans la carte bayésienne du mur.

E.1.2.1 Carte du mur : module de localisation

1. Spécification des connaissances préalables $CP_{mur-loc}$
   (a) Variables :

   $\boldsymbol{P}\ Px\ : \mathcal{D}_{Px} = \{0, 1, \ldots, 15\}^{16}, k_{Px} = 16^{16}$,

   $\boldsymbol{L_t}$ Nos deux variables pour décrire la position du mur relativement au robot :

- Θ_t : $\mathcal{D}_{\Theta_t} = \{-180, -150, \ldots, 150\}, k_{\Theta_t} = 12$,
- $Dist_t$: $\mathcal{D}_{Dist_t} = \{0, 1, 2\}, k_{Dist_t} = 3$.

(b) Décomposition : nous appliquons un schéma de fusion capteur :

$$P(\Theta_t\ Dist_t\ Px) = P(\Theta_t\ Dist_t) \prod_i P(Px_i \mid \Theta_t\ Dist_t).$$

(c) Formes paramétriques :

$P(\Theta_t\ Dist_t)$ À ce terme est associée une loi uniforme, car nous n'avons pas d'a priori sur la position du mur par rapport au robot.

$P(Px_i \mid \Theta_t\ Dist_t)$ Nous associons à chacun de ces 16 termes une loi gaussienne, dont les moyennes et écart-types sont paramétrés par la valeur des variables Θ_t et $Dist_t$: $\mathbf{G}_{\mu(\Theta_t, Dist_t), \sigma(\Theta_t, Dist_t)}(Px_i)$.

2. Identification : Nous identifions expérimentalement les gaussiennes. En pratique, il s'agit de poser le robot dans l'environnement à un angle et une distance donnée, d'indiquer ces valeurs au robot, et de lui faire enregistrer un ensemble de lectures capteurs (dans notre expérience, nous enregistrons 50 données par capteur). Les triplets collectées sont donc $\langle px_i, \theta, dist$ pour chaque capteur, ce qui permet l'identification des moyennes et écart-types des gaussiennes.

E.1.2.2 Carte du mur : module de contrôle

1. Spécification des connaissances préalables $CP_{mur-cont}$

(a) Variables :

L $_t$ Nous reprenons nos deux variables pour décrire le mur :
- Θ_t : $\mathcal{D}_{\Theta_t} = \{-180, -150, \ldots, 150\}, k_{\Theta_t} = 12$,
- $Dist_t$: $\mathcal{D}_{Dist_t} = \{0, 1, 2\}, k_{Dist_t} = 3$.

L $_{t'}$ Les mêmes à un pas de temps ultérieur :
- $\Theta_{t+\Delta t}$: $\mathcal{D}_{\Theta_{t+\Delta t}} = \{-180, -150, \ldots, 150\}, k_{\Theta_{t+\Delta t}} = 12$,
- $Dist_{t+\Delta t}$: $\mathcal{D}_{Dist_{t+\Delta t}} = \{0, 1, 2\}, k_{Dist_{t+\Delta t}} = 3$.

A Les deux variables de contrôle moteur du robot :
- $Vrot$: $\mathcal{D}_{Vrot} = \{-50, -49, \ldots, 50\}, k_{Vrot} = 101$,
- $Vtrans$: $\mathcal{D}_{Vtrans} = \{-50, -49, \ldots, 50\}, k_{Vtrans} = 101$.

(b) Décomposition : Nous choisissons ici de baser notre décomposition sur le terme de contrôle :

$$\begin{aligned} & P(\Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Vrot\ Vtrans) \\ = \ & P(\Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}) \\ & P(Vrot \mid \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}) \\ & P(Vtrans \mid \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}). \end{aligned}$$

(c) Formes paramétriques :

$P(\Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t})$ À ce terme est associée une loi uniforme, car nous n'avons pas d'a priori ni sur les angles et distances perçues par le robot, ni par les angles et distances consignes que le robot aura.

$P(Vrot \mid \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t})$ Nous associons à ce terme un ensemble de lois gaussiennes, dont les moyennes et écart-types sont paramétrés par les valeurs de Θ_t, $Dist_t$, $\Theta_{t+\Delta t}$ et $Dist_{t+\Delta t}$:

$$\mathbf{G}_{\mu(\Theta_t,Dist_t,\Theta_{t+\Delta t},Dist_{t+\Delta t}),\sigma(\Theta_t,Dist_t,\Theta_{t+\Delta t},Dist_{t+\Delta t})}(Vrot).$$

$P(Vtrans \mid \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t})$ De manière similaire, nous définissons ce terme par des lois gaussiennes, dont les moyennes et écart-types sont paramétrés par les valeurs de Θ_t, $Dist_t$, $\Theta_{t+\Delta t}$ et $Dist_{t+\Delta t}$:

$$\mathbf{G}_{\mu(\Theta_t,Dist_t,\Theta_{t+\Delta t},Dist_{t+\Delta t}),\sigma(\Theta_t,Dist_t,\Theta_{t+\Delta t},Dist_{t+\Delta t})}(Vtrans).$$

2. Identification : Programmation a priori des moyennes et écart-types des gaussiennes, afin de faire réaliser au robot les tâches spécifiées.

E.1.2.3 Carte du mur : intégration

1. Spécification des connaissances préalables c^{mur}

(a) Variables :

P Px : $\mathcal{D}_{Px} = \{0, 1, \ldots, 15\}^{16}, k_{Px} = 16^{16}$,

L_t Nous reprenons nos variables de lieux, communes aux modules de localisation et contrôle :
- Θ_t : $\mathcal{D}_{\Theta_t} = \{-180, -150, \ldots, 150\}, k_{\Theta_t} = 12$,
- $Dist_t$: $\mathcal{D}_{Dist_t} = \{0, 1, 2\}, k_{Dist_t} = 3$.

L_t' Les mêmes variables à un pas de temps ultérieur :
- $\Theta_{t+\Delta t}$: $\mathcal{D}_{\Theta_{t+\Delta t}} = \{-180, -150, \ldots, 150\}, k_{\Theta_{t+\Delta t}} = 12$,
- $Dist_{t+\Delta t}$: $\mathcal{D}_{Dist_{t+\Delta t}} = \{0, 1, 2\}, k_{Dist_{t+\Delta t}} = 3$.

A Les variables de contrôle qui apparaissent dans $CP_{mur-cont}$:
- $Vrot$: $\mathcal{D}_{Vrot} = \{-50, -49, \ldots, 50\}, k_{Vrot} = 101$,
- $Vtrans$: $\mathcal{D}_{Vtrans} = \{-50, -49, \ldots, 50\}, k_{Vtrans} = 101$.

(b) Décomposition :

$$\begin{aligned} &P(Px\ \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}\ Vrot\ Vtrans) \\ &= P(Px)P(\Theta_t\ Dist_t \mid Px)P(\Theta_{t+\Delta t}\ Dist_{t+\Delta t}) \\ &\quad P(Vrot \mid \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t})P(Vtrans \mid \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t}) \end{aligned}$$

(c) Formes paramétriques :

$P(Px)$ À ce terme est associée une loi uniforme.

$P(\Theta_t\ Dist_t \mid Px)$ Nous définissons ce terme comme étant une question probabiliste posée au module de localisation.

$P(\Theta_{t+\Delta t}\ Dist_{t+\Delta t})$ À ce terme est associée une loi uniforme.

$P(Vrot \mid \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t})$ Ce terme est une question probabiliste au module de contrôle.

$P(Vtrans \mid \Theta_t\ Dist_t\ \Theta_{t+\Delta t}\ Dist_{t+\Delta t})$ Ce terme est une question probabiliste au module de contrôle.

2. Identification : Pas de paramètres libres.

Ceci termine la définition de la description de la carte de la situation « mur ». Nous allons maintenant voir comment utiliser cette description dans le cadre d'un programme de contrôle robotique, puis nous discuterons des résultats obtenus expérimentalement.

E.1.3 Utilisation

E.1.3.1 Programme et question probabiliste

Grâce à c^{mur}, nous pouvons générer différents programmes de contrôle robotique, en fixant des valeurs consignes pour les variables $\Theta_{t+\Delta t}$ et $Dist_{t+\Delta t}$. Par exemple, pour obtenir un suivi de mur où le robot garde le mur sur sa droite (comportement $SuiviD$), nous fixons comme consigne $\Theta_{t+\Delta t} = 90$ et $Dist_{t+\Delta t} = 1$, afin que le robot cherche à atteindre cette situation, où l'angle du mur perçu est de 90°, et la distance est modérée. Cela correspond à poser et résoudre la question probabiliste suivante :

$$P(Vrot\ Vtrans \mid [\Theta_{t+\Delta t} = 90]\ [Dist_{t+\Delta t} = 1]\ Px).$$

De manière similaire, nous définissons un comportement $SuiviG$ de suivi du mur sur la gauche (consigne $\Theta_{t+\Delta t} = -90$, $Dist_{t+\Delta t} = 1$), un comportement $ÉloignerMur$ (consigne $\Theta_{t+\Delta t} = -180$, $Dist_{t+\Delta t} = 2$), un comportement $FaceAuMur$ (consigne $\Theta_{t+\Delta t} = 0$, $Dist_{t+\Delta t} = 0$), un comportement $DosAuMur$ (consigne $\Theta_{t+\Delta t} = -180$, $Dist_{t+\Delta t} = 0$), et enfin, un comportement $SuiviGD$ de suivi du mur dans un sens puis dans l'autre, en changeant de sens toutes les 20 secondes (alternance des consignes qui définissent $SuiviD$ et $SuiviG$).

E.1.3.2 Résultats

Nous rapportons ici des résultats expérimentaux très satisfaisants : cette carte contient suffisamment d'informations pour faire atteindre au robot n'importe quel lieu dans l'espace des angles et distances représentés. Nous montrons Figure E.3 une copie d'écran du module de visualisation des variables internes Θ_t, $Dist_t$ (angle et distance au mur à l'instant t), Θ_{obj}, et $Dist_{obj}$ (angle et distance au mur donnés en consigne au terme de contrôle), et Θ_{pred}, $Dist_{pred}$, (angle et distance au mur prédit pour le pas de temps suivant). Enfin, nous pouvons également, en combinant cette carte avec des connaissances simples supplémentaires, générer les comportements $SuiviD$ et $ÉcarterMur$.

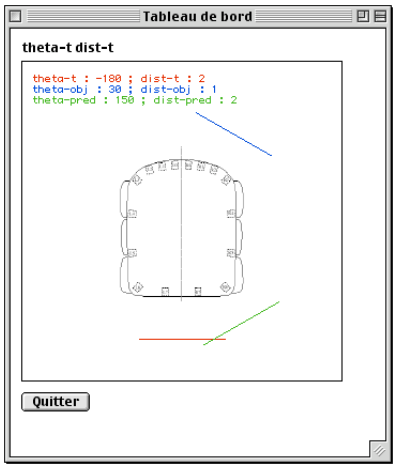


FIG. E.3: Copie d'écran du module de visualisation pour la carte C_{mur}.

E.1.4 Enseignements et discussion

Si nous n'avons pas utilisé de terme de récurrence dans cette carte bayésienne du mur, c'est qu'il n'est pas besoin de mémoriser la localisation du pas de temps précédent pour aider à se localiser à l'instant t. En effet, les données capteurs sont suffisamment riches, et le module de localisation suffisamment précis, pour que le robot se relocalise sans mémoire à chaque instant. Cela est plus simple à mettre en pratique, et est satisfaisant expérimentalement dans ce cas.

E.2 Exemple 2 : carte d'un coin

E.2.1 Objectifs et protocole expérimental

Dans cette expérience, nous souhaitons donner au robot la capacité de reconnaître et de naviguer en présence d'un coin, c'est-à-dire deux murs s'intersectant avec un angle proche de 90°.

E.2.1.1 Choix d'une variable de lieux

Nous choisissons ici une représentation très peu fine de ce type d'environnement, en n'utilisant qu'une variable symbolique, $Coin_t$, qui représentera quatre situations caractéristiques : AvG lorsque le coin est devant et à gauche du robot, AvD lorsque le coin est devant et à droite du robot, ArG lorsque le coin est derrière et à gauche du robot, et ArD lorsque le coin est derrière et à droite du robot. Ces différentes situations sont présentées Figure E.4. Rappelons ici que dans notre premier exemple de carte bayésienne, Section 6.3, nous avions défini par programme déterministe un détecteur pour chacune de ses situations. Ces détecteurs ont été repris pour définir a priori les termes $P(Px_i \mid Coin_t)$ qui

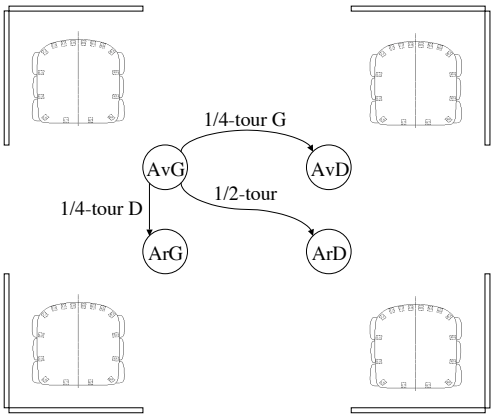


FIG. E.4: La variable retenue pour la description d'un coin, ses valeurs, et un extrait du graphe induit.

apparaissent dans la carte C_{coin}. Les inclure ainsi dans le cadre de la carte bayésienne a l'avantage de pouvoir ensuite lier ces représentations au actions (ici $Vrot$ et $Vtrans$).

E.2.1.2 Graphe induit

Nous montrons Figure E.4 un extrait du graphe induit par la carte bayésienne de la situation « coin » que nous allons définir.

E.2.2 Description

1. Spécification des connaissances préalables c^{coin}
   (a) Variables :
   $\boldsymbol{P}$ Les capteurs habituels $Px \ : \mathcal{D}_{Px} = \{0, 1, \ldots, 15\}^{16}, k_{Px} = 16^{16}$,
   $\boldsymbol{L_t}$ $Coin_t \ : \mathcal{D}_{Coin_t} = \{AvG, AvD, ArG, ArD\}, k_{Coin_t} = 4.$
   $\boldsymbol{L_{t'}}$ $Coin_{t+\Delta t} \ : \mathcal{D}_{Coin_{t+\Delta t}} = \{AvG, AvD, ArG, ArD\}, k_{Coin_{t+\Delta t}} = 4.$
   $\boldsymbol{A}$ Les variables d'action habituelles $Vrot \ : \mathcal{D}_{Vrot} = \{-50, -49, \ldots, 50\}, k_{Vrot} = 101$, et $Vtrans \ : \mathcal{D}_{Vtrans} = \{-50, -49, \ldots, 50\}, k_{Vtrans} = 101.$
   (b) Décomposition :

$$\begin{aligned} & P(Px\ Coin_t\ Coin_{t+\Delta t}\ Vrot\ Vtrans) \\ = \ & P(Coin_t)\left(\prod_i P(Px_i \mid Coin_t)\right) P(Coin_{t+\Delta t} \mid Coin_t) \\ & P(Vrot \mid Coin_t\ Coin_{t+\Delta t}) P(Vtrans \mid Coin_t\ Coin_{t+\Delta t}). \end{aligned}$$

   (c) Formes paramétriques :

$P(Coin_t)$ À ce terme est associée une loi uniforme.

$P(Px_i \mid Coin_t$ À chacun de ces termes est associé un ensemble de lois gaussiennes, dont les moyennes et écart-types sont paramétrés par les valeurs de $Coin_t$: $\mathbf{G}_{\mu(Coin_t),\sigma(Coin_t)}(Px_i)$.

$P(Coin_{t+\Delta t} \mid Coin_t)$

$P(Vrot \mid Coin_t\ Coin_{t+\Delta t})$ $\mathbf{G}_{\mu(Coin_t,Coin_{t+\Delta t}),\sigma(Coin_t,Coin_{t+\Delta t})}(Vrot)$.

$P(Vtrans \mid Coin_t\ Coin_{t+\Delta t}))$ $\mathbf{G}_{\mu(Coin_t,Coin_{t+\Delta t}),\sigma(Coin_t,Coin_{t+\Delta t})}(Vtrans)$.

2. Identification : Programmation a priori.

E.2.3 Utilisation

E.2.3.1 Programme et question probabiliste

La question probabiliste est $P(Vrot\ Vtrans \mid Coin_{t+\Delta t}\ Px)$.

La carte c^{coin} permet d'obtenir, par les comportements élémentaires qui ne consistent qu'en une consigne à atteindre pour la variable $Coin_t$ (il y en a 4 possibles), les comportements suivant :

- nous obtenons un comportement qui permet de « passer » le coin et longer le mur gauche en posant la consigne $[Coin_{t+\Delta t} = ArG]$ au robot (comportement $PasserG$) ;
- de manière similaire nous obtenons un comportement $PasserD$:
- enfin, nous obtenons deux comportements où le robot s'arrête dans le coin, en posant les consignes $[Coin_{t+\Delta t} = AvG]$ et $[Coin_{t+\Delta t} = AvD]$.

E.3 Exemple 3 : carte de l'espace libre

E.3.1 Objectifs et protocole expérimental

E.3.1.1 Choix d'une variable de lieux

Dans cette carte, il s'agit d'identifier les cas où le robot est en espace libre, c'est-à-dire où aucun de ses capteurs de proximétrie n'est excité. Malheureusement, ce phénomène est tellement simple qu'il ne varie jamais : nous ne pouvons pas *différencier* plusieurs aspects de ce phénomène « espace libre » (alors que c'est effectivement le rôle des variables internes des cartes bayésienne en général : les variables Θ_t et $Dist_t$ permettent de caractériser différentes facettes du même phénomène « mur »). Nous choisissons donc ici de représenter l'espace libre, par opposition à l'espace encombré en général. Nous nous donnons donc une variable $Espace_t$, qui prend les valeurs $\{vide, encombré\}$.

Dans cette carte, nous résolvons les tâches $Stop$ et $ToutDroit$, qui consistent respectivement à s'arrêter et aller tout droit.

E.3.1.2 Graphe induit

E.3.2 Description

1. Spécification des connaissances préalables c^{libre}
   (a) Variables :
   $\boldsymbol{P}$ Px : $\mathcal{D}_{Px} = \{0, 1, \ldots, 15\}^{16}, k_{Px} = 16^{16}$,
   $\boldsymbol{L_t}$ $Espace_t$: $\mathcal{D}_{Espace_t} = \{vide, encombré\}, k_{Espace_t} = 2$,
   $\boldsymbol{L_{t'}}$ $Espace_{t+\Delta t}$: $\mathcal{D}_{Espace_{t+\Delta t}} = \{vide, encombré\}, k_{Espace_{t+\Delta t}} = 2$,
   $\boldsymbol{A}$ Nos deux variables habituelles de contrôle :
   – $Vrot$: $\mathcal{D}_{Vrot} = \{-50, -49, \ldots, 50\}, k_{Vrot} = 101$.
   – $Vtrans$: $\mathcal{D}_{Vtrans} = \{-50, -49, \ldots, 50\}, k_{Vtrans} = 101$.
   (b) Décomposition :

$$\begin{aligned} & P(Px\ Espace_t\ Espace_{t+\Delta t}\ Vrot\ Vtrans) \\ = \ & P(Espace_t) \left(\prod_i P(Px_i \mid Espace_t) \right) P(Espace_{t+\Delta t}) \\ & P(Vrot \mid Espace_t\ Espace_{t+\Delta t}) P(Vtrans \mid Espace_t\ Espace_{t+\Delta t}). \end{aligned}$$

   (c) Formes paramétriques :
   $P(Espace_t)$ À ce terme est associé une loi uniforme.
   $P(Px_i \mid Espace_t)$ Pour les cas où $[Espace_t = vide]$, ces 16 termes sont des Diracs qui valent 1 lorsque le proximètre rend 0 : $P(Px_i \mid [Espace_t = vide]) = \boldsymbol{\delta}_0(Px_i)$. En revanche, nous ne souhaitons pas contraindre la situation $[Espace_t = encombré]$, donc nous définissons les termes correspondant par des uniformes : $P(Px_i \mid [Espace_t = encombré]) = \mathbf{U}_{k_{Px_i}}(Px_i)$.
   $P(Espace_{t+\Delta t})$ À ce terme est associé une loi uniforme.
   $P(Vrot \mid Espace_t\ Espace_{t+\Delta t})$ Ce terme est soit une loi uniforme, soit une loi gaussienne, selon la valeur de la partie droite : lorsque $[Espace_t = vide]$ et $[Espace_{t+\Delta t} = encombré]$, c'est une loi gaussienne (de moyenne 0 et d'écart-type 2), sinon c'est une uniforme.
   $P(Vtrans \mid Espace_t\ Espace_{t+\Delta t})$ Ce terme est soit une loi uniforme, soit une loi gaussienne, selon la valeur de la partie droite : lorsque $[Espace_t = vide]$ et $[Espace_{t+\Delta t} = encombré]$, c'est une loi gaussienne (de moyenne 40 et d'écart-type 2), lorsque $[Espace_t = vide]$ et $[Espace_{t+\Delta t} = vide]$, c'est une loi gaussienne (de moyenne 0 et d'écart-type 2), sinon c'est une loi uniformes.
2. Identification : Nous avons inclus les définitions des paramètres libres dans la phase de spécification, pour plus de lisibilité.

E.3.3 Utilisation

Nous réalisons les comportements élémentaires *ToutDroit* et *Stop* en posant les questions probabilistes

$$P(Vrot\ Vtrans \mid [Espace_{t+\Delta t} = encombré]\ Px),$$

et

$$P(Vrot\ Vtrans \mid [Espace_{t+\Delta t} = vide]\ Px),$$

respectivement.