Utilisation de treillis de Galois pour structurer la généralisation des politiques pour l'Apprentissage par Renforcement

Marc Ricordeau

UNIVERSITÉ MONTPELLIER II
SCIENCES ET TECHNIQUES DU LANGUEDOC

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ MONTPELLIER II

Discipline : Informatique

Ecole Doctorale : Information Structures Systèmes

présentée et soutenue publiquement par

Marc Ricordeau

le 16 juillet 2009

Titre

Utilisation de treillis de Galois pour structurer la généralisation des politiques pour l'Apprentissage par Renforcement

Jury

M Jean Sallantin (Directeur de Thèse)
M Michel Liquière (co-directeur de Thèse)
M Engelbert Mephu Nguifo (Raporteur)
M Bruno Apolloni (Examinateur)
M^{me} Marianne Huchard (Examinatrice)
M^{me} Celine Rouveirol (Examinatrice)

Remerciements

Presque huit ans se sont écoulés depuis la décision de commencer ma thèse et la conclusion de celle-ci. Aujourd'hui que je suis (enfin) arrivé enfin au bout du chemin - et du bon – ce n'est pas sans émotion et soulagement que je mets un point final à ce doctorat. Avant de passer à une autre étape, il me reste néanmoins beaucoup de personnes à remercier.

S'il y a de nombreux remerciements à faire, évidemment, mes tous premiers, pour l'aboutissement de ce travail, vont à Michel Liquière, mon directeur de thèse. Pendant toutes ces années, il m'a aidé, soutenu, conseillé, enrichi scientifiquement, et surtout fait confiance et traité en égal. Je ne ferai pas la liste de tous les moments où il m'a apporté, je résumerai cela par le fait qu'il a fait beaucoup plus que ce qu'on attend d'un directeur de thèse et qu'il est devenu, de fait, beaucoup plus qu'un directeur de thèse.
Je remercie également Jean Sallantin, non seulement pour avoir accepté d'être mon directeur pour cette soutenance, mais pour les appuis réguliers qu'il a mis en place le long de ces années. Il me faut également citer de riches et longues conversations que peu de gens peuvent offrir.
Même si au final, la collaboration n'a pas été au bout, je remercie également Jacques Ferber pour m'avoir permis de me lancer dans cette aventure en tant que premier directeur de thèse.

Bien sûr, il me faut aussi remercier les membres du jury qui se sont penchés sur mon travail et me permettent de soutenir ma thèse.
François Charpillet, directeur de recherche à l'équipe MAIA du Loria, pour le temps passé ainsi que pour les précieux conseils qu'il m'a donné il y a quelques années et qui m'ont permis à l'époque des avancées importantes dans mes travaux. Je suis d'ailleurs honoré qu'il soit un de mes rapporteurs, notamment compte tenu de son expertise sur l'apprentissage par renforcement ; aspects de ma thèse sur lequel je n'ai eu que peu de retours experts directs en dehors du sien.
Je remercie également Engelbert Mephu Nguifo, de l'université de Clermont Ferrand, non seulement pour avoir accepté d'être également mon rapporteur, mais aussi pour les discussions lors des congrès dans lesquels nous nous sommes croisés. Je voudrais adresser un remerciement spécial à Bruno Apolloni, professeur à l'université de Milan qui a accepté d'être membre de mon jury, malgré la langue et la distance. De plus, il a été un des vecteurs d'une publication importante de mes travaux.
Un merci également à Marianne Huchard de l'université Montpellier 2 pour avoir accepté d'être membre de mon jury, mais également pour être la personne à travers qui, en tant qu'enseignante, j'ai découvert les treillis de Galois.
Enfin, merci à Céline Rouveirol de l'université Paris nord, pour avoir accepté de faire partie du jury et donc de se pencher sur mes travaux.

Pour les remerciements dans le cadre académique, je voulais également citer l'équipe KAYOU du LIRMM en général et Frédéric Koriche pour avoir proposé de faire partie de mon jury d'une part ainsi que pour ses conseils et discussions enrichissantes d'autre part.
Même si tout cela commence à être loin dans le temps, ma thèse a eu ses débuts qui ont été facilités par les conseils amicaux d'Olivier Simonin. Comme ce n'est pas parce que le temps a passé que ce n'était pas important à l'époque, je l'en remercie également.

Pour qu'un travail d'écriture ait un point final, il faut parfois forcer la main de l'écrivain en lui donnant des perspectives. Eric Jallas d'ITK a en partie joué ce rôle, c'est ce dont je l'en remercie.

La thèse comprend une grosse composante solitaire, tout un travail qu'on est seul a pouvoir accomplir et faire avancer. Cependant, celui-ci ne peut être réalisé sans un important soutien de l'entourage, surtout quand cela dure de longues années, avec des moments de difficultés et de doutes.
En premier lieu, je dois remercier mes parents ainsi que mon frère Olivier pour m'avoir non seulement soutenu matériellement et moralement, mais également pour avoir créé les conditions

nécessaires me permettant d'être parfaitement libre de mes choix.

Le doctorat ne touche pas que son prétendant, dans le sens où il y procède à une véritable et inévitable invasion de la vie privée. Ainsi, celles et ceux qui a un titre ou a un autre ont partagé des bouts de ma vie ont été de fait impliqués dans ce parcours. C'est pour cela que je remercie tout particulièrement ma compagne Souria non seulement pour son aide et son précieux soutien, mais également pour avoir supporté une bonne partie du fardeau.
Il me faut également citer le trio Nadia, Justine et Léa qui ont aussi pris leur part.

Si ce qui fait la richesse, c'est la diversité, je peux dire qu'en amitié je suis riche. Bien sûr, un bout de ce travail est à dédier au soutien sans faille, à tous les niveaux de Gilles et Nanou. La liste des amis et camarades qui ont joué un rôle soit par leur aide soit par leurs points de vue sur mes décisions, soit simplement pour avoir su trouver la bonne distance est longue, mais je voudrais citer nommément Stéphane, Cécile, Sylvie, JC et Édouard.
Je n'oublie pas également tous les coups de main logistiques catalans qui m'ont enlevé une épine du pied : Mathieu, Edward et Martine, Sam et Mimi.

De manière générale, je remercie toutes celles et ceux qui ont permis à un moment où à un autre que cette aventure se termine de la bonne façon.

Table des matières

1 Introduction

L'introduction est divisée en deux parties. Premièrement, partie 1.1, nous replacerons nos travaux à partir d'éléments globaux autour de la conception *agent* et de l'apprentissage artificiel (ou apprentissage machine). Deuxièmement, partie 1.2, nous donnerons une trame synthétique de nos travaux.

1.1 Concepts autour de l'apprentissage machine

Plutôt que de définir formellement l'ensemble des concepts utilisés en Apprentissage Machine, nous allons les présenter au travers d'un exemple. Évidemment, les concepts que nous utiliserons spécifiquement dans nos travaux seront, eux, définis formellement. Le lecteur intéressé par les fondements de l'apprentissage machine pourra se référer aux chapitres introductifs des ouvrages suivants : [Russel and Norvig, 2003] pour une présentation assez complète et historique, [Cornuéjols and Miclet, 2002] pour une introduction en français et [Mitchell, 1997] pour une introduction plus courte.
Depuis peu de temps, une technique issue de l'apprentissage machine s'est répandue rapidement jusqu'à devenir essentielle pour de nombreux utilisateurs ; il s'agit des filtres permettant de distinguer les courriers électroniques dits indésirables des courriers électroniques classiques.
Précisons ce que font ces programmes : ils servent de filtres pour le courrier électronique, ils séparent pour un utilisateur donné, les mails qu'il reçoit en les répartissant en deux catégories, les mails « normaux » et les mails n'étant pas sollicités et étant généralement issus de démarches de prospection agressives, généralement commerciales. Premièrement, il n'existe pas de définition claire et univoque de ce que sont les courriers indésirables. Deuxièmement, les caractéristiques des mails dits indésirables peuvent variées d'un individu à l'autre (imaginons par exemple que les caractéristiques de filtrage pour un sexologue de profession doivent être beaucoup plus fines que pour d'autres utilisateurs). De plus, pour un même individu, la définition d'un courrier électronique indésirable peut varier au cours du temps. Il n'est donc pas question compte tenu de ces caractéristiques d'envisager des règles de filtrages figées et programmées une bonne fois pour toute.
En fait, c'est l'utilisateur qui va **entraîner** le programme de filtre à l'aide d'**exemples** de ce que sont pour lui des courriers électroniques désirables et indésirables, qui serviront, eux, de **contre-exemples**. L'ensemble des exemples et contre-exemples constitue un **échantillon**. A partir de ces données, le programme essaiera de se faire une représentation de ce que sont pour l'utilisateur ces deux catégories. On dit que le programme **modélise** les caractéristiques des courriers désirables et indésirables. A partir de ces données spécifiques, le programme effectuera une **généralisation** ou une **induction** sur les caractéristiques particulières à chaque courrier, afin de trouver celles qui sont communes à chacune des deux **classes**.

La façon dont l'utilisateur classerait ses courriers et que la machine tente de reproduire, constitue l'**hypothèse cible**. L'utilisateur serait dans bien des cas incapable de formuler explicitement cette hypothèse cible ! La machine, elle, doit pourtant le faire. Elle le fait en utilisant une **représentation interne** de cette hypothèse. Cette représentation interne ne peut pas prendre n'importe quelle forme. L'ensemble des représentations dont la machine dispose pour décrire l'hypothèse cible se nomme l'**espace des hypothèses**. On le voit, le programme de classement n'apprend pas « en soit » pour apprendre, ici, il s'agit d'apprendre à différencier deux catégories de courriers. Il apprend donc dans un objectif. Celui-ci constitue la **tâche d'apprentissage**.

Ces programmes de filtre utilisent généralement des contraintes sur des probabilités d'apparitions de mots entre autre, appelées réseaux bayesiens. Quand il s'agit de modéliser un problème en utilisant, comme ici, des calculs sur des valeurs numériques, on parlera d'**apprentissage numérique**. Si au contraire, il s'agit de manipuler, d'induire ou de déduire sur des structures (comme des graphes par exemple), on parlera plutôt d'**apprentissage symbolique**.[1]

[1] La frontière entre ces deux formes d'apprentissage n'est pas aussi claire. Notamment les travaux proposés dans cette thèse utilisent un apprentissage basé sur les deux types de méthode

Se pose la question de savoir si le programme est ici capable seulement de produire un **comportement**, en classant bien ou mal, ou s'il est aussi capable de communiquer son hypothèse de façon compréhensible par un humain et éventuellement dans le but que celui-ci puisse directement intervenir dessus. On parle alors de l'**intelligibilité** du résultat. Enfin, si l'utilisateur ne sait pas lui-même quelles sont les raisons qui le font classer ici plutôt que là, et que la machine, par les exemples successifs permet de donner les éléments qui engendrent ce classement, on parlera d'**extraction de connaissances**.

Il se peut, et c'est fréquent, que l'hypothèse cible ne puissent pas être représentée dans l'espace des hypothèses. Par exemple, les raisons poussant un utilisateur à rejeter ou non un courrier peuvent-elles être représentées sous la forme de probabilités sur des apparitions de termes ? Nous reviendrons ultérieurement sur l'écart qu'il peut y avoir dû à l'impossibilité de représenter l'hypothèse cible dans un certain espace de représentation, mais disons ici que cette erreur se nomme **erreur d'approximation**.

Au fur et à mesure que les mails sont classés, il y a donc **transformation** de l'hypothèse. Nous le savons pour les êtres humains, le langage utilisé pour représenter un fait et les méthodes de raisonnement utilisées pour modifier l'appréhension de ces faits ne sont pas neutres et sont mêmes fortement corrélés. Il en va de même pour les méthodes d'apprentissage machine. La façon particulière de décrire les faits (espace des hypothèses) associée aux méthodes permettant de passer d'une hypothèse à une autre représente un **biais**.
Le système peut commettre des erreurs en classant ; s'il classe un courrier désirable alors que l'utilisateur l'aurait classé indésirable et considérant que l'on souhaite qu'il modélise ce qu'est un courrier indésirable, il s'agit d'un **faux-positif**. Inversement, si un courrier n'est pas classé indésirable alors que l'utilisateur aurait fait le contraire, il s'agit d'un **faux-négatif**.
Si le programme ne classe pas correctement, qu'il commet des erreurs dûes à l'échantillon fourni (celui-ci ne pouvant pas représenter l'ensemble des mails possibles), cette erreur est appelée **erreur d'estimation**, ou **variance**.
Enfin, l'utilisateur peut se tromper en classant des mails pour de multiples raisons, classant un courrier indésirable alors que pour lui, il ne l'était pas en se trompant de bouton, ou en étant distrait,... De telles erreurs sur l'échantillon constituent l'**erreur intrinsèque**, ou **bruit**. Trois types d'erreurs, intrinsèque, d'estimation, d'approximation ont été recensées. La somme de ces erreurs constitue l'**erreur totale**.

Le rapport entre le nombre de courriers à classer et les courriers bien classés peuvent déterminer un **critère de réussite**, permettant de mesurer la qualité de l'apprentissage. Nous avons donc dit que le programme classe les mails dans un sens ou dans un autre ; il effectue des actions (ici deux actions possibles). De plus, le programme **interagit** avec un utilisateur, qui fournit des exemples et à qui en retour il pré-classe son courrier, et le tout, très certainement à l'intérieur d'un autre logiciel. Toutes ces composantes, (humain, autres programmes) constituent son **environnement**. Même si l'**autonomie** de ce programme est faible, on pourrait imaginer que celle-ci soit supérieure en allant chercher sur internet des informations, en communiquant avec d'autres programmes,... Il pourrait aussi par exemple, en utilisant le taux de réussite et sa représentation interne, savoir quel type de mail lui pose problème. Il pourrait donc par exemple créer un mail dont le classement par l'utilisateur serait très utile pour améliorer sa représentation. La capacité de créer une instance à partir d'un modèle constitue une **abduction**. De plus, le fait d'essayer de trouver des exemples permettant de s'améliorer dans sa tâche constitue l'**exploration**.

Une telle entité artificielle, pouvant agir sur son environnement, ayant une représentation propre de celui-ci, bénéficiant d'une certaine autonomie et pouvant éventuellement communiquer avec d'autres entités, artificielles ou non, est appelée **agent**. Cet agent modifiant sa représentation interne ainsi que ses actions au fur et à mesure de ses expériences et dans le but de s'améliorer dans l'accomplissement d'une tâche est appelé **agent apprenant**.

Nous avons vu les caractéristiques principales des moyens utilisés pour aborder une tâche d'apprentissage. Voyons maintenant des concepts relatifs à l'environnement ou à la tâche d'apprentissage elle même. On peut premièrement considérer que l'utilisateur, si on lui présente toujours le même courrier, aura toujours la même façon de le classer. L'environnement de l'agent dans ce

type de cadre est alors dit **déterministe**. Si on contraire, l'utilisateur classe en respectant des distributions probabilistes (par exemple, un même mail sera classé 70% de fois en désirable, 30% en indésirable), l'environnement sera appelé **stochastique**. Poursuivons, si l'utilisateur change ses probabilités de classement au cours du temps, l'environnement sera **dynamique**. Si au contraire, celui-ci conserve toujours son comportement, il sera **statique**. Attention, toutes les combinaisons sont permises un utilisateur peut par exemple d'abord toujours classer le même mail en désirable puis toujours en indésirable. Il sera alors et déterministe et dynamique.

Questions classiques de l'apprentissage machine

Nous avons donc vu, à l'aide d'un exemple, les concepts les plus courants utilisés en apprentissage machine. Nous allons maintenant rappeler trois paradigmes relatifs à l'apprentissage machine et dans lequel nous inscrivons en partie ces travaux.

1. Il s'agit premièrement d'un problème que nous avons déjà évoqué : celui du rapport entre biais et variance traitant de la richesse de l'espace des hypothèses et confronté au fait que l'on ne dispose pas de l'ensemble des exemples. Il faut donc que les hypothèses puissent être valables sur les exemples non-vus. C'est la question du **compromis biais-variance**.
2. Deuxièmement, Nous évoquons un problème aux sources de l'apprentissage machine, développé par un de ses créateurs, Minsky ([Minsky, 1961]) et connu sous le nom de **Temporal Difference Learning** (que l'on pourrait traduire par Différenciation Temporelle de l'Apprentissage). Le problème est le suivant : les effets, bénéfiques ou problématiques dans un environnement donné, ne sont généralement pas les conséquences de l'action qui vient d'être effectuée, ni même les conséquences d'une seule action. La question, est donc de trouver, pour une conséquence, quelles sont les actions ayant concourues à un résultat, et selon quelle proportion pour chacune d'elles. Plus formellement, on devrait même écrire selon quelle fonction, puisqu'il dans le cas général, le rôle des actions ne suit pas une fonction paramétrique.
3. Enfin, évoquons une autre hypothèse qui est celle du **rasoir d'Ockham** (voir une analyse de sa pensée dans [Biard, 1997]), hypothèse attribuée au philosophe Anglais du XIVème siècle William of Ockham. Celle-ci consiste à dire que parmi l'ensemble des hypothèses consistantes avec un ensemble de faits, il faut choisir la plus simple. Comme il est rappelé dans [Russel and Norvig, 2003], intuitivement, il est nécessaire que les hypothèses généralisant des faits soient plus simples que les faits, sinon, comment extraire des connaissances de ceux-ci.

Apprentissage par renforcement, conception agent et apprentissage machine

Nous décrirons précisément les propriétés, formalismes et algorithmes dans la partie suivante. Contentons nous ici de rapprocher les concepts liés à l'apprentissage par renforcement relativement aux notions évoquées à propos de l'apprentissage machine. Nous évoquerons également les éléments qui sont spécifiques à cette forme particulière d'apprentissage.

L'apprentissage par renforcement se propose d'obtenir un certain comportement d'un agent, en lui donnant seulement comme indications des valeurs numériques lui indiquant la qualité de l'état dans lequel il se trouve. Bien sûr, l'agent dispose également de descriptions, lui permettant d'obtenir des informations sur son environnement. Cette forme d'apprentissage machine a bien sûr une partie de ses origines en biologie. On fera l'analogie avec l'apprentissage par essais-erreurs que l'on trouve dans le monde animal. Les récompenses négatives pouvant être vues comme des punitions ou des signaux négatifs telle que la douleur, les valeurs positives pouvant elle être reliées à des récompenses ou à des signaux positifs tel que le plaisir. L'objectif de l'agent apprenant, comme celui de l'animal peut être vu comme l'accumulation de valeurs positives (la recherche des états de plaisir ou des récompenses) et d'éviter les valeurs négatives (l'évitement de situations pénibles, des punitions). Si cette analogie est pratique pour expliquer l'inspiration de l'apprentissage par renforcement, comparaison n'étant pas raison, il faut assez vite s'en éloigner. On peut dire ici que l'abus de l'analogie peut conduire aux mêmes écueils que de garder comme ligne de mire l'évolution des espèces pour les algorithmes génétiques ou que d'essayer de limiter l'aéronautique à l'imitation des oiseaux. En effet, les formalismes mathématiques utilisés, la structures des espaces de recherche et même les contraintes spécifiques dûes à la nécessaire numérisation en informatique entraînent

des comportements bien différents de ceux observables dans la nature. Et surtout, comme nous le verrons dans ces travaux, au delà d'obtenir un comportement souhaitable pour un agent, celui-ci va pouvoir extraire des connaissances relatives à son apprentissage.

L'apprentissage par renforcement est une forme d'apprentissage artificiel. La saisie par le monde de la recherche en apprentissage artificiel de ce domaine, auparavant étudié dans un cadre mathématique est relativement récente (voir par exemple l'historique proposé dans [Sutton and Barto, 1998]). De ce fait, les connexions avec les autres domaines de l'apprentissage machine de sont pas très explorées. Il faut quand même relever l'exception notable de [Russel and Norvig, 2003] dans lequel, contrairement à beaucoup d'autres ouvrages généralistes d'apprentissage artificiel, l'apprentissage par renforcement n'est pas traité « à part », c'est à dire dans un chapitre contenant peu de ponts en terme de notations ou de concepts avec les autre chapitres. Ce n'est pas l'objet ici d'étudier systématiquement ces relations. Nous allons cependant en donner quelques éléments sur le plongement de l'apprentissage par renforcement dans le cadre de l'apprentissage artificiel et des conceptions agent.

Tout d'abord, on peut noter qu'avec la popularité croissante de la vision agent de la recherche informatique à l'heure actuelle, cette forme d'apprentissage a connu un intérêt croissant. En effet, les notions d'agent, se fondent complètement dans l'apprentissage par renforcement.
Premièrement, on dispose bien d'une entité ayant une représentation interne et interagissant avec son environnement. On a donc bien cette notion d'intérieur et d'environnement séparés. L'agent agit sur son environnement au travers d'actions et perçoit celui-ci au travers de deux perceptions, l'une qualitative (une valeur numérique) et l'autre descriptive (la perception de l'état du monde). Deuxièmement, la tâche d'apprentissage n'est pas nécessairement explicitée pour l'agent. On ne dit pas à l'agent de remplir telle tâche, on ne le programme pas non plus dans cette optique. C'est lui qui va adapter son comportement, ce qui donne à ce type d'algorithme, à l'horizon, de concevoir des interactions avec ce type d'agent, sans avoir à connaître quoi que ce soit de l'apprentissage machine.

La notion d'apprentissage de la différence temporelle donne également des arguments en faveurs de la vision agent de l'apprentissage par renforcement. En effet, beaucoup d'algorithmes d'apprentissage n'intègrent pas cette notion, ceci implique qu'il faut de façon externe, c'est à dire sans que cela ne soit intégré aux algorithmes, prendre en compte cet élément. L'apprentissage en ligne, c'est à dire la non séparation entre une phase d'apprentissage proprement dite et une phase d'utilisation de l'apprentissage d'autre part, intègre directement le rapport entre l'acquisition de la connaissance l'utilisation de celle-ci dans la théorie. Là encore, pour accroître l'autonomie de l'agent, il n'y a pas à faire de réglages extérieurs aux algorithmes.

Finalement, cette autonomie théorique pose les bases d'une utilisation d'agent de ce type dans le cadre de systèmes multi-agents. Dans ce cadre, des travaux émergent (voir notamment les travaux de l'équipe *Maia* au Loria de Nancy : [Dutech et al., 2001] et [Dutech et al., 2003]).
Cependant, l'intégration théorique de l'ensemble de ces problèmes (apprentissage de la différence temporelle, apprentissage en ligne) ne les résout pas pour autant. Les preuves de convergence des algorithmes d'apprentissage par renforcement se font avec des conditions très limitatives, surtout du point de vue multi-agents. Par exemple, la base de la preuve de convergence d'algorithmes classiques comme le *Q-Learning* se fait sur une omniscience de l'agent, ce qui est évidemment loin d'une conception multi-agents. On peut aussi citer le fait que l'environnement est supposé statique au sens ou son comportement suit toujours la même probabilité de répondre à une même action. Or, par définition, si on a deux agents apprenants (chacun constitue une partie de l'environnement pour l'autre), ceux-ci vont modifier leurs comportements. Les utilisations des méthodes d'apprentissage par renforcement se heurtent donc à l'ensemble de ces problèmes théoriques et à un fossé entre les restrictions apportées si l'on veut rester dans un cadre mathématique formel et convergent et l'utilisation théorique potentielle qui pourrait en être faite.

Nous avons vu que le cadre de l'apprentissage par renforcement s'insérait correctement dans le cadre de la conception agent. Évidemment, étant une forme d'apprentissage machine, il s'intègre également dans ce cadre-ci. Mais comme nous allons le voir les ponts sont pour l'instant assez ténus quant à la comparaison avec les autres algorithmes.

Existent bien sûr dans l'apprentissage par renforcement des notions communes avec les autres formes d'apprentissage. On identifie une tâche d'apprentissage, une hypothèse cible (le comportement désiré par l'agent), une hypothèse courante (le comportement courant de l'agent), un espace d'hypothèses (l'ensemble des comportements possibles). On a bien une transformation, puisque le comportement est modifié et un critère de réussite, qui est constitué par le rapport entre les récompenses maximalement potentiellement obtenues et les récompenses effectivement obtenues. Les trois formes d'erreurs évoquées sont également identifiées :

- l'erreur de biais, qui découle d'une distance entre les représentations possibles (un comportement stochastique) et l'hypothèse cible (un comportement optimal)
- l'erreur de variance (la façon dont les exemples sont fournis, ici, la façon dont l'agent explore son environnement)
- les erreurs intrinsèques (nous le verrons ultérieurement, mais l'apprentissage par renforcement peut tout à fait s'accommoder de bruits)

Cependant, pour certaines des notions, des précisions sont nécessaires. En effet, il convient avant tout de constater que l'apprentissage par renforcement implique deux formes d'apprentissages, que sont l'apprentissage symbolique et l'apprentissage numérique. L'apprentissage numérique vient de deux éléments.
Premièrement, on cherche à optimiser une succession de valeurs que l'on accumule et, comme nous le verrons, dont on fait une moyenne temporellement pondérée.
Deuxièmement, pour chacun des états que l'agent rencontre, on effectue un échantillonnage permettant de mesurer la qualité de cet état. L'apprentissage symbolique vient du fait que les états, fournis à l'agent à l'aide de descriptions, restent symboliquement séparés et l'on peut, comme nous le verrons au cours de ces travaux, appliquer des techniques d'apprentissage sur leur représentation. La façon de transmettre la tâche d'apprentissage à l'agent, c'est à dire sous la forme de valeurs numériques, permet d'avoir des tâches d'apprentissage très générales, qui peuvent même se passer d'objectifs autre que « survivre » ou faire de son mieux dans un environnement donné. Ce qui est assez loin d'une tâche de classification par exemple.

Concernant les données d'entraînement, il y a là un problème important. Si on reprend le cas de notre classifieur de mail, les données sont fournies de façon externe, soit par l'utilisateur, soit par l'ordre d'arrivée des mails. Dans le cadre de l'apprentissage par renforcement, les données d'apprentissage influent certes sur l'apprentissage, mais également par corollaire sur l'acquisition des données d'apprentissage suivantes. Ce genre de phénomène d'interaction où les données biaisent les données futures reste toujours un problème délicat à étudier. Il y a modélisation dans l'apprentissage par renforcement, mais comme nous l'avons vu, celle-ci intervient à deux niveaux. Les techniques classiques d'apprentissage n'interviennent qu'à un seul de ces deux niveaux. C'est à dire que l'ensemble des états reste dans les algorithmes classiques un ensemble, donc sans structure. Dès que l'on cherche à utiliser les algorithmes, on est obligé de faire intervenir le deuxième niveau, sans quoi les algorithmes convergent mais, dans la pratique, dans des temps inutilisables. Et comme nous le verrons, l'interaction de ces deux niveaux de généralisation pose des problèmes théoriques pour la plupart non maîtrisés actuellement. Les relations entre l'hypothèse cible (un comportement optimal) et l'espace des hypothèses sont là encore bien connues dans un cadre formel. En effet, il est démontré que dans un environnement stochastique statique modélisé par des Processus de Décision Markoviens, le comportement optimal est une politique déterministe. Non seulement l'hypothèse cible fait partie de l'espace des représentations, mais en plus, on est sûr que l'algorithme converge. Cependant, si on rajoute des propriétés à l'environnement que laisserait envisager les potentialités de l'apprentissage par renforcement, c'est à dire un système dynamique par exemple, ces résultats deviennent faux.

En revanche, un des résultats attendu en apprentissage numérique est l'intelligibilité du résultat, c'est à dire la possibilité pour un humain d'appréhender, voire d'analyser et de modifier le résultat de l'algorithme. Dans un cadre général, on peut dire que pour l'apprentissage par renforcement, cette propriété n'est pas respectée. Le résultat des algorithmes, est en général peu ou pas interprétable. C'est d'autant plus vrai si la représentation de l'environnement ou du comportement est elle-même numérique. Cependant, en pratique, les représentations de

l'environnement sont sélectionnées pour donner à l'agent des motifs connus par expertise pour être intéressant dans le cadre de la tâche désirée. (voir comme exemple parmis de nombreux autres [Stone et al., 2001]). L'intérêt chiffré, que l'agent estime pour une action dans un état possédant tel ou tel motif réputé significatif pour une tâche donnée, peut alors se révéler intelligible. En effet, l'utilisateur de l'algorithme peut extraire des informations du type « dans cette configuration là que je connais, il vaut mieux agir comme ceci ». Cependant, et c'est une partie de l'objet de nos travaux, il faudrait que ces motifs caractérisant les éléments intéressants du point de vue de la tâche pour une situation donnée ne soient pas fournis, mais trouvés par l'agent. L'intelligibilité pourrait alors être vue comme « l'agent a trouvé que ces caractéristiques étaient intéressantes ; et dans ce cas là, il vaut mieux agir comme ça ». Non seulement, c'est un pas vers l'intelligibilité, mais également vers l'extraction de connaissances.

Enfin l'incorporation dans la théorie de la résolution de l'apprentissage de la différence temporelle n'est pas sans poser de problèmes. La résolution de ceux-ci est actuellement principalement étudiée sous l'angle probabiliste et en utilisant judicieusement les paramètres contrôlant la myopie de l'agent. Donc, et contrairement à la question de l'exploration, cette question n'est pas traitée par une méthode directe. Cependant, le chapitre liant planification et renforcement de ([Sutton and Barto, 1998]) ainsi que les travaux menés sur l'apprentissage par renforcement relationnel (([Dzeroski et al., 2001]) vont sûrement remettre en cause cet état de choses.
L'étude de ces champs de recherche de l'apprentissage par renforcement pourra donc potentiellement amener à ce qu'au-delà d'un bon comportement pour un agent dans un environnement donné, on puisse obtenir des résultats sur la dynamique sous-jacente de l'environnement. En effet, l'agent disposera alors de résultats du type « cette succession d'actions à partir d'un état ayant telles propriété mène dans un état ayant celles-ci, il vaut alors mieux faire ça ».

Ces dernières réflexions nous permettent de récapituler les objectifs ainsi que les cadres de nos travaux.

Objectifs Nos travaux se situent donc dans le cadre des algorithmes d'apprentissage par renforcement. Nous nous plaçons dans la perspective de conserver une représentation symbolique de l'environnement. L'absence de structuration des représentations successives de l'environnement, appelées états de l'environnement, limite l'intérêt, voir les possibilités de l'apprentissage par renforcement. Nous nous proposons donc d'étudier une forme de structuration basée sur le langage de description de ces états. Des opérateurs associés à un langage de description d'un ensemble d'objets impliquent une structuration sur l'espace de ces objets. Nous considérerons les états comme ces objets et utiliserons des opérateurs d'inégalité et d'équivalence pour doter d'une structure d'ordre partiel à l'ensemble de ces états, forme de structuration largement utilisée en apprentissage machine pour ses propriétés de généralisation.
La généralisation pour l'apprentissage par renforcement n'est pas une initiative nouvelle, nous verrons comment divers tentatives ont déjà été faites, que ce soit d'une manière numérique, algébrique, ou dans le cadre de la discrétisation d'un espace continu. Nous analyserons ensuite les implications de cette structure quant aux prises de décisions et proposerons des algorithmes prenant en compte cette structuration. De plus, nous verrons comment ces résultats peuvent être interprétés d'une point de vue extraction de connaissances ou autrement, donnerons la possibilité d'utiliser une connaissance préalable sur l'environnement par le biais de la description de l'environnement.
Une expérimentation mettant en oeuvre ces idées sera proposée. Enfin, les questions soulevées par ces travaux et non encore résolues ainsi que des perspectives de recherche seront avancées.

Cadre et limitations Comme nous l'avons évoqué, l'apprentissage par renforcement dispose d'un horizon très vaste d'utilisation. Cependant, les propriétés mathématiques assurant la convergence de ses algorithmes se basent sur des cadres a priori qui restreignent fortement les utilisations pratiques. De nombreuses recherches visent à s'affranchir de ces cadres, que ce soit dans le traitement de cas continus ([Munos, 1997a]), dans l'utilisation de connaissances a priori ([Dzeroski et al., 2001]) ou dans l'application aux systèmes multi-agents.
Comme nous introduisons une notion formelle dont les effets ne sont pas étudiés à l'heure actuelle, nous allons suivre le processus inverse. Nos allons maintenir des cadres relativement restrictifs

dans le but d'étudier déjà les problèmes dans ceux-ci. Même si bien entendu et comme nous le montrerons, les idées exposées ont vocations à s'étendre au-delà. C'est pourquoi nous conserverons les restrictions suivantes :

- Environnement markovien, statique et discret
- Agent unique et omniscient

Cependant, nous conserverons une des propriétés forte de l'apprentissage par renforcement et qui consiste à limiter les connaissances a priori sur l'environnement. Même si par essence, notre algorithme permet l'introduction de connaissances par le biais de la représentation, nous bornerons l'introduction de connaissances a priori à cet élément.

1.2 Résumé - Problématique

Nous allons résumer ici la structure de nos travaux afin de donner une vue d'ensemble et synthétique au lecteur des problématiques posées et des champs de réponse envisagés.

Problème de l'apprentissage par renforcement

Le cadre général de notre travail se situe dans le cadre de l'apprentissage par renforcement ([Sutton and Barto, 1998]). L'apprentissage par renforcement est un cadre d'apprentissage artificiel ayant pour objectif de faire apprendre un comportement à un agent, pouvant percevoir et agir sur un environnement. L'agent reçoit des récompenses ou des punitions sous forme de valeurs numériques en fonction de la qualité de l'état de l'environnement dans lequel il se trouve (voir figure 1.1). L'agent doit alors faire varier son comportement en renforçant la sélection des actions le conduisant à maximiser le flux de récompenses qu'il reçoit, et inversement à éviter sa sélection des actions l'amenant dans des états peu intéressants, voire problématiques.

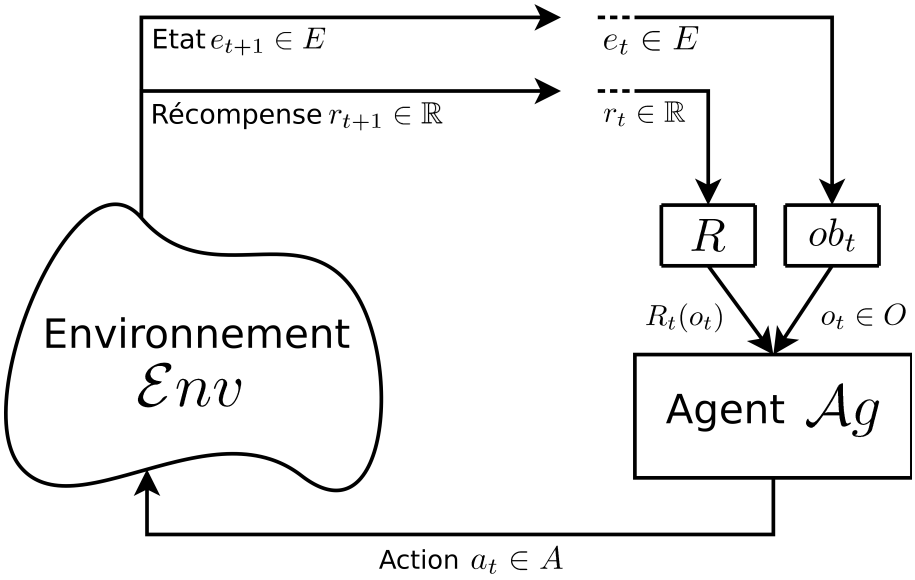


FIG. 1.1 – Apprentissage par renforcement : l'agent apprend un comportement en fonction de ses interactions avec son environnement avec l'objectif de maximiser une fonction de récompense.

Parmi les caractéristiques les plus importantes de ce cadre général, citons les suivantes :

- C'est bien le flux de récompenses et non pas une seule récompense qui est à considérer par l'agent, cela signifie que les algorithmes doivent intégrer la notion de récompense « à long terme ». De plus, il faut prendre en compte le fait qu'une action peut avoir des effets décalés dans le temps sur l'acquisition des récompenses et le fait qu'une récompense est généralement la résultante de plusieurs actions.
- Il n'y a pas nécessairement de séparation entre une phase d'apprentissage et une phase d'utilisation de l'apprentissage, celles-ci se font conjointement. Par conséquent, il faut gérer

le compromis entre 1) la recherche immédiate de récompenses en se basant sur la connaissance courante et 2) l'amélioration de la connaissance permettant de les accroître.
- Les environnements considérés sont généralement stochastiques et non déterministes, ce qui implique que les actions n'ont pas systématiquement les mêmes effets sur l'environnement, ceux-ci étant régis par des lois probabilistes.

Problème de la généralisation des comportements pour l'apprentissage par renforcement

Si dans ses formes premières, l'apprentissage par renforcement répond à l'apprentissage d'un comportement optimal pour un agent pour tous les états d'un environnement donné, dans ses extensions ainsi que dans son utilisation pratique, il est nécessaire d'étendre le problème. En effet, confronter un agent à tous les états possibles d'un environnement un grand nombre fois est en pratique souvent infaisable ; on désirerait plutôt que l'agent puisse généraliser un comportement appris à des situations similaires, quitte à ce que celui-ci ne soit pas optimal. Ce problème est celui de la *généralisation* en apprentissage par renforcement.
La plupart des travaux concernant cette question ont été produits dans le cadre de l'approximation de fonction. Les algorithmes utilisant les réseaux de neuronaux en sont probablement les exemples les plus connus. D'autres approches ont toutefois été développées. Actuellement et depuis quelques années, c'est l'utilisation de représentations relationnelles ([Dzeroski et al., 2001]) pour décrire l'environnement qui fait l'objet de nombreux travaux. C'est dans ce type de démarches plus sémantiques et symboliques que nous plaçons nos travaux.
Pour notre part, nous nous proposons d'adapter les treillis de Galois, structures utilisées par ailleurs dans le cadre de l'apprentissage artificiel et notamment de l'apprentissage par induction, pour donner un cadre algébrique à la question de la généralisation pour l'apprentissage par renforcement. Nous proposons également des algorithmes basés sur cette approche.

Utilisation des treillis de Galois pour structurer un espace de généralisation

Considérons un ensemble d'objets $\mathcal{O}bj$ et un langage L ayant une structure de demi-treillis permettant de décrire ces objets ainsi que certaines parties de leur ensemble. Les treillis de Galois ([Ganter and Wille, 1999]) et plus précisément les treillis de concepts sont des objets mathématiques permettant de structurer sous forme de treillis, l'ensemble des façons de regrouper les éléments de $\mathcal{O}bj$ de manière compatible avec L ([Liquière, 2006]). On peut également le qualifier comme regroupant l'ensemble des parties de $\mathcal{O}bj$ pouvant être décrites par L.

Les Treillis de Galois sont de plus en plus utilisés dans un cadre informatique et notamment en apprentissage artificiel. Ils permettent de structurer l'espace des regroupements possibles en fonction d'un langage de description (voir figure 1.2 page suivante). Selon un procédé identique, il est possible de définir l'ensemble des partitionnements possibles en utilisant un langage de description sur l'ensemble des objets, c'est une structure particulière que nous appelons le treillis de Galois des partitions (voir figure 1.3 page 12). Elle permet de définir avec une structure de treillis, l'ensemble des partitions de $\mathcal{O}bj$ compatibles avec le langage L.

Reformulation du problème de la généralisation pour l'apprentissage par renforcement à l'aide des treillis de Galois et des treillis de Galois des paritions

Nos travaux consistent donc à proposer une formalisation particulière du problème de généralisation de l'apprentissage par renforcement. Celui-ci peut se voir comme le regroupement des états similaires du point de vue de l'action à mener. Cette formalisation implique deux structures : le treillis des parties (l'ensemble des regroupements possibles des états entre eux) et le treillis des partitions de l'ensemble des états.
Le treillis de concepts permet précisément de biaiser l'espace des généralisations possibles d'un ensemble d'objets en le contraignant à l'aide d'un langage de description. C'est la formalisation de cet espace que nous proposons ici. Ainsi, un comportement optimal appris sur plusieurs états de l'environnement se ressemblant conformément au langage utilisé pour les décrire, sera généralisé et attribué à tous les autres états dont le comportement optimal est inconnu et qui correspondent à la même généralisation (voir figure 1.4 page 13).
Dans la même démarche que pour les treillis de Galois, nous considérons un deuxième espace de généralisation avec le treillis de Galois des partitions, permettant de considérer les partitions

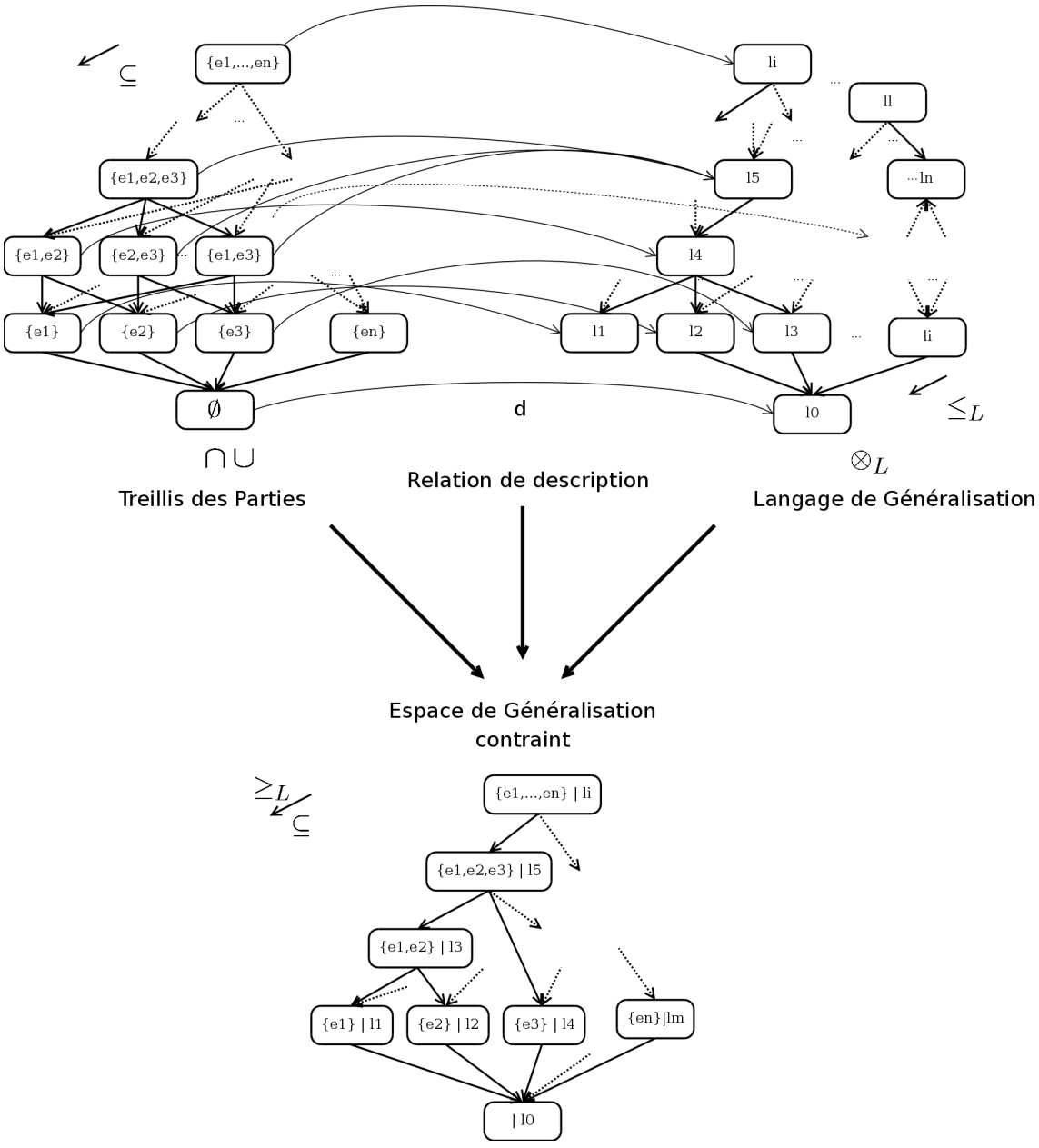


FIG. 1.2 – A partir du treillis des parties d'un ensemble d'objets (à gauche) et d'un langage muni d'un ordre partiel, permettant de décrire un sous-ensemble de ces parties (à droite), on produit le treillis de Galois (en bas), comme étant l'ensemble partiellement ordonné des parties de l'ensemble d'objets que l'on peut décrire avec le langage. Il implique également des règles du type « si on regroupe deux objets obj_1 et obj_2, alors le langage me contraint à les regrouper également avec obj_i ».

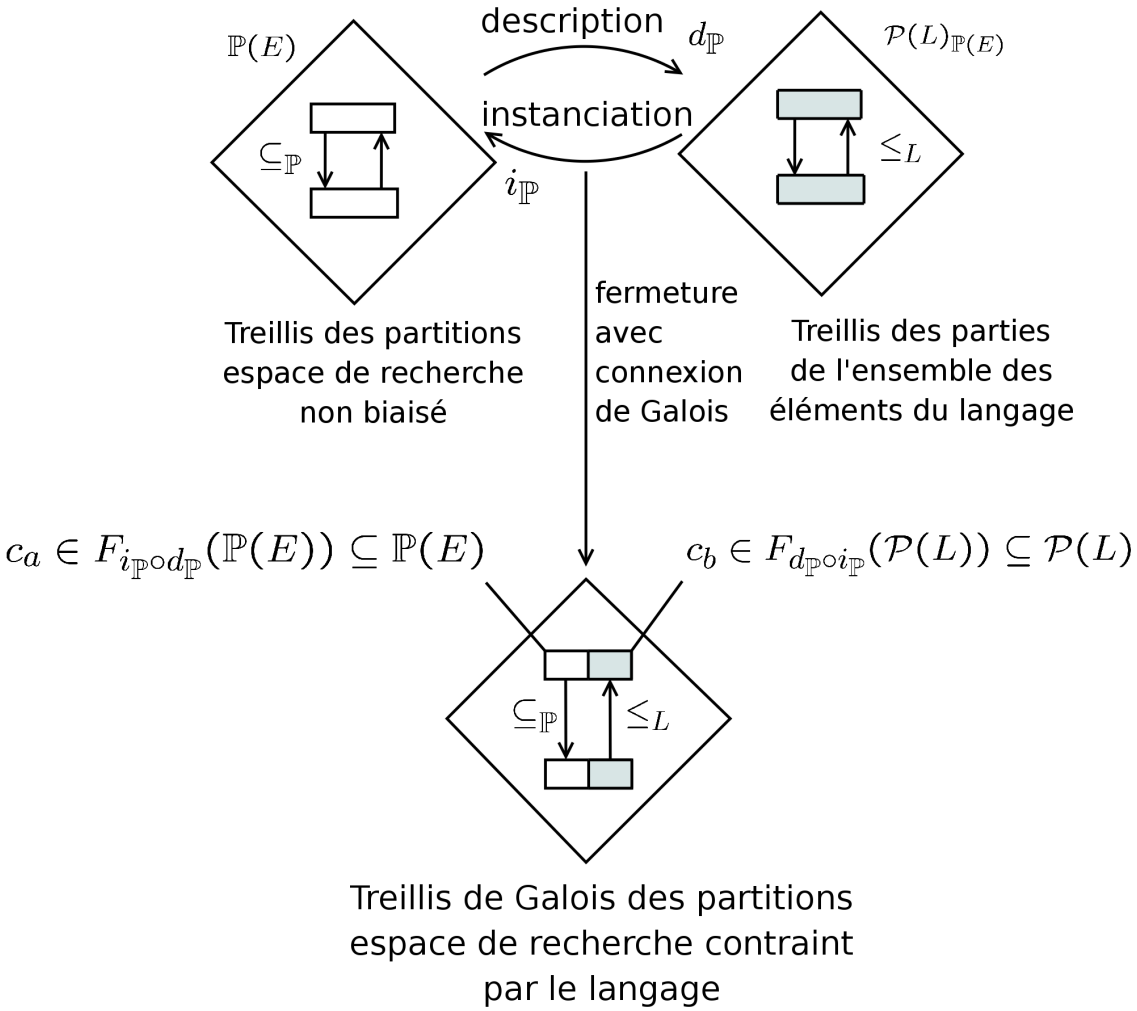


FIG. 1.3 – Le treillis des partitions constitue un espace de recherche non biaisé parmi les partitions possibles d'un ensemble d'objets. Avec le treillis des parties de l'ensemble des éléments du langage permettant de décrire une partie de l'ensemble des parties de l'ensemble des objets, on peut construire le treillis de Galois des partitions. Celui-ci structure l'ensemble des partitions descriptibles par le langage de description.

compatibles avec le langage de description des états. Toutes les façons de partitionner les états ne sont pas admises, seules celles compatibles avec le langage de description des états le sont.

Algorithmes

Le cadre que nous proposons est très général, puisqu'il fournit des outils algébriques pour structurer l'espace de généralisation pour l'apprentissage par renforcement. Par conséquent, il peut être utilisé algorithmiquement de nombreuses façons et nous ne les avons pas toutes explorées. Néanmoins, nous proposons deux algorithmes en nous basant sur ces concepts.

Q-Concept-Learning Une des utilisations possibles de la structure de treillis de Galois pour les états de l'environnement est de ne pas reporter, après l'obtention d'une récompense, la mise à jour de la fonction de qualité seulement sur l'état concerné, mais sur l'ensemble des généralisations possibles à partir de l'état concerné ([Ricordeau, 2003, Ricordeau, 2004] et voir figure 1.4 page ci-contre). Cette méthode, que nous nommons *Q-Concept-Learning* a deux aspects. Premièrement, elle permet une généralisation de l'apprentissage, puisque l'expérience acquise sur un état est reportée sur les états corrélés par le langage. Deuxièmement, elle permet de faire la distinction entre les

généralisations significatives et les généralisations abusives dans le cadre de la tâche considérée. En effet, les valeurs estimées des actions pour les généralisations significatives auront tendance à converger vers un même ensemble d'actions optimales contrairement aux généralisations abusives.

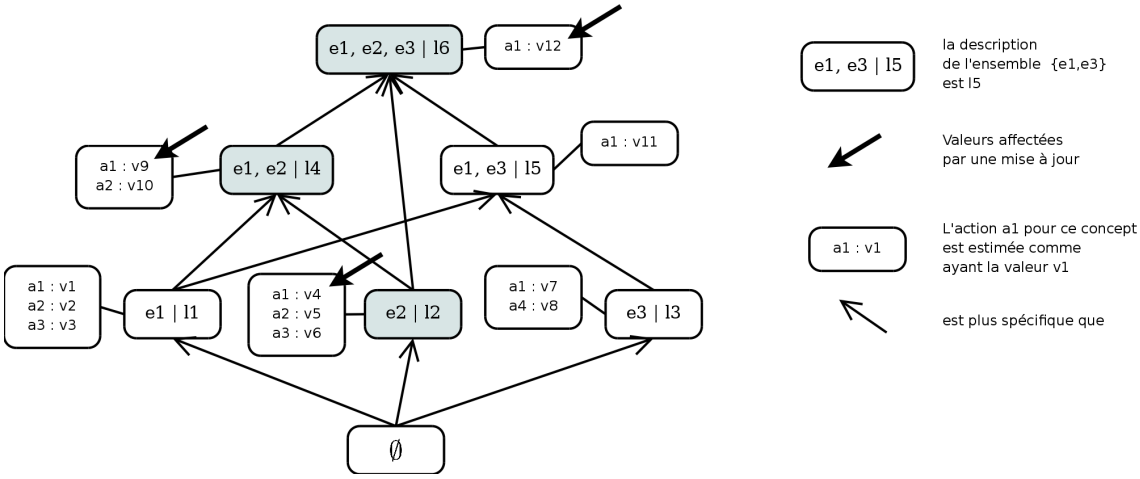


FIG. 1.4 – Nous voyons sur cet exemple un environnement à 3 états : e_1, e_2, e_3. Le langage de description ne permet pas de décrire $\{e_2, e_3\}$. Une mise à jour suivant une interaction avec l'état e_2 et l'action a_1 impliquera également une mise à jour sur les concepts dont les extensions incluent e_2 : $\{e_1, e_2\}$ et $\{e_1, e_2, e_3\}$. Si l'agent découvre un état inconnu e_4 qui se généralise avec e_1 et e_2, l'agent pourra sélectionner son action à partir du concept $\{e_1, e_2\}$ plutôt que de considérer qu'il ne connaît rien sur e_4.

Recherche du langage optimal / algorithme en généralisation sur le treillis de Galois des Partitions La généralisation pour l'apprentissage par renforcement peut se voir comme le regroupement des états similaires du point de vue de leur action optimale. Ceci est équivalent à un partitionnement de l'ensemble des états en une partition $\mathcal{P}^*$ ([Ricordeau and Liquière, 2006]).

Nous proposerons un algorithme de généralisation de la politique ainsi apprise à l'aide du langage L_E de description des états de l'environnement.
De plus, si on considère le treillis de Galois des partitions $\mathcal{TGP}$ de l'ensemble des états de l'environnement et après convergence d'un apprentissage pas renforcement, on peut comparer les différents cas de figures suivant le fait que $\mathcal{P}^*$ appartienne ou non au $\mathcal{TGP}$.
On pourra alors proposer le cas échéant, d'enrichir le langage L_E de termes permettant à $\mathcal{P}^*$ d'être exprimé. La figure 1.5 page suivante donne un aperçu de ces notions.

Nos travaux sont organisés comme suit. Dans la partie 2, nous présenterons les bases de l'apprentissage par renforcement. Nous y définirons notamment toutes les notions de bases que nous utiliserons.
Ensuite, partie 3, nous présenterons les notions pour la généralisation des politiques pour l'apprentissage par renforcement. Nous présenterons la généralisation par approximation de fonction, l'apprentissage par renforcement relationnel, puis des travaux plus algébriques utilisant directement le modèle de l'environnement pour effectuer les généralisations.
Après avoir introduit les notions autour de l'apprentissage par renforcement, il nous faudra faire de même avec celles autour des treillis de Galois. C'est ce que nous ferons partie 4.
La partie 5 présentera une façon de formaliser les notions de généralisation des politiques en apprentissage par renforcement en utilisant les treillis de Galois. Nous y proposerons également une première contribution algorithmique visant à généraliser une politique après un apprentissage par renforcement.
Partie 6, nous présenterons les algorithmiques que nous nommons *Q-Concept-Learning*, utilisant également le treillis de Galois comme outil de généralisation des politiques, cependant, l'algo-

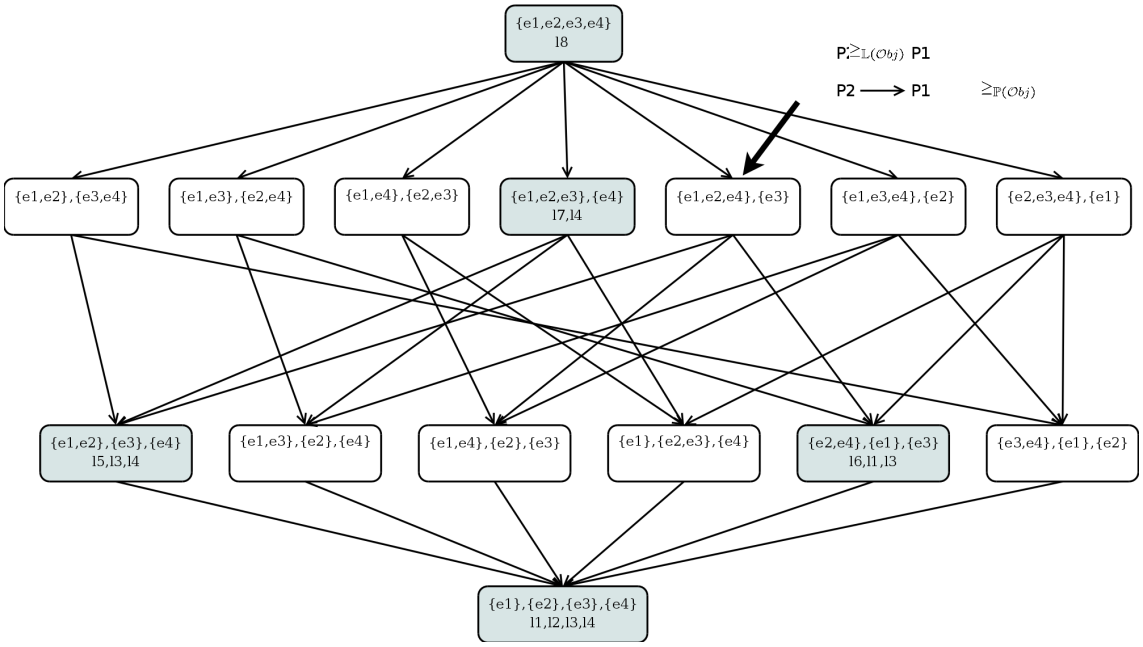


FIG. 1.5 – On voit ici le treillis des partitions de l'ensemble des 4 états d'un environnement. Seules les partitions avec le fond grisé peuvent être décrites dans le langage de description des états. Le treillis de Galois des partitions serait donc composé de celles-ci. Dans cet exemple, la partition marquée d'une flèche est la partition recherchée, c'est à dire qu'aprés convergence de l'apprentissage par renforcement, e_1, e_2, e_4 ont les même actions optimales. Le langage ne permettant pas de la décrire, on devra choisir entre deux partitions plus fines. Ceci suggère également qu'il faudrait rajouter un terme au langage permettant de décrire $\{e_1, e_2, e_4\}$

rithme proposé sera conçu pour opérer cette généralisation durant l'apprentissage (en ligne), plutôt qu'après. Nous proposerons finalement partie 7, une conclusion ainsi qu'une discussion et des perspectives sur les problèmes ouverts que posent nos travaux.

2 Apprentissage Machine - Apprentissage par Renforcement

Dans ce chapitre, nous allons présenter les bases et les formalismes de l'apprentissage par renforcement. Nous poserons d'abord le problème partie 2.1, puis nous le formaliserons partie 2.2.
Nous commencerons par donner les méthodes classiques de résolution exacte partie 2.3, avant de donner les algorithmes basiques d'apprentissage par renforcement partie 2.4. Le lecteur ou la lectrice connaissant le sujet pourra sauter cette partie de rappel en regardant toutefois éventuellement les formalismes utilisés, ceux-ci pouvant différer légèrement des approches classiques.
Le lecteur ou la lectrice souhaitant une introduction plus approfondie pourra se référer à [Mitchell, 1997] pour une approche rapide et à [Sutton and Barto, 1998] ou [Kaelbling and Littman, 1996] pour une introduction plus complète. Nous n'aborderons les différentes méthodes de généralisation pour l'apprentissage par renforcement que dans le prochain chapitre.

2.1 Introduction

L'apprentissage artificiel[2] s'est inspiré de l'étude de l'apprentissage humain (neurosciences, psychologie) ou animal, comme du reste de nombreuses branches de la science se sont inspirées d'observations de phénomènes naturels ou des « inventions » des différentes espèces. Ainsi, on peut citer les apprentissages génétiques ou neuronals pour ce qui est de l'inspiration biologique ou l'apprentissage supervisé pour ce qui est de l'inspiration relative à l'apprentissage humain. Évidemment, tout comme l'aéronautique ne s'est jamais contentée d'imiter le vol des oiseaux, l'apprentissage artificiel une fois formalisé en domaine de recherche s'est immédiatement émancipé de ses inspirations biologiques.
L'apprentissage par renforcement est donc issu de l'inspiration biologique de l'apprentissage par essais-erreurs permettant à un certain nombre d'animaux de renforcer leurs comportements ayant pour effet d'accroître leur plaisir ou de limiter leur déplaisir ou souffrance.
De manière similaire, l'apprentissage par renforcement pose le problème d'un agent artificiel, disposant d'un ensemble de comportements possibles, et recevant par valeur numérique un plaisir ou déplaisir, l'objectif étant de faire en sorte que l'agent apprenne à produire un comportement qui accroisse les valeurs numériques représentant son plaisir et diminue les valeurs numériques représentant son déplaisir. La biologie n'étant pas notre domaine et la comparaison entre les champs de recherche se bornant à l'inspiration initiale, nous nous arrêtons ici pour ce propos. Limitons-nous à définir le problème dans le cadre de l'apprentissage artificiel.

L'agent doit apprendre de son interaction avec son environnement Commençons par définir de manière générale, la notion d'apprentissage par renforcement, résumée figure 2.1 page suivante.

Définition 2.1 (Problème de l'apprentissage par renforcement)
Soit un **agent** $\mathcal{A}g$ et un **environnement** $\mathcal{E}nv$. Avec $\mathcal{A}g$:

- agissant au cours d'interactions avec son environnement $\mathcal{E}nv$ par des **actions** $a_i \in A$
- percevant au cours des interactions les **états** $e_i \in E$ de son environnement $\mathcal{E}nv$ modifiés par une **fonction d'observation** $ob_t : E \rightarrow O$
- percevant au cours des interactions une évaluation numérique $r_i \in \mathbb{R}$ de l'état observé appelée **récompense**
- étant paramétré par une **fonction de récompense** R à maximiser.

On appelle apprentissage par renforcement les algorithmes permettant à l'agent $\mathcal{A}g$ d'améliorer au sens de R, la sélection des actions a_i au fur et à mesure de ses interractions avec $\mathcal{E}nv$.

[2] voir le chapitre introductif de [Sutton and Barto, 1998] pour un intéressant rappel historique sur les origines de l'Apprentissage par Renforcement

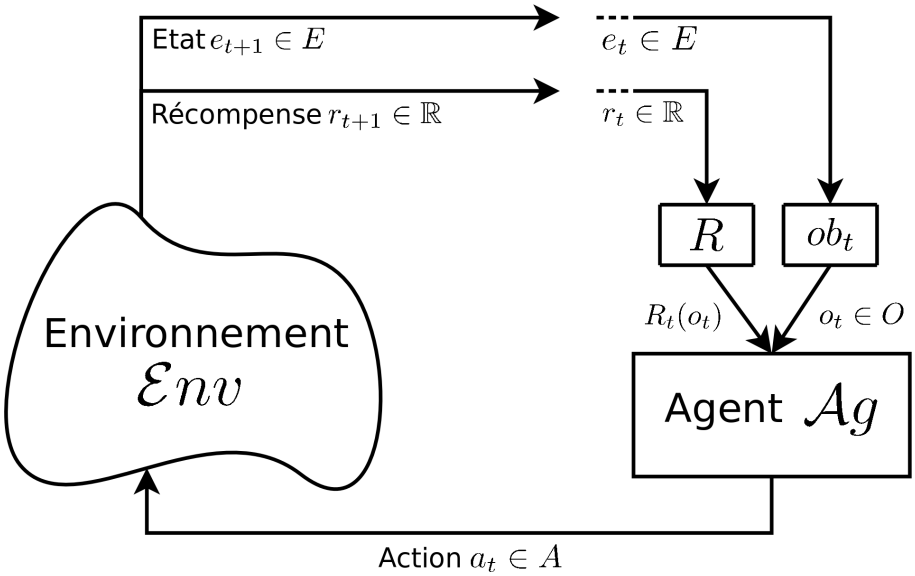


FIG. 2.1 – Apprentissage par renforcement : l'agent apprend un comportement en fonction de ses interactions avec son environnement avec l'objectif de maximiser une fonction de récompense

Notons qu'une des propriétés importantes de l'apprentissage par renforcement est que le cadre d'apprentissage indiqué n'implique pas que l'agent ait des connaissances préalables sur son environnement.

Fonction d'observation de l'environnement Comparé au schéma classique de l'apprentissage par renforcement, nous avons introduit la notion de fonction d'observation. Si l'on veut conserver le formalisme « agent », il est en effet essentiel de séparer dans le modèle les états de l'environnement proprement dits de la perception que peut en avoir l'agent. La fonction d'observation permet donc dans le modèle cette séparation. Elle peut par exemple formaliser le fait que plusieurs états de l'environnement soient perçus de la même façon par l'agent ceci étant dû aux limites des percepteurs de l'agent.
Ainsi, la fonction d'observation formalisera les limites « physiques » de l'agent concernant la perception de son environnement, limites dans notre cadre, qu'il ne peut pas modifier.

Le contexte d'un agent interagissant avec son environnement et modifiant son comportement en fonction des récompenses et des punitions qu'il reçoit est un contexte séduisant dans le sens où il est très général et peut donc s'adapter à nombre de problèmes. Cependant, pour des raisons à la fois historiques et par nécessité d'un formalisme mathématique, l'apprentissage par renforcement est souvent traité dans le cadre du formalisme des Processus de Décision Markovien (abrégé en PDM ou MDP en anglais). C'est donc celui-ci qui nous servira à modéliser la dynamique de l'environnement.
Nous décrirons le formalisme de **Processus de Décision Markovien** dans le prochain paragraphe ainsi que les notations utiles à nos travaux. Nous donnerons également la notion de **politique**, traduction formelle du comportement d'un agent dans un tel environnement. Nous poursuivrons par des méthodes de résolution classiques pour les PDM. Ensuite, nous présenterons l'apprentissage par renforcement dans ce contexte avec une présentation des méthodes désormais classiques de ce domaine. Enfin, nous présenterons les problèmes posés par l'apprentissage par renforcement. Notons que d'autres formalismes pour l'apprentissage par renforcement ont été développés pour pouvoir intégrer une plus vaste gamme de problèmes ainsi, on peut trouver les Processus de Décision Semi-Markoviens (voir [Sutton et al., 1999]) ainsi que plus récemment, les Représentations Prédictives des Etats (voir [Singh et al., 2004]).

2.2 Formalisme

Les notations utilisées dans le cadre des travaux sur l'Apprentissage par Renforcement varient selon les ouvrages. Le livre [Sutton and Barto, 1998] constituant maintenant la référence du domaine, nous essaierons de nous conformer le plus possible à ses notations. Dans le cadre de nos travaux, nous considérerons que l'environnement ne peut se trouver que dans un nombre fini d'états. Nous considérons donc que l'agent évolue dans un environnement discret. De même, ses interactions avec l'environnement seront elles aussi discrètes.
L'étude du « cas continu », c'est à dire l'étude d'un monde dont le nombre d'états et/ou d'actions peut être infini en étant représentés par des variables à valeur dans des espaces continus, fait l'objet de travaux, notamment dans [Munos, 1997a]. Nous y reviendrons dans la partie 6 car ces études ont cependant des relations avec les méthodes exposées ici.

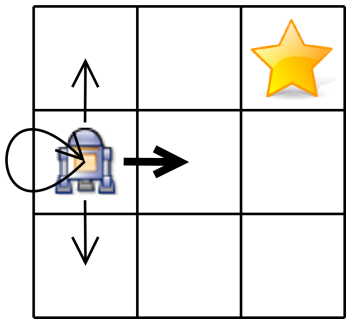


FIG. 2.2 – Un monde grille classique où la tâche à apprendre est de se déplacer sur la case où se trouve l'étoile

Exemple 2.1 *Nous allons illustrer notre propos par l'exemple classique du monde grille (figure 2.2). Il s'agit d'un échiquier sur lequel l'agent peut se déplacer dans les quatre directions cardinales. Si l'action de déplacement amène hors des limites définies, l'agent reste sur place. Le comportement que l'on désire obtenir de l'agent est qu'il se dirige le plus rapidement possible en direction d'une case particulière, ici celle comportant une étoile. Nous détaillerons certains aspects de la formalisation qui peuvent paraître évidents au premier abord et qui pourtant recèlent quelques pièges, notamment dans les abus d'écriture qui peuvent nuire ultérieurement à la compréhensions de certains phénomènes.*

2.2.1 Processus de Décision Markovien

Nous allons définir l'ensemble des élements permettant de formaliser un environnement sous la forme d'un processus de décision markovien.
Commençons d'abord par décrire les éléments qui composent un environnement : les **états**, qui sont les différentes configurations possibles de l'environnement puis les **actions**, qui sont le moyen qu'a l'agent pour interagir et ainsi modifier l'état de l'environnement.

Notation 2.1 (E : ensemble des états possibles de l'environnement)
On note E l'ensemble des états possibles de l'environnement. On note $E_0 \in E$, l'état initial de l'environnement et $E_T \in E$ l'état terminal de l'environnement s'il existe.

L'état terminal E_T n'existe pas nécessairement pour tous les mondes. Un monde avec plusieurs états initiaux ou terminaux est trivialement transformable en monde avec un seul état initial et terminal.

Remarque 2.1 *Pour la suite de ce chapitre, nous allons considérer que la perception qu'a l'agent d'un état de son environnement est exactement l'état de l'environnement tel qu'il est dans le modèle de l'environnement lui-même, pour ne pas alourdir le propos ainsi que les notations. Ainsi, nous considérerons donc que $E = O$ et $\forall e \in E, o_t = e$. Nous reviendrons sur cet a priori chapitre 5 avec la notion de Processus de Décision de Markov décrit.*

Exemple 2.2 *L'ensemble des états de l'environnement pour l'agent est l'ensemble des cases de l'échiquier :*
$E = \{e_1, e_2, ..., e_9\}, E_0 = e_1, E_T = e_9, |E| = 9$
Pour des questions pratiques, il est classique d'étiqueter les états par leurs coordonnées :
$E = \{e_{(0,0)}, e_{(0,1)}, ..., e_{(2,2)}\}, E_0 = e_{(0,0)}, E_T = e_{(2,2)}$
Cependant, il faut se souvenir que pour l'agent, il ne s'agit que d'un ensemble d'états non corrélés. Il ne faut pas considérer que la structuration implicite des états donnée par un tel étiquetage est exploitable pour l'agent. Pour celui-ci, ils sont simplement différents. Dans le même esprit, la notion de distance, qui pourrait être pratique pour la résolution du problème, est absente. Nous détaillons ce point car c'est précisément l'utilisation d'une structuration des états basée sur l'étiquetage pour biaiser et accélérer l'apprentissage qui fait l'objet des présents travaux.
Pour notre exemple, l'agent pourra observer complètement son environnement et donc différencier tous les états. Ainsi, $\forall t, \forall e, ob_t(e) = e$ *et* $E = O$.

L'agent, comme présenté figure 2.1 page 16, peut modifier l'état de l'environnement dans lequel il évolue. Il peut le faire par l'intermédiaire d'actions.

Notation 2.2 (A : ensemble des actions possibles pour l'agent)
Soit A l'ensemble discret des actions que l'agent peut sélectionner pour agir sur son environnement.

Exemple 2.3 *L'ensemble des actions pour notre exemple est* $A = \{$"*Déplacement Nord*", "*Déplacement Sud*", "*Déplacement Est*", "*Déplacement Ouest*"$\}$ *pour chacun des états. Cependant, comme pour l'étiquetage des états, il ne s'agit ici que d'un étiquetage pratique pour donner du sens à l'humain.*
Pour l'agent, il s'agit là encore d'un ensemble d'éléments différents. L'action "*Déplacement Ouest*" *dans l'état* $e_{(1,1)}$ *n'est pas plus en rapport avec l'action* "*Déplacement Ouest*" *dans l'état* $e_{(1,2)}$ *qu'avec l'action* "*Déplacement Sud*" *dans l'état* $e_{(1,1)}$. *Ainsi, ici* $|A| = 4 \times 8 = 32$. *L'agent conserve cependant l'information que telle action est effectuée dans tel état.*

L'apprentissage se fait donc par une suite d'interactions entre l'agent et l'environnement. Cette succession d'interactions peut être vue comme une mesure du temps qui s'écoule, même si chacune des interactions n'a pas nécessairement la même durée. Suivant la tâche, le nombre d'interactions peut être infini, borné par un nombre N (la tâche s'arrête après N interactions), soit les interactions se terminent lorsque l'état terminal E_T est atteint.

Notation 2.3 (e_t, a_t)
On note e_t, $t \in \mathbb{N}$, l'état de l'environnement après t interactions. On note a_t, $t \in \mathbb{N}$ l'action sélectionnée par l'agent dans l'état e_t.

Définition 2.2 (épisode)
Si l'environnement est muni d'un état final E_T, quand l'agent se trouve dans celui-ci, il revient dans l'état de l'environnement initial E_0. De même, lorsque le nombre d'interactions est borné, quand le nombre d'interactions maximal N est atteint, l'agent revient dans l'état de l'environnement initial E_0. Dans les deux cas, une telle séquence s'appelle un **épisode**.

Exemple 2.4 *Prenons par exemple la succession d'interaction de la figure 2.3 page suivante elle est représentée par la suite* $e_0 = E_0 = e_{(0,0)}$, $a_0 =$ *Déplacement Nord*, $e_1 = e_{(0,1)}$, $a_1 =$ *Déplacement Est*, $e_2 = e_{(1,1)}$, $a_2 =$ *Déplacement Nord*, $e_3 = e_{(2,1)}$, $a_3 =$ *Déplacement Est*, $e_4 = E_T = e_{(2,2)}$.

Modélisation de l'environnement Nous avons vu que l'agent pouvait agir sur son environnement, c'est à dire modifier l'état de celui-ci par l'intermédiaire d'actions. Nous allons maintenant préciser la mécanique interne qui permet à un environnement de passer d'un état $e_i \in E$ à un état $e_j \in E$ en fonction d'une action $a_i \in A$. Ce passage est modélisé par une fonction de probabilité appelée **modèle de transition** ou **dynamique** de l'environnement.

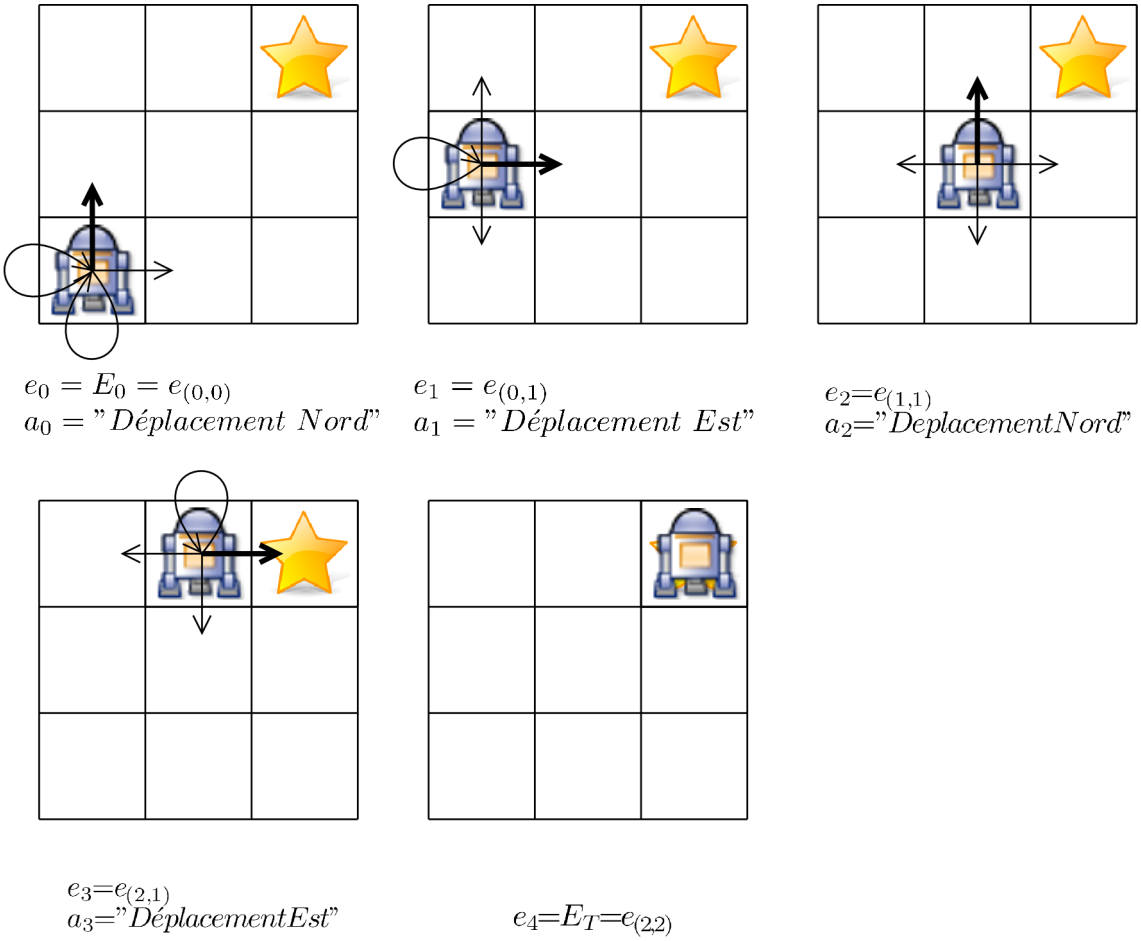


FIG. 2.3 – Exemple d'épisode dans l'environnement du monde grille. Les flèches représentent les actions disponibles pour l'état considéré. Celle qui est en gras représente l'action effectivement sélectionnée.

Définition 2.3 (Modèle de transition T)
Soit $T : E \times \cdots \times E \times A \times E \longrightarrow [0,1]$. T est appelée modèle de transition (ou dynamique) du système (ou de l'environnement). Il est défini par un ensemble de probabilités définissant la distribution des évènements en fonctions des états rencontrés et des actions sélectionnées :

$$T(e_0, \ldots, e_t, a_t, e'_{t+1}) = P(e'_{t+1} \mid e_0, \ldots, e_t, a_t) \tag{2.1}$$

où $P(a|b)$ dénote la probabilité que l'évènement a se produise dans le contexte b. Nous avons de plus la propriété suivante :

$$\sum_{e' \in E} T(e_0, \ldots, e_t, a_t, e'_{t+1}) \in \{0, 1\} \tag{2.2}$$

Notation 2.4 ($Acc(e_t)$, $A(e_t), \psi$)
Soit $e_t \in E$ un état de l'environnement. Soit $Acc(e_t)$, l'ensemble des états $e'_{t+1} \in E$ et $A(e_t)$, l'ensemble des actions $a_t \in A$ tels qu'il existe $e_0, \ldots, e_{t-1}$ avec $T(e_0, \ldots, e_{t-1}, e_t, a_t, e'_{t+1}) \neq 0$.
$Acc(e_t)$ représente l'ensemble des états *accessibles* à partir de e_t et $A(e_t)$ représente l'ensemble des actions que l'agent peut *sélectionner* dans l'état e_t.
Soit $\psi \subseteq E \times A$ tel que $\forall (e_t, a_t) \in \psi,\ a_t \in A(e_t)$. ψ est nommé l'ensemble des couples $(état, action)$ admissibles.

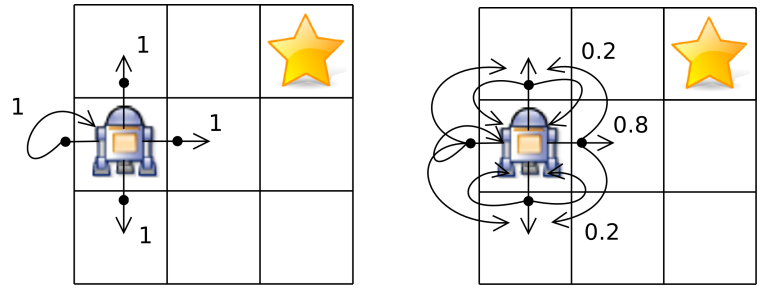
Exemple 2.5 *Dans notre exemple,* $\forall (i,j) \in ([0,2],[0,2])$, $A(e_{(i,j)})$={ *"Déplacement Nord", "Déplacement Est", "Déplacement Sud", "Déplacement Ouest"*}. $Acc(e_{(0,0)}) = \{e_{(0,0)}, e_{(0,1)}, e_{(1,0)})\}$.

Le modèle de transition T définit la probabilité que le système soit dans l'état e'_{t+1} suite à l'interaction t étant données :

1. la succession des états pris par le système les état $e_0, \ldots, e_t$,
2. l'action a_t sélectionnée par l'agent à l'interaction t

La condition posée sur la somme des probabilités pour un même historique $(e_0, \ldots, e_t)$ et une même action (a_t) (équation 2.2 page précédente) vise simplement à conserver la cohérence du système. Elle exprime deux choses :

1. Premièrement qu'une action peut ne pas être disponible pour chaque état de l'environnement (si la somme des probabilités est nulle, voir notation 2.4 page précédente)
2. Deuxièmement que si une action a_t est disponible à l'interaction t et étant donné un historique, alors la somme des probabilités pour l'état du système à l'interaction $t+1$, l'action a_t étant sélectionnée, doit être égale à 1.



(a): exemple d'environnement déterministe

(b): exemple d'environnement stochastique

FIG. 2.4 – Deux exemples de dynamique de l'environnement

Exemple 2.6 *Dans le cadre de notre exemple, on peut proposer plusieurs modèles de transition.*

1. *Premièrement, un exemple de transition* ***déterministe*** *: L'action "Déplacement Nord" conduit toujours l'agent dans la case au nord sauf si cette case n'existe pas. Dans ce cas, il reste dans la même case. On applique ce principe à l'ensemble des actions et des états.*
2. *Deuxièmement, l'effet des actions sur l'environnement peut ne pas être déterministe. Ainsi, l'action "Déplacement Nord" conduira l'agent dans la case au nord avec une probabilité de 80%, dans la case à l'est avec une probabilité de 10% et dans la case à l'ouest avec une probabilité de 10%. L'effet des actions est dit* ***stochastique****.*
3. *Troisièmement, on peut considérer que si l'agent est passé par l'état $e_{(2,2)}$, le modèle de transition se comporte comme au premièrement, sinon, comme au deuxièmement.*

Propriété de Markov Suivant les caractéristiques (stochastique, déterministe, ...) de l'environnement modélisant la tâche d'apprentissage, les algorithmes d'apprentissage permettant de les résoudre en seront différents. Nous nous restreindrons dans notre cadre à des algorithmes adaptés aux environnements correspondants aux deux premiers de l'exemple, c'est à dire des environnement ayant la propriété de Markov.

Propriété 2.1 (Propriété de Markov)
Un système a la propriété de Markov si et seulement si

$$\forall e' \in E, P(e'_{t+1} \mid e_0, \ldots, e_t, a_t) = P(e'_{t+1} \mid e_t, a_t) \tag{2.3}$$

La fonction du modèle de transition T devient alors :

$$\begin{aligned} T &: E \times A \times E \longrightarrow [0,1] \\ T(e_t, a_t, e'_{t+1}) &= P(e'_{t+1} \mid e_t, a_t) \end{aligned} \tag{2.4}$$

L'équation 2.4 page ci-contre est à comparer avec l'équation 2.1 page 19 : la probabilité que le système soit dans l'état e' à l'interaction $t+1$, ne dépend que de l'état précédent e_t et de l'action sélectionnée par l'agent a_t et doit donc être indépendante de la succession des états précédents. On dit aussi que l'évolution du système est **indépendante du chemin suivi**.
On peut citer le jeu d'échec comme système ayant la propriété de Markov. Peu importe la suite d'actions qui ont mené l'échiquier dans l'état où il se trouve. Seuls comptent

- la position courante des pièces sur l'échiquier
- les mouvements possibles

pour analyser la situation.
Cependant, cette affirmation n'est vraie que s'il existe une façon optimale de jouer aux échecs, ce qui reste à notre connaissance une question aujourd'hui ouverte. Sinon, elle reste vraie si l'autre joueur adopte toujours le même comportement (éventuellement stochastique), c'est à dire qu'il ne modifie pas sa façon, de jouer au cours du temps en fonction des coups précédemment joués. Notamment, l'autre joueur ne doit pas modifier son comportement par apprentissage.

Exemple 2.7 *Parmi les modèles de transition donnés dans le cadre de notre exemple, seules les modèles de transition 1 et 2 peuvent être associés à un système ayant la propriété de Markov. En effet, le troisième modèle de transition tient compte de l'historique, c'est à dire tient compte du fait que l'agent soit ou non passé par un état précis.*

Pour la suite de notre présentation, nous nous placerons dans des environnements markoviens.

Notion de Stationnarité La succession des interactions entre un agent et son environnement influe sur ce dernier par deux facteurs :

- Le choix des actions par l'agent
- La succession des interactions, que l'on peut se représenter comme un écoulement du temps

Nous avons vu que les environnements ne modifiant par leur dynamique en fonction des actions de l'agent étaient appelés markoviens. Les environnements ne modifiant pas leur dynamique en fonction de la succession des interactions seront appelés **stationnaires**.

Propriété 2.2 (Système stationnaire)
Un système est stationnaire si et seulement si $\forall t \in \mathbb{N}, t' \in \mathbb{N}$,

$$T(e_t, a_t, e'_{t+1}) = T(e'_t, a_{t'}, e'_{t'+1})$$

On peut donc omettre la notion temporelle pour la fonction de transition. On note alors $T(e, a, e')$.

La plupart des travaux concernant l'apprentissage par renforcement se place dans le cadre de systèmes stationnaires, c'est à dire dont la dynamique n'évolue pas au cours du temps. Nous nous plaçons également dans ce cadre, même si comme nous le verrons, nos méthodes peuvent avoir une efficacité sur des systèmes non-stationnaires. On peut néanmoins trouver dans [Sutton and Barto, 1998] un algorithme nommé *Dyna-Q* applicable aux systèmes non-stationnaires. De plus, comme il est rappelé dans [Kaelbling and Littman, 1996], beaucoup des algorithmes d'apprentissage par renforcement pourraient avoir une validité dans un système *faiblement* non-stationnaire, étant donné qu'ils incluent un compromis entre les connaissances nouvellement acquises par l'expérience et les connaissances antérieurement acquises lors de l'apprentissage. Cependant, il n'existe pas à notre connaissance d'analyse théorique de cette affirmation, notamment sur la formalisation du terme *faiblement* non-stationnaire.

Récompense L'agent modifie donc l'état de l'environnement au travers d'actions et perçoit de celui-ci un nouvel état à chaque interaction. En plus de celui-ci, l'agent reçoit une valeur lui permettant d'évaluer le bénéfice ou l'inconvénient d'être dans le nouvel état qu'il vient de percevoir. Il peut ainsi comparer numériquement l'avantage d'être dans tel ou tel état de l'environnement. Par analogie avec une certaine forme d'éducation, les valeurs positives peuvent être vues comme des récompenses, et les valeurs négatives comme des punitions.

Définition 2.4 (fonction de récompense)
Soit $\mathcal{R} : \psi \times E \longrightarrow \mathbb{R}$. $\mathcal{R}$ est appelée **fonction de récompense**. r_{t+1} est la récompense obtenue par l'agent après avoir sélectionné l'action a_t dans l'état e_t, l'environnement ayant évolué vers l'état e_{t+1}.

La fonction de récompense ne variant pas au cours du temps, on peut noter $r(e, a, e')$ la récompense qu'obtiendrait l'agent dans l'état $e \in E$ en sélectionnant une action $a \in A(e)$ et en se retrouvant dans l'état e'.

Rappelons que l'information qui est donnée par le biais de la récompense concerne exclusivement l'état en cours. La récompense ne donne aucune indication sur le fait que ce soit précisemment l'action qui vient d'être effectuée qui est la source de la récompense. La source de la récompense peut très bien être une autre action plus éloignée dans le temps où plus généralement, une suite d'actions.

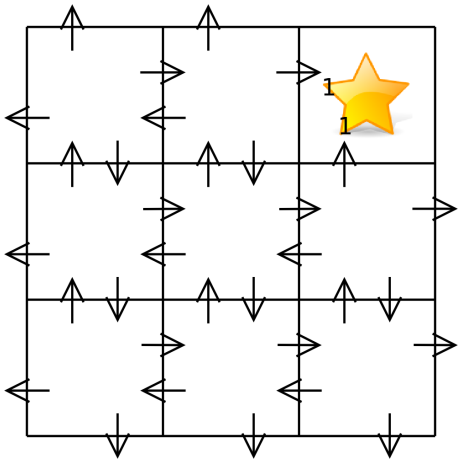


FIG. 2.5 – Fonction de récompense $\mathcal{R} : E \times A \rightarrow \mathbb{R}$ pour l'exemple du monde grille. Les flèches non étiquetées ont une valeur de 0.

Exemple 2.8 *Dans notre monde grille, nous pouvons définir la fonction de récompense suivante :*

$$\begin{cases} \bullet\ r(e,a,e') = 0,\ \forall (e,a), e' \in \psi \times E, \\ (e,a) \notin \{(e_{(2,1)}, \textit{Déplacement Nord}), (e_{(1,2)}, \textit{Déplacement Est})\} \\ \bullet\ r(e_{(2,1)}, \textit{Déplacement Nord}, e_{(2,2)}) = r(e_{(1,2)}, \textit{Déplacement Est}, e_{(2,2)}) = 1 \end{cases}$$

L'épisode de l'exemple 2.4 doit être réécrit pour tenir compte de la succession des récompenses. $e_0 = e_{(0,0)}$, $r_0 = 0$, $a_0 = \textit{Déplacement Nord}$, $e_1 = e_{(0,1)}$, $r_1 = 0$, $a_1 = \textit{Déplacement Est}$, $e_2 = e_{(1,1)}$, $r_2 = 0$, $a_2 = \textit{Déplacement Nord}$, $e_3 = e_{(2,1)}$, $r_3 = 0$, $a_3 = \textit{Déplacement Est}$, $e_4 = E_T = e_{(2,2)}$, $r_4 = 1$

Dans l'épisode décrit exemple 2.4, l'action "Déplacement Est" qui mène dans la case de récompense est bien sûr en partie la cause de la récompense, mais l'ensemble des autres actions de l'épisode également. Pourtant, les états $e_{(0,0)}$, $e_{(0,1)}$, $e_{(1,1)}$, *et* $e_{(2,1)}$ *ont une récompense nulle.*

La définition de la fonction de récompense implique que l'ensemble des récompenses $\{r_i\}$ est borné. Cet élément sera important par la suite lors de la définition de la fonction d'utilité.

Remarque 2.2 *Il n'existe pas à notre connaissance de travaux étudiant un autre ensemble (ou sous-ensemble) que* $\mathbb{R}$. *Il pourrait être néanmoins intéressant d'en envisager d'autres, notamment des ensembles totalement ou partiellement ordonnés conservant une description symbolique, y compris dans la récompense. En effet, le fait de pouvoir quantifier la qualité des états de l'environnement à l'aide d'une valeur réelle est déjà un supposé fort, puisque cette quantification entraîne toute la topologie de* $\mathbb{R}$ *: ordre total, propriétés algébriques des opérateurs additifs et multiplicatifs,... Or, dans les problèmes « réels », il est en général bien difficile de pouvoir dire qu'une situation est 2,5 fois meilleure qu'une autre !*

Processus de Décision de Markov Finalement, un environnement ainsi modélisé par un ensemble d'états, un ensemble d'actions, une fonction de transition ainsi qu'une fonction de récompense forment un Processus de Décision de Markov ou markovien.

Définition 2.5 (Processus de Décision de Markov)
Un Processus de Décision de Markov (PDM) est un tuple $\langle E, A, \psi, T, \mathcal{R} \rangle$ tel que

- $E = \{e_1, e_2, \ldots, e_n\}$ est un ensemble d'états
- $A = \{a_1, a_2, \ldots, a_n\}$ est un ensemble d'actions
- $\psi \subseteq E \times A$ est l'ensemble des couples (état,action) admissibles tels que l'action $a \in A(e)$
- $T(e, a, e') : \psi \times E \longrightarrow [0, 1]$ est l'ensemble des probabilités de transition de l'état e vers l'état e' en ayant effectué l'action a.
- $\mathcal{R}(e, a, e') : \psi \times E \longrightarrow \mathbb{R}$ est la fonction de récompense affectant une valeur numérique à chacune des transitions.

2.2.2 Comportement formalisé par une politique

Dans l'introduction, nous avons défini l'apprentissage par renforcement comme étant un mécanisme permettant de faire apprendre à un agent dans un environnement donné, un certain comportement. Avec les Processus de Décision de Markov, nous venons de formaliser la notion d'environnement. Nous avons notamment vu que l'agent agissait sur celui-ci par l'intermédiaire d'actions. Nous allons maintenant formaliser la notion de comportement de l'agent, c'est à dire la méthode grâce à laquelle celui-ci sélectionne ses actions.

Le comportement est formalisé par la notion de **politique** :

Définition 2.6 (Politique)
On considère un PDM $M\langle E, A, \psi, T, \mathcal{R} \rangle$ tel que défini définition 2.5. Soit la fonction :

$$\pi(e, a) : \psi \longrightarrow [0, 1]$$

telle que

$$\forall e \in E, \sum_{a_i \in A(e)} \pi(e, a_i) = 1$$

La fonction $\pi(e, a)$ est nommée **politique** et donne la probabilité pour l'agent de sélectionner, parmi les actions disponibles, l'action a dans l'état e. Pour chaque état, pour des raisons de cohérence, la somme des probabilités de sélection parmi les actions disponibles doit être égale à 1.

Ainsi, une politique est définie par une **fonction stochastique** permettant à l'agent de sélectionner, pour chacun des états de l'environnement, une action en faisant un tirage aléatoire parmi l'ensemble des actions admissibles et suivant une certaine distribution.

L'apprentissage dans ce formalisme est effectué par une modification de cette distribution. Les algorithmes d'apprentissage par renforcement, de façon directe ou indirecte consistent donc en une modification de $\pi(e, a)$.

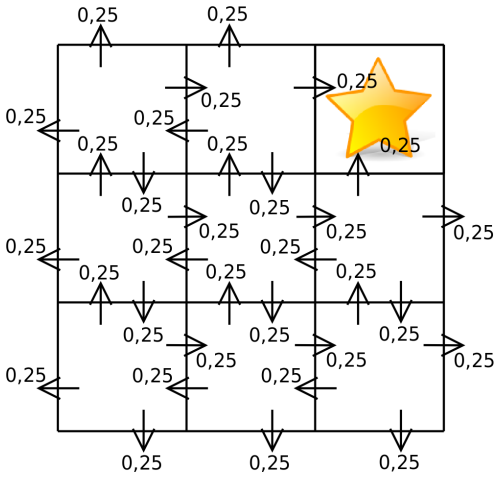
Exemple 2.9 *La figure 2.6 page suivante présente différentes politiques pour notre exemple :*

1. *Une politique aléatoire*
2. *Une politique déterministe qui consiste à parcourir tout l'échiquier en suivant les colonnes.*
3. *Une politique permettant à l'agent de se comporter de façon optimale*[3].

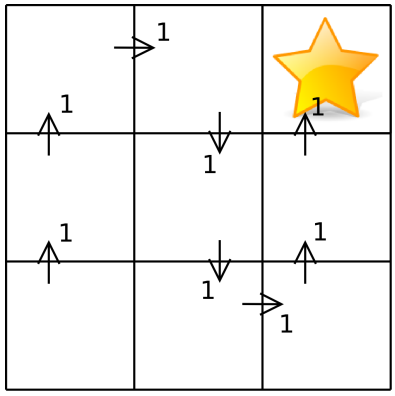
2.2.3 Mesure de gain

Nous venons de voir les modélisations de la dynamique de l'environnement ainsi que du comportement de l'agent. Précisons maintenant l'objectif de l'apprentissage. Nous avons présenté celui-ci comme étant le fait de faire adopter un certain comportement à un agent sans lui donner d'objectifs explicites, mais seulement des récompenses ou punitions quand la situation dans laquelle il se trouve est jugée adéquate ou non. L'agent doit alors apprendre un comportement lui permettant de maximiser ses récompenses et de minimiser les punitions.

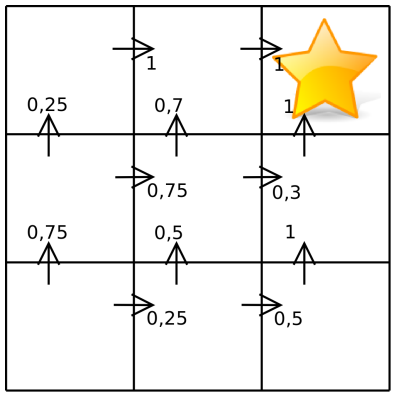
[3]La définition précise de l'optimalité d'une politique est définie définition 2.9 page 28



(a): exemple de politique aléatoire



(b): exemple d'une politique déterministe



(c): exemple d'une politique optimale

FIG. 2.6 – Exemples de politiques pour le monde grille. Les valeurs indiquent pour chacune des actions la probabilité que celle-ci soit sélectionnée. Les flèches omises ont une probabilité égale à 0.

Il nous faut maintenant préciser la notion de maximisation des récompenses ; c'est à dire que ne sera pas seulement considérée la récompense immédiate mais l'accumulation de celles-ci au cours des interactions. Ces diverses formes de mesures se nomme **gain**. Cependant, il existe plusieurs façons d'apprécier cette accumulation et celles-ci n'entraînent pas le même comportement optimal. Nous allons introduire les calculs de **gains** les plus courants : la récompense additive et la récompense temporellement pondérée.

Récompenses additives C'est la forme de mesure de gain la plus simple. Elle consiste à additionner les récompenses reçues à partir de l'état e_t jusqu'à l'état final E_T.

$$R_t = r_{t+1} + \cdots + r_T = \sum_{i=t+1}^{T} r_i$$

Pour avoir une mesure comparable aux valeurs des récompenses, on peut également utiliser la moyenne des récompenses par exemple.

$$R_t = \frac{1}{n} \sum_{i=t}^{n} r_i$$

Cette façon de considérer la récompense implique que la tâche à effectuer puisse se représenter par une série d'épisodes distincts comportant un état initial et un état final.

On peut également considérer les n prochaines récompenses obtenues par l'agent.

$$R_t = \underbrace{r_{t+1} + \cdots + r_{t+n}}_{n} = \sum_{i=t+1}^{t+n} r_i$$

Deux problèmes se posent avec cette fonction de gain. Premièrement, les récompenses au-delà de n interactions seront ignorées et deuxièmement toutes les récompenses jusqu'à n interactions sont considérées de façon identiques, quelques soit leur proximité en terme de nombre d'interactions.

Si l'agent est motivé par ce genre de récompense, on dit qu'il évolue avec un **horizon fini**. On peut trouver des travaux basés sur ce type d'environnements dans les travaux de Garcia : [Garcia and Ndiaye, 1997] par exemple.

Les tâches ne peuvent cependant pas systématiquement se découper en épisodes, comme celles n'ayant pas d'état final, mais pas seulement. Par exemple, les tâches décrivant un évènement à éviter[4] se modélisent mal en épisodes, étant donné que l'objectif est précisémment que celui-ci ne se termine jamais. Nous allons donc présenter la mesure de récompense généralement utilisée en apprentissage par renforcement et qui tient compte de ce problème.

Récompenses temporellement pondérées La récompense temporellement pondérée d'un état e_t est évaluée à partir de la somme potentiellement infinie des récompenses obtenues par l'agent à partir de e_t et au cours des interactions suivantes. Comme on ne se donne pas a priori de bornes sur le nombre d'interactions, cet ensemble est potentiellement infini.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$
$$0 \leqslant \gamma < 1$$

De plus, nous rajoutons une pondération temporelle γ. Le principe est de considérer que les récompenses proches en terme de nombre d'interactions sont plus importantes que les récompenses éloignées. On introduit donc un facteur d'estompement qui décroît en fonction du nombre des interactions ($0 \leqslant \gamma < 1$).
Un facteur γ proche de 1 amènera l'agent à considérer de façon quasiment égale les récompenses à venir, on dit qu'il est *prévoyant*. Si γ est proche de 0, à l'inverse, seules les récompenses proches compteront. On dit que l'agent est plutôt *myope*. Un agent qui évolue en considérant ce genre de récompense évolue avec un **horizon infini** ou **horizon estompé**.
Même si R_t est définie dans ce cas comme une somme d'un nombre potientiellement infini d'éléments, elle converge quand $n \to \infty$ vers un nombre réel si $\{r_i\}$ est un ensemble borné et si $0 \leqslant \gamma < 1$.
Cette mesure peut également convenir pour un PDM comprenant un état final E_T. En effet, il suffit de considérer qu'à partir de l'état E_T, une seule action est disponible et rapporte une récompense $r_{E_T} = 0$.

Exemple 2.10 *Dans notre exemple, avec l'épisode* $e_0 = e_{(0,0)}$, $a_0 = Déplacement\ Nord$, $e_1 = e_{(0,1)}$, $r_1 = 0$, $a_1 = Déplacement\ Est$, $e_2 = e_{(1,1)}$, $r_2 = 0$, $a_2 = Déplacement\ Nord$, $e_3 = e_{(2,1)}$, $r_3 = 0$, $a_3 = Déplacement\ Est$, $e_4 = E_T = e_{(2,2)}$, $r_4 = 1$ *et* $\gamma = 0,9$ *:*

$$\begin{aligned} R_0 &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 \\ &= 0 + 0,9 \times 0 + 0,9^2 \times 0 + 0,9^3 \times 1 \\ R_0 &= 0,729 \end{aligned}$$

D'autres mesures de gain existent. Notamment dans le cadre du $R - Learning$, il est étudié la moyenne des récompenses à l'infini sans estompement :

$$\rho^\pi = \lim_{n \to \infty} \frac{1}{n} \sum_{t=1}^{n} E_\pi\{r_i\}$$

[4] comme pour un des exemples les plus classiques de l'apprentissage par renforcement : le problème du pendule inversé où une barre verticalement posée sur un chariot ne doit pas tomber. Le chariot peut alors régler son sens de marche ainsi que sa vitesse. De plus, l'environnement a des bornes infranchissables, obligeant le chariot à changer de direction.

Notons que la mesure de gain pour un état donné dépend de la politique suivie à la différence de la fonction de récompense $\mathcal{R}$.
Le choix de la mesure de gain est dans le cas général un élément important pour la résolution de problèmes par apprentissage par renforcement. Pour s'en convaincre, des exemples de PDM, montrant que le comportement optimal dépend de la fonction de gain choisie, sont donnés dans [Kaelbling and Littman, 1996].

A chacun des états $e \in E$ de l'environnement, on peut donc associer, en fonction d'une politique donnée, une mesure de gain. L'ensemble des états $e \in E$ associé à l'ensemble de ces valeurs réelles constituent une fonction nommée *fonction d'utilité*.

2.2.4 Fonction d'utilité

Nous avons vu la notion de politique (définition 2.6 page 23) qui permettait à un agent de déterminer quelle action choisir en fonction de l'état où il se trouve. De plus, nous avons vu des méthodes, avec les notions de gain, pour évaluer l'ensemble des récompenses obtenues à partie d'un état donné. Il est clair que les gains accumulés dépendent de la politique suivie par l'agent. Voyons donc le lien fonctionnel qui existe entre une politique et la mesure de gain impliquée par celle-ci dans un environnement donné.

Définition 2.7 (Fonction d'utilité)
La **fonction d'utilité** V d'un état e est la valeur de gain espérée à partir de l'état e et suivant une politique π est définie par :

$$V^\pi : E \longrightarrow \mathbb{R},\ V^\pi(e) = E_\pi\{R_t | e_t = e\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid e_t = e\right\} \tag{2.5}$$

On peut également définir la **fonction de qualité** Q^π définissant la valeur de gain espérée à partir de l'état e et en effectuant l'action a puis suivant une politique π :

$$\begin{aligned} Q^\pi : \psi \longrightarrow \mathbb{R},\ Q^\pi(e,a) &= E_\pi\{R_t | e_t = e, a_t = a\} \\ &= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid e_t = e, a_t = a\right\} \end{aligned} \tag{2.6}$$

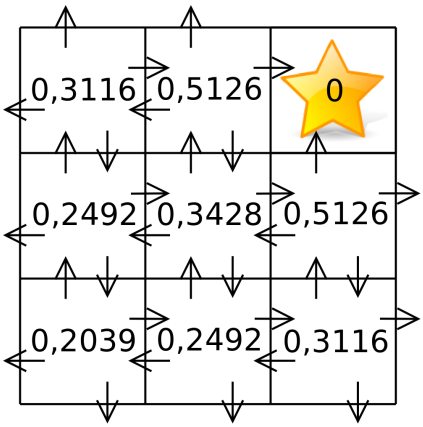
E_π représente l'espérance de gain en suivant une politique π. En effet, le résultat des actions étant stochastique et non déterministe dans le cas général, une même politique peut amener dans des états différents et donc rapporter des récompenses différentes. De plus, la politique π comme nous l'avons vu peut elle-même être stochastique. Ainsi, du point de vue de l'agent, il ne peut y avoir qu'une espérance de gain.
La différence entre ces deux fonctions peut ne pas paraître évidente au premier abord, mais nous verrons plus loin que la fonction $Q(e,a)$ permettra un apprentissage et la production d'une politique sans que l'agent ne dispose de de modèle de son environnement.

Equation de Bellman Une des propriétés fondamentales de la fonction d'utilité V^π est qu'elle est exprimable de façon récursive, définition connue sous le nom d'équation de Bellman pour V^π. Partons de la définition de la fonction d'utilité :

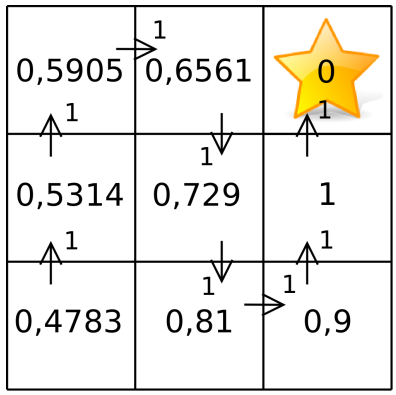
$$\begin{aligned} V^\pi(e) &= E_\pi\left\{R_t | e_t = e\right\} \\ &= E_\pi\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | e_t = e\right\} \\ &= \sum_{a \in A(e)} \pi(e,a) \sum_{e' \in E} T(e,a,e')\left[\mathcal{R}(e,a,e') + \gamma E_\pi\left\{r_{t+2} + \gamma r_{t+3} + \ldots | e_{t+1} = e'\right\}\right] \\ V^\pi(e) &= \sum_{a \in A(e)} \pi(e,a) \sum_{e' \in E} T(e,a,e')\left[\mathcal{R}(e,a,e') + \gamma V^\pi(e')\right] \end{aligned} \tag{2.7}$$

L'équation 2.7 donne donc une définition récursive de la fonction d'utilité $V^\pi(e)$. Il faut noter que outre la politique suivie $\pi(e,a)$, celle-ci utilise la fonction de transition $T(e,a,e')$ ainsi que la

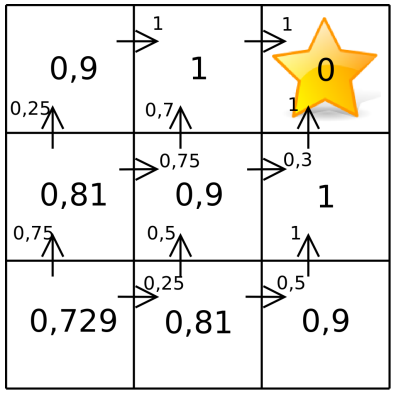
fonction de récompense $\mathcal{R}(e, a, e')$. La dynamique de l'environnement doit donc être complètement connue pour se servir de cette définition.
Ainsi, si l'on suppose inconnue la fonction V et connus la politique π, la fonction de transition T, la fonction de récompense $\mathcal{R}$ et le facteur d'estompement γ, appliquée à l'ensemble des $|E|$ états du système, l'équation 2.7 forme un système de $|E|$ équations à $|E|$ inconnues.



(a): $V^{\pi}(e)$ pour une politique aléatoire, toutes les actions sont sélectionnées de façon équiprobable



(b): $V^{\pi}(e)$ pour une politique déterministe



(c): $V^{\pi}(e)$ pour une politique optimale

FIG. 2.7 – Exemples de fonctions d'utilité $V^{\pi}(e)$ pour les politiques définies figure 2.6 page 24

Exemple 2.11 *La figure 2.7 donne les valeurs de la fonction $V^{\pi}(e) : E \rightarrow \mathbb{R}$, pour les politiques présentées figure 2.6 page 24. La méthode pour calculer la fonction est donnée plus loin par l'algorithme 2.3 page 32.*

2.2.5 Ordre sur les politiques

Dans le cadre d'un PDM, nous pouvons maintenant comparer les politiques en fonction des gains qu'elle rapporte, c'est à dire en fonction des valeurs de $V^{\pi}(e)$. Définissons donc l'opérateur d'inégalité suivant :

Définition 2.8 (Ordre sur les Politiques)

$$\pi(e, a) \geq_V \pi'(e, a) \iff V^{\pi}(e) \geq V^{\pi'}(e), \forall e \in E$$

Une politique $\pi(e, a)$ est donc supérieure à une politique $\pi'(e, a)$ si pour chacun des états, les récompenses espérées en suivant π sont supérieures ou égales aux récompenses espérées en suivant π'. De même, on définit un opérateur d'équivalence entre politiques :

$$\pi(e, a) \equiv_V \pi'(e, a) \iff V^{\pi}(e) = V^{\pi'}(e), \forall e \in E$$

Nous pouvons donc comparer la qualité de deux politiques en nous basant sur les fonctions de d'utilité engendrées par chacune d'elles. Nous allons maintenant qualifier les politiques qui engendrent les meilleures fonctions d'utilité, c'est à dire les politiques qui impliquent l'espérance de gains la plus élevée.

Définition 2.9 (politique optimale)
Soit Π, l'ensemble des politiques $\pi(e,a)$ muni des opérateurs $\geq_V$, $\equiv_V$. Π est un ordre partiel et il existe au moins une politique supérieure à toutes les autres au sens de $\geq_V$. L'ensemble de ces plus grands éléments de Π est noté Π^*. Les politiques $\pi \in \Pi^*$ sont notées π^* et sont caractérisées par :

$$\pi^* \in \Pi^* \iff \forall \pi \in \Pi, \pi^* \geq_V \pi$$

L'ensemble de ces politiques π^* sont appelées **politiques optimales**.

Elles sont en effet optimales dans le sens où il n'existe pas de meilleures politiques en accord avec la maximisation de la fonction de gain R.

Propriété 2.3 (V^*, Q^*)
Les fonctions d'utilité et de qualité pour l'ensemble des politiques optimales partagent les mêmes valeurs.

$$\forall \pi_1, \pi_2 \in \Pi^*, \forall e \in E, V^{\pi_1}(e) = V^{\pi_2}(e)$$

et

$$\forall e \in E, \forall a \in A, Q^{\pi_1}(e,a) = Q^{\pi_2}(e,a)$$

On note alors la fonction d'utilité pour les politiques optimales V^* et la fonction de qualité Q^*.

Preuve 2.1
Soit $\pi_1, \pi_2 \in \Pi^*$.
$\forall e \in E, V^{\pi_1}(e) \geq V^{\pi_2}(e)$ et $V^{\pi_2}(e) \geq V^{\pi_1}(e)$ donc $\forall \pi_1, \pi_2 \in \Pi^*, \pi_1 \equiv_V \pi_2$.

□

2.2.6 Politiques optimales et fonction d'utilité

Nous allons maintenant montrer comment, à partir de la fonction d'utilité pour les politiques optimales, comment en déduire une politique optimale.
Reprenons la définition récursive de Bellman de la fonction d'utilité (équation 2.7) qui est également valable pour les politiques optimales.

$$V^*(e) = \sum_{a \in A(e)} \pi^*(e,a) \sum_{e' \in E} T(e,a,e') \Big[\mathcal{R}(e,a,e') + \gamma V^*(e') \Big] \tag{2.8}$$

L'équation ci-dessus permet de définir les politiques optimales en fonction de la fonction d'utilité optimale. En effet, pour que $V^*(e)$ soit optimale, il ne faut pas qu'il existe une politique, c'est à dire une fonction stochastique de sélection des actions, qui produise une fonction d'utilité supérieure, considérant la fonction de gain.
Ainsi, comme décrit algorithme 2.1 page ci-contre, une politique optimale doit sélectionner en étant dans l'état e, parmi les actions $A(e)$ qui produisent d'un point de vue probabiliste (la fonction $T : A \times E \times A \to [0,1]$ n'étant pas nécessairement déterministe), une action $a \in A(e)$ maximisant le terme :

$$\sum_{e' \in E} T(e,a,e') \Big[\mathcal{R}(e,a,e') + \gamma V^*(e') \Big]$$

Cette action est alors optimale, compte-tenu de la fonction de transition $T : E \times A \times E \to [0,1]$, de la fonction de récompense $\mathcal{R} : E \times A \times E \to \mathbb{R}$, du facteur d'estompement γ et des valeurs de la fonction d'utilité optimales des états accessibles e' à partir de e.

On a alors

$$V^*(e) = \max_{a \in A(e)} \sum_{e' \in E} T(e,a,e') \Big[\mathcal{R}(e,a,e') + \gamma V^*(e') \Big] \tag{2.9}$$

Données :
- Un PDM $M\langle E, A, \psi, T, \mathcal{R}\rangle$
- La fonction d'utilité des politiques optimales pour M, $V^*_M : E \times A \to \mathbb{R}$
- $0 \leq \gamma \leq 1$, le paramètre de dégressivité

Résultat :
- $\pi^*_M : E \times A \to [0, 1]$, une politique optimale pour M

début
  pour $e \in E$ **faire**
    $max = \max_{a_i \in A(e)} \sum_{e' \in E} T(e, a_i, e')\big[\mathcal{R}(e, a_i, e') + \gamma V^*(e')\big]$;
    pour $a \in A(e)$ **faire**
      $Q(e, a) = \sum_{e' \in E} T(e, a, e')\big[\mathcal{R}(e, a, e') + \gamma V^*(e')\big]$;
      si $Q(e, a) = max$ **alors** $\pi^*(e, a) \leftarrow v$ avec $1 \geq v \geq 0$;
      sinon $\pi^*(e, a) \leftarrow 0$;
    fin
  fin
fin

Algorithme 2.1: Production d'une politique optimale pour un PDM à partir de V^*

Reprenons l'équation 2.6 page 26 donnant la définition de la fonction de qualité appliquée aux politiques optimales et montrons le rapport entre la fonction de qualité des politiques optimales $Q^*(e, a)$ et la fonction d'utilité des politiques optimales $V^*(e, a)$.

$$\begin{aligned}
Q^*(e, a) &= E_{\pi^*}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid e_t = e, a_t = a\right\} \\
&= E_{\pi^*}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid e_t = e, a_t = a\right\} \\
&= E_{\pi^*}\left\{r_{t+1} + \gamma V^*(e_{t+1}) \mid e_t = e, a_t = a\right\} \\
&= \sum_{e' \in E} T(e, a, e') \times [\mathcal{R}(e, a, e') + \gamma V^*(e')] \\
\max_{a \in A(e)} Q^*(e, a) &= \max_{a \in A(e)} \sum_{e' \in E} T(e, a, e') \times [\mathcal{R}(e, a, e') + \gamma V^*(e')] \\
\max_{a \in A(e)} Q^*(e, a) &= V^*(e) \qquad (2.10)
\end{aligned}$$

L'équation 2.10 est une équation centrale pour l'apprentissage par renforcement. Elle a été notamment utilisée dans [Watkins and Dayan, 1992]. L'apprentissage de la fonction d'utilité Q^* donnera son nom à une méthode particulière d'apprentissage par renforcement et peut-être la plus connue : **Q-Learning** que nous détaillerons dans la partie suivante.
En effet, si l'équation 2.9 sous-tend une méthode permettant de produire une politique optimale à partir de la connaissance de $V^*(e), T, \mathcal{R}$ et γ, l'équation 2.10 nous montre que la seule connaissance de $Q^*(e, a)$ permet également de produire une politique optimale et ce, sans connaissance de la dynamique interne de l'environnement. Ceci est décrit par l'algorithme 2.2 page suivante.
C'est ce mécanisme qui permet d'insérer l'apprentissage par renforcement dans le cadre d'une conception agent, plutôt que dans un cadre strictement mathématique ; il y a une séparation entre le modèle mathématique de l'environnement et les éléments permettant de résoudre le problème de recherche de politique optimale. Ceux-ci se résument à des perceptions sous forme d'états et de récompenses dans le sens environnement → agent et à des applications d'actions dans l'autre.
De plus, la méthode est extrêmement simple (une fois la fonction $Q^*(e, a)$ connue !), puisqu'elle consiste à sélectionner l'action ayant la valeur maximale parmi les actions disponibles.
Dit autrement, on peut considérer que la connaissance de $Q^*(e, a)$ permet une sélection des actions optimales, relativement à la fonction de récompense considérée, de façon **gloutonne**. Ainsi, la valeur attribuée à chacun des couples (e, a) par la fonction Q^*, tient directement compte d'une vision à long-terme (impliquée par le facteur γ) ainsi que du caractère stochastique de l'environnement

Données :
- Un PDM $M\langle E, A, \psi, T, \mathcal{R}\rangle$ avec $T, \mathcal{R}$ inconnues
- La fonction d'utilité des politiques optimales pour M, $Q^*_M : E \times A \to \mathbb{R}$
- $0 \leq \gamma \leq 1$, le paramètre de dégressivité

Résultat :
- $\pi^*_M : E \times A \to [0, 1]$, une politique optimale pour M

début
  pour $e \in E$ **faire**
    $max = \max_{a_i \in A(e)} Q(e, a_i)$;
    pour $a \in A(e)$ **faire**
      si $Q(e, a) = max$ **alors** $\pi^*(e, a) \leftarrow v$ avec $1 \geq v \geq 0$;
      sinon $\pi^*(e, a) \leftarrow 0$;
    fin
  fin
fin

Algorithme 2.2: Production d'une politique optimale pour un PDM à partir de Q^*

en ce qui concerne l'accumulation des récompenses.
Pour un même état, plusieurs actions peuvent être optimales. On peut ainsi définir l'ensemble des actions optimales d'un état.

Notation 2.5 (ensemble des actions optimales d'un état $A^*(e)$)
Soit un Processus de Décision Markovien $M\langle E, A, \psi, T, \mathcal{R}\rangle$ et un état $e \in E$. $A^*(e) \subseteq A(e)$ est défini par :

$$\left\{ a \in A(e) \text{ telles que } Q^*(e, a) = \max_{a_i \in A(e)} Q^*(e, a_i) \right\}$$

$A^*(e)$ est l'**ensemble des actions optimales** pour l'état e dans le PDM M.

Une politique optimale est une politique qui sélectionne dans chaque état une action optimale pour celui-ci. Ainsi, toutes les politiques optimales ont la propriété suivante :

Propriété 2.4
Soit un Processus de Décision Markovien $M\langle E, A, \psi, T, \mathcal{R}\rangle$. $\pi(e, a)$ est une politique optimale pour M si et seulement si $\forall e \in E$,

$$\begin{cases} \pi(e, a) \geq 0 \text{ si } a \in A^*(e) \\ \pi(e, a) = 0 \text{ sinon} \end{cases}$$

Une action optimale peut ne jamais être sélectionnée ($\pi(e, a) = 0$ est possible même si $Q^*(e, a) = \max_{a_i \in A(e)} Q^*(e, a_i)$), mais le fait que π soit une politique impose que la somme des probabilités de sélection des actions optimales soit égale à 1 (définition 2.6). Au moins un action optimale par état a donc une probabilité non nulle d'être sélectionnée.

Exemple 2.12 *La figure 2.8 montre les valeurs de la fonction d'utilité $V^*(e)$ et $Q^*(e, a)$ pour notre exemple du monde grille. On peut notamment voir l'llustration du fait que la connaissance de ces fonctions permet de trouver une politique optimale : en sélectionnant dans chaque état e l'action amenant dans l'état ayant la valeur la plus élevée (en utilisant $V^*(e)$), en sélectionnant l'action avec la valeur la plus élevée (en utilisant $Q^*(e, a)$).*
Par exemple, considérons la case centrale. Pour trouver une action optimale en utilisant la fonction d'utilité $V(e)$, il faut considérer les 4 actions possibles : "Déplacement Nord", "Déplacement Est", "Déplacement Sud", "Déplacement Ouest". Ensuite, il faut trouver grâce à la dynamique de l'environnement, l'ensemble des états dans lesquels l'agent se retrouve après avoir sélectionner chacune de ces actions (ici, l'environnement est déterministe, ce qui facilite la tâche), ainsi que les récompenses associées. Finalement, il faut sélectionner une des actions maximales, selon l'équation

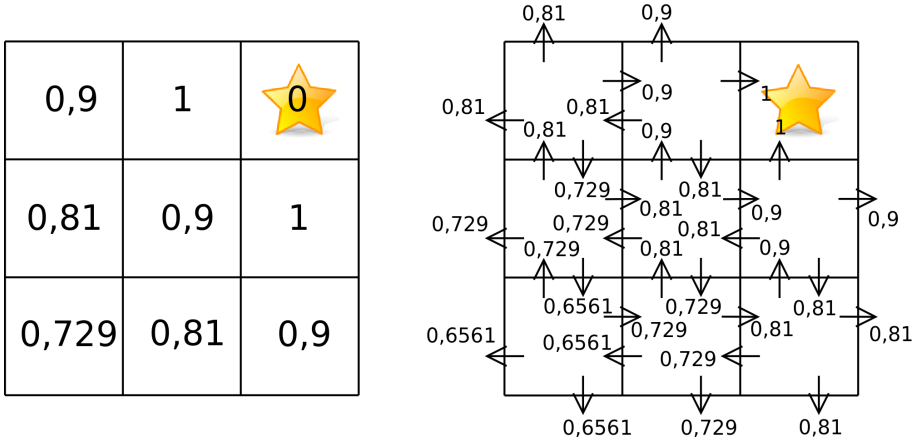


FIG. 2.8 – Valeurs de $V^*(e)$ et de $Q^*(e,a)$ pour l'exemple du monde grille avec $\gamma = 0,9$

2.9.
Avec la fonction de qualité $Q(e,a)$, il suffit de considérer les 4 mêmes actions et de sélectionner une de celles qui ont la valeur maximale, ici 0,9 correspondant à "Déplacement Nord" et "Déplacement Est".

L'objectif de l'apprentissage de l'apprentissage par renforcement peut se définir comme la recherche d'une de ces politiques optimales en un nombre d'interactions minimal. L'implication réciproque entre la connaissance de la fonction de qualité $Q^*(e,a)$ et les politiques optimales est en général à la base des algorithmes d'apprentissage par renforcement, comme nous le verrons partie 2.4 ; le problème étant évidemment, qu'au début de l'apprentissage, la fonction $Q^*(e,a)$ est inconnue et qu'il faut l'approximer durant les interactions.
Nous allons voir dans la partie suivante, un exemple classique de calcul exact d'une politique optimale par programmation dynamique.

2.3 Programmation Dynamique

La résolution des Processus de Décision Markoviens tels que nous venons de les décrire peut se faire de manière exacte par les algorithmes dits de Programmation Dynamique. Ces méthodes développées à la fin des années 50, notamment dans [Bellman, 1957] ne sont applicables que sur de petits problèmes compte-tenu de leur complexité algorithmique ainsi que du présupposé qu'ils exigent de connaissance parfaite de la dynamique de l'environnement. Cependant, nous reproduisons ici ces idées car elles restent importantes d'un point de vue théorique. En effet, ces méthodes s'accompagnent de preuves de convergence, bien que celles-ci soient garanties par une omniscience de l'agent. Enfin, elles donnent un point de comparaison pour les algorithmes d'apprentissage par renforcement.

Evalutation itérative d'une politique

Bien que l'on dispose de méthodes pour résoudre un tel système, le coût en complexité par des méthodes classiques les rendent inutilisables sur les problèmes d'apprentissage par renforcement que nous nous proposons de traiter ici.
Il existe cependant un algorithme nommé *évaluation itérative d'une politique*, permettant d'approcher aussi prés qu'on le souhaite vers les valeurs de la fonction $V^\pi(e)$ sans avoir à résoudre un tel système.

Principe Le principe est d'initialiser pour l'étape $k = 0$ avec une valeur quelconque chacune des valeurs de la fonction $V(e) : E \to \mathbb{R}$. Puis, à chaque étape $k > 0$, de les mettre à jour selon l'équation de Bellman 2.7, jusqu'à obtenir la précision souhaitée. La formule de mise à jour est alors :

$$V_k(e) \leftarrow \sum_{a \in A(e)} \pi(e,a) \sum_{e' \in E} T(e,a,e')[\mathcal{R}(e,a,e') + \gamma V^{\pi}(e')] \tag{2.11}$$

Données :
- $\langle E, A, T, \psi, \mathcal{R} \rangle$, un Processus de Décision de Markov
- $\pi(e,a)$, $(e,a) \in \psi$ une politique à évaluer
- $0 \leq \gamma \leq 1$, le paramètre de dégressivité
- $\epsilon > 0$, un critère d'arrêt

Résultat :
- $V(e) \approx V^{\pi}(e)$

début
  Initialiser $V(e) = v_0 \in \mathbb{R}$, $\forall e \in E$;
  répéter
    $\Delta \leftarrow 0$;
    pour *chaque* $e \in E$ **faire**
      $v \leftarrow V(e)$;
      $V(e) \leftarrow \sum_{a \in A(e)} \pi(e,a) \sum_{e' \in E} T(e,a,e')[\mathcal{R}(e,a,e') + \gamma V(e')]$;
      $\Delta \leftarrow max(\Delta, |v - V(e)|)$;
    fin
  jusqu'à $\Delta < \epsilon$;
fin

Algorithme 2.3: Evaluation itérative d'une politique

Algorithme L'algorithme est présenté algorithme 2.3. Pratiquement, il peut s'implémenter grâce à l'utilisation de deux tableaux, contenant à chaque itération k, respectivement les anciennes et les nouvelles valeurs de $V_k(e)$. Chaque itération appliquant l'équation de Bellman à l'ensemble des $V_k(e)$.

Convergence et terminaison Sous la condition que $\alpha < 1$, la suite des valeurs de $V_k(e)$ va converger vers $V^{\pi}(e)$ quand $k \to \infty$. Mathématiquement, il s'agit d'une contraction de suite vers un point fixe ; le point fixe étant ici $V^{\pi}(e)$. La démonstration est disponible pour le lecteur intéressé dans [Bellman, 1957].
Pour le principe de convergence, il faut constater que pour chaque étape, quelles que soient les $V_0(e)$ initiales :

$$\max_{e \in E} |V_{k+1}(e) - V_k(e)| \leq \gamma \max_{e \in E} |V_k(e) - V_{k-1}(e)| \tag{2.12}$$

Ainsi, si $\alpha < 1$, $\max_{e \in E} |V_k(e) - V_{k+1}(e)|$ converge vers 0 quand $k \to \infty$.
Si $\max_{e \in E} |V_k(e) - V_{k+1}(e)|$ est considérée comme la correction apportée à la valeur d'utilité $V(e)$, alors cette correction tend vers 0 au fur et à mesure des itérations.

Typiquement, le critère d'arrêt de l'algorithme sera une fonction de la valeur de $\max_{s \in E} |V_k(e) - V_{k+1}|$ voir [Even-Dar and Mansour, 2003].

Nous disposons donc d'une méthode pour évaluer une politique en fonction des récompenses qu'elle permet d'obtenir, c'est à dire la fonction d'utilité. Nous allons maintenant voir que cette fonction d'utilité va nous permettre de hiérarchiser les politiques, de définir l'existence de politiques optimales, et finalement de proposer un algorithme permettant d'approcher cette politique optimale. Bien sûr nous restons ici dans le cadre théorique d'une connaissance parfaite de la dynamique de l'environnement.

2.3.1 Amélioration d'une politique

Poursuivons sur la méthode classique en présentant un algorithme permettant de d'améliorer une politique.

Principe Le principe de l'amélioration de la politique est à partir d'une politique $\pi(e,a)$, d'obtenir une politique strictement meilleur en cherchant à maximiser par une méthode gloutonne, la fonction d'utilité $V(e)$.

Notation 2.6
Soit un PDM $\langle E, A, T, \psi, \mathcal{R}\rangle$, $\pi(e,a) \in \Pi$ une politique sur celui-ci et $Q^\pi(e,a)$, sa fonction de qualité. Définissons à partir de la politique $\pi(e,a)$ une politique $\pi_Q(e,a)$ telle que :

$$\begin{cases} \pi_Q(e,a) \geq 0 \text{ si } a = \arg \max\limits_{a \in A(e)} Q^\pi(e,a) \\ \pi_Q(e,a) = 0 \text{ sinon} \end{cases}$$

Théorème 2.1 (amélioration d'une politique) *Soit un PDM $\langle E, A, T, \psi, \mathcal{R}\rangle$, $\pi(e,a) \in \Pi$ une politique sur celui-ci et $Q^\pi(e,a)$, sa fonction de qualité. Soit $\pi'(e,a) = \pi_Q(e,a)$.*
Alors $\pi'(e,a) \geq \pi(e,a)$ et
si $\pi'(e,a) \equiv_v \pi(e,a)$ alors $\pi'(e,a) \in \Pi^*$ et $\pi(e,a) \in \Pi^*$.

Autrement dit, pour chaque état $e \in E$, l'action choisie pour la nouvelle politique $\pi'(e,a)$ est l'action qui amène dans l'état rapportant la récompense espérée la plus élevée en utilisant la politique $\pi(e,a)$. Il faut aussi noter que la politique améliorée $\pi'(e,a)$ est déterministe. Une seule action est choisie systématiquement quelque soit l'état e.
Comme nous allons le montrer après, ceci améliore $\pi(e,a)$ dans le sens où $\pi'(e,a) \geq \pi(e,a)$.
Mieux, si $\pi'(e,a) \equiv_v \pi(e,a)$, c'est à dire si les deux politiques ont des fonctions d'utilité identiques (pas au sens où elles ont les mêmes probabilités d'action en fonction des états), alors $\pi(e,a)$ et $\pi'(e,a)$ sont optimales.
Reproduisons ici la preuve donnée dans [Sutton and Barto, 1998] :

Preuve 2.2
Pour chaque état $e \in E$, l'action a_1 sélectionnée de façon déterministe pour $\pi'(e,a)$ est telle que $a_1 = arg \max\limits_{a \in A(e)} Q^\pi(e,a)$ donc $V^\pi(e) \leq Q^\pi(e,a_1)$.

$$\begin{aligned}
V^\pi(e) &\leq Q^\pi(e,a_1) \\
&= E_{\pi'}\{r_{t+1} + \gamma V^\pi(e_{t+1}) | e_t = e\} \\
&\leq E_{\pi'}\left\{r_{t+1} + \gamma Q^\pi(e_{t+1}, \max_{a \in A(e)} Q^\pi(e,a)) | e_t = e\right\} \\
&= E_{\pi'}\{r_{t+1} + \gamma E_{\pi'}\{r_{t+2} + \gamma V^\pi(e_{t+2})\} | e_t = e\} \\
&= E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 V^\pi(e_{t+2}) | e_t = e\} \\
&\leq E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^\pi(e_{t+3}) | e_t = e\} \\
&\vdots \\
&\leq E_{\pi'}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots | e_t = e\} \\
&= V^{\pi'}(e) \\
V^\pi(e) &\leq V^{\pi'}(e) \\
\pi(e,a) &\leq \pi'(e,a)
\end{aligned}$$

De plus, si $\forall e \in E, V^\pi(e) = V^{\pi'}(e)$, on a

$$V^{\pi'}(e) = \max_{a \in A(e)} E\left\{r_{t+1} + \gamma V^{\pi'}(e_t + 1) | e_t = e, a_t = a\right\} \tag{2.13}$$

$$= \max_{a \in A(e)} \sum_{e' \in E} T(e,a,e')[\mathcal{R}(e,a,e') + \gamma V^{\pi'}(e')] \tag{2.14}$$

L'équation 2.14 correspond exactement à une solution de l'équation d'optimalité de Bellman 2.8. On a donc $\pi(e,a) \in \Pi^*$ et $\pi^*(e,a) \in \Pi^*$.

□

2.3.2 Itération sur les politiques

Principe Nous disposons donc d'une méthode permettant d'évaluer une politique et une autre permettant d'améliorer strictement une politique. Ces deux méthodes sont combinées dans un algorithme appelé **itération sur les politiques**, permettant ainsi de trouver une politique optimale pour un PDM dont on connaît la dynamique.

Données :
- $\langle E, A, T, \psi, \mathcal{R} \rangle$, un Processus de Décision de Markov
- $0 \leq \gamma \leq 1$, le paramètre de dégressivité
- $\epsilon > 0$, un critère d'arrêt

Résultat :
- $\pi(e,a) \in \Pi^*$, une politique optimale

début
  Initialisation de la politique
  Initialiser $V(e) = v_0 \in \mathbb{R}$, $\forall e \in E$;
  Initialiser $\pi(e,a)$ arbitrairement $\forall (e,a) \in \psi$;
  $\pi(e,a)$ *stable* $\leftarrow$ Faux;
  tant que $\neg\pi(e,a)$ *stable* **faire**
    Evaluation de la politique
    répéter
      $\Delta \leftarrow 0$;
      pour *chaque* $e \in E$ **faire**
        $v \leftarrow V(e)$;
        $V(e) \leftarrow \sum_{a \in A(e)} \pi(e,a) \sum_{e' \in E} T(e,a,e')[\mathcal{R}(e,a,e') + \gamma V(e')]$;
        $\Delta \leftarrow max(\Delta, |v - V(e)|)$;
      fin
    jusqu'à $\Delta < \epsilon$;
    Amélioration de la politique
    $\pi(e,a)$ *stable* $\leftarrow$Vrai;
    pour *chaque* $e \in E$ **faire**
      $a_1 \leftarrow arg \max_{a \in A(e)} \pi(e,a)$;
      $a_2 \leftarrow arg \max_{a \in A(e)} \sum_{e' \in E} T(e,a,e')[\mathcal{R}(e,a,e') + \gamma V(e')]$;
      pour *chaque* $a \in A(e)$ **faire**
        si $a = a_2$ **alors** $\pi(e,a) = 1$;
        sinon $\pi(e,a) = 0$;
        si $a_1 \neq a_2$ **alors** $\pi(e,a)$ *stable* $\leftarrow$Faux;
      fin
    fin
  fin
fin

Algorithme 2.4: Itération sur les politiques

Algorithme On part avec une politique quelconque $\pi_0(e,a)$, on l'évalue grâce à la fonction d'utilité $V_0(e)$ puis on l'améliore vers une politique strictement meilleur $\pi_1(e,a)$. On recommence ainsi jusqu'à ce qu'il n'y ait plus d'amélioration. On est alors sur d'après le théorème 2.1 que la politique ainsi obtenue est optimale. On peut noter que cet algorithme présenté algorithme 2.4 ne permet pas de trouver l'ensemble des politiques optimales.

Si cet algorithme est intéressant car il donne des méthodes de comparaison entre politique et montre l'existence de politiques optimales, il ne reste néanmoins intéressant seulement d'un point de vue théorique. En effet, premièrement il repose sur le présupposé fort que l'ensemble de la dynamique du PDM soit connu et de plus, il repose sur deux processus coûteux en terme calculatoire.

2.4 Algorithmes d'apprentissage par renforcement

2.4.1 Introduction

Nous allons, dans cette partie, présenter quelques un des algorithmes principaux d'apprentissage par renforcement : *Q-Learning*, *TD(λ)*, *Sarsa*. On pourra les retrouver dans [Sutton and Barto, 1998]. Nous proposons ceux-ci car ils sont relativement connus et permettent de montrer les éléments de variabilité dans cette famille d'algorithmes. Nous commencerons par présenter leur différence d'avec la méthode de recherche exhaustive qu'est la programmation dynamique, vue partie 2.3. Ensuite, nous détaillerons l'ensemble des concepts (compromis exploration-exploitation, politique d'exploration, formules de mise à jour, apprentissage dépendant ou indépendante de la politique, paramètres des algorithmes) variant dans les différents algorithmes. Au fur et à mesure de la présentation de ces concepts, nous expliciterons les algorithmes proprement dits.

2.4.2 Programmation dynamique et apprentissage par renforcement

Nous avons vu partie 2.3 l'algorithme de programmation dynamique consistant à évaluer une politique, améliorer cette politique grâce à cette évaluation, puis recommencer jusqu'à converger vers un politique optimale. Cette méthode a deux caractéristiques qui la rend en pratique peu utilisable : Premièrement, algorithmiquement, elle est très coûteuse : il faut estimer la politique à chaque étape. Deuxièmement, elle nécessite la connaissance du modèle de transition $T(e, a, e)$ et de récompense $\mathcal{R}(e, a, e')$. C'est à dire que la dynamique de l'environnement doit-être entièrement connue, ce qui n'est pas le cas pour de nombreux problèmes que se propose de traiter l'apprentissage machine. Si on considère une approche *agent* notamment, ce genre de pré-requis sur la connaissance de l'environnement est inenvisageable. On peut voir figure 2.9 un résumé de la comparaison entre les deux approches.

Notation 2.7 ($\hat{Q}(e, a)$)
L'estimation courante de la fonction $Q(e, a)$ se note $\hat{Q}(e, a)$.

Les algorithmes d'apprentissage par renforcement classiques et que nous allons présenter ici : *Q-Learning*, *TD(λ)*, *Sarsa*, dans leur essence, poursuivent la même démarche :

1. Partir d'une fonction de qualité $\hat{Q}_0(e, a)$ arbitraire.
2. Utiliser une politique basée sur l'estimation courante de la fonction de qualité $\hat{Q}(e, a)$ tout en améliorant conjointement celle-ci à l'aide des récompenses effectivement obtenue et d'une prédiction sur les récompenses à long terme à venir. L'apprentissage amène $\hat{Q}(e, a)$ à converger vers $Q^*(e, a)$.

Ainsi, par le biais de l'amélioration de l'estimation de la fonction de qualité, on retrouve le même processus d'évaluation – amélioration de la politique.

Si ce processus est similaire, il existe également des différences essentielles séparant ces familles d'algorithmes tant sur la gamme des problèmes traités que sur les mécanismes algorithmiques concrets. Ces différences tentent de répondre aux deux problèmes que nous venons de mentionner : complexité algorithmique et connaissance de la dynamique de l'environnement.

Apprentissage en ligne et incrémental La première caractéristique des algorithmes d'apprentissage par renforcement est qu'ils sont généralement prévus pour un apprentissage *en ligne* et *incrémental*, c'est à dire qu'il n'y a pas de séparation entre une phase d'apprentissage et une phase d'utilisation de l'apprentissage. L'incrémentalité vient du fait suivant. L'objectif est bien de converger vers un comportement optimal, cependant, les comportements, même sous-optimaux, restent utilisables à tout moment, en l'état actuel de l'apprentissage. Ainsi, ils sont adaptés à une conception *agent*, étant données les caractéristiquent d'autonomie qu'ils proposent.

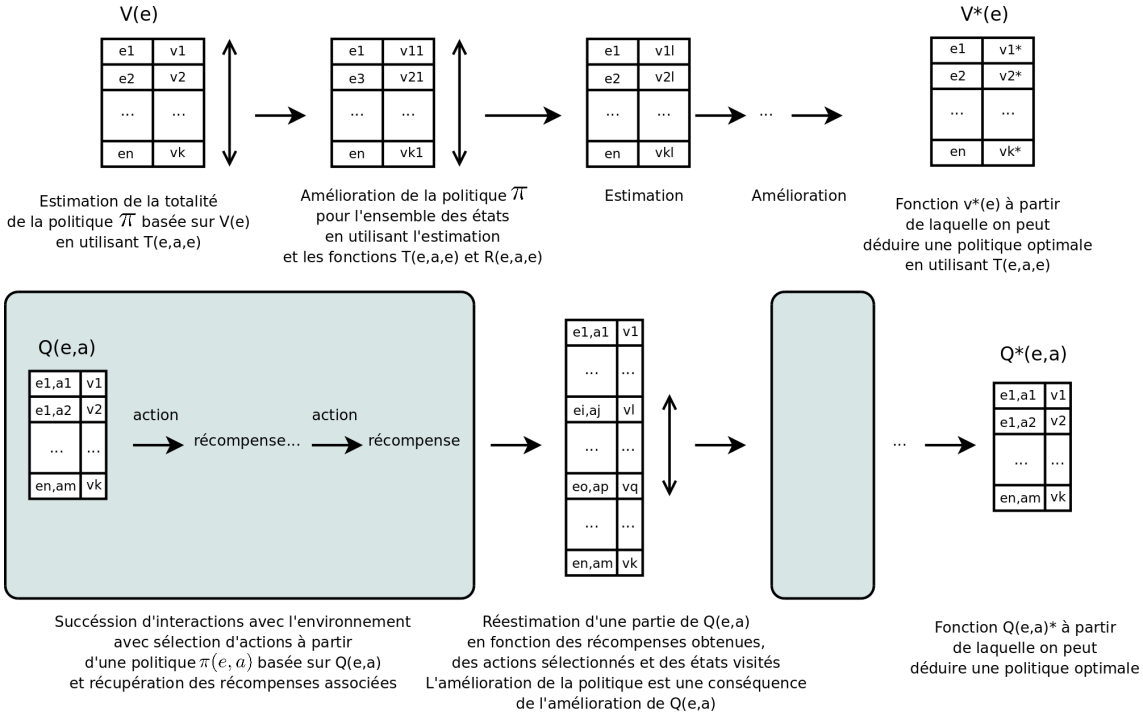


FIG. 2.9 – Comparaison entre les mécanismes d'apprentissage par renforcement et de programmation dynamique. L'apprentissage par renforcement estime la fonction $Q(e,a)$ plutôt que $(V(e)$, n'utilise pas les fonctions $T(e,a,e)$ et $R(e,a,e)$ (l'agent n'a donc pas à connaître la dynamique de l'environnement), seule une sélection de couples (e,a) sont mis à jours, ceux qui ont été rencontrés. De plus, ce ne sont pas des valeurs exactes qui sont calculées, mais des estimations basées sur l'expérience.

Fonction de qualité $Q(e,a)$ Les méthodes de programmation dynamique travaillent sur la fonction d'utilité $V(e)$. La connaissance de la fonction d'utilité $V^*(e)$ permet en effet, de manière gloutonne, de trouver une politique optimale.
Rappelons la définition de la fonction de qualité :

$$Q^\pi : E \times A \to \mathbb{R}$$
$$Q^\pi(e_t, a_t) = E_{\pi'}\{r_{t+1} + \gamma V^\pi(e_{t+1}) | e_t = e\}$$

Celle-ci associe la récompense espérée par l'agent en poursuivant la politique π et en ayant sélectionné l'action a_t dans l'état e_t. En fait, il ne s'agit plus de mesurer la qualité d'un état, mais la qualité des actions dans chacun des états.
De la même manière que avec la fonction $V^*(e)$, la connaissance de $Q^*(e,a)$ permet de trouver une politique optimale π^* de façon gloutonne en sélectionnant à chaque état e, l'action ayant la valeur $Q^*(e,a)$ la plus élevée :

$$\begin{cases} \pi^*(e,a) \geq 0 \text{ si } a = \underset{a}{max}\ Q^*(e,a) \\ \pi^*(e,a) = 0 \text{ sinon} \end{cases}$$

La recherche de la fonction $Q^*(e,a)$ à la place de la recherche de la fonction $V^*(e)$ constitue un des aspects particuliers de l'apprentissage par renforcement[5]. Ceci permet, une fois cette fonction connue, de s'affranchir de la nécessité de connaître le modèle de l'environnement pour déduire une politique optimale.

[5]cette modification d'objectif est à l'origine du nom du célèbre algorithme *Q-Learning*, apprentissage de la fonction Q.

Estimation en fonction des interactions avec l'environnement Comme l'agent n'a pas accès à la dynamique de son environnement, il ne peut réévaluer sa politique que par rapport aux récompenses qu'il obtient effectivement, il ne peut donc pas réestimer systématiquement l'ensemble des couples $Q(e,a)$ existant pour son environnement comme dans le cadre de la programmation dynamique, mais seulement ceux concernés par les récompenses effectivement obtenues.
Ce qui peut être un désavantage dû au manque de certaines informations devient un avantage de réduction de complexité algorithmique si on considère les éléments suivants. Il est important de noter que l'objectif de l'agent n'est pas de connaître la fonction $Q^*(e,a)$. Ce n'est qu'un moyen. L'objectif est de maximiser le flux des récompenses. Ainsi, une approximation raisonnable de la fonction $Q^*(e,a)$ peut-être suffisante. En particulier, il n'est pas essentiel de connaître parfaitement la fonction pour les états de l'environnement et les actions ne rapportant que très peu de récompenses, l'agent aura précisément comme objectif de les éviter. De même, il n'est pas « grave » de ne pas connaître celle-ci pour les états très peu rencontrés par l'agent, l'erreur n'entraînant pas, statistiquement, de pertes importantes. Ainsi, les algorithmes d'apprentissage par renforcement vont privilégier, par leur fonctionnement, l'approximation de la fonction $Q^*(e,a)$ pour les états les plus visités par l'agent.

2.4.3 Q-learning - Techniques de mises à jour de $\hat{Q}(e,a)$

Comme suggéré figure 2.9, une des différences entre la programmation dynamique et l'apprentissage par renforcement est le fait que l'hypothèse courante $\hat{Q}(e,a)$ n'est pas réestimée pour l'ensemble des couples $(e,a) \in E \times A$, mais seulement pour les couples concernés par les interactions faites entre l'agent et l'environnement. C'est à dire que ne sont mises à jour que les valeurs des états rencontrés des actions sélectionnées durant les interactions et en ayant reçu des récompenses correspondantes.
Un des paramètres caractérisant les différents types d'algorithmes d'apprentissage par renforcement est constitué du mécanisme de mise à jour de $\hat{Q}(e,a)$. Commençons par présenter l'équation de mise à jour de l'algorithme *Q-Learning*, car c'est la plus simple et la plus connue, une fois ce mécanisme de base expliqué, nous aborderons les équations de mises à jours de *TD(λ)* pour ajouter la notion de traces d'éligibilité et de répartition de la récompense dans le temps. Ensuite, nous aborderons la notion de dépendance de l'apprentissage par rapport à la politique avec l'algorithme *Sarsa*.

Pour l'algorithme de *Q-Learning* (voir algorithme 2.5), la valeur affectée par une opération de mise à jour se fait immédiatement après chaque itération, c'est à dire après avoir sélectionné une action a_{t-1} dans l'état e_{t-1} et après avoir reçu la récompense r_t correspondant au nouvel état e_t dans lequel l'agent se retrouve. La valeur du couple $Q(e_{t-1}, a_{t-1})$ est alors modifiée comme suit :

$$\hat{Q}(e_{t-1}, a_{t-1}) \leftarrow (1-\alpha)\hat{Q}(e_{t-1}, a_{t-1}) + \alpha(r_t + \gamma \max_a \hat{Q}(e_t, a_t)) \tag{2.15}$$
$$0 < \gamma < 1 \text{ et } 0 < \alpha < 1$$

La mise à jour est donc une combinaison entre l'ancienne valeur estimée :

$$(1-\alpha)\hat{Q}(e_{t-1}, a_{t-1})$$

et la nouvelle estimation :

$$\alpha(r_t + \gamma \max_a \hat{Q}(e_t, a_t))$$

Celle-ci est elle même composée de deux éléments :

- la récompense effectivement obtenue :

$$r_t$$

- et une estimation des récompenses espérées en suivant une politique optimale :

$$\gamma \max_a \hat{Q}(e_t, a_t)$$

L'élément principal de compréhension par rapport à la convergence vers $Q^*(e,a)$ de $\hat{Q}(e,a)$ grâce aux mises à jour est que l'estimation courante $\hat{Q}(e,a)$ est composée au fur et à mesure des interactions de trois éléments :

Données :
- $\mathcal{M}\langle E, A, T, \psi, \mathcal{R}\rangle$, un Processus de Décision de Markov avec un état initial E_0 et un état final E_T
- Un agent évoluant dans $\mathcal{M}$
- $0 \leq \gamma \leq 1$, le paramètre de dégressivité
- $0 < \alpha < 1$, le paramètre de remise en cause de la connaissance

Résultat :
- $\pi^*(e, a) \in \Pi^*$, une politique optimale

début
Initialisation de la politique
Initialiser $\hat{Q}(e, a) = v_0 \in \mathbb{R}$, $\forall (e, a) \in \psi$;
pour *Chaque épisode* **faire**
$e \leftarrow E_0$;
tant que $e_t \neq E_T$ **faire**
$a =$ choisir une action en utilisant $\hat{Q}(e, a)$;
appliquer a;
$r \leftarrow$ recevoir une récompense;
$e' \leftarrow$ recevoir un nouvel état;
$\hat{Q}(e, a) \leftarrow (1 - \alpha)\hat{Q}(e, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(e', a'))$;
$e \leftarrow e'$;
fin
fin
fin

Algorithme 2.5: *Q-Learning*

- La valeur initial de la fonction $\hat{Q}_0(e, a)$, soumise à une erreur initiale.
- L'estimation de la somme estompée des récompenses futures espérées en suivant une politique optimale $\gamma \max_a \hat{Q}(e, a)$ également soumise à une erreur, puisque $\hat{Q}(e, a)$ est une estimation.
- Les récompenses r_{t_i} réellement obtenues qui ne sont, elles, pas soumises à l'erreur car elles sont fournies directement par l'environnement.

L'idée principale de convergence est donc que si l'agent fait un nombre infini d'interactions (nous verrons ci-après comment l'obtenir), pour l'ensemble des couples (e, a), la valeur initiale ainsi que l'erreur d'estimation deviendront infiniment petits devant les valeurs des récompenses effectivement obtenues.

Théorème 2.2 *(convergence de l'algorithme de* Q-Learning*) Soit un agent apprenant $\mathcal{A}g_Q$ utilisant l'algorithme de Q-Learning 2.5 pour un processus de décision markovien $\langle E, A, \psi, T, \mathcal{R}\rangle$ ayant des récompenses bornées, $\forall (e, a), r(e, a) \leq c$, $c \in \mathbb{R}$. Si la politique de $\mathcal{A}g_Q$ implique que chaque couple $(e, a) \in E \times A$ soit visité infiniment souvent alors $\forall e, a \in E \times A$, $\hat{Q}_n(e, a)$ converge vers $Q^*(e, a)$ quand $n \to \infty$.*

Pour comprendre le théorème ci-dessus, il faut comprendre ceci : soit l'erreur maximale à l'interation n, $\Delta_n = \max_{e,a} |\hat{Q}_n(e, a) - Q^*(e, a)|$ l'erreur maximale entre la fonction $\hat{Q}_n(e, a)$ et $Q^*(e, a)$, et e_Δ et a_Δ l'état et l'action pour lesquels cette erreur est maximale, $\Delta_n = |\hat{Q}_n(e_\Delta, a_\Delta) - Q^*(e_\Delta, a_\Delta)|$. Lorsque l'agent visite (e_Δ, a_Δ), Δ_n est diminué d'un facteur γ car la récompense alors obtenue est exacte. Comme tous les états sont potientiellement visités infiniment souvent, l'erreur maximale diminuera nécessairement. Une preuve complète peut se trouver dans [Mitchell, 1997]. Ainsi, il faut deux conditions que l'on retrouve en règle générale pour les preuve de convergence en apprentissage par renforcement :

- que les récompenses aient une valeur maximale finie : $\forall (e, a), r(e, a) \leq c, c \in \mathbb{R}$
- que l'agent visite l'ensemble des couples (e, a) infiniment souvent

Exemple 2.13 *La figure 2.10 montre un exemple de session de Q-learning sur l'exemple du monde grille.*

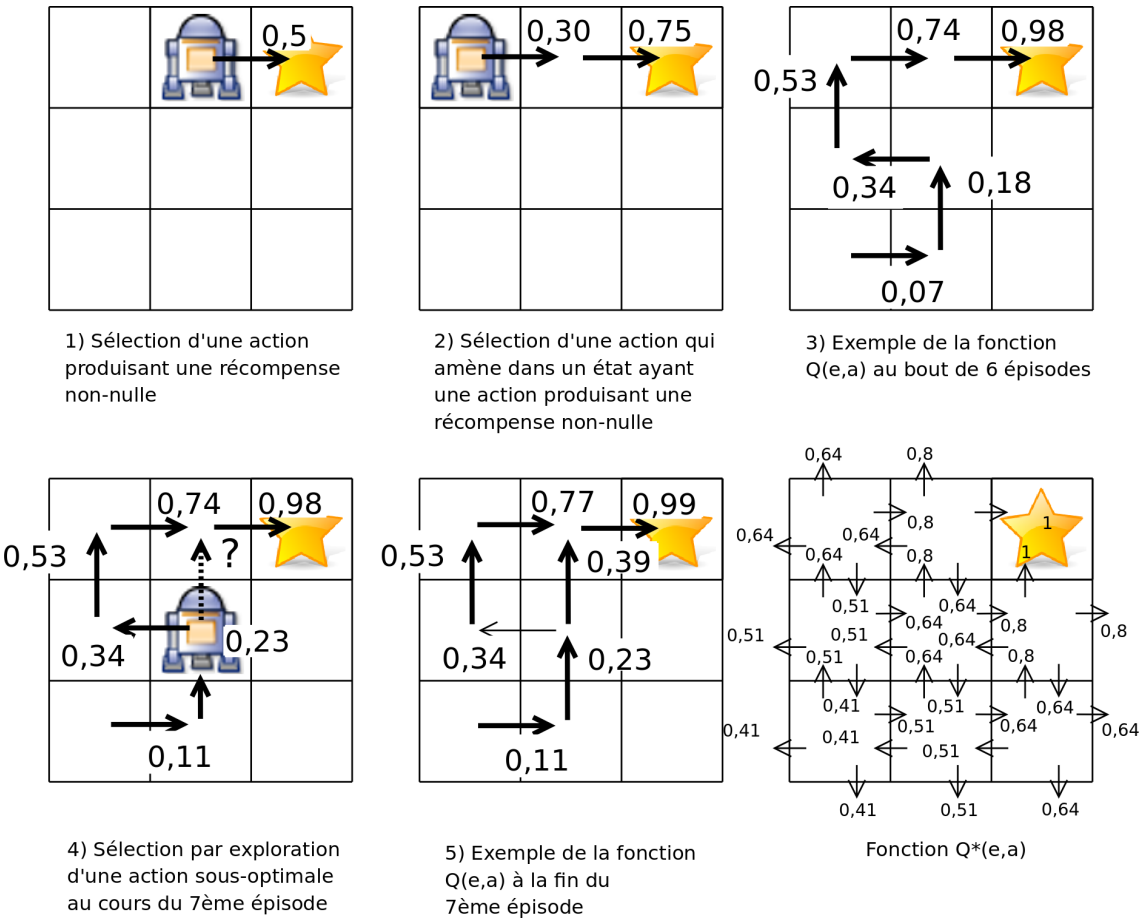


FIG. 2.10 – Exemple de session de *Q-Learning* avec $\alpha = 0,5$, $\gamma = 0,8$ et $\hat{Q}_0(e,a) = 0$, $\forall (e,a) \in E \times A$. L'agent sélectionne toujours l'action optimale sauf à la $4^{ème}$ vignette. Quand il y a plusieurs actions optimales (par exemple au début de l'apprentissage), il en sélectionne aléatoirement une. • Vignette 1), l'agent par son parcours aléatoire sélectionne l'action qui l'amène à la fin de l'épisode et à la récompense non nulle. La valeur de $0,5$ est obtenue par le calcul suivant : $\alpha \times Q_0(e_{(1,2)}, Est) + (1-\alpha) \times r_{(e_{(1,2)},Est)} = 0,5 \times 0 + (1-0,5) \times 1 = 0,5$. • Vignette 2), après un parcours aléatoire, l'agent sélectionne une action qui l'amène dans un état qui a une récompense nulle, mais dont il existe une action dont la récompense à long terme est non nulle. $\alpha \times Q_0(e_{(0,2)}, Est) + (1-\alpha) \times (r_{(e_{(0,2)},Est)} + \gamma(\max_a \hat{Q}_1(e_{(1,2)}, a))) = 0,5 \times 0 + (1-0,5) \times (0 + 0,8 * 0,5) = 0,3$, puis $\alpha \times Q_1(e_{(1,2)}, Est) + (1-\alpha) \times r_{(e_{(1,2)}}, Est) = 0,5 \times 0,5 + (1-0,5) \times 1 = 0,75$. • Vignette 3), au bout de 6 épisodes, l'agent connaît un chemin entre l'état initial et la récompense, mais celui-ci n'est pas optimal. De plus, pour certaines valeurs, la fonction $\hat{Q}(e,a)$ a presque convergé vers $Q^*(e,a)$. • Vignette 4), l'agent décide de ne pas sélectionner l'action optimale en l'état de $\hat{Q}(e,a)$, mais de sélectionner une action sous-optimale. • Vignette 5), à la fin de l'épisode, l'agent connaît un chemin optimal entre l'état initial et la récompense.

Soulignons que pour que la fonction $\hat{Q}_n(e,a)$ converge vers $Q^*(e,a)$, il n'est pas nécessaire qu'il suive une politique elle même optimale, il faut seulement qu'il visite infiniment souvent l'ensemble des couples (e,a). Nous reviendrons sur ce point en étudiant l'algorithme $Sarsa$ ci-après. Nous avons néanmoins affirmé que les algorithmes d'apprentissage par renforcement ne faisait pas de différence entre une phase d'apprentissage et une phase d'utilisation de l'apprentissage pour accumuler des récompenses. Ceci pose une contradiction que nous allons étudier maintenant.

2.4.4 Compromis exploration – exploitation

Les algorithmes d'apprentissage par renforcement classiques sont donc basés sur l'utilisation de l'estimation courante de la fonction $Q^*(e,a)$, la fonction $\hat{Q}(e,a)$, pour établir une politique. L'amélioration de $\hat{Q}(e,a)$ dépend des valeurs des récompenses collectées par l'agent au fur et à mesure de ses interactions avec l'environnement. Or, celles-ci dépendent directement de la politique suivie par l'agent. Par conséquent, $\hat{Q}(e,a)$ sert de base pour s'améliorer elle même. Cette utilisation de l'estimation $\hat{Q}(e,a)$ déjà effectuée pour acquérir de nouvelles données afin d'améliorer l'estimation $\hat{Q}(e,a)$ s'appelle l'*amorçage*[6].
Poser des conditions aussi peu contraignantes que possible pour que ce mécanisme aboutisse, c'est à dire qu'il y ait convergence de $\hat{Q}(e,a)$ vers $Q^*(e,a)$ est une des difficultés majeurs de l'apprentissage par renforcement.

Nécessité de l'exploitation Une estimation $\hat{Q}(e,a)$ va engendrer une certaine politique donnant à l'agent « un comportement optimal en l'état actuel des connaissance ». Or, il n'y a pas de séparation de phase d'apprentissage et de phase d'utilisation de l'apprentissage en apprentissage par renforcement. L'objectif de l'agent durant l'utilisation de son apprentissage devrait être d'obtenir le maximum de gains. Il ne peut faire cette acquisition maximale de gain qu'en utilisant sa connaissance actuelle de son environnement représentée sous la forme $\hat{Q}(e,a)$. Quand l'agent agit ainsi, c'est ce qu'on appelle l'*exploitation* de la connaissance acquise. L'incorporation d'exploitation est une nécessité, car c'est l'objectif *in fine* de l'apprentissage : utiliser la connaissance de la fonction de qualité de façon gloutonne.

Nécessité de l'exploration On peut également voir l'apprentissage par renforcement comme un apprentissage artificiel par induction classique, mais dont l'échantillon des valeurs de la fonction à apprendre n'est pas donné par un oracle, mais à collecter par l'agent lui-même[7]. Il faut, dans cette perspective, que cet échantillon soit suffisamment volumineux et varié si l'on veut avoir une bonne approximation de la fonction en question.
D'un point de vue sémantique, l'agent doit avoir suffisamment exploré l'ensemble des situations pour pouvoir trouver une politique optimale « dans l'absolu », et notamment être sûr qu'il ne suit pas des actions sous-optimales par manque de connaissances. Cela se traduit algorithmiquement par le fait que l'agent doit régulièrement faire des choix d'actions qui peuvent être sous-optimales conformément à $\hat{Q}(e,a)$ afin d'acquérir de nouvelles valeurs. C'est ce qu'on appelle l'*exploration* de l'environnement.

Compromis exploitation – exploitation Il y a donc nécessité pour qu'il y ait convergence de $\hat{Q}(e,a)$ vers $Q^*(e,a)$ que chacun des couples (e,a) soient visités suffisamment souvent (potentiellement infiniment). Il faut également que l'agent mette à profit son approximation, même imparfaite, de $Q^*(e,a)$ pour acquérir le maximum de récompenses. Le compromis entre ces deux comportements, en règle générale incompatibles, s'appelle le compromis *exploitation – exploitation*.

Le compromis exploration-exploitation a surtout été étudié sur le problème du bandit à n-bras[8]. Celui-ci consiste en un lot de n machines à sous chacune ayant sa propre probabilité de donner ses propres gains. A chaque étape, l'agent a le choix de la machine sur laquelle jouer. Il peut se contenter de jouer sur la machine qu'il connaît comme lui apportant potentiellement le plus de bénéfices, ou alors essayer d'améliorer sa connaissance sur ce que peuvent rapporter les autres

[6] bootstrapping en anglais
[7] Nous formaliserons ce propos partie 3
[8] n-armed bandit, en référence à un groupe que constituerait n bandits manchots, surnom donné aux machine à sous aux Etats-Unis

machines. Ce problème est largement étudié en statistiques, ingénierie voire psychologie. Une approche des principales questions que pose ce problème se trouve dans [Sutton and Barto, 1998]. La méthode optimale pour traiter ce compromis d'un point de vue général fait appel à des notions statistiques relativement complexes. Il faut tenir compte du nombre de valeurs à envisager, de l'incertitude de celles-ci, éventuellement le nombre d'essais encore possibles. Les études mathématiques de ce problème font des suppositions sur des éléments de connaissance de l'environnement qui ne sont pas envisageables dans le cadre dans lequel nous nous plaçons. Pour finir, ces études n'abordent pas la question de l'apprentissage avec différence temporelle, c'est à dire la prise en compte d'actions dont les effets sont répartis dans le temps.

Voyons ici les façons les plus courantes de traiter ce problème dans le cadre de l'apprentissage par renforcement.

2.4.5 Politiques exploratoires et gloutonnes

Définissons d'abord une politique d'exploitation simple, c'est à dire sans exploration.

Politique gloutonne Un agent suivant une politique qui cherche systématiquement à maximiser ses gains à chaque étape se nomme *politique gloutonne*[9].

$$\pi(e,a)\begin{cases} > 0 \text{ si } a \in \underset{a}{max}\ \hat{Q}(e,a) \\ = 0 \text{ sinon} \end{cases} \tag{2.16}$$

Même pour des problèmes simples, cette politique n'est pas efficace. L'agent utilise une politique qui est potentiellement sous-optimale si $\hat{Q}(e,a) \neq Q^*(e,a)$. Cependant, même une politique gloutonne permet l'amélioration $\hat{Q}(e,a)$, puisque des récompenses sont effectivement obtenues. Par contre, elle ne permettra d'augmenter la précision de $\hat{Q}(e,a)$ seulement pour les couples (e,a) rencontrés, c'est à dire potentiellement toujours les mêmes. Ainsi, à cause des couples (e,a) ignorés, $\hat{Q}(e,a)$ convergera vers une politique potentiellement sous-optimale. Voici deux mécanismes utilisés pour remédier à ce problème.

Politique ϵ-gloutonne Une première façon de considérer le problème et de séparer de façon explicite l'exploration et l'exploitation dans la politique, ce sont les politiques *ϵ-gloutonnes* :

$$\pi(e,a)\begin{cases} \epsilon\%\ \{\ \frac{1}{n} \text{ avec } n = |A(e)|,\ \forall a \in A(e) \\ \\ (1-\epsilon)\%\begin{cases} > 0 \text{ si } a \in \underset{a}{max}\ \hat{Q}(e,a) \\ = 0 \text{ sinon} \end{cases} \end{cases} \tag{2.17}$$

L'agent aura donc un comportement aléatoire dans $\epsilon\%$ des cas et glouton dans $(1-\epsilon)\%$.
Ce type de politique permet en premier lieu de palier aux problèmes cités pour les politiques gloutonnes. Si le nombre de sélections d'actions est infini, et que le monde est ergodique[10], toutes les actions seront sélectionnées potentiellement infiniment souvent. Du point de vue mathématique, les conditions de convergence de $\hat{Q}(e,a)$ vers $Q^*(e,a)$ sont réunies.
De plus, dans un système non stationnaire, c'est à dire si l'environnement se modifie en fonction du temps[11] la poursuite de l'exploration permet de découvrir d'éventuels changements de l'environnement.

Par contre, comme l'agent ne sélectionnera pas l'action optimale $\epsilon\%$ du temps, la politique effectivement suivie par l'agent ne sera optimale que $(1-\epsilon)\%$ du temps. Ainsi, même si la connaissance de $Q^*(e,a)$ est parfaite, l'agent n'aura pas un comportement optimal.
Une des méthodes employée pour résoudre ce problème est d'avoir une politique dont le paramètre ϵ est dégressif au cours du temps et $\epsilon \to 0$ quand le nombre d'interactions $n \to \infty$. Cependant,

[9] greedy en anglais
[10] tous les états sont potentiellement atteignables à partir de n'importe quel autre
[11] Rappelons que nous avons exclu ce cas de notre étude, mais que l'ouverture à ce type de problème reste un enjeu important pour l'apprentissage par renforcement

la vitesse de décroissance du paramètre ϵ reste posée pour cette méthode. En effet, cette question est rattachée à un problème important en apprentissage par renforcement et que nous retrouverons dans nos travaux : *comment sans connaissances sur l'environnement l'agent peut-il évaluer la précision de son apprentissage* ?

Sélection d'action SoftMax Avec une politique ϵ-gloutonne, quand l'agent explore, il le fait en choisissant une action au hasard. Du point de vue de la maximisation du flux de récompenses, ceci entraîne la sélection de temps en temps d'actions connues comme étant franchement sous-optimales au même titre que des actions peu explorées. Du point de vue sémantique, ceci peut-être franchement problématique si le monde contient des actions dangereuses. Imaginons par exemple un robot décidant de se jeter dans un gouffre de temps en temps pour voir si quand même, il n'y aurait pas une récompense à la clé.
Pour palier à ce problème, on utilise une sélection d'actions ayant une composante aléatoire mais également une pondération en relation avec la valeur estimée des actions. Plus une action est jugée intéressante, plus elle va avoir de probabilité d'être sélectionnée et réciproquement, les actions estimées comme amenant dans un état très défavorable auront tendance à être écartées, y compris durant la phase d'exploration.
Pour ce faire, la méthode la plus utilisé est le tirage des actions selon la distribution de *Boltzmann* aussi appelée sélection *SoftMax* :

$$\pi(e,a) \left\{ \frac{e^{\tilde{Q}(e,a)/\tau}}{\sum_{b=1}^{n} e^{\tilde{Q}(e,b)/\tau}} \text{ avec } n = |A(e)| \right.$$

Le paramètre τ est appelé la *température*. Plus la température est haute, plus la sélection d'actions devient équiprobable. Plus la température est proche de 0, plus la sélection d'action devient gloutonne.
L'utilisation de cette méthode de sélection d'action conserve la propriété de convergence des algorithmes, étant donné que chaque action dans chaque état reste visitée potentiellement infiniment souvent.

2.4.6 TD(λ) - Traces d'éligibilité

Nous avons vu que les récompenses envisagées par l'agent n'étaient pas les récompenses immédiates après une action, mais les récompenses « à long terme ». Ainsi, réciproquement, quand une récompense est obtenue, celle-ci ne dépend pas seulement de l'action précédente mais de la suite d'actions précédentes. Il convient donc de reporter cette récompense non pas seulement sur le couple $(état, action)$ précédent, mais sur l'ensemble des couples $(état, action)$ précédents.

De plus on peut rajouter une composante temporelle indiquant que l'effet d'une action est d'autant plus faible qu'elle a été sélectionnée il y a longtemps. Pour cela, on rajoutera un facteur $0 \leq \lambda \leq 1$ servant à formaliser la dégressivité de l'importance de l'action au cours du temps. C'est une façon de traiter l'apprentissage avec différence temporelle[12] posé par Minsky dans [Minsky, 1961].

Concrètement, à chaque interaction, le couple $(état, action)$ venant d'être visité reçoit une marque numérique, une *trace d'éligibilité*. Tous les autres couples couples $(état, action)$ voient leur trace diminuer d'un facteur λ.

$$tr_t(e,a) = \left\{ \begin{array}{l} \gamma\lambda tr_{t-1}(e,a) + 1 \text{ si } e = e_t \text{ et } a = a_t \\ \gamma\lambda tr_{t-1}(e,a) \text{ sinon} \end{array} \right.$$

Ensuite, la mise à jour composée de la récompense r_t et de la prévision de récompense en suivant une politique gloutonne, $\gamma Q(e',a)$ est reportée sur l'ensemble des couples $(état, action)$, en fonction de leur trace. Ainsi, le couple $(état, action)$ juste visité a une mise à jour classique, comme dans le *Q-Learning*. Les actions précédemment sélectionnées recevront également la mise à jour, mais atténué par le paramètre λ.

[12] Temporal Difference Learning, d'où le « TD » dans *TD(λ)*. Le terme de *TD(λ)* ne désigne pas à proprement parler un algorithme d'apprentissage par renforcement. Il sert en effet à évaluer une politique plutôt que de combiner l'évaluation et l'amélioration.

Finalement, il faut prendre en compte le fait que l'algorithme de *Q-Learning* ne prend pas systématiquement l'action optimale. Il ne faut donc mettre à jour les couples $(état, action)$ seulement lors des interactions ayant une politique gloutonne. Pour cela, les traces sont remises à zéro quand une action sous optimale est prise. L'algorithme complet nommé $Q(\lambda)$ est présenté algorithme 2.6.

Données :
- $\mathcal{M}\langle E, A, T, \psi, \mathcal{R}\rangle$, un Processus de Décision de Markov avec un état initial E_0 et un état final E_T
- Un agent évoluant dans $\mathcal{M}$ en suivant une politique $\pi(e, a) \in \Pi$
- $0 \leq \gamma \leq 1$, le paramètre de dégresivité
- $0 < \alpha < 1$, le paramètre de remise en cause de la connaissance
- $0 \leq \lambda < 1$, le paramètre de différence temporelle

Résultat :
- $\pi^*(e, a) \in \Pi^*$, une politique optimale

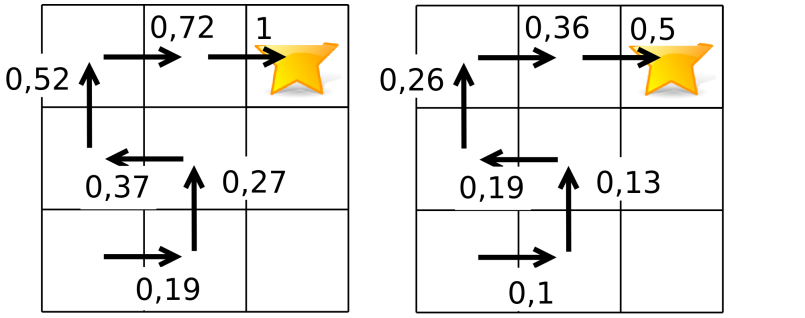
début
Initialisation de la politique
Initialiser $\hat{Q}(e, a) = v_0 \in \mathbb{R}$, $\forall (e, a) \in \psi$;
pour *Chaque épisode* **faire**
$e \leftarrow E_0$;
$a =$ choisir une action en utilisant $\hat{Q}(e, a)$;
tant que $e_t \neq E_T$ **faire**
appliquer a;
$r \leftarrow$ recevoir une récompense;
$e' \leftarrow$ recevoir un nouvel état;
$a' =$ choisir une action dans l'état e' en utilisant $\hat{Q}(e, a)$;
$a^* \leftarrow arg \max_b \hat{Q}(e', b)$;
$tr(e, a) \leftarrow tr(e, a) + 1$;
pour *Chaque couple* $(e_i, a_j) \in E \times A$ **faire**
$\hat{Q}(e_i, a_j) \leftarrow (1 - \alpha)\hat{Q}(e_i, a_j) + \alpha.tr(e_i, a_j).(r + \gamma\ \hat{Q}(e', a^*))$;
si $a' = a^*$ **alors**
$tr(e_i, a_j) \leftarrow \gamma.\lambda.tr(e_i, a_j)$;
sinon
$tr(e_i, a_j) \leftarrow 0$;
fin
fin
$e \leftarrow e'$;
$a \leftarrow a'$;
fin
fin
fin

Algorithme 2.6: *Q(λ)*

Exemple 2.14 *Figure 2.11 page suivante donne un exemple de session d'apprentissage par renforcement en utilisant l'algorithme* Q(λ).

2.4.7 Sarsa - Dépendance ou indépendance de l'apprentissage et de la politique

Nous avons montré qu'en apprentissage par renforcement, il n'y avait pas de distinction entre une phase d'apprentissage et une phase d'utilisation de cet apprentissage. Un des corollaires est qu'il faut que l'agent explore, c'est à dire qu'il sélectionne de temps en temps des actions sous-optimales par rapport à la connaissance courante de l'environnement. Le problème de cette technique est que l'agent continuera à sélectionner des comportements sous-optimaux y compris après avoir convergé vers une politique optimale. Nous avons donné une première réponse à ce phénomène avec la



1) Valeur des traces d'éligibilité Pour les actions sélectionnées après être arrivé par hasard sur la case "récompense"

2) Fonction Q(e,a) mise à jour en foncion des traces d'éligibilité et suite à l'obtention de la récompense

FIG. 2.11 – Exemple de session de $Q(\lambda)$ avec $\alpha = 0,5$, $\gamma = 0,8$, $\lambda = 0,9$ et $\hat{Q}_0(e,a) = 0$, $\forall (e,a) \in E \times A$. L'agent sélectionne toujours l'action optimale. Quand il y a plusieurs actions optimales (par exemple au début de l'apprentissage), il en sélectionne aléatoirement une. • Vignette 1), l'agent par son parcours aléatoire sélectionne l'action qui l'amène à la fin de l'épisode et à la récompense non nulle. Les valeurs proposées sont celles de la fonction de trace : $tr(e,a)$. La valeur de la trace pour l'action finale est 1 car elle vient d'être sélectionnée. La valeur de l'action précédente $0,72$ est obtenue par le calcul suivant : $tr_{t-1}(e,a) \times \gamma \times \lambda = 1 \times 0,8 \times 0,9 = 0,72$. • Vignette 2), la fonction $\hat{Q}(e,a)$ après la sélection de l'action menant à la récompense et donc à la fin de l'épisode. La valeur $0,36$ est obtenue comme suit : $\alpha \times Q_1(e_{(0,2)}, Est) + (1-\alpha).tr(e_{(0,2)}, Est) \times (r_{(e_{(1,2)}, Est)} + \gamma(\max_a \hat{Q}_1(e_{(1,2)}, a))) = 0,5 \times 0 + 0,5 \times 0,72 \times (1+0)$. Cette vignette est à comparer avec la première vignette de la figure 2.10 page 39 où seule l'action qui venait d'être sélectionnée était mise à jour. L'agent, dès sa première rencontre de récompense non-nulle obtient un chemin partant de l'état initial et menant à l'état final.

sélection d'action *softmax* permettant, même pour un choix aléatoire, de pondérer la sélection en fonction de la récompense attendue. Un seconde réponse classique est de diminuer le facteur exploratoire en fonction du temps.
En fait, le problème vient du fait que la politique effectivement suivie (politique de sélections d'actions exploratoires de temps en temps) n'est pas la politique évaluée (politique de sélection exploitrice tout le temps). Les algorithmes comme le *Q-Learning* ayant cette différence entre politique suivie et politique estimée sont qualifiés d'**indépendants de la politique**[13]. Selon nous, l'enjeu réel qu'il y a derrière ce problème est la capacité à estimer le niveau de convergence de $\hat{Q}(e,a)$ vers $Q^*(e,a)$, car la technique consistant à diminuer la part exploratoire ne résout finalement rien, la question de la vitesse de décroissance du paramètre exploratoire restant posée.
Plutôt que d'intégrer des éléments permettant la diminution de l'exploration au cours du temps pour faire converger la politique effective vers la politique optimale, il est un algorithme qui prend le parti de rechercher le comportement optimal « compte-tenu de l'exploration régulière ». C'est ce qu'on appelle les algorithmes **dépendants de la politique suivie**[14]. Ainsi l'algorithme *Sarsa* (algorithme 2.7) contrairement au *Q-Learning*, ne met pas à jour la fonction $\hat{Q}(e,a)$ par rapport à la politique optimale (équation 2.15 page 37), mais par rapport à la politique effectivement suivie :

$$\hat{Q}(e_{t-1}, a_{t-1}) \leftarrow (1-\alpha)\hat{Q}(e_{t-1}, a_{t-1}) + \alpha(r_t + \gamma\, \hat{Q}(e_t, a_t)) \qquad (2.18)$$
$$0 < \gamma < 1 \text{ et } 0 < \alpha < 1$$

Il faut donc attendre quelle sera l'action effectivement sélectionnée pour pouvoir faire la mise à jour. La mise à jour s'applique toujours au couple $(état, action)$ précédent (e_{t-1}, a_{t-1}), en incluant la récompense obtenue (r_t) et la prévision de récompense à long terme à partir de l'état suivant l'action effectivement sélectionnée (e_t, a_t)[15].

[13] *off-policy* en anglais
[14] *on-policy* en anglais
[15] état se dit *state* en anglais abrégé en s. La mise à jour impliquant un état s_{t-1}, une action $a_{t,1}$, la récompense

Données :
- $\mathcal{M}\langle E, A, T, \psi, \mathcal{R}\rangle$, un Processus de Décision de Markov avec un état initial E_0 et un état final E_T
- Un agent évoluant dans $\mathcal{M}$ en suivant une politique $\pi(e, a) \in \Pi$
- $0 \leq \gamma \leq 1$, le paramètre de dégresivité
- $0 < \alpha < 1$, le paramètre de remise en cause de la connaissance

Résultat :
- $\pi^*(e, a) \in \Pi^*$, une politique optimale

début
  Initialisation de la politique
  Initialiser $\hat{Q}(e, a) = v_0 \in \mathbb{R}$, $\forall (e, a) \in \psi$;
  $a =$ choisir une action en utilisant $\hat{Q}(e, a)$;
  pour *Chaque épisode* **faire**
    $e \leftarrow E_0$;
    tant que $e_t \neq E_T$ **faire**
      appliquer a;
      $r \leftarrow$ recevoir une récompense;
      $e' \leftarrow$ recevoir un nouvel état;
      $a' =$ choisir une action en utilisant $\hat{Q}(e, a)$;
      $\hat{Q}(e, a) \leftarrow (1 - \alpha)\hat{Q}(e, a) + \alpha(r + \gamma\ \hat{Q}(e', a'))$;
      $e \leftarrow e'$;
      $a \leftarrow a'$;
    fin
  fin
fin

Algorithme 2.7: Sarsa

Notons qu'il existe une version *Sarsa(λ)* qui applique directement les principes vus au paragraphe précédent.

Exemple 2.15 *L'exemple fourni dans [Sutton and Barto, 1998] à ce sujet est très parlant. La tâche pour l'agent consiste à se rendre vers un objectif dont la route optimale, la plus courte, est bordée d'un ravin. La politique optimale est en effet de longer le ravin, mais si on prend en compte l'exploration, il vaut mieux s'éloigner un peu du ravin !* Sarsa *obtient donc de meilleurs résultat que le* Q-Learning *dans des problèmes de ce type.*

Nous venons de présenter les aspects de l'apprentissage par renforcement permettant de se familiariser avec le sujet, de donner le cadre général que prétend traiter ce type d'apprentissage et d'introduire les notions et les formalismes de bases ainsi que ceux nécessaires à nos travaux.
De nombreux aspects n'ont pas été abordés, comme les aspects pratiques de modélisation, les aspects techniques d'implémentation, les applications concrètes, l'extension du problème aux systèmes multi-agents, les aspects théoriques de la vitesse de convergence en fonction des paramètres des algorithmes, ... Comme ces notions n'ont pas été prises en considération pour l'instant dans notre travail, nous les laissons ici de côté.

Nous allons dans le chapitre suivant aborder un aspect essentiel de l'apprentissage par renforcement. Nous avons en effet présenter le problème comme si l'agent pouvait distinguer chacun des états de son environnement, sélectionner chacune des actions potentiellement infiniment souvent. Dans le cas général, ces contraintes sont trop fortes et irréalistes. Nous voudrions pouvoir faire en sorte que l'agent apprenne un comportement sur une sous-partie des états de l'environnement et puisse l'appliquer à des états nouveaux sans avoir à réapprendre de rien. De plus, nous avons représenter la fonction $Q(e, a)$ sous une forme tabulaire, c'est à dire qu'à chacun des états de l'environnement et pour chacune des actions, nous avons attribué une valeur. Nous allons voir

r_t, le nouvel état s_t et l'action ayant menée dans cette état a_t, cela donne *Sarsa*, d'où le nom de l'algorithme.

d'autres formes de représentation permettant que l'apprentissage sur un état influe sur l'apprentissage sur d'autres états. Ce problème que nous allons aborder est celui de la **généralisation de l'apprentissage en apprentissage par renforcement**.

3 Généralisation de l'apprentissage et apprentissage par renforcement

Après avoir présenté le problème de la généralisation en apprentissage par renforcement sous l'angle de l'induction d'un comportement, nous allons présenter divers méthodes utilisées pour traiter ce problème. Premièrement, partie 3.2, nous présenterons la généralisation sous l'angle de l'*approximation de fonction*. Cette forme de généralisation est relativement éloignée de nos méthodes, mais compte tenu du fait qu'elle est la plus étudiée et la plus employée en pratique, nous en donnerons les principes généraux.
Deuxièmement, partie 3.3, nous aborderons un point de vue différent, appelée *généralisation par abstraction algébrique*. Même si le cadre de travail ne correspond pas totalement au nôtre (il suppose une connaissance de la dynamique de l'environnement contrairement à nos travaux) il offre néanmoins un cadre algébrique intéressant et des démarches comparables à celles que nous développerons dans les chapitres suivants. Nous nous baserons principalement sur deux approches : [Ravindran and Barto, 2003b, Givan et al., 2003].
Troisièmement, partie 3.4, nous présenterons la généralisation abordée sous l'angle de l'*apprentissage par renforcement relationnel*. Cet apport pour l'apprentissage par renforcement est relativement récent (article fondateur : [Dzeroski et al., 1998] et un atelier sur le sujet : [Tad, 2004]). Il consiste à amener les techniques de Programmation Logique Inductive[16] pour la généralisation dans l'apprentissage par renforcement. Nous le présentons ici car c'est probablement la démarche qui est la plus corrélée à la notre.
Finalement, nous ferons un tour d'horizon d'autres techniques de généralisation d'apprentissage partie 3.5. Nous évoquerons trois démarches (étude du cas continu, algorithmes à mémoire, abstraction temporelle) posant des problèmes similaires que la généralisation.

3.1 Problématique

Nous avons vu dans la partie précédente les algorithmes classiques d'apprentissage par renforcement permettant à un agent d'apprendre un comportement optimal dans un environnement, en se basant sur les perceptions successives des états de son environnement ainsi que sur les récompenses numériques successives associées.
Si la convergence asymptotique de l'agent vers un comportement optimal, grâce à l'utilisation de tels algorithmes, a été démontrée, les *a priori* nécessaires ainsi que l'application concrète des algorithmes posent problème pour une utilisation sur des cas non-académiques.
En effet, l'approximation au cours des itérations de la fonction de qualité $Q^*(e,a) : E \times A \rightarrow \mathbb{R}$ implique que l'ensemble des états $e \in E$ de l'environnement soient **observables** et **différentiables** par l'agent, ce qui, dans le cas général est un *a priori* fort.
Deuxièmement, du point de vue de la sémantique des algorithmes, l'approximation de la fonction $Q^*(e,a)$ en considérant tous les couples $(état, action)$ distincts et non corrélés implique que toutes les situations rencontrées par l'agent doivent être considérées de façon complètement séparées. L'agent doit ainsi trouver un comportement adéquat pour chacune des situations, sans en induire un comportement général.
Cette conception est donc problématique à plusieurs points de vue :

Exploration exhaustive de l'environnement Le cadre que nous avons donné pour l'apprentissage par renforcement s'en trouve réduit. En effet, nous avons avancé que cette famille d'algorithmes permettait à un agent d'apprendre un comportement sans connaissances préalables de son environnement. Si les algorithmes impliquent un parcours exhaustif des actions et des états, on conserve le caractère attractif de la non connaissance par l'agent de la *dynamique* de l'environnement, mais on exige quand même la connaissance de l'ensemble des états et des actions possibles de celui-ci. Il est évidemement souhaité une famille d'algorithmes qui puisse produire pour un agent

[16] ILP, Inductive Logic Programming en anglais

un comportement idéalement optimal, sinon proche de l'optimal, tout en apprenant que sur un sous-ensemble des états de l'environnement.

Réutilisation des solutions Comme corollaire à cette première remarque, on peut dire que l'on souhaiterait également, après l'apprentissage d'un comportement sur un problème, pouvoir réutiliser celui-ci sur un problème similaire, idéalement en ayant un comportement optimal, ou au moins sans avoir à recommencer l'apprentissage complètement. C'est en effet un corollaire si l'on considère qu'il n'y a pas des problèmes similaires, mais un seul problème que l'on peut découper. Ainsi, la réutilisation de l'apprentissage d'un comportement d'un problème à l'autre revient à apprendre un comportement sur un sous-ensemble des états.

Vitesse de convergence Si deux états de l'environnement différents ont un comportement optimal identique, le fait d'apprendre deux fois de façon séparée ce même comportement optimal est problématique. Ceci est corrélé à la vitesse de convergence des algorithmes d'apprentissage par renforcement. En effet, si un comportement est appris deux fois là où il pourrait être appris qu'une seule, la vitesse de convergence vers un comportement optimal en est réduit d'autant.

Interprétabilité du résultat et extraction de règles de comportement Un des critères régulièrement utilisé pour juger les algorithmes d'apprentissage artificiel est l'*interprétabilité du résultat*. Dans bien des cas, obtenir « seulement » un bon comportement de la part d'un agent artificiel n'est pas satisfaisant. En effet, on veut avoir une explication de ce comportement, que ce soit pour intervenir directement dessus ou pour pouvoir l'analyser et/ou le cautionner avec un regard d'expert « humain » (ce critère est par exemple essentiel dans le domaine de l'application médicale).
Globalement, ce critère est peu utilisé en apprentissage par renforcement où un « bon » comportement suffit. Ceci est certainement dû aux origines de l'apprentissage par renforcement qui s'est précisément fixé comme objectif d'obtenir un comportement optimal par modifications successives du comportement courant orienté par la seule récompense, c'est à dire en éloignant tout ce qui est raisonnement symbolique et / ou planification.
Particulièrement, dans le cadre basique, où tous les états de l'environnement sont dissociés, l'interprétabilité est à peu près nulle. Ce constat est également vrai pour beaucoup des espaces de représentations utilisés en apprentissage par renforcement, par exemple par l'utilisation de réseaux de neurones. Cependant, nous pensons, pour les raisons évoquées, que c'est un critère de l'apprentissage artificiel à ne pas négliger, *a fortiori* de l'apprentissage par renforcement que celui de pouvoir expliquer un comportement. De plus, on peut voir l'apprentissage par renforcement comme un pont possible entre l'apprentissage purement statistique d'un comportement et son explication, ou l'extraction de règles sur l'environnement suite à un apprentissage.

En fait, l'utilisation d'une représentation tabulaire de la fonction $Q(e, a)$ est très rare. On trouve alors toute sorte d'implémentation pour celle-ci (voir figure 3.1 page suivante) : fonction paramétrique, dont les réseaux de neurones, arbres de décision, k-tree, treillis de Galois (dans nos travaux),...

3.1.1 Généralisation de l'apprentissage

Tous ces éléments recouvrent la question appelée classiquement *généralisation* dans le sens où l'on requiert le fait que l'agent apprenne sur un sous-ensemble des états de l'environnement un comportement optimal, et qu'il soit capable d'étendre le comportement appris, idéalement, à l'ensemble des états de l'environnement tout en restant évidemment optimal.

Nous avons présenté partie 1 la question de l'apprentissage artificiel par induction. Nous proposons ici la notion de généralisation pour l'apprentissage par renforcement sous un angle similaire quoique présentant certaines singularités.

Faisons quelques remarques concernant ce formalisme. Premièrement, les questions relatives aux sources d'erreurs : *biais*, *variance*, *bruit* sont conservées.
L'erreur d'approximation ou biais signifie que la fonction $Q^*(e, a)$ et par conséquent le comportement optimal engendré par celle-ci peut ne pas être exprimable dans l'espace de représentation. Comme le phénomène d'apprentissage est ici dynamique, ceci n'entraîne pas seulement au bout

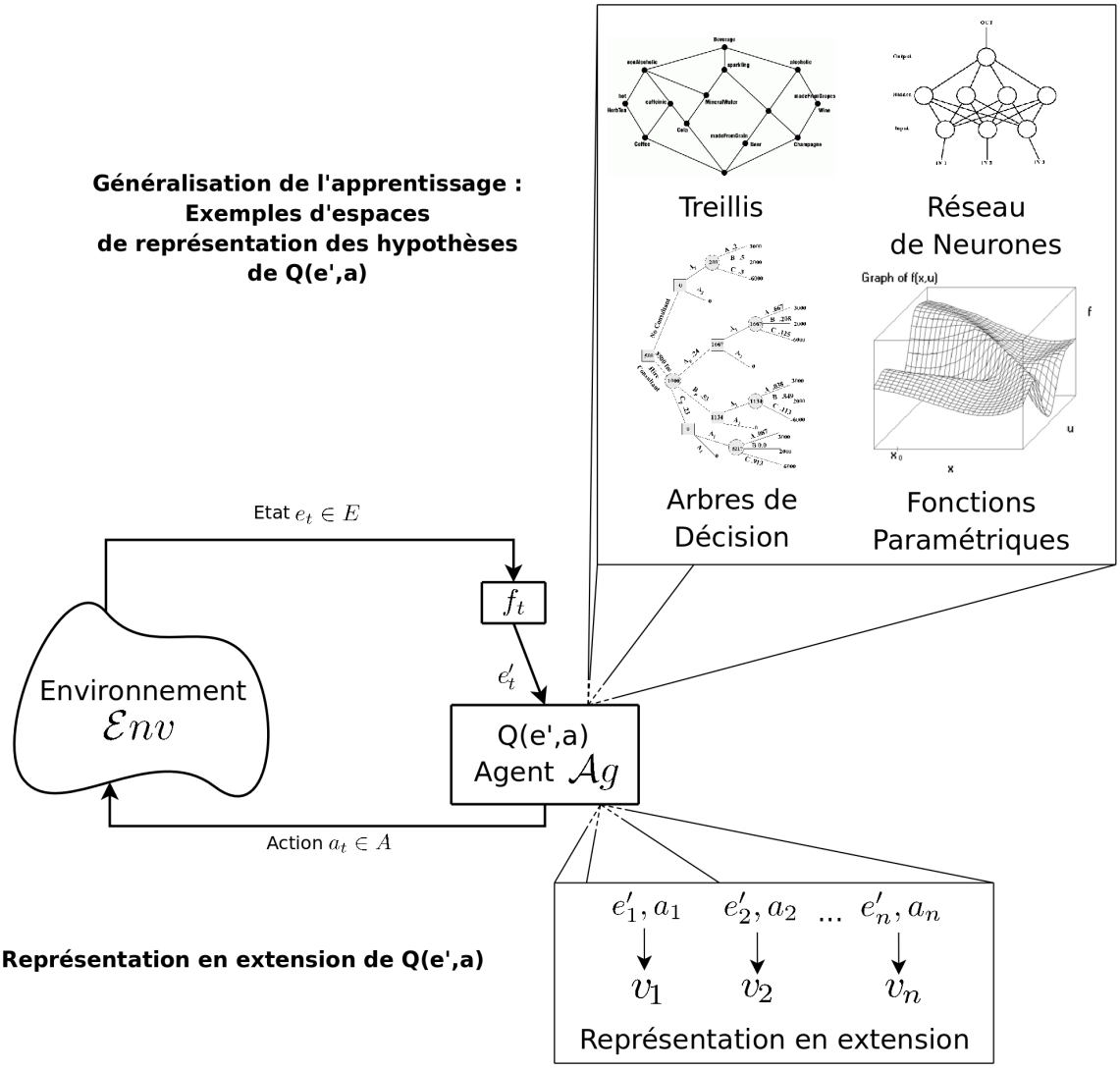


FIG. 3.1 – Dans les algorithmes basiques de l'apprentissage par renforcement, la fonction d'utilité $Q(e, a)$ donnant une valuation pour chacune des actions dans chacun des états est représentée en extension, par exemple en utilisant une table. Une représentation par intention permet de généraliser le comportement appris. Les méthodes dépendent des espaces de représentation utilisés.

du compte une mauvaise classification, mais toute sorte de comportements différents qui font en soit l'objet d'études (voir [Boyan and Moore, 1995] par exemple). Dans le cas général, il n'y a pas convergence vers un comportement sous optimal. Il y a souvent absence de convergence, voire totale divergence.

Dans le cadre de nos travaux, nous le rappellerons ultérieurement, nous ferons l'hypothèse que l'erreur d'approximation peut être rendue nulle, c'est à dire que l'espace de représentation des hypothèses permet de représenter le comportement optimal. L'erreur d'estimation, ou la variance est alors montrée comme étant nulle asymptotiquement. C'est à dire que l'hypothèse sur la fonction $\hat{Q}(e, a)$ va converger vers la fonction $Q^*(e, a)$ engendrée par un comportement optimal.

Nous n'avons pas traité la question du bruit ou de l'erreur intrinsèque, mais compte tenu du mécanisme par nature statistique de l'apprentissage par renforcement, sans développer, disons que celui-ci est relativement robuste au bruit qui pourrait par exemple être dû à une mauvaise observation de l'environnement.

Données :
- Un agent $\mathcal{A}g$ évoluant dans un environnement $\mathcal{E}nv$ par interactions discrètes
- Un espace de représentation $\mathcal{H}$ pour la fonction $Q(e, a)$
- Les triplets $(état, action, récompense)$ successifs produits par les interactions de l'agent avec son environnement, conséquences du choix des actions en fonction de l'hypothèse courante de la fonction $Q(e, a)$

Résultat :
- Un comportement optimal pour l'agent dans son environnement exprimé dans l'espace d'hypothèses $\mathcal{H}$

Algorithme 3.1: Apprentissage par renforcement vu sous l'angle de l'induction

Après ces éléments similaires à un apprentissage classique par induction, il nous faut poser une différence fondamentale.
Dans l'apprentissage par induction, les exemples ainsi que leur statut (exemple - contre-exemple) permettant d'affiner l'hypothèse sont en règle générale apportés par un oracle. Dans le cadre de l'apprentissage par renforcement, ces deux éléments sont différents.
L'équivalent des exemples sont les couples $(état, récompense)$ fournis à l'agent après chaque interaction. Or, la succession de ces couples est fonction de la partie *sélection d'action* des algorithmes d'apprentissage par renforcement (partie 2.4.5 page 41). Rappelons que celle-ci est fonction de trois facteurs :

- La dynamique de l'environnement
- Un facteur aléatoire (par exemple le facteur ϵ pour une sélection $\epsilon-$gloutonne)
- D'une fonction de l'hypothèse courante sur la fonction $Q^*(e, a)$.

Par conséquent, il faut que les méthodes développées soient très robustes quant au fait qu'il est impossible d'affirmer quoi que ce soit concernant la distribution des couples (état, action) fréquentés.

Ainsi, l'algorithme 3.1 propose le problème d'une tâche d'apprentissage par renforcement sous l'angle de l'induction classique de l'apprentissage machine.

3.2 Généralisation par approximation de fonctions

Nous avons évoqué en introduction le fait qu'il existait plusieurs méthodes explorées dans le cadre de la généralisation pour l'apprentissage par renforcement. Nous allons dans cette partie aborder une généralisation sur les états par utilisation d'approximation de fonctions. Cette méthode est la plus populaire pour un certain nombre de raisons. Premièrement, elle est la plus étudiée d'un point de vue théorique. Deuxièmement, elle est la plus utilisée d'un point de vue pratique pour des questions de performance. Nous donnerons des références quant à des utilisations de telles techniques d'apprentissage par renforcement dans des cas « réels ».
Enfin, l'approximation de fonctions n'étant pas une discipline propre à l'apprentissage par renforcement, ce domaine ne faisant que rajouter des contraintes spécifiques, il existe de nombreux résultats et techniques préexistants.
L'objectif de ce chapitre n'est pas de faire un état de l'art exhaustif sur la généralisation par approximation de fonctions. En effet, le coeur de nos travaux ne se situe pas dans ce domaine. Notre objectif est plutôt de donner une présentation de ces méthodes, car elles sont incontournables dans l'apprentissage par renforcement, ainsi que de donner des références pour un éventuel approfondissement.

Nous commencerons par montrer comment la fonction $Q(e, a)$ peut être représentée par une fonction paramétrée. Ensuite, nous présenterons une technique classique dans le domaine : la descente de gradient. Nous finirons par quelques résultats de convergence.

3.2.1 Représentation de la fonction $Q(e, a)$

Nous allons ici décrire les mécanismes de base de l'approximation de fonctions et montrer comment ils constituent bien une généralisation sur l'ensemble des états, le lien peut ne pas être évident au premier abord. L'idée et les mécanismes sous-jacents sont exposés de façon assez complète dans [Sutton and Barto, 1998] et de façon très pédagogique dans [Russel and Norvig, 2003]. On pourra également se réréfer à [Baird, 1995] pour un exposé succin.

Comme nous l'avons déjà évoqué, dans une utilisation naïve de l'apprentissage par renforcement, tous les états sont traités indépendamment les uns des autres sans aucune structuration. Pratiquement, c'est ce qu'on appelle une représentation tabulaire de la fonction $Q(e, a)$. A chaque couple (e, a), on associe la valeur de qualité estimée. Ceci s'implémente à l'aide d'un tableau par exemple.
Dans le cadre de l'approximation de fonctions, un état e n'est plus décrit par un identifiant le distinguant simplement des autres, mais à l'aide d'un ensemble de $n \in \mathbb{N}$ motifs[17] ayant chacun une valeur numérique.

$$f_i(e) : e \to \mathbb{R}, i \in n$$

Exemple 3.1 *Donnons différentes références de tâches d'apprentissage par renforcement ainsi que les motifs associés à chacune d'elles.*

Dans le cadre du monde grille (exemple 2.2 page 17), on peut poser que $f_1(e)$ soit l'abcisse de la case dans laquelle se trouve l'agent et $f_2(e)$ son ordonnée, considérant une origine quelconque. C'est ce qui est proposé dans [Russel and Norvig, 2003].

Dans le cadre de la RoboCup simulée, décrite dans [Stone et al., 2001], les motifs utilisés seront les distances entre joureurs et balons ainsi que les angles formés par les éléments (joueurs et ballon).

Dans le cadre du célèbre TD-Gammon ([Tesauro, 1992]), application de l'apprentissage par renforcement au jeu de Backgammon, les 198 fonctions sont les suivantes :

- *4 fonctions pour chacune des 24 cases et pour chacune des couleurs, avec $j \in \{1..24\}$ et $c \in \{blanc, noir\}$*
  $\begin{cases} f_{1jc}(e) = 0, f_{2jc}(e) = 0, f_{3jc}(e) = 0, f_{4jc}(e) = 0 \\ \text{s'il n'y a aucun pion de couleur } c \text{ sur la case } j. \\ f_{1jc}(e) = 1, f_{2jc}(e) = 0, f_{3jc}(e) = 0, f_{4jc}(e) = 0 \\ \text{s'il y a un pion de couleur } c \text{ sur la case } j. \\ f_{1jc}(e) = 1, f_{2jc}(e) = 1, f_{3jc}(e) = 0, f_{4jc}(e) = 0 \\ \text{s'il y a deux pions de couleur } c \text{ sur la case } j. \\ f_{1jc}(e) = 1, f_{2jc}(e) = 1, f_{3jc}(e) = 1, f_{4jc}(e) = 0 \\ \text{s'il y a trois pions de couleur } c \text{ sur la case } j. \\ f_{1jc}(e) = 1, f_{2jc}(e) = 1, f_{3jc}(e) = 1, f_{4jc}(e) = (n-3)/2 \\ \text{s'il y a } n > 3 \text{ pions de couleur } c \text{ sur la case } j. \end{cases}$
  Soit un total de 192 fonctions.
- *2 fonctions pour le nombre de pions noirs et blancs en stock*
  $f_{192}(e)$ =nb blancs/2 et $f_{193}(e)$ =nb noirs/2
- *2 fonctions pour le nombre de pions noirs et blancs déjà sortis*
  $f_{194}(e)$ =nb blancs/15 et $f_{195}(e)$ =nb noirs/15
- *2 fonctions pour indiquer le tour $f_{196}(e) = 1$ et $f_{197}(e) = 0$ si c'est aux blancs de jouer, l'inverse sinon.*

Chacun de ces motifs est ainsi pondérés par un ensemble de paramètres $\theta_i \in \mathbb{R}$ souvent représentés sous forme de vecteur noté $\overrightarrow{\theta}$. La fonction d'utilité d'un état $V(e)$ est alors une fonction des motifs pondérés par leur paramètre :

$$V_{\overrightarrow{\theta}}(e) = g(\theta_1 f_1(e), \theta_2 f_2(e), \ldots, \theta_n f_n(e))$$

[17] *features* en anglais

Si on se place comme dans le cadre d'une absence de connaissance du model, la fonction $Q(e,a)$ est alors définie par :

$$Q_{\overrightarrow{\theta}}(e,a) = h(\theta_1 f_1(e), \theta_2 f_2(e), \ldots, \theta_n f_n(e), a)$$

Exemple 3.2 *Dans le cas du monde grille, avec $f_0(e) = 1, f_1(e) = x, f_2(e) = y$, $V(e)$ peut par exemple être définie par :*

$$V_{\overrightarrow{\theta}}(e) = \theta_0 + \theta_1 x + \theta_2 y$$

Dans le cas de la Robocup ([Stone et al., 2001]), la fonction est un arbre de décision portant sur des conditions sur les paramètres.

Dans le cas de TD-Gammon *([Tesauro, 1992]), la fonction est un réseau de neurones multi-couche. θ est alors l'ensemble des poids du réseau de neurones.*

Approximation de fonctions L'approximation de fonctions étant un domaine très étudié, faisons en sorte de transposer le problème de l'apprentissage par renforcement dans ce cadre, comme un cas particulier.
Considérons que l'on cherche à approximer la fonction $Q^*(e,a)$ à l'aide d'une fonction $Q_{\overrightarrow{\theta}}(e,a)$ dont les paramètres sont $\overrightarrow{\theta}$. L'objectif idéal est alors de trouver les paramètres $\overrightarrow{\theta^*}$ tels que $Q_{\overrightarrow{\theta^*}}(e,a)$ soit la plus proche possible de $Q^*(e,a)$. A chaque interaction t, l'agent fait une expérience, c'est à dire qu'il observe une récompense qu'il combine avec la récompense estimée à long terme. Par exemple, dans le cas du *Q-Learning*, $r_t + \gamma \max_{a'} Q_{\overrightarrow{\theta_t}}(e_{t+1}, a')$. Chacune des expériences de l'agent peut être alors considérée comme un exemple de la fonction $Q_{\overrightarrow{\theta^*}}(e,a)$. Ainsi, à chaque interaction, l'agent modifiera les paramètres $\overrightarrow{\theta}$ pour les faire coïncider un peu plus avec ce qu'il estime être la fonction $Q^*(e,a)$.
A chaque étape t, $\overrightarrow{\theta}$ est modifié, c'est pourquoi on l'indice temporellement : $\overrightarrow{\theta_t}$.

La forme de la fonction (les motifs $\overrightarrow{\theta}$ ainsi que la fonction les liant les uns aux autres) est alors fixée par le concepteur. Il peut s'agir, comme nous l'avons proposé, d'une fonction linéaire des motifs, d'un arbre de décision, d'un réseau de neurones,... Il s'agit là d'une étape cruciale quant à la convergence de l'algorithme comme nous le verrons un peu plus loin.

Arrêtons nous sur quelques remarques importantes.

Bootstraping Rappelons que les expériences faites par l'agent sont fortement dépendantes de l'algorithme par renforcement utilisé. Elles le sont à deux titres qu'il est nécessaire de prendre en compte ici.

- Premièrement, la distribution des expériences dépend de façon importante de la politique suivie. Si par exemple une politique dans un état e_1 évite la sélection d'une action a_1 et privilégie la sélection d'une action a_2, non seulement le couple (e_1, a_2) sera plus fréquemment présent dans l'échantillon, mais tous les états rencontrés après e_1 en sélectionnant a_2 seront plus fréquemment présents dans l'échantillon également. Ainsi, la façon dont l'algorithme par renforcement engendrera une politique (politique fixée, politique aléatoire, politique ϵ-gloutonne, selection *soft-max*,...) influencera fortement la distribution.

- Deuxièmement, la récompense estimée par l'agent $(r_t + \gamma \max_{a'} Q_{\overrightarrow{\theta_t}}(e_{t+1}, a'))$ est une estimation de la récompense à long terme en suivant ce que l'agent estime être une politique optimale. Il s'agit donc d'une estimation faite à partir d'une donnée réelle (la récompense r_t) et d'une autre estimation $(\max_{a'} Q_{\overrightarrow{\theta_t}}(e_{t+1}, a'))$. En anglais, c'est ce qu'on appelle le *bootstraping*. Or, comme nous venons de l'exprimer ci-dessus, la politique de l'agent est généralement fonction de l'estimation courante $Q(e,a)$. La façon dont l'algorithme actualise son estimation en fonction des expériences influence également la politique de l'agent, donc l'acquisition des expériences suivantes.

Ainsi, toutes les méthodes d'approximation de fonctions faisant des assomptions sur la distribution des exemples ne sont pas adéquates pour l'apprentissage par renforcement.
Notons au passage que ceci est l'une des difficultés majeures de l'apprentissage par renforcement. En fait, on ne peut pas se servir de méthodes faisant intervenir de conditions quant à la distribution des couples $(état, action)$, celle-ci ne pouvant être connue ou maîtrisée par définition.

Remarque 3.1 *Montrons qu'il s'agit bien d'une généralisation de l'apprentissage. Pour chaque interaction, l'agent modifiera la fonction $\hat{Q}(e,a)$ pour que la fonction pour le couple concerné soit « un peu plus proche » de ce qu'elle était. Cependant, comme l'ensemble des paramètres $\overrightarrow{\theta}$ sera modifié, il s'agit bien d'une modification de la fonction $\hat{Q}(e,a)$ pour l'ensemble des couples (e,a), y compris ceux qui n'ont pas ou peu été fréquentés. Ainsi, ce qui a été appris sur l'ensemble des couples fréquentés (e,a) implique une fonction $\hat{Q}(e,a)$ pour l'ensemble des couples de $\psi \subseteq E \times A$.*

3.2.2 Erreur quadratique et descente de gradient

Voyons maintenant le mécanisme général de l'approximation de fonctions.
Tout d'abord, il nous faut préciser la mesure de l'erreur entre une fonction donnée et la fonction réelle qu'elle est censée approcher. Classiquement, on utilise l'erreur quadratique moyenne [18] ou **erreur quadratique**.

Définition 3.1 (Erreur Quadratique (Moyenne) pour $Q(e,a)$)
Soit une tâche d'apprentissage par renforcement modélisée par un PDM $\mathcal{M}\langle E, A, T, \psi, \mathcal{R}\rangle$. E est perçu par l'agent avec n motifs f_i, tels que $\forall i \in n, f_i : E \to \mathbb{R}$. La fonction $Q(e,a)$ est représentée par la fonction $Q_{\overrightarrow{\theta}}(e,a) = g(\theta_i f_i(e))$. L'**erreur quadratique moyenne** est définie par :

$$MSE(\theta) = \sum_{(e,a)\in\psi} P(e,a)\big(Q_{\overrightarrow{\theta}}(e,a) - Q^*(e,a)\big)^2 \tag{3.1}$$

Pour la version tabulaire de l'apprentissage par renforcement que nous avons exposée chapitre 2, il s'agissait de faire converger $\hat{Q}(e,a)$ vers $Q^*(e,a)$, la connaissance de $Q^*(e,a)$ permettant d'engendrer de façon simple une politique optimale.
De la même manière, il s'agira ici par apprentissage de diminuer l'erreur quadratique entre la fonction $Q_{\overrightarrow{\theta}}(e,a)$ et les exemples observés.
Arrêtons nous encore sur une remarque. L'erreur quadratique est pondérée par la probabilité qu'un couple $(état, action)$ soit sélectionné, ce qui est bien entendu dépendant de la politique suivie. Ceci implique que le résultat de l'approximation de fonctions ne devrait pas être le même en fonction de la façon dont l'algorithme d'apprentissage génère une politique : aléatoire, fixe, ϵ-gloutonne, *softmax*,... Ce n'est en soit pas « choquant », étant donné que c'est bien sur la politique effectivement suivie que l'on veut diminuer l'erreur, mais cela pose des problèmes en terme de réutilisabilité du résultat.

Le mécanisme classiquement utilisé pour diminuer cette erreur quadratique est la descente de gradient. Nous renvoyons le lecteur au chapitre 8.2 de [Sutton and Barto, 1998].
L'idée générale est d'ajuster les paramètres de pondération $\overrightarrow{\theta}$ progressivement à chaque expérience dans l'objectif de diminuer l'erreur quadratique. Ainsi, à chaque interaction, la fonction $Q_{\overrightarrow{\theta}}(e,a)$ est modifiée par l'intermédiaire de la modification de $\overrightarrow{\theta}$ par la formule suivante :

$$\overrightarrow{\theta}_{t+1} = \overrightarrow{\theta}_t - \frac{1}{2}\alpha\nabla_{\overrightarrow{\theta}_t}\big(r_{t+1} + \gamma \max_{a'} Q_{\overrightarrow{\theta}_t}(e_{t+1},a') - Q_{\overrightarrow{\theta}_t}(e_t,a_t)\big)^2 \tag{3.2}$$

$$= \overrightarrow{\theta}_t + \alpha\big(r_{t+1} + \gamma \max_{a'} Q_{\overrightarrow{\theta}_t}(e_{t+1},a') - Q_{\overrightarrow{\theta}_t}(e_t,a_t)\big)\nabla_{\overrightarrow{\theta}_t} Q_{\overrightarrow{\theta}_t}(e_t,a_t) \tag{3.3}$$

$\nabla_{\overrightarrow{\theta}_t} Q_{\overrightarrow{\theta}_t}(e_t,a_t)$ est le vecteur des dérivées partielles de $Q_{\overrightarrow{\theta}_t}(e_t,a_t)$ selon $\overrightarrow{\theta}_t$. Ceci implique que $Q_{\overrightarrow{\theta}}(e,a)$ soit une fonction dérivable.

L'algorithme 3.2 page suivante fournit l'algorithme de *Q-Learning* adapté à la descente de gradient. Le problème, c'est qu'un tel algorithme est connu pour diverger dans le cas général.

[18] Mean Squared Error (MSE) en anglais

Pour un algorithme assuré de converger avec une probabilité de 1 et ne tenant pas compte de la politique suivie (*off-policy*) comme c'est le cas du *Q-learning*, il faut se référer au papier [Precup et al., 2001]. Voir également [Tsitsiklis and van Roy, 1997] pour d'autres résultats sur le sujet.

Données :
- $\mathcal{M}\langle E, A, T, \psi, \mathcal{R}\rangle$, un Processus de Décision de Markov avec un état initial E_0 et un état final E_T
- Un agent évoluant dans $\mathcal{M}$
- $0 \leq \gamma \leq 1$, le paramètre de dégressivité
- $0 < \alpha < 1$, le paramètre de remise en cause de la connaissance

Résultat :
- $\pi^*(e, a) \in \Pi^*$, une politique optimale

début
  Initialisation de la politique
  Initialiser $\hat{Q}(e, a) = v_0 \in \mathbb{R}$, $\forall (e, a) \in \psi$;
  pour *Chaque épisode* **faire**
    $e \leftarrow E_0$;
    tant que $e_t \neq E_T$ **faire**
      $a =$ choisir une action en utilisant $\hat{Q}(e, a)$;
      appliquer a;
      $r \leftarrow$ recevoir une récompense;
      $e' \leftarrow$ recevoir un nouvel état;
      $\hat{Q}(e, a) \leftarrow (1 - \alpha)\hat{Q}(e, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(e', a'))$;
      $e \leftarrow e'$;
    fin
  fin
fin

Algorithme 3.2: *Q-Learning* avec descent de gradient. Cet algorithme est dans le cas général divergent

Nous laissons au lecteur le soin d'aller voir les références pour les nombreuses précisions concernant ces méthodes.

Convergence Nous avons vu chapitre 2 que sous la condition que tous les couples (état,action) soient visités potentiellement infiniment souvent, $Q(e, a)$ convergeait vers $Q^*(e, a)$. En est-il de même pour les fonctions paramétrées ? La réponse est non dans le cas général. Pire, dans le cas général, il n'y a même pas convergence vers une solution sous-optimale, il peut y avoir absence de convergence, ou même divergence dans le pire des cas. Deux contre-exemples connus sont présentés dans [Baird, 1995] et [Tsitsiklis and van Roy, 1997].
C'est d'ailleurs ce problème qui mobilise une partie de la recherche en apprentissage par renforcement. Quelles sont les classes de fonctions et/ou les conditions impliquant une convergence ?
A noter quand même que dans le cas simple d'une fonction linéaire en les motifs, comme celle que nous avons présentée exemple 3.2 page 52, il existe des résultats de convergence (voir : [Precup et al., 2001, Tsitsiklis and van Roy, 1997]) pour les fonctions linéaires. Dans [Gordon, 2001] on trouve des résultats intéressants concernant les algorithmes *on-line*, montrant qu'ils oscillent dans une région bornée. Pour une ralation avec le PAC-Learning ainsi qu'un résultat général, on pourra voir [Papavassiliou and Russell, 1999].

3.3 Généralisation par abstraction algébrique des états

Nous venons de montrer comment fonctionne le principe de généralisation pour l'apprentissage par renforcement le plus connu, c'est à dire en utilisant l'approximation de fonctions. Nous allons maintenant montrer une approche complètement différente, que nous nommerons *par abstraction*

algébrique des états. Si la méthode précédente était plutôt orientée par le côté pratique et utilisable de l'apprentissage par renforcement, quitte à converger vers des solutions sous-optimales, la généralisation par abstraction algébrique est plutôt orientée par le côté théorique et recherche exacte de solutions. En effet, en pratique elle est peu utilisable sans heuristiques car elle nécessite une connaissance du modèle de l'environnement et la complexité algorithmique des méthodes sont très importantes. Cependant, elle reste très intéressante dans sa démarche comme modèle théorique permettant de modéliser algébriquement la notion de généralisation. Elle nous permet également de poser des bases dont nous nous resservirons pour nos travaux que nous présenterons dans les chapitres suivants.
Les notions données ici viennent principalement de [Ravindran and Barto, 2003b] et [Givan et al., 2003].

Le principe général de la généralisation par abstraction algébrique des états est de considérer que des états de l'environnement peuvent être similaires du point de vue du modèle de l'environnement, similaires dans le sens où, à une modification algébrique près, la dynamique de l'environnement reste la même. Ainsi, l'apprentissage sur un état de l'environnement e_1 peut être reporté sur un état e_2. Il convient alors de les regrouper pour ne pas apprendre deux fois, de façon distincte la même chose.

Nous allons d'abord voir ce point de vue sous l'angle développé dans [Givan et al., 2003] qui considère la généralisation comme un partitionnement de l'ensemble des états de l'environnement. Nous le présentons en premier car nous aborderons le même point de vue pour nos travaux. Ensuite, nous verrons l'angle développé dans [Ravindran and Barto, 2003b], car le propos y est plus général d'un point de vue algébrique.

3.3.1 Généralisation par partitionnement de l'ensemble des états de l'environnement

Nous allons tout d'abord présenter la méthode dite de minimisation du modèle développée dans [Givan et al., 2003]. Toutes les notions entre partitions d'un ensemble et relation d'équivalence seront présentées en détail chapitre 4. Il peut être utile au lecteur non-familiarisé avec ces notions de lire la partie 4.6 page 94.
L'idée de l'algorithme de minimisation de modèle est de regrouper les états de l'environnement similaires afin d'obtenir un PDM dont l'apprentissage sur un état peut être appliqué à l'ensemble des états similaires.
Plus formellement, regrouper les états similaires revient algébriquement à définir une relation d'équivalence entre les états. Cette relation d'équivalence définissant elle-même une partition de l'ensemble des états de l'environnement.
Dans ce cadre, la relation d'équivalence nommée bissimulation est basée sur le fait que deux états sont équivalents s'ils jouent le même rôle dans le modèle de l'environnement. Ainsi, deux états sont équivalents si la suite d'interactions possibles à partir de ceux-ci sont les mêmes. On dit alors qu'ils sont équivalents du point de vue des séquences.

Définition 3.2 (équivalence du point de vue des séquences)
Soit un Processus de Décision Markovien $\mathcal{M}\langle E, A, T, \psi, \mathcal{R}\rangle$. La relation d'équivalence du point de vue des séquences $\equiv_{trans}$ définie sur E est définie par :
$\forall e_1, e_2 \in E, e_1 \equiv_{trans} e_2 \Leftrightarrow A(e_1) = A(e_2)$ et $\forall a \in A(e_1), T(e_1, a, E_1) = T(e_2, a, E_1)$
avec $E_1 \subseteq E$ tel que $\forall e_i, e_j \in E_1$, $e_i \equiv_{trans} e_j$

Comme on le voit, la relation d'équivalence est définie de manière récurrente, il faut que les actions de deux états soient les mêmes et amènent, avec les mêmes probabilités, vers des groupes d'états eux-mêmes équivalents.
Cette équivalence ne suffit pas, car l'effet des actions sur l'environnement peut être le même en ce sens qu'elles amènent statistiquement vers les mêmes états, mais les récompenses associées peuvent différer. Ainsi, il faut rajouter une deuxième condition à l'équivalence : que les récompenses associées aux actions sont les mêmes[19].

[19]La définition est légèrement modifiée comparée à l'originale pour garder une unité de formalisme.

Définition 3.3 (équivalence du point de vue des récompenses)
Soit un Processus de Décision Markovien $\mathcal{M}\langle E, A, T, \psi, \mathcal{R}\rangle$. La relation d'équivalence du point de vue des récompenses $\equiv_{rec}$ définie sur E est définie par :
$\forall e_1, e_2 \in E, e_1 \equiv_{rec} e_2 \Leftrightarrow \{e \in E \text{ tels que } e_1 \in Acc(e)\} = \{e \in E \text{ tels que } e_2 \in Acc(e)\} = E_1$ et $\forall e_i \in E_1, \forall a_i \in Acc(e_i), \mathcal{R}(e_i, a_i, e_1) = \mathcal{R}(e_i, a_i, e_2)$

Remarquons que cette dernière équivalence n'est pas non plus suffisante. En effet, elle a pour conséquence de regrouper les états dont les actions procurent les mêmes récompenses. Ceci peut n'être qu'une pure coïncidence. Ainsi, l'équivalence de bissimulation est basée sur les deux notions d'équivalence que nous venons de décrire.

Définition 3.4
Soit un Processus de Décision Markovien $\mathcal{M}\langle E, A, T, \psi, \mathcal{R}\rangle$. La relation d'équivalence du point de vue des séquences $\equiv_{bis}$ définie sur E est définie par :

$$\forall e_1, e_2 \in E, e_1 \equiv_{bis} e_2 \Leftrightarrow e_1 \equiv_{rec} e_2 \text{ et } e_1 \equiv_{trans} e_2$$

L'algorithme en spécialisation fourni dans [Givan et al., 2003] dont nous donnons simplement les spécifications algorithme 3.3, permet par divisions successives de la partition initiale (tous les états sont dans la même partie), de produire un PDM minimisé. Les parties divisées sont celles dont les états ne respectent pas l'équivalence $\equiv_{bis}$.

Ainsi, les états regroupés forment un nouvel état dans le PDM minimisé $\equiv_{bis}$ implique des classes d'équivalence et chacun des états est dit stochastiquement bisimilaire à l'état

Il est également mentionné mais non implémenté la méthode en généralisation, c'est à dire celle qui consiste à fusionner les états équivalents. Les actions ayant le même nom sont également fusionnées et les probabilités maintenues.

Données :
- Un PDM $M\langle E, A, \psi, T, \mathcal{R}\rangle$

Résultat :
- La partition $\mathbb{P}$ de E telle que $\mathbb{P}$ soit la plus générale des partitions telle que $\forall \mathcal{P}_i \in \mathbb{P}$, $\forall e_j, e_k \in \mathcal{P}_i$, $e_j \equiv_{bis} e_j$.

Algorithme 3.3: Spécifications de l'algorithme de minimisation de modèle

Remarque 3.2 *Notons cependant, et nous y reviendrons partie 4.6, que cette factorisation nécessite un langage d'équivalence (au sens où nous le définirons définition 4.35 page 97) sur les actions. En effet, la définition 3.2 pose le fait que deux états sont équivalents si une action a dans l'état e_1 a le même effet qu'une action a dans l'état e_2. Ceci implique que les actions a dans l'état e_1 et e_2 aient quelque chose de commun. En fait, il est sous-entendu que deux actions portant la même étiquette sont similaires. Ce sous-entendu n'est pas explicité dans le papier original des auteurs.*

Premièrement, un PDM minimisé, par définition contient moins d'états que le PDM original. Deuxièmement, il conserve les politiques optimales, c'est à dire que toute politique optimale pour un PDM est également optimale sur un modèle minimisé et réciproquement.

Une fois le modèle minimisé, c'est à dire que tous les états similaires du point de vue de la dynamique de l'environnement ont été fusionnés, il suffit d'appliquer l'apprentissage sur le modèle minimisé. Ainsi, toute expérience faite sur un couple (e, a) sera appliquée sur chacun des couples (e', a) tels que e' soit un état appartenant à la même partition que e, c'est à dire $\forall e' \in E$ tels que $e \equiv_{bis} e'$. Ainsi, une politique optimale π^* sera trouvée pour le PDM minimisé et pour le PDM original, mais plus rapidement, étant donné que chaque expérience donnera lieu à plusieurs, et non une seule, mises à jour de la fonction $Q(e, a)$.

Il s'agit bien d'une généralisation, car quand deux états sont regroupés dans la même partie, l'apprentissage suite à une interaction sur un état, influe sur d'autres.

Malheureusement, la recherche du modèle minimisé par cette méthode est NP-difficile.

3.3.2 Généralisation par homomorphisme de PDM

Nous allons maintenant voir une extension de cette conception, et c'est pour cela que nous la présentons avec la notion d'homomorphisme de PDM. Les articles [Ravindran and Barto, 2001, Ravindran and Barto, 2002, Ravindran and Barto, 2003b, Ravindran and Barto, 2003a, Ravindran and Barto, 2004] exposent ce sujet largement.

Le résultat de la généralisation est comme précédemment un modèle de l'environnement factorisé permettant un apprentissage plus rapide.
Nous allons reprendre ici la définition donnée dans [Ravindran and Barto, 2003b] pour appliquer la notion d'homomorphisme aux Processus de Décision Markovien. La définition originale s'applique également aux Processus de Décision Semi-Markoviens[20], formalisme développé dans [Sutton et al., 1999] et ajoutant aux PDM une notion d'abstraction temporelle par le biais d'options. Comme nos travaux ne se situent pas dans ce cadre, nous omettrons la partie abstraction temporelle. Avant de donner la notion d'homomorphisme pour les Processus de Décision Markovien, nous allons rappeler les notions d'homomorphismes classiquement utilisées pour les graphes. Les homomorphismes pour les processus déterministes ont déjà été développés auparavant. Ici, la définition s'étend au PDM stochastiques.

Définition 3.5 (Homomorphisme)
Soit $G(S_G, A_G)$ et $H(S_H, A_H)$ deux graphes ayant respectivement pour ensemble de sommets S_G et S_H et pour ensemble d'arêtes $A_G \in S_G \times S_G$ et $A_H \in S_H \times S_H$. Une fonction $\alpha : S_G \to S_H$ est un **homomorphisme** de G vers H s'il préserve les arêtes de G dans H ; c'est à dire que $\forall [u, v] \in A_G$, $[\alpha(u), \alpha(v)] \in A_H$.

Exemple 3.3 *La figure 3.2 donne un exemple d'homomorphisme de graphe. Avec $\alpha(a_i) = a$ et $\alpha(b_i) = b$. Par exemple, on a bien conservation de l'arête $[b_1, a_3]$ qui donne l'arête $[b, a]$ dans le second graphe.*

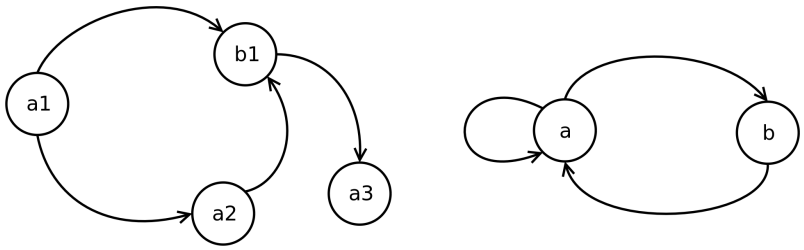


FIG. 3.2 – Exemple d'homomorphisme de graphes. Chaque sommet du graphe de gauche est projeté sur le sommet de même étiquette sur le graphe de droite sans son indice.

Présentons maintenant la notion d'homomorphisme pour les Processus de Décision de Markov.[21]

Définition 3.6 (homomorphisme de PDM)
Soit deux Processus de Décision Markoviens $\mathcal{M}(S, A, \psi, P, R)$ et $\mathcal{M}'(S', A', \psi', P', R')$. Un **homomorphisme de PDM** $h : \mathcal{M} \to \mathcal{M}'$ est une surjection $h_\psi : \psi \to \psi'$ définie par :
- une fonction f sur les ensembles d'état $f : S \to S'$
- une fonction g_s sur les actions admissibles pour chaque état $g_s : A(s) \to A'(f(s))$

[20] Semi-Markov Decision Processes, SMDP en anglais, nous réévoquerons les SMDP dans la dernière partie de ce chapitre.
[21] Dans la définition originale, on trouve un paramètre supplémentaire utile pour les Processus de Décision Semi-Markovien.

– un tuple de surjections $\langle f, g_1, g_2, \ldots, g_n \rangle$, $n \in |S|$
– une fonction pour les couples (états,actions) admissibles $h(s,a) : \psi \to \psi'$ définie par

$$h(s,a) = (f(s), g_s(a)) \text{ telle que } \forall s \in S, s' \in S', a \in A(s) :$$

$$P'(f(s), g_s(a), f(s')) = \sum_{s'' \in \{S'\}_f} P(s,a,s'') \tag{3.4}$$

$$R'(f(s), g_s(a)) = R(s,a) \tag{3.5}$$

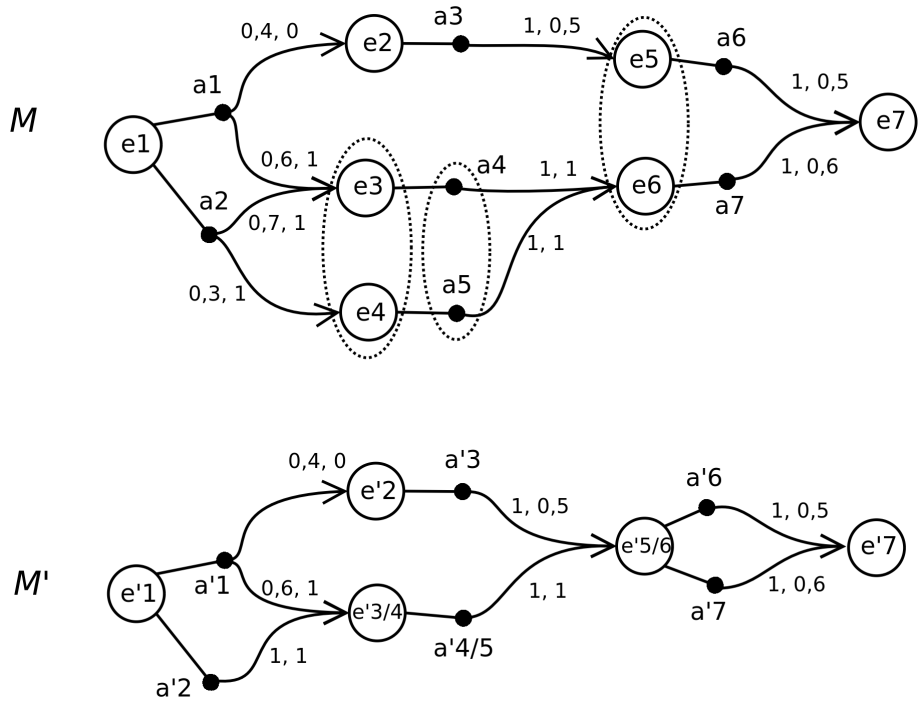


FIG. 3.3 – Exemple d'homomorphisme de PDM

Exemple 3.4 *La figure 3.3 donne un exemple d'homomorphisme de PDM. On a par exemple* $f(e_3) = f(e_4) = e'_{3/4}$ *et* $g_3(a_4) = a'_{4/5}$.
Ainsi, $h(e_3, a_4) = (f(e_3), g_3(a_4)) = (e'_{3/4}, a'_{4/5})$.
De plus, on a bien $P'(e'_{3/4}, a'_{4/5}, e'_{5/6}) = P(e_3, a_4, e_6) = 1$ *et*
$P'(e'_1, a'_2, e'_{3/4}) = P(e_1, a_2, e_3) + P(e_1, a_2, e_4) = 1$.
Finalement : $R'(e'_{3/4}, a'_{4/5}, e'_{5/6}) = R(e_3, a_4, e_6) = R(e_4, a_5, e_6) = 1$

La préservation du voisinage pour l'homomorphisme de graphes est ici traduite par une préservation des propriétés de la fonction de transition qui à chaque état associe une probabilité de passer dans un autre état en ayant sélectionné une action. Ainsi, il s'agit ici d'une préservation des probabilités de passage d'état à état.

Ces propriétés pour la généralisation sur les états se retrouvent pour la généralisation sur les actions par l'ensemble de surjection sur les couples (actions-états) valides $\langle f, g_1, g_2, \ldots, g_n \rangle$.
Les propriétés 3.4 et 3.5 à vérifier sur la préservation de la fonction de transition et la préservation des fonctions de récompenses impliquent une conservation de la dynamique du système. Elles donnent en fait les contraintes nécessaires pour que la fusion d'états et d'actions respectent la fonction de transition $T : S \times A \times S$.

L'homomorphisme de PDM est plus général que la minimisation de modèle vue précédemment dans le sens où l'on peut toujours transcrire la minimisation de modèle par bisimulation comme un

homomorphisme. La réciproque est fausse comme montré dans l'article original. L'homomorphisme est notamment adéquat pour représenter des structures de symétrie dans le modèle de l'environnement ou la répétition d'un ensemble d'états équivalents à une fonction près (plusieurs pièces similaires orientées différemment desquelles l'agent doit sortir par exemple). Un apprentissage sur le modèle minimisé a la même propriété intéressante de diminuer d'autant le nombre d'interactions nécessaires pour converger vers une politique optimale.

Ces méthodes exactes, gourmandes d'un point de vue du calcul et exigeant la connaissance du modèle de l'environnement, sont surtout intéressantes par le fait qu'elles posent le cadre algébrique pour la généralisation. C'est dans ce même cadre que nous situerons nos travaux dans les chapitres suivants, en nous replaçant toutefois dans le cadre agent que nous avons fixé.
Cependant, ces techniques font l'objet de travaux pour soulager les contraintes qu'elles posent. Notons par exemple ([Ravindran and Barto, 2004]) qui propose d'assouplir la notion d'homomorphisme.

3.4 Apprentissage par Renforcement Relationel

Nous avons précédemment vu la généralisation pour l'apprentissage par renforcement par *approximation de fonctions* ainsi que par *abstraction algébrique des états*. Nous allons maintenant développer un autre cadre de généralisation : *l'apprentissage par renforcement relationnel*. Contrairement à la première méthode, basée sur une généralisation numérique, l'apprentisage par renforcement relationnel conserve l'aspect symbolique des représentations de l'agent dans sa généralisation. Ce champ d'investigation est plus récent et très actif ; les articles fondateurs ([Dzeroski et al., 1998] et [Dzeroski et al., 2001]) datent de 1998 et 2001 ; de plus, un atelier lui a été consacré lors de la conférence ICML 2004 ([Tad, 2004]) renouvelé lors de la session 2005 de la même conférence. La partie 3 de [Kersting, 2006] lui est également consécrée. Son objectif est d'apporter le cadre bien étudié de la Programmation Logique Inductive (PLI)[22] à l'apprentissage par renforcement ; ceci dans le but notamment d'enrichir le langage de description des états à des expressions relationnelles, de pouvoir utiliser de la connaissance *a priori*, de généraliser en utilisant le mécanisme de liaison des variables inhérent au langage relationnel et de pouvoir réutiliser de la connaissance apprise sur d'autres tâches. Enfin, c'est une façon de relier le domaine de l'apprentissage par renforcement au domaine de la planification. Nous commencerons par développer les motivations de l'apprentissage par renforcement relationnel, puis nous le présenterons formellement, en nous appuyant principalement sur l'article ([Dzeroski et al., 2001]). Nous montrerons notamment le lien fait entre apprentissage par renforcement et planification. Nous poursuivrons en indiquant les avancés les plus importantes dans ce domaine.

Cette partie décrit les techniques de bases de l'apprentissage par renforcement relationnel. Pour plus de détails, se référer à [Dzeroski et al., 2001] ou [Dzeroski et al., 1998]. Pour une introduction et l'ensemble des formalismes concernant la Programmation Logique Inductive, voir [Muggleton, 1992, Muggleton et al., 1994, Lavrac and Dzeroski, 1993].

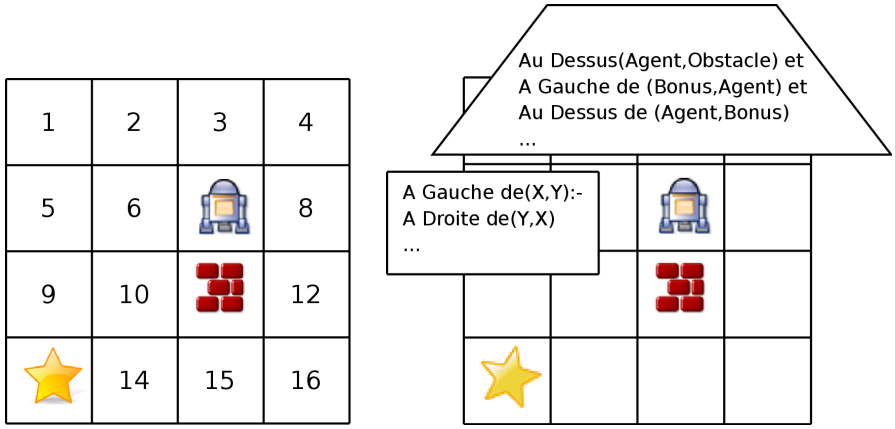
3.4.1 Description relationnelle de l'environnement

Une des formes la plus utilisées pour la représentation des états dans l'apprentissage par renforcement est le vecteur d'attributs multivalués. Un des problèmes de cette forme de représentation est son manque d'expressivité. Notamment, si l'on peut décrire les qualités d'un objet, il est impossible ou très peu commode de décrire les relations concernant celles-ci. Le langage relationnel, permet lui de décrire les relations entre ces attributs. Dans notre contexte les objets sont des états de l'environnement. Grâce au langage relationnel, nous allons pouvoir les décrire à l'aide de relations entre les différents éléments. Formellement, la description en langage relationnel est une description en logique du premier ordre.

Chaque état de l'environnement est décrit en utilisant des conjonctions de termes fondés (ne contenant pas de variables). On fait l'hypothèse du monde clos, c'est à dire que tout fait qui n'est

[22] Inductive Logic Programming (ILP) en anglais

pas explicitement cité dans la description est supposé faux.



(a): Description de l'environnement par extension

(b): Description de l'environnement relationnelle

FIG. 3.4 – Illustration de la différence entre la représentation par extension (à gauche) et la représentation relationnelle (à droite).

Exemple 3.5 *Si nous reprenons l'exemple du classique monde grille, une représentation de tabulaire des états de l'environnement se fera comme sur la figure 3.4 (a). Chacun des états est représenté par extension, c'est à dire par une étiquette différente de toutes les autres étiquettes des autres états (ici une numérotation). De plus, ces étiquettes n'ont aucun lien entre elles.*
Dans une représentation relationnelle (figure 3.4(b)), les états sont décrits par une conjonction de formules logiques (à l'intérieur du trapèze). De plus, les environnements peuvent être munis d'une base de connaissances (à l'intérieur du rectangle) permettant d'étendre la description de l'environnement par des implications logiques. Par exemple, la symétrie gauche-droite et traduite par les formules `A_Droite(X,Y) :-A_Gauche(Y,X)` *et* `A_Gauche(X,Y) :-A_Droite(Y,X)`*. Ainsi, même si la description de l'environnement n'inclut pas le fait* `A_Droite(Agent,Bonus)`*, ce fait est déduit à l'aide du fait* `A_Gauche(Agent,Bonus)` *et de la formule* `A_Gauche(X,Y) :-A_Droite(Y,X)`

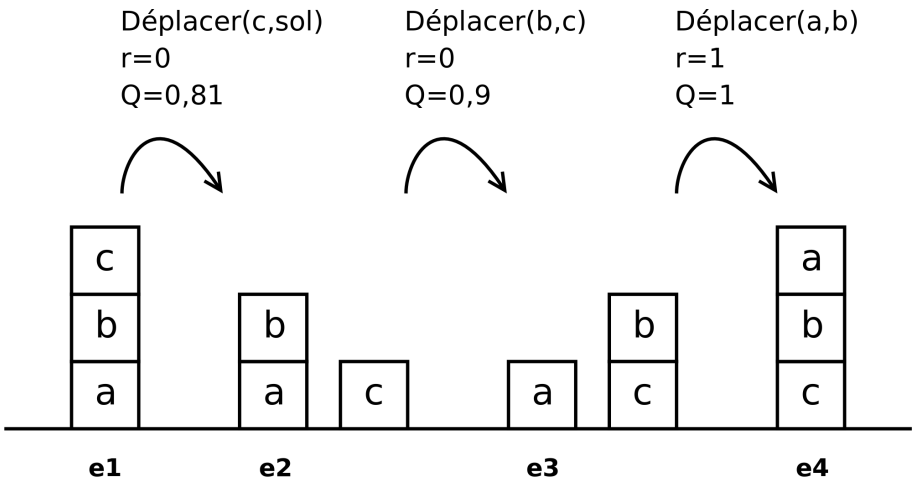


FIG. 3.5 – Exemple d'un monde des blocs à 3 blocs où l'objectif est par exemple d'empiler les blocs c, b et a (dans cet ordre). Un agent ne peut déplacer un bloc que si celui-ci n'a aucun bloc au-dessus. Il peut alors le mettre sur une pile déjà constituée ou sur le sol

Exemple 3.6 *Prenons l'exemple classique du domaine de l'apprentissage par renforcement relationel : le monde des blocs. Il s'agit comme suggéré figure 3.5 d'envisager un monde composé de blocs que l'agent peut empiler ou dépiler. Chaque état est décrit comme une liste de termes fondés. Par exemple, pour la figure 3.5, la description du deuxième état est :*
`rien_sur(b), sur(b,a), sur(a,sol), rien_sur(c), sur(c,sol)`

Base de connaissances Une des propriétés intéressante et bien connue du langage relationnel est la possibilité d'intégrer des connaissances *a priori* sur le domaine de la tâche à effectuer sous la forme d'une base de connaissances. Les connaissances *a priori* peuvent être des concepts connus comme pertinents par des experts du domaine, ou découverts lors de précédentes expériences. Plus généralement, elles peuvent représenter une certaine « intuition » qu'auraient les expérimentateurs sur des éléments de résolution de la tâche.
Une base de connaissances est représentée sous forme d'un ensemble de formules logiques. Celles-ci peuvent être des constantes, des faits toujours vrais ou bien des concepts utiles représentés sous forme d'implications logiques. Nous verrons que la possibilité de formaliser une connaissance *a priori* sur le domaine constitue l'une des deux avancées importantes de l'utilisation du langage relationnel pour l'apprentissage par renforcement. En effet, si la convergence de l'apprentissage d'une politique pour une tâche par renforcement dans les conditions classiques est mathématiquement assurée sans connaissances *a priori* sur la tâche, le traitement de cas non triviaux nécessite l'introduction de biais.

Exemple 3.7 *La base de connaissances du monde des blocs présentée dans [Dzeroski et al., 2001] contient par exemple les formules :*
`audessus(X,Y) :- sur(X,Y).`
`audessus(X,Y) :- sur(X,Z),audessus(Z,Y).`
Permettant d'exprimer le fait qu'un bloc soit situé au dessus d'un autre dans une pile.
On y trouve également le prédicat :
`nombredeblocsssur(X,N)`
Représentant le fait qu'il y a exactement `N` *blocs sur le bloc* `X`*. Les auteurs montrent expérimentalement que l'utilisation de ce concept accroit de façon significative l'efficacité de l'apprentissage, voire la faisabilité de celui-ci pour les algorithmes particuliers qu'ils utilisent.*

Arbres de régression logiques - fonction $Q(e,a)$ Comme dans toute forme de généralisation dans l'apprentissage par renforcement, l'ensemble des couples $(état, action) \in E \times A$ n'est pas représenté pour $\hat{Q}(e,a)$. Pour l'apprentissage par renforcement relationnel, il s'agit de représenter la fonction $\hat{Q}(e,a)$ à l'aide d'un arbre de régression logique.

	Exemple 1	Exemple 2	Exemple 3	Exemple 4
Valeur	qvaleur(0,81)	qvaleur(9)	qvaleur(1)	qvaleur(0)
Action	déplacer(c,sol)	déplacer(b,c)	déplacer(a,b)	déplacer(a,sol)
Objectif	objectif(sur(a,b))	objectif(sur(a,b))	objectif(sur(a,b))	objectif(sur(a,b))
Etat	rien_sur(c)	rien_sur(b)	rien_sur(a)	rien_sur(a)
	sur(c,b)	rien_sur(c)	rien_sur(b)	sur(a,b)
	sur(b,a)	sur(b,a)	sur(b,c)	sur(b,c)
	sur(a,sol)	sur(a,sol)	sur(a,sol)	sur(c,sol)
		sur(c,sol)	sur(c,sol)	

FIG. 3.6 – Description des couples *(état,action)* de la figure 3.5

Habituellement, les *arbres de régression* sont une structure de données servant à partionner un ensemble d'objets selon leurs propriétés. Les *arbres de régression logiques* sont leurs équivalents avec des exemples décrits à l'aide d'un langage relationnel. Ici, un exemple sera un quadruplet $(valeur, action, objectif, état) \in \mathbb{R} \times A \times E \times E$ décrit sous forme relationnelle. Les descriptions

font l'hypothèse du monde clos, c'est à dire que chacun des exemples est décrit complètement. Autrement dit, tout ce qui n'est pas spécifié dans une description est supposé faux. On notera que l'objectif de l'apprentissage fait partie de la description de l'état, ce qui n'est pas habituel. Nous verrons plus loin l'intérêt du point de vue de la généralisation. On notera également que la valeur du couple *(état,action)* fait également partie intégrante de la description.

Exemple 3.8 *La figure 3.6 page précédente montre les descriptions des quatre exemples décrits figure 3.5 page 60.*

Les définitions relatives aux arbres de régression classiques (propositionnels) peuvent être trouvés dans [Breiman et al., 1984]. Une étude complète sur cette structure et notamment une comparaison avec les arbres de régression classiques peut être trouvée dans [Blockeel and Raedt, 1998] ; voir également [Raedt and Blockeel, 1997] et [Kramer, 1996]. L'usage explicite pour l'apprentissage par renforcement relationnel est décrite dans [Dzeroski et al., 2001].

Donnons les caractéristiques importantes de cette structure de données :
- Chaque noeud interne est une requête prolog dont le résultat est `vrai` ou `faux`. Les arbres de décision logiques sont donc toujours des arbres binaires.
- Les variables portant le même nom sont partagées dans l'ensemble des noeuds de l'arbre. Nous verrons que c'est un élément permettant la généralisation sur les états.
- Les feuilles de l'arbre contiennent la valeur du couple *(état,action)* décrit par les noeuds du chemin menant de la racine à la feuille en question.

Données :
- Un ensemble d'exemples sous forme de quadruplets $(q - valeur, action, objectif, états) \in \mathbb{R} \times A \times E \times E$ décrits sous forme relationnelle.

Résultat :
- Un arbre de décision logique partitionnant les quadruplets $(q - valeur, action, objectif, états)$ et dont chacune des classes correspond à une $q - valeur$.

Algorithme 3.4: Entrées-Sorties de l'algorithme TILDE-RT ([Blockeel and Raedt, 1998])

La construction de tels arbres est faite à l'aide de l'algorithme TILDE-RT ([Blockeel and Raedt, 1998]), basé sur l'algorithme TDIDT et dont les entrées sorties sont rappelées algorithme 3.4.

```
root : objectif_sur(A,B),nombredeblocs(C),actiondeplacer(D,E)
sur(A,B) ?
+--oui : [0]
+--non : rien_sur(A) ?
         +--oui [1]
         +--non : rien_sur(E) ?
                  +--oui : [0.9]
                  +--non : [0.81]
```

FIG. 3.7 – exemple d'arbre de régression logique généré par l'algorithme TILDE-RT à partir des exemples de la figure 3.5

Exemple 3.9 *La figure 3.7 montre l'arbre de régression logique produit par les exemples de la figure 3.5 page 60 et décrits relationellement figure 3.6 page précédente.*

3.4.2 Q-Learning Relationnel

Nous venons de décrire la façon dont les états et les actions étaient décrits en apprentissage par renforcement relationnel, ainsi que la structure de données permettant de stocker la fonction $Q(e, a)$. Voyons maintenant le mécanisme d'apprentissage proprement dit.

L'utilisation d'une description relationnelle des couples $(état, action)$ se fait dans l'algorithme *Q-Learning Relationnel* (algorithme 3.5). Sa structure est similaire à celle du *Q-Learning* classique (algorithme 2.5 page 38) : initialisation de la politique, puis succession d'interractions avec l'environnement avec conjointement utilisation et amélioration de la connaissance de la fonction $Q(e, a)$. Notons cependant des différences :

- 1 : L'initialisation de la politique
- 2 : Boucle classique d'apprentissage par renforcement. Dans l'article original, la politique est une sélection $softmax$ ([Sutton and Barto, 1998]) avec une température $\tau = 1$. Notons que comme la fonction $\hat{Q}(e, a)$ n'est définie que pour les exemples déjà vus, le nouvel état e_{i+1} peut ne pas être défini.
- 3 : Cette partie de l'algorithme est la partie spécifique à l'apprentissage par renforcement relationnel. En effet, chaque épisode (succession d'interractions entre l'agent et l'environnement d'un état initial à un état final) est traité dans son ensemble comme pour les techniques de type Monte-Carlo ([Sutton and Barto, 1998]). Les récompenses sont alors distribuées à chaque couple $(état, action)$, selon la formule de répartition classique.
  La fonction $\hat{Q}(e, a)$ est reconstruite à l'aide de l'algorithme TILDE-RT en ajoutant les nouveaux couples $(état, action)$ et en actualisant les valeurs. Celui-ci n'étant pas incrémental, l'incrémentalité est simulée par reconstruction à partir de l'ensemble des exemples.

Données :
– Un environnement $\mathcal{E}nv$ dont la dynamique est modélisée par un Processus de Décision de Markov $\langle E, A, T, \psi, R\rangle$

début

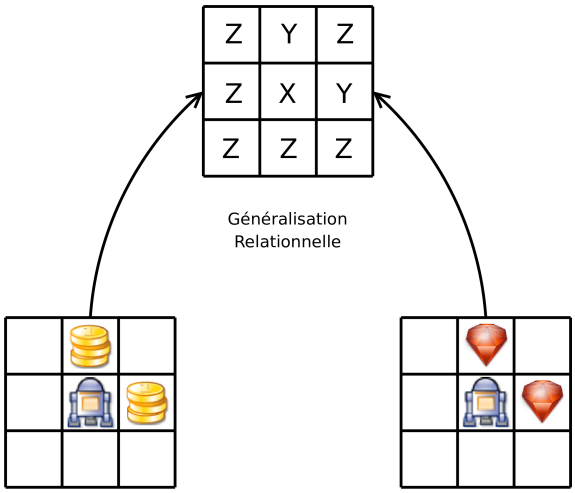
1 *Initialisation de la politique*
Initialiser $\hat{Q}_0(e, a) = v_0 \in \mathbb{R}$, $\forall (e, a) \in E \times A$;
Initialiser $Exemples \leftarrow \emptyset$;
pour *toujours* **faire**
$e \leftarrow e + 1$;
Initialisation d'un nouvel épisode
$i \leftarrow 0$;
$e \leftarrow e_0$;
2 **tant que** $e \notin E_T$ (e n'est pas le but) **faire**
$a_i \leftarrow$ Sélectionner une action en utilisant une politique basée sur $Q(e, a)$;
Exécuter (a_i, $\mathcal{E}nv$);
$r_i \leftarrow$ Réception d'une récompense($\mathcal{E}nv$);
$e_{i+1} \leftarrow$ Observation d'un nouvel état($\mathcal{E}nv$);
$i \leftarrow i + 1$;
fin
3 **pour** j **de** $i - 1$ **à** 0 **faire**
$\hat{q}_j \leftarrow r_j + \gamma \max_{a' \in A(e_{j+1})} \hat{Q}_e(e_{j+1}, a')$;
$x \leftarrow$ Générer un exemple $(e_j, a_j, \hat{q}_j)$;
si $(e_j, a_j, \hat{q}_{old}) \in Exemples$ **alors** $\hat{q}_{old} \leftarrow \hat{q}_j$;
Mise à jour de $\hat{Q}_e$ en $\hat{Q}_{e+1}$ en utilisant $Exemples$ et $TILDE - RT$;
fin
fin
fin

Algorithme 3.5: Q-Learning Relationnel

3.4.3 Généralisation

Voyons maintenant en quoi cet algorithme permet effectivement une généralisation des politiques pour l'apprentissage par renforcement.

Nous pouvons distinguer deux formes de généralisation dans l'apprentissage par renforcement relationnel : par la liaison des variables dans l'arbre de décision logique d'une part et dans l'utilisation de celui-ci sur les exemples non-vus d'autre part.



(a): Généralisation relationnelle

$d(e_1) =$
`NordOuest(case_vide,agent)`
`NordEst(case_vide,agent),`
`Nord(pièces,agent),`
`Ouest(pièces,agent),`
`Est(case_vide,agent),`
`SudOuest(case_vide,agent),`
`Sud(case_vide,agent),`
`SudEst(case_vide,agent)`

$d(e_2) =$
`NordOuest(case_vide,agent),`
`NordEst(case_vide,agent),`
`Nord(diamant,agent),`
`Ouest(diamant,agent),`
`Est(case_vide,agent),`
`SudOuest(case_vide,agent),`
`Sud(case_vide,agent),`
`SudEst(case_vide,agent)`

$Gen_{\mathcal{R}}(d(e_1)) = Gen_{\mathcal{R}}(d(e_2)) =$
$\exists$ `X,Y,Z` telles que
`NordOuest(Z,X),`
`NordEst(Z,X),`
`Nord(Y,X),`
`Ouest(Y,X),`
`Est(Z,X),`
`SudOuest(Z,X),`
`Sud(Z,X),`
`SudEst(Z,X)`

(b): Description relationnelle des états

FIG. 3.8 – Principe de la généralisation relationnelle : Les atomes des termes fondés (sans variables) composant les descriptions des états sont généralisés par des variables. Les états, dont les descriptions sont égales structurellement et différentes à une permutation prêt sur les constantes, ont la même généralisation relationnelle.

Généralisation par liaison des variables Le mécanisme de généralisation par liaison des variables représente sûrement l'apport déterminant de la représentation relationnelle pour l'apprentissage par renforcement. Il s'agit, comme le suggère la figure 3.8, de généraliser les égalités entre les constantes d'une description. Pour ce faire, on utilise les propriétés de liaison de variables d'une formule logique. Plus formellement :

Définition 3.7 (Généralisation relationnelle)
Soit un objet e décrit par la description $d(e)$ composée d'une conjonction de termes fondés p_i. La généralisation relationnelle de la description de e notée $Gen_{\mathcal{R}}(d(e))$ est construite en remplaçant les *atomes* de $d(e)$ par des *variables* telles que $d(e)$ soit une instance de $Gen_{\mathcal{R}}(d(e))$ et qu'il y ait une bijection entre l'ensemble des constantes de $d(e)$ et l'ensemble des variables de $Gen_{\mathcal{R}}(d(e))$.

Il suffit de reprendre l'ensemble des termes fondés (termes n'ayant pas de variables) de $d(e)$ et de remplacer chaque atome par une variable en remplaçant le même atome toujours par la même variable.
Notons que les variables ainsi introduites sont par définition quantifiées existentiellement sur l'ensemble des atomes.

Exemple 3.10 *Prenons un état de l'environnement issu du monde des blocs (figures 3.5 page 60 et 3.6 page 61) :*
$d(e_1)$=`{objectif_sur(a,b), rien sur(b), rien sur(c), sur(b,a), sur(a,sol), sur(c,sol)}`.
$Gen_{\mathcal{R}}(d(e))$ = $\exists$ `W,X,Y,Z,` *telles que* `objectif_sur(W,X), rien sur(X), rien sur(Y), sur(X,W), sur(W,Z), sur(Y,Z).`

Ainsi, si l'agent sait comment agir dans un état de l'environnement e_1 et détecte une situation similaire e_2, avec seulement un changement dans les constantes utilisées mais avec la même structure de description, l'agent agira dans e_2 comme il aurait agit dans l'état e_1.
Dans le monde des blocs, cette généralisation est très utile. En effet, la tâche d'empiler le bloc a sur le bloc b ou le bloc c sur le bloc d correspond à la même politique optimale, à une substitution de constantes prêt.
C'est ici qu'on voit l'intérêt d'inclure dans la description de chacun des états l'objectif de la tâche. l'arbre ne contient aucune variable instanciée. Par exemple, si l'agent apprend avec l'objectif `sur(a,b)` qui consiste à poser le bloc a directement sur le bloc b, cet état ne sera jamais représenté en tant que tel dans l'arbre. il sera représenté sous la forme `objectif_sur(W,X)`, qui représente aussi bien l'état `sur(a,b)` que `sur(b,a)` ou que `sur(c,b)`. Que le `sur(W,X)` corresponde bien au but recherché vient du fait que les variables portent le même nom que `objectif_sur(W,X)`.

Exemple 3.11 *La figure 3.8 page précédente propose un exemple proche de notre monde grille montrant l'intérêt d'une telle généralisation sur les états et les actions.*

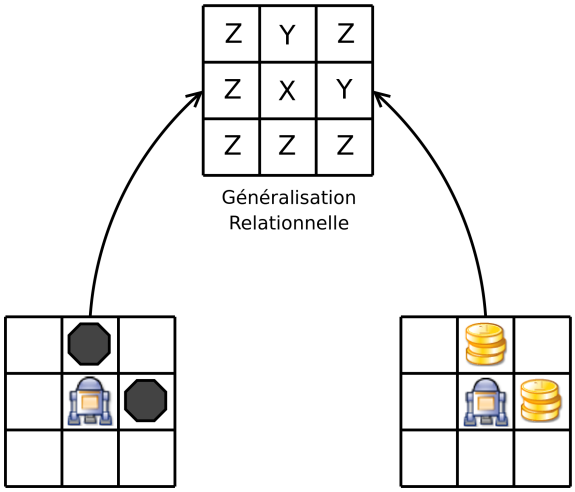


FIG. 3.9 – Exemple avec lequel la généralisation relationnelle donne une mauvaise politique. En effet, si dans l'état de droite, une des actions optimales est d'aller soit au nord, soit à l'est, pour se retrouver sur un trésor, dans l'état de gauche, cette politique conduit l'agent dans un gouffre

Cependant, pour d'autres tâches, cette généralisation peut s'avérer inutile voire dangereuse. Considérons l'exemple illustré figure 3.9. Il est clair que la confusion entre les deux états de l'environnement e_1 et e_2 peut conduire à des politiques non-optimales.
Comme toujours en apprentissage, on traite ce type de problème avec un langage de description approprié. Dans ce cas, il faut rajouter des éléments de descriptions permettant de différencier les généralisations de e_1 et e_2. Par exemple, on peut rajouter, dans la base de connaissances ou dans

la description, les propositions suivantes :
`danger(gouffre)`
`bonus(trésor)`
Ainsi, $Gen_{\mathcal{R}}(d(e_1)) \neq Gen_{\mathcal{R}}(d(e_2))$.

Le problème n'est pas pour autant complètement résolu. En effet, ces deux cas sont maintenant totalement différenciés alors que dans certaines circonstances, il pourrait être utile de les considérer similaires (par exemple pour les considérer en tant qu'obstacles).

Nous pointons là une limite de ce système de généralisation qui est très efficace quand l'environnement s'y prête. C'est à dire quand la politique optimale pour la tâche donnée ne dépend que de critères structurels, mais constitue une généralisation abusive dans le cas contraire. Nous verrons que ce sont des problèmes que nous retrouvons dans nos propres travaux ultérieurement.

Généralisation par arbre de régression logique La deuxième généralisation effectuée par l'algorithme RRL est une conséquence de l'utilisation des arbres de régression logiques comme structure de données pour stocker la fonction $Q(e, a)$.
Contrairement à l'algorithme basique de *Q-learning*, tous les couples possibles *(état,action)* ne sont pas présents dans cette représentation. Cependant, comme la fonction $\hat{Q}(e, a)$ est utilisée sur des couples qui potentiellement ne sont pas représentés dans l'arbre, cela constitue une généralisation des états vus sur les états non vus.

Comme nous venons de le voir sur l'exemple, un exemple non-vu est traité comme s'il appartenait à l'arbre. Rappelons que pour trouver la valeur d'un couple *(état,action)*, il s'agit de tester chaque noeud de l'arbre contenant une formule bien formée en commençant par la racine. Pour chaque noeud, si la formule bien formée composée par la conjonction des noeuds atteints est une instance du couple *(état,action)* à tester, la descente dans l'arbre continue. Sinon, la valeur de l'exemple est trouvée. La racine contenant l'objectif à atteindre, elle est nécessairement vraie sur l'ensemble des couples *(état,action)*, même sur ceux qui n'ont pas été vus. Assurant ainsi que $\hat{Q}(e, a)$ peut retourner une valeur pour tout couple *(état,action)*, même ceux qui n'ont pas encore été rencontrés.

La structure de l'arbre partitionne l'ensemble des exemples vus, mais également des exemples non-vus.

Les noeuds de test de l'arbre ainsi que leur ordre ont donc une importance capitale pour le résultat de cette généralisation. Cependant, le choix des noeuds de l'arbre n'est pas fait avec l'objectif d'améliorer la généralisation sur les exemples. Les tests sont réalisés avec, comme critères des propriétés de séparabilité sur l'ensemble, des exemples utilisés pour la construction de l'arbre.

Il y a donc implicitement une hypothèse sur la distribution des exemples qui suppose que les termes, ayant la meilleure qualité de classification pour les exemples, conservent cette propriété sur les exemples non vus.

Cependant, deux éléments font que cette généralisation n'intervient que peu dans le résultat des algorithmes. Premièrement, les épisodes non vus sont entrés dans $\hat{Q}(e, a)$ dès qu'un épisode est terminé ; une mauvaise utilisation de cette généralisation sur un état de l'environnement n'a donc d'effet que durant un épisode. Deuxièmement, l'incrémentalité de l'algorithme de construction de l'arbre est simulée, *TILDE-TD* n'étant pas incrémental. Par conséquent, les valeurs des critères de séparabilité pour les termes composant les noeuds de l'arbre sont réactualisés à chaque épisode.

Néanmoins, la difficulté sous-jacente est réelle. En effet, les extensions à venir des algorithmes par renforcement utilisant une description relationnelle de l'environnent devront nécessairement éviter de stocker l'ensemble des états de l'environnement pour pouvoir traiter des problèmes non-académiques, et ce même compte tenu de la généralisation effectuée par la liaison de variables. La question du traitement des états non vus, c'est à dire de la généralisation, se reposera alors. Comme pour les autres formes de représentation de la fonction $Q(e, a)$, le problème de la sélection des motifs pertinents (ici les termes) reste à considérer.

Ce champ de recherche est probablement le plus actif aujourd'hui pour l'apprentissage par renforcement, peut-être plus d'ailleurs par le fait qu'il propose une expression plus riche pour la description des états de l'environnement que par les possibilités de raisonnement automatique qu'offre le lien avec le domaine de l'ILP. Il n'est pas dans l'objectif du présent travail de faire un état de l'art exhaustif sur le sujet. Néanmoins, nous renvoyons le lecteur sur les publications issues des deux ateliers [Dri, 2005] et [Tad, 2004] sur la question et notamment sur l'état de l'art [Tadepalli et al., 2004].

3.5 Domaines corrélés à la question de la généralisation

Nous venons de voir différentes approches pour la généralisation des états ou des politiques pour l'apprentissage par renforcement, il existe d'autres techniques qui ne s'inscrivent pas explicitement dans ce cadre, mais qui y sont fortement correlées. Nous allons donc rapidement évoquer les travaux suivants : [Reynolds, 2002, Reynolds, 2000] pour ce qui est d'une représentation à échelle variable de la fonction $Q(e, a)$, [Munos, 2003, Munos and Moore, 2002, Munos, 1997a, Munos, 1997b] pour le traitement du **cas continu**, [McCallum, 1996b, McCallum, 1996a] pour le traitement des **algorithmes à mémoire** et [Sutton et al., 1999, Dietterich, 2000, Parr and Russel, 1997, Barto and Mahadevan, 2003] pour l'**abstraction temporelle**.

Représentation continue de l'environnement Nous avons jusqu'ici considéré que l'environnement dans lequel évoluait l'agent était modélisé par un Processus de Décision de Markov, dont les états sont discrets et par conséquent la succession des actions que l'agent sélectionne à chaque étape de temps. Cependant, nombre de problèmes posés sont des environnements où une telle discrétisation n'est pas naturelle. On peut ici citer de nombreux exemples tests comme l'acrobot ([DeJong and Spong, 1994, Boone, 1997, Sutton and Barto, 1998, Munos, 2001]), le pendule renversé ([Munos, 2001, Boone, 1997, Sutton, 1996]), la voiture devant atteindre le sommet d'une colline ([Munos, 2001, Sutton and Barto, 1998]) et plus généralement, nombre d'applications en robotique.
Le cas continu, pour l'apprentissage par renforcement, est défini par le fait que l'ensemble des états de l'environnement n'est pas composé d'un ensemble discret comme nous l'avons proposé jusque là, mais d'un ensemble continu. Notamment, l'ensemble des états peut être une surface continue sur laquelle se situe l'agent. L'ensemble des actions peut également être considéré comme un ensemble continu. Dans ce cas, l'ensemble des directions que peut prendre l'agent constitue l'ensemble des actions possibles.
Historiquement, cette question est à rattacher à la discipline du contrôle optimal, qui elle-même est rattachée aux mathématiques et qui considère un tel cadre avec des différences notables avec le nôtre : 1) l'environnement doit être parfaitement connu, ce qui est en contradiction avec le modèle agent 2) la dynamique de l'environnement est déterministe, alors que les PDM sont des processus stochastiques.
Ainsi, les travaux de Munos ([Munos and Moore, 2002]) considèrent la question du contrôle optimal mais procèdent par une discrétisation de l'ensemble des états transformant ainsi un problème de contrôle optimal en problème de résolution de PDM.
Le lien avec la généralisation vient du fait qu'il utilise un processus en spécialisation, découpant autant que nécessaire l'ensemble des états de l'environnement. La question de l'opportunité de diviser un ensemble continu d'états se pose, ce qui est un corollaire de la problématique de la généralisation. Nous reviendrons sur cette question chapitre 6 quand nous poserons la question des critères de division pour un ensemble d'états.
De plus, la structure de la fonction $Q(e, a)$ proposée est faite sous forme de *triangulation de Kuhn* ([Davies, 1997]), pour représenter une généralisation de précision variable.
Pour ce qui est de notre problèmétique, le principal désavantage des techniques exposées est qu'elles se basent sur deux mesures, l'influence et la variance qui ne sont calculables qu'avec une connaissance de la dynamique de l'environnement, ce que nous avons exclu.

Le même problème a également été traité par Reynolds ([Reynolds, 2002, Reynolds, 2000]), mais dans le cadre de l'apprentissage par renforcement. Il s'agit là d'avoir une précision variable de la fonction $Q(e, a)$ en utilisant des *k-tree* comme structure de représentation.

Là encore, des problèmes similaires à la question de la généralisation sont posés pour les mêmes raisons que celle évoquées ci dessus.

Algorithmes à mémoire Dans les travaux de Mc Callum ([McCallum, 1996b]), il s'agit de considérer le cas des Processus de Décisions de Markov Partiellement Observables (PDMPO). Nous avons indiqué, que l'agent n'avait pas accès au modèle de son environnement. Il n'a donc pas accès aux états de l'environnement, mais à une observation de ceux-ci. Dans les PDMPO, plusieurs états peuvent potentiellement produire une observation similaire pour l'agent. Ainsi, celui-ci sera incapable de distinguer un état d'un autre dans certains cas. Il existe une vaste littérature à ce sujet, [Lovejoy, 1991] et [Bertsekas, 1976] fournissent une introduction à ce sujet. Ce formalisme convient particulièrement bien à une vision agent, et surtout multi-agents (voir [Bernstein et al., 2002] et de nombreux travaux de l'équipe Maïa du Loria.

Pour résoudre ce problème, Mc Callum propose des algorithmes à mémoire, permettant de distinguer des états produisant la même observation à l'aide de la succession récente des états observés par l'agent. Notons que ces travaux font toujours l'objet d'une recherche active, [Shani et al., 2005] par exemple.

Ce problème a un étroit rapport avec celui de la généralisation. En effet, il se pose, pour les POMDP, la question de savoir si une même observation correspond à plusieurs états. Cela se fait en analysant les valeurs observées relativement à celle-ci.
Dans le cas de la généralisation, il se pose la question de savoir si deux états peuvent être fusionnés sans conséquence. Dans ce cas, on considère de la même manière leurs observations, et on cherche si ceci est problématique par rapport aux valeurs observées.
Dans un cas, on cherche donc à vérifier si plusieurs états sont fusionnés, dans le second, on cherche à vérifier s'il est sans conséquence de fusionner plusieurs états.

Abstraction Temporelle Pour un état de l'art et une synthèse sur la question de l'abstraction temporelle, on se référera à [Barto and Mahadevan, 2003]. Pour présenter rapidement cette question, on notera que le formalisme de l'apprentissage par renforcement pose le fait que l'agent interagisse avec son environnement régulièrement. C'est bien la nature séquentielle des prises de décision qui est essentielle, et non pas le temps écoulé entre chacune d'elle.
Ainsi, plutôt que de sélectionner une action, un agent peut sélectionner une sous-politique, ou un comportement correspondant à un sous-objectif. Par exemple, pour une tâche consistant à ouvrir une porte fermée à clé, plutôt que de considérer la tâche dans son ensemble, il s'agit de décomposer la tâche en « trouver la clé » puis « aller à la porte ». On peut aussi voir cette conception comme l'utilisation de macro-actions.

Le formalisme sous-jacent de ces techniques sont les *Processus de Décision Semi-Markovien* (SMDP). Ceux-ci incluent une variable temporelle de durée indéterminée pour une action sélectionnée, durée qui est en fait celle de la succession des actions qui constituent la macro-action. Il existe trois principaux travaux menés en parallèle dans cette optique [Sutton et al., 1999] avec le concept d'*options*, [Parr and Russel, 1997, Parr, 1998] avec le concept de *Hiérarchies de Machines Abstraites* (HAM) et [Dietterich, 2000] avec un cadre nommé *MAXQ*.

Cette problématique enrichie considérablement l'apprentissage par renforcement, notamment parce qu'elle fait un pont (de manière différente que celui que nous avons déjà cité *Dyna-Q*) avec la planification. La recherche est très active sur le sujet [Diuk et al., 2006] par exemple, avec la question des options multiples ([Mehta and Tadepalli, 2005]) qui permet à un agent de se donner plusieurs buts parallèles simultanément et la découverte automatique des sous-objectifs ([Asadi and Huber, 1994, Asadi and Huber, 1995] par exemple).

Ce problème est également corrélé à celui de la généralisation. En effet, une sous-politique est une politique applicable à des ensembles d'états similaires. Il faut donc détecter que des états sont similaires et en quoi ils le sont à l'exclusion des autres.

Notons que ce problème est aussi vu sous l'angle du transfert des connaissances, c'est à dire le transfert de politiques apprises à d'autres problèmes similaires dans [E.Taylor and Stone, 2007].

3.6 Conclusion

Nous avons vu dans ce chapitre différentes formes de généralisation sur les états de l'environnement pour l'apprentissage par renforcement : généralisation par approximation de fonction, généralisation algébrique sur le modèle et apprentissage par renforcement relationnel. De plus, nous avons évoqué d'autres formes importantes de généralisation : le traitement du cas continu ainsi que les algorithmes à mémoire que nous n'avons pas développés. Pour finir, nous avons évoqué d'autres formes de généralisation, comme la généralisation temporelle qui est fortement corrélée à la généralisation sur les états.

La figure 3.10 page 71 montre un résumé des méthodes que nous avons développées. Nous y avons rajouté nos propres méthodes pour pouvoir nous y référer ultérieurement.

La question de la généralisation des politiques pour l'apprentissage par renforcement est *de facto* un sujet primordial pour cette discipline, car elle conditionne en effet les principales qualités qui sont exigibles d'un apprentissage artificiel : réutilisabilité du résultat, transfert de connaissances ou utilisation de connaissances *a priori*,...
Outre les difficultés techniques ou mathématiques propres à chacune des méthodes évoquées ou développées, on retrouve dans chacune d'elles les problèmes transversaux à l'apprentissage artificiel :

Choix du langage de description Cette question est primordiale pour tout type d'apprentissage artificiel. Pour l'approximation de fonctions, elle se pose sous la forme des motifs utilisés, pour l'apprentissage par renforcement relationnel, c'est l'objectif même de cette méthode que d'enrichir le langage de description. Pour la généralisation par abstraction algébrique, cette question est moins présente. Ceci est dû au fait de travailler directement sur la dynamique de l'environnement. Cette question est directement liée au *compromis biais – variance* de la théorie générale de l'apprentissage artificiel.

Structure de l'espace de recherche Là encore, il s'agit d'une question classique de la théorie générale de l'apprentissage artificiel. Pour l'approximation de fonctions, c'est la forme de la fonction ainsi que la formule de mise à jour qui influent sur le parcours au sein de l'espace de recherche. Pour l'apprentissage par renforcement relationnel, tous les travaux concernant l'ILP sur la structure des espaces de généralisation traitent de cette question. Pour l'abstraction algébrique des états, dans les travaux de [Givan et al., 2003], on se retrouve avec un parcours systématique de l'espace de recherche, ce qui implique d'ailleurs les problèmes de complexité de cette méthode. Pour ce qui est des travaux de [Ravindran and Barto, 2003b], il en va de même, toutes les possibilités ne sont pas explorées compte tenu de la taille des espaces de recherche. Cependant, vouloir tester la symétrie par exemple plutôt qu'une autre propriété revient in fine à structurer l'espace de recherche. En fait, la distinction faite entre exploration syntaxique et exploration sémantique faite en apprentissage artificiel classique est également applicable ici.

Connaissance *a priori* Cette question est très liée aux deux premiers points. En effet, des motifs « adéquats » pour décrire les états de l'environnement peuvent être considérés comme de la connaissance a priori. De même, la base de connaissances importée pour l'apprentissage par renforcement relationnel influence le parcours de l'espace de recherche. Cependant, si la question du transfert des connaissances est importante, on voit que pour l'approximation de fonction comme pour l'abstraction algébrique, l'apport de connaissances a priori est faible ou au moins limitée.

Choix des exemples Si les premiers points mentionnés peuvent être traités de manière assez classique, en les rapprochant de la théorie générale de l'apprentissage artificiel, la question du choix des exemples est cruciale pour l'apprentissage par renforcement. En effet, le cadre que nous

nous somme donné est celui d'un agent pouvant dans le même temps utiliser une politique pour agir sur son environnement et utiliser les retours de son environnement pour améliorer celle-ci. Ceci implique que l'utilisateur des algorithmes n'a pas la possibilité d'influencer de manière directe la succession des exemples de situation que devrait traiter l'agent. En effet, c'est la politique de l'agent dans le modèle qui détermine l'arrivée des expériences. Cette politique dépend principalement de la façon dont l'algorithme met à jour $Q(e, a)$ en fonction des expériences ainsi que de la façon dont est géré le compromis exploration-exploitation. De plus, dans le cadre fixé, l'agent ne connaît pas son environnement. A fortiori, il ne connaît pas l'ensemble des états de l'environnement et encore moins la façon d'obtenir des distributions choisies de ceux-ci. Il faut donc que les méthodes de généralisation pour l'apprentissage par renforcement soient robustes concernant cette question. En terme statistique, elles doivent pouvoir converger quelle que soit la distribution des exemples. Ceci explique d'ailleurs les conditions de convergence classiques qui sont que l'ensemble des couples (état,action) soit visité potentiellement infiniment souvent. Comme nous le verrons plus loin, ces dispositions particulières sont à l'origine de nombreuses difficultés pour les algorithmes d'apprentissage par renforcement. Notamment à cause d'un corollaire de cette question : comment évaluer son apprentissage si on ne connaît pas la qualité de son exploration ?

Après ce tour d'horizon, nous allons développer nos propres techniques de généralisation des politiques pour l'apprentissage par renforcement. Pour la nature symbolique des techniques utilisées, nos méthodes sont à rapprocher de l'apprentissage par renforcement relationnel. D'ailleurs, nos techniques peuvent utiliser, sous certaines conditions, une utilisation de description relationnelle de l'environnement. En ce qui concerne l'aspect algébrique de notre travail, c'est plutôt vers le formalisme d'abstraction algébrique des états qu'il faut se tourner. En effet, comme nous le verrons, l'ensemble des partitions des états de l'environnement constituera explicitement notre espace de recherche. Finalement, il faut voir un rapprochement avec les techniques présentées dans [Munos, 2001] ou même [Reynolds, 2001] dans le fait que nous ferons varier notre niveau de généralisation comme parcours de l'espace de recherche.

Le prochain chapitre va rappeler comment structurer un espace de généralisation à l'aide des treillis de Galois. Nous verrons comment appliquer cette technique à l'apprentissage par renforcement dans les deux chapitres suivants.

Comparaison des méthodes de généralisation pour l'apprentissage par renforcement

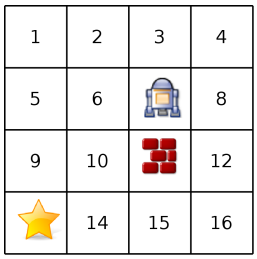
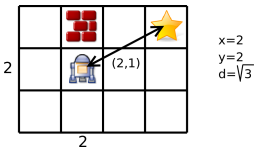
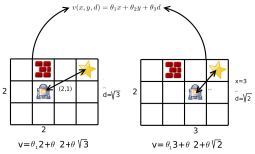
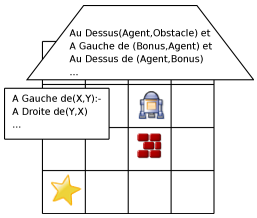
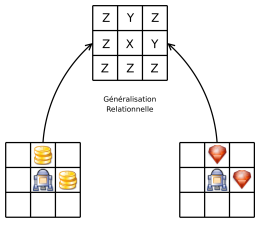
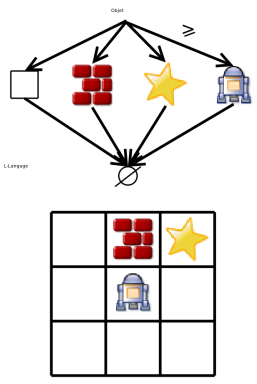
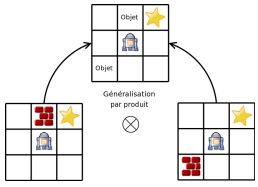
Types d'algorithmes	**Description des états de l'environnement**	**Exemple de description**	**Généralisation de l'apprentissage lors d'une mise à jour**
Algorithmes de base	Description en extension des états : $d(e_i) = e_i$		$\emptyset$
Algorithmes par approximation de fonctions	Description numérique des états : $d(e_i) = \{a_1 = v_1, \dots a_n = v_n\}$ avec chaque a_i étant un attribut à valeur dans $\mathbb{R}$		
Apprentissage par renforcement relationnel	Description des états en logique du premier ordre : $d(e_i) = f_1 \wedge \dots \wedge f_n$ avec chaque f_i étant un fait fondé en logique du premier ordre avec hypothèse de monde clos		
Q-Concepts Learning	Description des états dans un L-Langage : $d(e_i) = l$ avec $l \in L(\leq, \simeq, \bigotimes)$		

FIG. 3.10 – Comparaison des différentes méthodes de généralisation.

4 Généralisation à l'aide de Treillis de Galois

Nous avons vu, dans le chapitre précédent, des méthodes utilisées pour généraliser l'apprentissage dans le cadre de l'apprentissage par renforcement. Nos travaux portent également sur un apport théorique et algorithmique afin de proposer des méthodes de généralisation pour l'apprentissage par renforcement, basées sur des propriétés algébriques des langages de description des états et des actions ; les algorithmes proposés pouvant alors s'adapter à tout langage de description respectant ces propriétés. Au coeur de nos méthodes, nous utiliserons principalement la notion de treillis de Galois[23] afin de structurer l'ensemble des descriptions des états de l'environnement.

Les treillis de Galois sont de plus en plus employés en apprentissage machine. Leur étude est également connue sous le nom d'analyse formelle de concepts[24]. Un certain nombre d'applications ont été proposées ces dernières années utilisant des méthodes basées sur ce concept mathématique. Citons par exemple [Maille, 1999], [Zenou, 2004], ou plus récemment [Sébastien, 2007] pour une application directe à l'organisation d'une collection de photos étiquetée, [Duquenne, 2007] pour un application pédagogique à l'enseignement des mathématiques ou bien [Ducrou and Eklund, 2007] pour une assistance à la recherche sur internet. Cependant, dans le cadre de l'apprentissage par renforcement, ces notions n'ont pour l'instant pas été explorée.
Donnons les intérêts principaux qui nous amènent à considérer l'utilisation des treillis de Galois pour l'apprentissage par renforcement.

Généralisation Comme nous le verrons au chapitre 5, ils nous permettrons de formaliser la généralisation de l'apprentissage, avec le même type de procédé que les algorithmes bien connus d'espace des versions (voir notamment [Mitchell, 1997], chapitre 2).

Connaissance a priori Les algorithmes d'apprentissage par renforcement permettent d'obtenir un comportement adéquat, idéalement optimal, pour un agent effectuant une tâche dans un environnement donné. Comme nous l'avons vu chapitre 3, il est possible d'apporter de la connaissance a priori aux algorithmes d'apprentissage par renforcement, permettant ainsi d'accélérer l'apprentissage. Généralement, cette connaissance se fait sous la forme d'éléments de description des états déterminés par expertise, utiles, voire nécessaires pour l'apprentissage.
Nous verrons, chapitre 5 page 109 comment de tels concepts peuvent être amenés via le langage de description des états et des actions à l'aide des notions autour des treillis de Galois. Nous y verrons également que de nouveaux concepts utiles peuvent être trouvés au cours de l'apprentissage. Finalement, nous y montrerons également que nos méthodes permettent de trouver un sous-ensemble des langages de descrition des états et des actions, pertinent pour la tâche, suite à l'apprentissage.

Méthodes générales et incrémentales Le formalisme algébrique très général des treillis de Galois associé à la souplesse de la description des objets par l'utilisation de langages de généralisation ainsi que l'existence d'algorithmes incrémentaux argumentent dans le sens de leur utilisation pour l'apprentissage par renforcement. Toutefois, il est clair que les complexités tant spatiales que temporelles restent importantes pour l'utilisation et la contruction des treillis. Ceci nous amènera dans les chapitres suivants à adapter nos algorithmes d'apprentissage par renforcement plutôt que d'envisager une utilisation directe.

Partie 4.1, nous commencerons par donner les définitions autour des relations d'ordre partiel et de treillis nécessaires à notre propos.
Comme nous l'avons vu partie 3, une méthode de généralisation dans le cadre de l'apprentissage

[23] Du nom d'Evariste Galois, mathématicien (25 octobre 1811 [Bourg-la-Reine] - 31 mai 1832 [Paris]) dont la vie est surprenante quand on met en rapport sa courte durée et son contenu tant sur le plan mathématique que sur son engagement politique. Originellement, les correspondances de Galois qui auraient été développées la veille au soir de sa mort, avaient pour but la résolution par radicaux d'équations de polynômes. On trouvera notamment des éléments bibilographiques dans [Auzias, 2001]

[24] Formal Concept Analysis, FCA en anglais

par renforcement, consiste à regrouper des état similaires du point de vue de la tâche. Nous allons donc formaliser, partie 4.2, l'espace de l'ensemble des regroupements possibles sous forme de **treillis des parties**.
Cependant, les états de l'environnement dans un PDM ne sont généralement pas exprimés par extention, c'est à dire en énumérant, à l'aide d'un identifiant unique, chacun des états. Ceux-ci sont au contraire généralement décrits à l'aide d'un langage de description. Nous verrons d'abord, partie 4.3, des méthodes classiques pour classifier un ensemble d'objets décrits par des attributs en utilisant le **treillis de Galois d'une relation binaire**. Nous définirons la notion de treillis de Galois à l'aide d'une relation binaire entre un ensemble d'attributs et un ensemble d'objets.
Partie 4.4, nous rappellerons les éléments algébriques permettant de contruire un **treillis de Galois** dans le cadre général. Nous verrons ensuite, partie 4.5 comment appliquer ces notions à l'apprentissage artificiel, c'est à dire comment les descriptions possibles d'un ensemble d'objets peuvent se structurer avec la notion de langage de description muni d'un ordre partiel $\leq$ et d'un opérateur produit $\otimes$. Nous nommerons ces langages, **langages de généralisation** car ils nous permettront de décrire avec cohérence, l'ensemble des généralisations possibles d'un ensemble d'objets.
Nous généraliserons ensuite notre propos sur la classification d'un ensemble d'objets, afin d'étendre les représentations possibles engendrant un treillis de Galois. Nous verrons que la confrontation du treillis des parties d'un ensemble d'objets et d'un langage de généralisation correspond effectivement à l'objet mathématique **treillis de Galois**, que nous présenterons comme **l'ensemble des regroupements possibles contraints par un langage**. Nous montrerons en effet que tout langage de généralisation sur un ensemble associé à une relation de description, permet de formaliser un tel espace.
La généralisation des politiques pour l'apprentissage par renforcement pouvant être considérée comme nous l'avons vu dans le chapitre précédent comme un partitionnement des états de l'environnement, nous allons présenter, partie 4.6 une structure originale, le **treillis de Galois des partitions**. Celui-ci possède les mêmes propriétés que le treillis de Galois classique, mais celles-ci s'appliquent aux partitions qu'aux parties.
Nous verrons au chapitre suivant comment ces méthodes peuvent s'inscrire dans notre problématique pour l'apprentissage par renforcement.

4.1 Treillis

Nous allons maintenant donner quelques définitions de base concernant les ensembles structurés par treillis et qui nous serviront par la suite.[25] Pour une étude complète sur le sujet, nous renvoyons le lecteur à l'ouvrage [Ganter and Wille, 1999], voire aux travaux historiques sur le sujet : [Birkhoff, 1973].

Commençons par quelques rappels d'algèbre. Nous considérons dans le cadre de nos travaux que les ensembles auxquels nous nous intéressons sont **finis**[26].

Définition 4.1 (Relation de pre-ordre)
Soit un ensemble E et une relation binaire R de E dans lui-même. R est une **relation de pre-ordre** notée $\preceq_E$ ou $\preceq$ s'il n'y a pas d'ambiguité, si est seulement si :

1. $\preceq$ est *réflexive* : $\forall e \in E, e \preceq e$
2. $\preceq$ est *transitive* : $\forall e_1, e_2, e_3 \in E, e_1 \preceq e_2$ et $e_2 \preceq e_3 \Rightarrow e_1 \preceq e_3$

Définition 4.2 (Relation d'ordre partiel)
Soit un ensemble E et une relation binaire R de E dans lui-même. R est une **relation d'ordre partiel** notée $\leq_E$ ou $\leq$ s'il n'y a pas d'ambiguité, si est seulement si :

1. $\leq$ est un *pre-ordre*
2. $\leq$ est *antisymétrique* : $\forall e_1, e_2 \in E, e_1 \leq e_2$ et $e_2 \leq e_1 \Rightarrow e_1 = e_2$

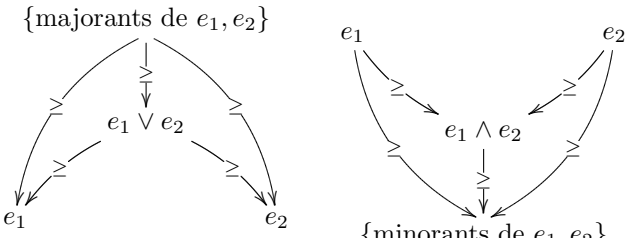
[25] Les notions les plus classiques comme la notion de *relation* sont fournies en annexe, partie 8 page 169
[26] Nous ne le rappellerons pas dans les définitions et preuves. Certaines peuvent ne pas être généralisables dans le cas d'ensembles potentiellement infinis

Un ensemble E muni d'une telle relation est appelée **ensemble partiellement ordonné** et noté $(E, \leq)$.

Définition 4.3 (voisin, comparabilité, chaîne, antichaîne, hauteur, largeur)
Soit un ensemble partiellement ordonné $(E, \leq)$. Soit $e_1, e_2 \in E$ tels que $e_1 \leq e_2$ et $\nexists\ e_3 \in E$ tel que $e_1 \leq e_3 \leq e_2$. e_1 est alors appelé **voisin inférieur** de e_2 et e_2 est appelé **voisin supérieur** de e_1. On écrit alors $e_1 \prec e_2$. Soit $e_1, e_2 \in E$ tel que $e_1 \leq e_2$ ou $e_2 \leq e_1$. e_1 et e_2 sont alors appelés **comparables**. Sinon, ils sont appelés **incomparables**.
Soit $F \subseteq E$ tel que tous les éléments de F soient comparables. F est alors appelé une **chaîne**.
Soit $F \subseteq E$ tel que tous les éléments de F soient incomparables. F est alors appelé une **antichaîne**.
Le cardinal de la chaîne de taille maximale de E moins 1 est appelé **hauteur** de E.
Le cardinal de l'antichaîne de taille maximale de E est appelé **largeur** de E.

Exemple 4.1 *Soit $\mathbb{C}$, l'ensemble des nombres complexes $c = (a, b)$ avec $a, b \in \mathbb{R}$. Soit la relation $\leq_{\mathbb{C}}$ sur $\mathbb{C}$ telle que $\forall (a_1, b_1), (a_2, b_2) \in \mathbb{C}$, $(a_1, b_1) \leq (a_2, b_2) \Leftrightarrow a_1 \leq a_2$ et $b_1 \leq b_2$. $(\mathbb{C}, \leq_{\mathbb{C}})$ est un ordre partiel.*

Définition 4.4 (majorant, supremum (resp. minorant, infimum))
Soit $(E, \leq)$ un ensemble partiellement ordonné et $A \subseteq E$.
Un **majorant** (resp. **minorant**) de A est un élément $e \in E$ tel que $\forall a \in A$, $a \leq e$ (resp. $e \leq a$).
S'il y a un plus petit élément dans l'ensemble des majorants (resp. un plus grand élément dans l'ensemble des minorants), il est appelé **supremum** (resp. **infimum**). On le notera $\bigvee A$ (resp. $\bigwedge A$).
Le supremum (resp. l'infimum) d'un ensemble de deux éléments $\{e_1, e_2\} \subseteq E$ est noté $e_1 \vee e_2$ (resp. $e_1 \wedge e_2$).



(a): $e_1 \vee e_2$ est le plus petit des majorants

(b): $e_1 \wedge e_2$ est le plus grand des minorants

FIG. 4.1 – le supremum et l'infimum de deux éléments

Nous basant sur ces définitions, nous allons maintenant donner deux définitions pour les treillis : une définition classique et une définition constructive. Pour la définition classique, on suppose que l'on a un ensemble muni d'une relation d'ordre partiel, et on montre qu'ils vérifient les propriétés d'un treillis. Pour la définition constructive, on considérera, que l'on fournit les opérateurs sur l'ensemble partiellement ordonné permettant de lui donner une structure de treillis.
Nous donnerons généralement les deux types de définition au cours de ce chapitre. Les définitions classiques servent à faire le lien avec les notions courantes sur le sujet. Dans le cadre de nos travaux, nous aurons à programmer en utilisant les structures que nous présentons, donc à les construire. C'est pourquoi nous nous attachons également à donner des définitions constructives.

Définitions classiques

Définition 4.5 (Treillis $\mathcal{T}(E, \leq)$ définition classique)
Soit un ensemble partiellement ordonné $(E, \leq)$. Si pour tout couple d'éléments $e_1, e_2 \in E$, il existe un infimum $e_1 \wedge e_2 \in E$ et un supremum $e_1 \vee e_2 \in E$ alors $(E, \leq)$ est un **treillis** noté $\mathcal{T}(E, \leq)$.
De plus, si pour tout sous-ensemble de E, $A \subseteq E$, $\bigvee A$ et $\bigwedge A$ existent et appartiennent à E, alors $\mathcal{T}(E, \leq)$ est appelé **treillis complet**.
En particulier, $\bigvee E$ est le plus grand élément de E et est appelé **élément unité** de E et noté 1_E.
Respectivement, $\bigwedge E$ est le plus petit élément de E et est appelé **élément zéro** de E et noté 0_E.

Remarque 4.1 *Il est d'usage de donner une représentation sous forme de graphe pour les treillis suffisamment petit pour être représentables, plutôt que de donner la liste des éléments de $E \times E$ constituants la relation d'ordre partiel.*
Chaque élément de E est alors représenté par un sommet et les relations de voisinage $\prec$ sont représentées par des arcs orientés entre les voisins. La transitivité de la relation d'ordre impliquant que toute l'information du treillis est représentée en utilisant seulement les relations de voisinage.De telles représentations sont appelées graphes de Hass.
Pour nos représentations, les éléments les plus grands seront situés en haut et la relation d'ordre sera représentée par des arcs dirigés des éléments les plus grands vers les éléments les plus petits.

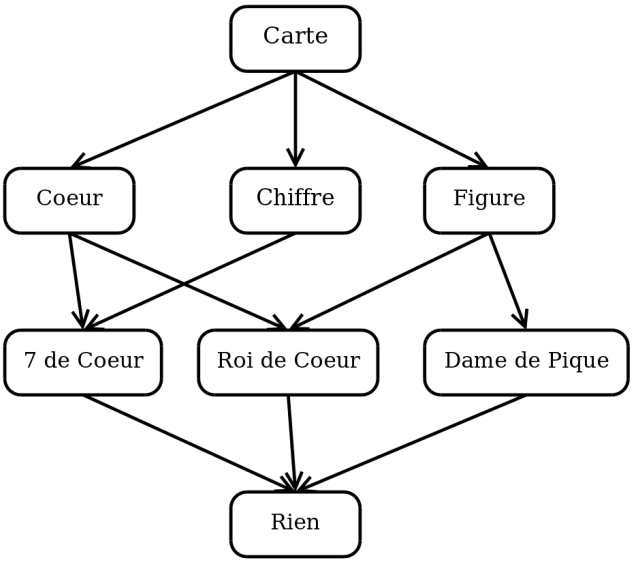


FIG. 4.2 – Exemple de treillis

Exemple 4.2 *La figure 4.1 est la représentation graphique d'un treillis $\mathcal{T}(E, \leq)$, avec $E = \{$Carte, Coeur, Chiffre, Figure, 7 de Coeur, Roi de Coeur, Dame de Pique, Rien$\}$ et muni de la relation d'ordre partiel $\leq$ définie par la relation de voisinage représentée par le graphe.*
Soit deux éléments e_1, $e_2 \in E$. S'il existe un chemin de e_1 vers e_2, alors $e_1 \geq e_2$.
Soit deux éléments e_1, $e_2 \in E$. Le supremum de e_1 et e_2, $e_1 \vee e_2$ est le plus petit ancêtre commun dans le graphe de e_1 et e_2. L'infimum de e_1 et e_2, $e_1 \wedge e_2$ est le plus petit ancêtre commun dans le graphe de e_1 et e_2 en inversant le sens des arcs.

Si l'on ne considère que les supremums ou que les infimums, on a les notions respectivement de sup-demi-treillis et inf-demi-treillis.

Définition 4.6 (sup-demi-treillis $\mathcal{T}_\vee(E, \leq)$)
Soit un ensemble partiellement ordonné $(E, \leq)$. Si pour tout couple d'éléments $e_1, e_2 \in E$, il existe un supremum $e_1 \vee e_2 \in E$ alors $(E, \leq)$ est un sup-demi-treillis noté $\mathcal{T}_\vee(E, \leq)$.

On peut définir de façon duale l'inf-demi-treillis $\mathcal{T}_\wedge(E, \leq)$ en considérant les infimums plutôt que les supremums.

Définitions constructives

Nous venons de donner une définition où l'existence du supremum $\vee$ et de l'infimum $\wedge$ pour chaque couple d'éléments est une propriété à vérifier. Nous allons maintenant donner une définition constructive du treillis, étant donnés des opérateurs[27] produit $\otimes$ et somme $\oplus$ permettant de construire respectivement le supremum et l'infimum de deux éléments.

[27] La définition d'opérateur est rappelée en annexe, définition 8.5 page 170

Définition 4.7 (Opérateur produit $\otimes$)
Soit un ensemble partiellement ordonné $(E, \leq)$ et un opérateur binaire $\otimes_E$

$$\otimes_E : E \times E \to E$$

$\otimes_E$ est un **opérateur produit sur** E si est seulement si $\forall e_1, e_2 \in E$:

1. $e_1 \otimes e_2$ est unique
2. $e_1 \leq e_1 \otimes e_2$ et $e_2 \leq e_1 \otimes e_2$
3. $\forall e \in E$ tel que $e_1 \leq e$ et $e_2 \leq e$ alors $e_1 \otimes e_2 \leq e$

On note $\otimes_E$ l'opérateur produit sur E ou $\otimes$, s'il n'y a pas d'ambiguïté.

Par définition, l'opérateur $\otimes$ construit **le supremum** de deux éléments de E.

Propriété 4.1 (commutativité et associativité de $\otimes$)
Soit un ensemble partiellement ordonné $(E, \leq)$ et un opérateur produit $\otimes$ sur E.
$\otimes$ est **commutatif** et **associatif**.

Preuve 4.1

- **commutativité**
  Il suffit de constater que l'inversion de e_1 et e_2 ne modifie pas la définition.
- **associativité**
  $(e_1 \otimes e_2) \otimes e_3$ est unique par définition et
  $\forall e \in G$ tel que $e \geq e_1$, $e \geq e_2$ et $e \geq e_3$, $e \geq (e_1 \otimes e_2) \otimes e_3$
  De la même façon, $e \geq e_1 \otimes (e_2 \otimes e_3)$
  Donc $(e_1 \otimes e_2) \otimes e_3 = e_1 \otimes (e_2 \otimes e_3)$

□

On définit dualement l'**opérateur somme** $\oplus$ qui construit **l'infimum** de deux éléments de E.

Définition 4.8 (Opérateur somme $\oplus$)
Soit un ensemble partiellement ordonné $(E, \leq)$ et un opérateur binaire $\oplus_E$

$$\oplus_E : E \oplus E \to E$$

$\oplus_E$ est un **opérateur somme sur** E si et seulement si $\forall e_1, e_2 \in E$:

1. $e_1 \oplus e_2$ est unique
2. $e_1 \oplus e_2 \leq e_1$ et $e_1 \oplus e_2 \leq e_2$
3. $\forall e \in E$ tel que $e \leq e_1$ et $e \leq e_2$ alors $e \leq e_1 \oplus e_2$

On note $\oplus_E$ l'opérateur somme sur E ou $\oplus$, s'il n'y a pas d'ambiguïté.

Par définition, l'opérateur $\oplus$ construit **l'infimum** de deux éléments de E.

On peut généraliser les définitions précédentes en appliquant successivement un opérateur produit ou somme à un ensemble d'éléments.

Définition 4.9 (Opérateur produit généralisé $\bigotimes$)
Soit un ensemble partiellement ordonné $(E, \leq)$, soit $A \subseteq E$ et un opérateur produit $\otimes$ sur E.
On note $\bigotimes A = e_1 \otimes ... \otimes e_n$ avec $A = \{e_1, \ldots, e_n\}$. On a $\bigotimes A \in E$.

Preuve 4.2

- **existence**
  $\forall e_1, e_2 \in A,\ e_1 \otimes e_2$ existe par définition et $e_1 \otimes e_2 \in E$ donc
  $\forall e_1, e_2, e_3 \in A, (e_1 \otimes e_2) \otimes e_3$ existe et $(e_1 \otimes e_2) \otimes e_3 \in E$.
- **unicité**
  Se déduit de l'associativité de $\otimes$.

□

On définit dualement l'**opérateur somme généralisé** $\bigoplus$ en généralisant l'opérateur $\oplus$.

Définition 4.10 (Opérateur somme généralisé $\bigoplus$)
Soit un ensemble partiellement ordonné $(E, \leq)$, soit $A \subseteq E$ et un opérateur somme $\oplus$ sur E.
On note $\bigoplus A = e_1 \oplus ... \oplus e_n$ avec $A = \{e_1, \ldots, e_n\}$. On a $\bigoplus A \in E$.

Exemple 4.3 *Si on se réfère aux expressions logiques du premier ordre, l'opérateur lgg*[28] *défini par Plotkin ([Plotkin, 1970] et [Plotkin, 1971]) de deux clauses est un opérateur produit.*

Définition 4.11 (Treillis $\mathcal{T}(E, \leq, \otimes, \oplus)$ définition constructive)
Soit un ensemble partiellement ordonné $(E, \leq)$ un opérateur produit $\otimes$ sur E et un opérateur de somme $\oplus$ sur E.
Par construction, $(E, \leq)$ est un **treillis complet** noté $\mathcal{T}(E, \leq, \otimes, \oplus)$.

Exemple 4.4 *L'ensemble $\mathcal{P}(E)$ des parties d'un ensemble E muni de la relation d'inclusion ensembliste $\subseteq$, de l'opérateur d'intersection $\cap$ ensembliste et de l'opérateur d'union ensembliste $\cup$ est un treillis.*

Une définition analogue permet de créer un demi-treillis seulement à partir d'un des opérateurs produit $\otimes$ ou somme $\oplus$.

4.2 Treillis des Parties

Présentons dans cette partie un treillis particulier qui permet de représenter l'ensemble des regroupements possibles d'un ensemble d'objets : le treillis des parties. Comme notre objectif et ici de formaliser les espaces pour la généralisation pour l'apprentissage par renforcement, nous l'appellerons **espace des regroupements**.

Définition 4.12 (Ensemble des Parties d'un ensemble $\mathcal{P}(E)$)
L'ensemble des parties d'un ensemble E, noté $\mathcal{P}(E)$, est l'ensemble des ensembles A tels que $A \subseteq E$.
Notons que $E \in \mathcal{P}(E)$ et $\emptyset \in \mathcal{P}(E)$.
Notons de plus que $|\mathcal{P}(E)| = 2^n$, n étant le nombre d'éléments de E.

Exemple 4.5 *Si on considère l'ensemble $E = \{e_1, e_2, e_3, e_4\}$,*
$\mathcal{P}(E) = \{\emptyset, \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_1, e_2\}, \{e_1, e_3\}, \{e_1, e_4\}, \{e_2, e_3\}, \{e_2, e_4\}, \{e_3, e_4\}, \{e_1, e_2, e_3\}, \{e_1, e_2, e_4\}, \{e_1, e_3, e_4\}, \{e_2, e_3, e_4\}, \{e_1, e_2, e_3, e_4\}\}$

Définition 4.13 (Treillis des parties d'un ensemble $\mathcal{T}_{\mathcal{P}}(E)$)
Soit $\mathcal{T}_{\mathcal{P}}(E) = (\mathcal{P}(E), \subseteq)$, $\subseteq$ étant la relation d'inclusion sur les ensembles. $\mathcal{T}_{\mathcal{P}}(E)$ est un treillis nommé treillis des parties de E.
Soit $A_1, A_2 \in \mathcal{P}(E)$. Le suppremum $A_1 \vee A_2$ est tel que $A_1 \vee A_2 = A_1 \cup A_2$. L'infimum $A_1 \wedge A_2$ est tel que $A_1 \wedge A_2 = A_1 \cap A_2$.
On peut également définir $\mathcal{T}_{\mathcal{P}}(E) = (\mathcal{P}(E), \supseteq)$. On a alors $A_1 \vee A_2 = A_1 \cap A_2$ et $A_1 \wedge A_2 = A_1 \cup A_2$.

Preuve 4.3 ($\mathcal{T}_{\mathcal{P}}(E)$ est un treillis)
Soit $A_1 \subseteq E$ et $A_2 \subseteq E$, $A_1 \cup A_2 = \{o \in A_1 \text{ ou } o \in A_2\}$.
Démontrons que $A_1 \cup A_2$ est le supremum de A_1 et A_2 en considérant la relation d'ordre partiel $\subseteq$.

- **domaine et existence**
  Remarquons simplement que $A_1 \cup A_2 \subseteq E$.
- **suppremum**
  Supposons $A_3 \subseteq E$ tel que $A_1 \subseteq A_3, A_2 \subseteq A_3$ et $A_3 \subseteq A_1 \cup A_2$.
  Soit $o \in A_1 \cup A_2 = \{o \in A_1 \text{ ou } o \in A_2\}$. Si $o \in A_1$ alors $o \in A_3$. Si $o \in A_2$ alors $o \in A_3$. Donc $A_1 \cup A_2 \subseteq A_3$. Donc $A_3 = A_1 \cup A_2$.

De même, on démontre que $A_1 \cap A_2 = A_1 \wedge A_2$.

[28] Least General Generalisation - généralisation la moins générale en français

□

On peut donner aussi une définition constructive du treillis des parties en utilisant la définition constructive 4.11 page ci-contre pour les treillis. On aura comme opérateur de produit $\otimes$, $\cap$ ou $\cup$ et comme opérateur de somme $\oplus$, $\cup$ ou $\cap$. Le treillis des parties peut-être ainsi noté

$$\mathcal{T}_{\mathcal{P}}(E) = (\mathcal{P}(E), \subseteq, \cap, \cup) \text{ ou } \mathcal{T}_{\mathcal{P}}(E) = (\mathcal{P}(E), \supseteq, \cup, \cap)$$

La preuve 4.3 page précédente du fait que $\mathcal{T}_{\mathcal{P}}(E)$ soit un treillis donne tous les éléments qui montrent que $\cup$ et $\cap$ sont des opérateurs produit $\otimes$ ou somme $\oplus$ sur $\mathcal{P}(E)$.

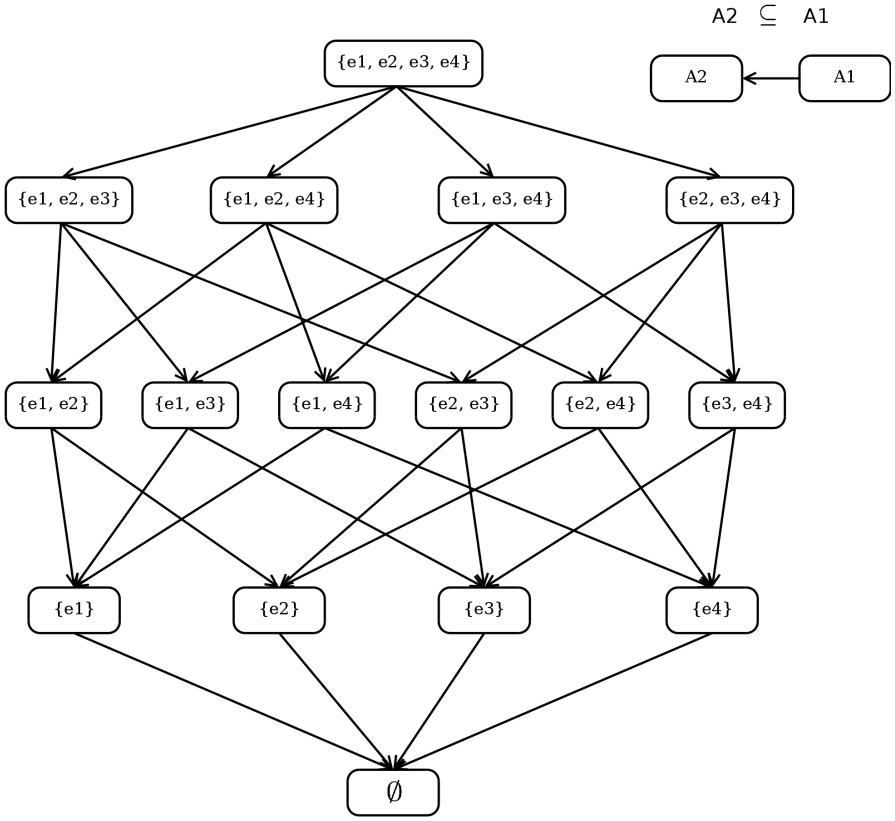


FIG. 4.3 – Treillis des Parties avec $E = \{e_1, e_2, e_3, e_4\}$

Exemple 4.6 *La figure 4.3 donne une représentation sous forme de graphe du treillis des parties* $\mathcal{T}_{\mathcal{P}}(E)$ *de l'ensemble* $E = \{e_1, e_2, e_3, e_4\}$*, avec* $\mathcal{T}_{\mathcal{P}}(E) = (\{e_1, e_2, e_3, e_4\}, \subseteq, \cup, \cap)$*.*

Le treillis des parties $\mathcal{T}_{\mathcal{P}}(E)$ d'un ensemble E est l'ensemble des regroupements possibles des éléments de E. Dit autrement, à chaque façon de regrouper les éléments de E entre eux, il existe un élément du treillis des parties.
De plus, l'opérateur d'inclusion $\subseteq$, donne une méthode de comparaison entre deux regroupements, basée sur la capacité de ceux-ci à inclure des éléments de E.
Cependant, ces regroupements possibles ne nous apportent rien du point de vue d'une généralisation possible de l'apprentissage. Comme nous l'avons vu partie 1.1 page 5, l'apprentissage machine, pour généraliser, nécessite un biais. Le treillis des parties représente l'espace des regroupements possibles d'un ensemble d'éléments sans biais.

Nous allons mainteant supposer que les objets à regrouper sont décrits par une relation binaire entre des attibuts et les objets. Nous montrerons alors que cette relation permet de construire un treillis particulier nommé **treillis de concepts** ou **treillis de Galois d'une relation binaire**, donnant l'ensemble des regroupements possibles des objets en fonction d'attributs, et ordonant ces regroupements.

4.3 Treillis de Galois d'une relation binaire

Les treillis de Galois sont classiquement présentés comme une double fermeture particulière entre un ensemble d'objets et un ensemble d'attributs reliés par une relation binaire. Nous allons présenter dans un premier temps cette approche classique. Dans un second temps, partie 4.4, nous préciserons d'une part notre propos en définissant les notions algébriques, notamment les connections de Galois, implicitement utilisées dans l'approche classique. D'autre part, tout en conservant la structure de treillis de Galois, nous montrerons que l'on peut utiliser n'importe quel langage de description muni d'une structure de sup-demi-treillis appelée *langage de généralisation*. On peut citer parmi de nombreuses autres [Birkhoff, 1973], [Nguifo, 1993], ou [Liquière, 2006] comme références pour les notions utilisées ci-après.

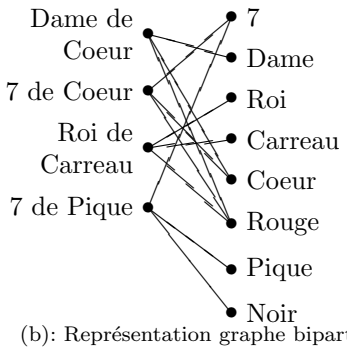
Nous allons dans cette partie donner les définitions et propriétés classiques amenant à la définition des treillis de Galois d'une relation binaire. Dans [Ganter and Wille, 1999], l'étude spécifique de cette structuration est ensuite connue sous le nom d'analyse formelle de concepts[29]. Les notions de base que nous donnons ici sont issues de cet ouvrage.

Tout d'abord, considérons la notion de contexte formel, associant un ensemble d'objets décrits par des attributs par une relation binaire.

Définition 4.14 (contexte formel)
Un **contexte formel** $\mathbb{C} := (\mathcal{O}bj, \mathcal{A}tt, R)$ consiste en deux ensembles $\mathcal{O}bj$ et $\mathcal{A}tt$ et une relation binaire R entre $\mathcal{O}bj$ et $\mathcal{A}tt$. Les éléments de $\mathcal{O}bj$ sont appelés les **objets** et les éléments de $\mathcal{A}tt$ sont appelés les **attributs** ou **motifs** du contexte. Pour dire qu'un objet o est en relation R avec un attribut a, nous écrirons **oRa** et lirons o a l'attribut a ou l'attribut a est vu sur l'objet o.

	7	Dame	Roi	Carreau	Coeur	Rouge	Noir	Pique
Dame de Coeur		×			×	×		
7 de Coeur	×				×	×		
7 de Pique	×						×	×
Roi de Carreau			×	×		×		

(a): Représentation matricielle



(b): Représentation graphe biparti

FIG. 4.4 – Contexte formel $\mathbb{C}$ avec $\mathcal{O}bj$ ={Dame de Coeur, 7 de Coeur, 7 de Pique, Roi de Coeur} et $\mathcal{A}tt$ ={Dame, 7, Roi, Coeur, Carreau, Noir, Pique, Rouge}

Exemple 4.7 *Considérons l'ensemble d'objets $\mathcal{O}bj$ ={Dame de Coeur, 7 de Coeur, 7 de Pique, Roi de Carreau} et pour le décrire, l'ensemble d'attributs suivant : $\mathcal{A}tt$ ={Dame, 7, Roi, Coeur, Carreau, Noir, Pique, Rouge}.*
La relation binaire de description établie entre les deux ensembles se trouve figure 4.4. Elle est représentée sous la forme matricielle (figure (a)) et à l'aide d'un graphe biparti (figure (b)).

[29]Formal Concept Analysis, FCA en anglais

Définissons maintenant deux opérations de fermeture[30] duales entre les objets et les attributs.

Remarque 4.2 *Dans l'ouvrage original [Ganter and Wille, 1999] les opérateurs que nous allons définir ci-après sont notés* $'$: $\mathcal{P}(\mathcal{O}bj) \rightarrow \mathcal{P}(\mathcal{A}tt)$ *et* $'$: $\mathcal{P}(\mathcal{A}tt) \rightarrow \mathcal{P}(\mathcal{O}bj)$. *Pour des soucis de cohérence de notations avec notre présentation, nous les nommerons respectivement* $d : \mathcal{P}(\mathcal{O}bj) \rightarrow \mathcal{P}(\mathcal{A}tt)$ *et* $i : \mathcal{P}(\mathcal{A}tt) \rightarrow \mathcal{P}(\mathcal{O}bj)$. d *étant l'abréviation de* description *et* i *d'*instanciation.

Définition 4.15 (opérateur de description d)
Pour un ensemble d'objets $O \in \mathcal{P}(\mathcal{O}bj)$[31], définissons l'opérateur d

$$d : \mathcal{P}(\mathcal{O}bj) \rightarrow \mathcal{P}(\mathcal{A}tt)$$

tel que :

$$d(O) = \{a \in \mathcal{A}tt \mid oRa, \forall o \in O\} \tag{4.1}$$

$d(O)$ est l'ensemble des attributs communs à tous les objets de O. On dit également que les éléments de O sont couverts par $d(O)$.

Définition 4.16 (opérateur d'instanciation i)
Pour un ensemble d'attributs $A \in \mathcal{P}(\mathcal{A}tt)$ l'opérateur i

$$i : \mathcal{P}(\mathcal{A}tt) \rightarrow \mathcal{P}(\mathcal{O}bj)$$

tel que :

$$i(A) = \{o \in \mathcal{O}bj \mid oRa, \forall a \in A\} \tag{4.2}$$

$i(A)$ est l'ensemble des objets qui ont tous les attributs de A.

Exemple 4.8 *En reprenant le contexte formel* $\mathbb{C}$ *de la figure 4.4 page ci-contre on a par exemple :*
$d\big(\{$*Dame de Coeur, 7 de Coeur*$\}\big) = \{$*Coeur, Rouge*$\}$
$i\big(\{7\}\big) = \{$*7 de Coeur, 7 de Pique*$\}$
$i\big(\{7,$ *Dame*$\}\big) = \emptyset$

Définissons des éléments particuliers nommés concepts. Ce sont les éléments stables par l'application successive de ces deux opérateurs.

Définition 4.17 (concept formel)
Un **concept formel** c du contexte formel $\mathbb{C}(\mathcal{O}bj, \mathcal{A}tt, R)$ est un couple (O, A) avec $O \in \mathcal{P}(\mathcal{O}bj)$, $A \in \mathcal{P}(\mathcal{A}tt)$ tels que :
$d(O) = A$ et $i(A) = O$ (Ce qui est équivalent à $i \circ d(O) = O$ et $d \circ i(A) = A$).
Nous appelons O l'**extension** et A l'**intention** ou la **description** du concept (O, A). O est noté $ext(c)$ et A est noté $int(c)$.
$\mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R)$ dénote l'ensemble des concepts formels du contexte formel $\mathbb{C}(\mathcal{O}bj, \mathcal{A}tt, R)$.

Exemple 4.9 *En reprenant le contexte formel de la figure 4.4 page précédente :*
$\big(\{$*Dame de Coeur, 7 de Coeur, Roi de Carreau*$\}$,$\{$*Rouge*$\}\big)$ *est un concept. En effet :*
$i \circ d\big(\{$*Dame de Coeur, 7 de Coeur, Roi de Carreau*$\}\big) = i\big(\{$*Rouge*$\}\big) = \{$*Dame de Coeur, 7 de Coeur, Roi de Carreau*$\}$.
Par contre, $\{$*Dame de Coeur, 7 de Pique*$\}$ *n'est pas l'extension d'un concept. En effet :*
$i \circ d\big(\{$*Dame de Coeur, 7 de Pique*$\}\big) = i(\emptyset) = \{$*Dame de Coeur, Roi de Carreau, 7 de Coeur, 7 de Carreau*$\}$.
De même, $\{$*Noir*$\}$ *n'est l'intention d'aucun concept. En effet :*
$d \circ i\big(\{$*Noir*$\}\big) = d\big(\{$*7 de Pique*$\}\big) = \{$*Noir, Pique, 7*$\}$
L'ensemble des concepts formels $\mathfrak{C}$ *de* $\mathbb{C}$ *sont représentés sur la figure 4.5 page 83.*

[30] La définition de fermeture, ainsi que la preuve que cette opération est bien une opération de fermeture, sera faite dans le paragraphe suivant

[31] Rappelons que $\mathcal{P}(E)$ désigne l'ensemble des parties de l'ensemble E (définition 4.12 page 78)

Définissons maintenant une relation d'ordre partiel sur les concepts basée sur l'inclusion ensembliste sur les ensembles d'objets et d'attributs. On dira qu'un concept est d'autant plus grand qu'il inclut d'objets.

Définition 4.18 (ordre partiel sur les concepts)
Soit un contexte formel $\mathbb{C}(\mathcal{O}bj, \mathcal{A}tt, R)$, $(O_1, A_1), (O_2, A_2) \in \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R)$, deux concepts tels que $A_2 \subseteq A_1$ (ce qui est équivalent à $O_1 \subseteq O_2$)[32].
On dit alors que (O_1, A_1) est un **sous-concept** de (O_2, A_2) et réciproquement que (O_2, A_2) est un **super-concept** de (O_1, A_1). On note cette relation d'ordre partiel $\leq$.

Le fait que cette relation soit un ordre partiel vient directement de sa définition. En effet, les relations d'inclusion des attributs ou des objets impliquent des relations d'ordres partiels. Montrons que cet ordre partiel est un treillis.

Propriété 4.2
$\forall O_1, O_2 \in \mathcal{O}bj$ et $A_1, A_2 \in \mathcal{A}tt$:

1. $O_1 \subseteq O_2 \Rightarrow d(O_2) \subseteq d(O_1)$ et $A_1 \subseteq A_2 \Rightarrow i(A_2) \subseteq i(A_1)$
2. $O_1 \subseteq i \circ d(O_1)$ et $A_1 \subseteq d \circ i(A_1)$
3. $d(O_1) = d \circ i \circ d(O_1)$ et $i(A_1) = i \circ d \circ i(A_1)$

Preuve 4.4

1. Soit $a \in d(O_2)$. On a $\forall o \in O_2, aRo$.
   Or $\forall o' \in O_1, o' \in O_2$, donc $\forall o' \in O_1, aRo'$.
   Donc $\forall o' \in O_1, a \in d(O_1)$.
2. Soit $o \in O_1$. On a $\forall a \in d(O_1)$, oRa par application de la définition de d. Donc, par application de la définition de i, $o \in i \circ d(O_1)$.
3. $d(O_1) \subseteq d \circ i \circ d(O_1)$ par application de $A_1 \subseteq d \circ i(A_1)$.
   Comme $O_1 \subseteq O_2 \Rightarrow d(O_2) \subseteq d(O_1)$ et $O_1 \subseteq i \circ d(O_1)$, $d \circ i \circ d(O_1) \subseteq d(O_1)$.

□

Propriété 4.3 (treillis de Galois d'une relation binaire)
$\mathcal{T}\big(\mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R), \leq\big)$ est un treillis. Ce **treillis de concepts** est aussi appelé **treillis de Galois d'une relation binaire**.
La version constuctive de ce treillis est $\mathcal{T}\big(\mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R), \leq, \vee, \wedge\big)$
avec $\vee : \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R) \times \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R) \rightarrow \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R)$ défini par :

$$\text{Soit } (O_1, A_1), (O_2, A_2) \in \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R), (O_1, A_1) \vee (O_2, A_2) = \big(i(A_1 \cap A_2), A_1 \cap A_2\big)$$

et $\wedge : \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R) \times \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R) \rightarrow \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R)$

$$\text{Soit } (O_1, A_1), (O_2, A_2) \in \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R), (O_1, A_1) \wedge (O_2, A_2) = \big((O_1 \cap O_2), d(O_1 \cap O_2)\big)$$

Preuve 4.5
Montrons que l'opérateur $\vee : \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R) \rightarrow \big(P(\mathcal{O}bj), P(\mathcal{A}tt)\big)$ défini par
$(O_\vee, A_\vee) = (O_1, A_1) \vee (O_2, A_2) = \big(i(A_1 \cap A_2), A_1 \cap A_2\big)$ définit un supremum pour tout élément de $\big(\mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R), \leq\big)$.

- **Domaine** : $(O_\vee, A_\vee) \in \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R)$
  Par application de 3. de la propriété 4.2, $i(A_1 \cap A_2) = i \circ d \circ i(A_1 \cap A_2)$.
  $a \in A_1 \cap A_2 \Leftrightarrow a \in A_1$ et $a \in A_2 \Leftrightarrow a \in d \circ i(A_1)$ et $a \in d \circ i(A_2) \Leftrightarrow a \in d \circ i(A_1) \cap d \circ i(A_2)$
- **Unicité** : par construction $(O_1, A_1) \vee (O_2, A_2)$ est unique.
- **Majorant** : on a bien $(O_1, A_1) \leq (O_\vee, A_\vee)$ et $(O_2, A_2) \leq (O_\vee, A_\vee)$ car
  $O_1 \subseteq O_1 \cup O_2$ et $d \circ i(A_1 \cap A_2) \subseteq A_1 \cap A_2 \subseteq A_1$ d'une part et
  $O_2 \subseteq O_1 \cup O_2$ et $d \circ i(A_1 \cap A_2) \subseteq A_1 \cap A_2 \subseteq A_2$ d'autre part.

[32] Cette propriété est une conséquence du fait que les opérateurs d et i forment une connexion de Galois entre $\mathcal{O}bj$ et $\mathcal{A}tt$, ce que nous montererons dans le prochain paragraphe.

- **Plus petit des majorants** : Soit (O_1, A_1) et $(O_2, A_2) \in \mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R)$. Supposons un élément $(O, A) \in \big(P(\mathcal{O}bj), P(\mathcal{A}tt)\big)$, tel que $(O_1, A_1) \leq (O, A)$, $(O_2, A_2) \leq (O, A)$ et $(O, A) \leq (O_1, A_1) \vee (O_2, A_2)$
  On a $A_1 \subseteq A \Rightarrow (\forall a \in A_1 \Rightarrow a \in A)$ et
  $A_2 \subseteq A \Rightarrow (\forall a \in A_2 \Rightarrow a \in A)$
  donc, $\forall a \in A_1 \cup A_2, a \in A$
  On a donc $(A_1 \cup A_2 \subseteq A$ et $A \subseteq A_1 \cup A_2) \Rightarrow A_1 \cup A_2 = A$.

De même, on montre que $(O_1, A_1) \wedge (O_2, A_2) = \big(i \circ d(O_1 \cap O_2), A_1 \cup A_2\big)$ définit un infimum pour tout élément de $\big(\mathfrak{C}(\mathcal{O}bj, \mathcal{A}tt, R), \leq\big)$.

□

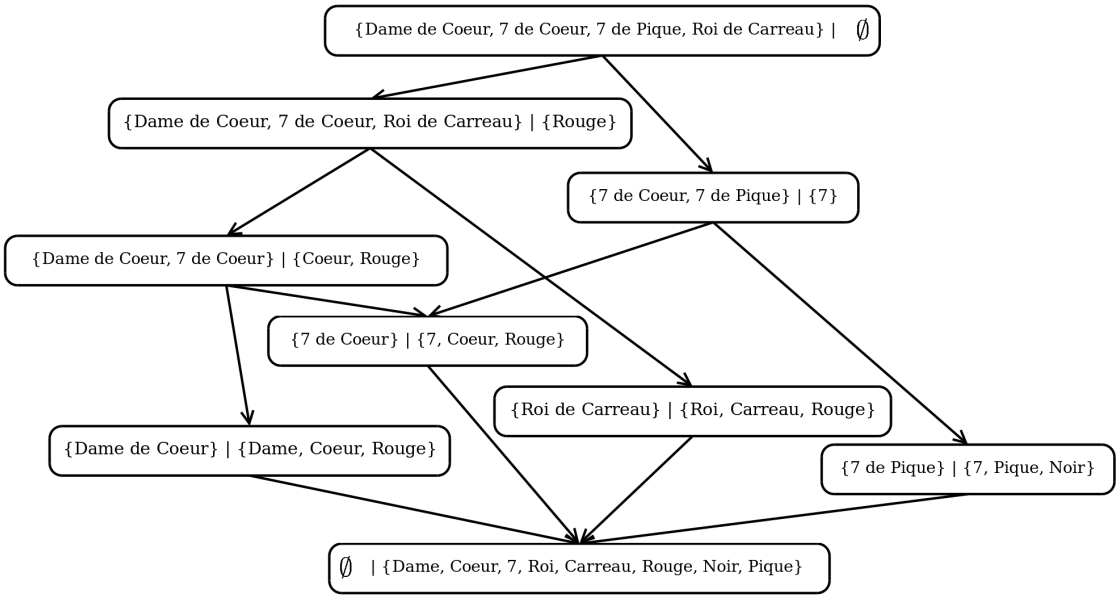


FIG. 4.5 – Treillis de Galois du contexte formel $\mathbb{C}$ avec $\mathcal{O}bj$ = {Dame de Coeur, 7 de Coeur, 7 de Pique, Roi de Carreau} et $\mathcal{A}tt$ = {Dame, 7, Roi, Coeur, Carreau, Pique, Noir}

Exemple 4.10 *La figure 4.5 représente le treillis de Galois du contexte formel présenté figure 4.4 page 80.*

4.4 Correspondance de Galois - Treillis de Galois

Nous venons de voir la définition classique de treillis de Galois défini par une une relation binaire entre un ensemble d'objets et un ensemble d'attributs. Nous allons voir dans ce paragraphe qu'il s'agit d'un cas particulier. Nous allons donner les propriétés algébriques nécessaires à deux ensembles partiellement ordonnés ayant une structure de demi-treillis et à leur relation pour qu'ils puissent constituer un treillis de Galois. Les relations respectant ces propriétés sont connues sous le nom de correspondance de Galois. Nous verrons ensuite, dans la partie 4.5 page 88, comment cette généralisation sur les propriétés algébriques peut être utilisée pour produire un espace de généralisation pour un ensemble d'objets.

4.4.1 Correspondance de Galois

Donnons d'abord la définition de fermeture, et montrons en quoi les correspondances de Galois sont une relation de double fermeture entre deux ensembles partiellement ordonnés.

Définition 4.19 (Fermeture)
Soit E un ensemble et $h : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ une application de l'ensemble des parties de E dans lui-même ayant les propriétés suivantes :

- **isotonie** (monotonie et croissance)
  $\forall e_1, e_2 \in \mathcal{P}(E), e_1 \subseteq e_2 \Rightarrow h(e_1) \subseteq h(e_2)$
- **extensivité** $\forall e_1 \in \mathcal{P}(E), e_1 \subseteq h(e_1)$
- **idempotence** $\forall e_1 \in \mathcal{P}(E), h(e_1) = h \circ h(e_1)$

h est alors appelée **fermeture**.

L'application de fermeture sur un ensemble permet notamment de produire des éléments particuliers de l'ensemble des partitions de E : les fermés, c'est à dire, les parties de E stable par l'application de fermeture.

Définition 4.20 (fermés)
Soit E un ensemble et $h : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ une fermeture sur cet ensemble.
Les éléments $x \in \mathcal{P}(E)$ tels que :

$$h(x) = x$$

sont appelés **fermés** de E par la fermeture h.
L'ensemble des fermés de E par h sera noté $F_h(E)$.

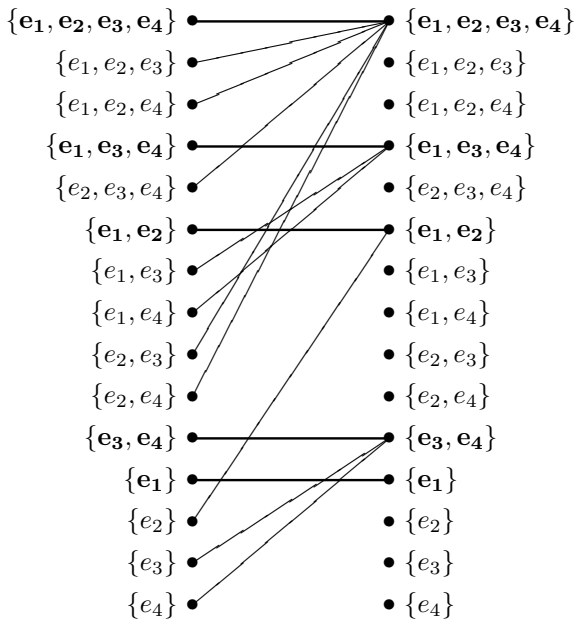


FIG. 4.6 – Exemple d'une fermeture $h : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ et de ses fermés sur l'ensemble $E = \{e_1, e_2, e_3, e_4\}$. Les fermés sont représentés en gras. On remarquera que toute image par h d'une partie de E est un fermé.

$F_h(E)$, l'ensemble des fermés est particulièrement notable pour la propriété suivante :

Propriété 4.4
h n'est pas définie $\mathcal{P}(E)$ vers $\mathcal{P}(E)$, mais de $\mathcal{P}(E)$ dans $F_h(E)$.

Preuve 4.6
Soit $h : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ une femeture de E.
h est idempotente c'est à dire $\forall x \in \mathcal{P}(E),\ h \circ h(x) = h(x)$.
Soit $y = h(x)$, on a alors $h(y) = y$, donc $y \in F_h(A)$.

□

Comme $F_h(E) \subseteq \mathcal{P}(E)$, l'application d'une fermeture sur un ensemble implique une sélection d'un sous-ensemble de celui-ci. Nous nous servirons de cette propriété plus loin.

Exemple 4.11 *La figure 4.6 page ci-contre présente un exemple de fermeture dans un ensemble à quatre éléments. On notera particulièrement l'ensemble des fermés qui sont stables par h et qui sont représentés en gras sur la figure.*

Voyons maintenant la composition de deux fermetures particulières que constitue la correspondance de Galois.

Définition 4.21 (Correspondance de Galois)
Soit $(A, \leq_A)$ et $(B, \leq_B)$ deux ensembles munis d'une relation d'ordre partiel. Soit deux fonctions $f : A \to B$ et $g : B \to A$. Le couple de fonctions (f, g) est appelé **correspondance de Galois** si :

1. $\forall a_1, a_2 \in A$ si $a_1 \leq_A a_2 \Rightarrow f(a_1) \geq_B f(a_2)$
2. $\forall b_1, b_2 \in B$ si $b_1 \leq_B b_2 \Rightarrow g(b_1) \geq_A g(b_2)$
3. $\forall a \in A$, $a \leq_A g \circ f(a)$
4. $\forall b \in B$, $b \leq_B f \circ g(b)$

Les fonctions f et g sont appelées duales l'une envers l'autre.

Proposition 4.1 *Soit une connexion de Galois (f, g) entre deux ensembles partiellement ordonnés $(A, \leq_A)$ et $(B, \leq_B)$.*
$g \circ f : A \to A$ et $f \circ g : B \to B$ sont des fermetures.

Preuve 4.7 ($f \circ g : A \to A$ est une fermeture)

- **isotonie**
  Soit $a_1, a_2 \in A$ avec $a_1 \leq_A a_2$, ce qui implique
  $f(a_1) \geq_B f(a_2)$ par application de (1) et finalement
  $g \circ f(a_1) \leq_A g \circ f(a_2)$ par application de (2).
- **extensivité**
  Par application de (3).
- **idempotence**
  Par application de (3) sur $g \circ f(a)$, on a $g \circ f(a) \leq_A g \circ f \circ g \circ f(a)$
  Par application de (4) sur $f(a)$, on a $f(a) \leq_B f \circ g \circ f(a)$ et finalement par application de (3), $g \circ f(a) \geq_A g \circ f \circ g \circ f(a)$.
  Donc $g \circ f(a) = g \circ f \circ g \circ f(a)$

De manière duale, on montre que $f \circ g : B \to B$ est une fermeture.

□

4.4.2 Correspondance de Galois et Treillis de Galois

Nous avons montré qu'une correspondance de Galois entre deux ensembles partiellement ordonnés $(A, \leq_A)$ et $(B, \leq_B)$ correspondait à deux fermetures duales. Intéressons-nous aux fermés de cette fermeture et montrons que ceux-ci associés aux relations d'ordre $\leq_A$ et $\leq_B$ définissent également un ensemble partiellement ordonné. Ensuite, nous verrons que si A et B sont respectivement des sup-demi-treillis et des inf-demi-treillis, l'ensemble des fermés forme un treillis, le treillis de Galois.

Dans un premier temps, remarquons qu'il y a bijection entre les fermés des deux fermetures duales formées par une correspondance de Galois.

Propriété 4.5
Soit deux ensembles munis d'ordre partiel $(A, \leq_A)$ et $(B, \leq_B)$ et une correspondance de Galois (f, g), $f : A \to B$, $g : B \to A$.
f est alors une bijection de l'ensemble des fermés de A par la fermeture $f \circ g$ vers l'ensemble des fermés de B par la fermeture $g \circ f$:

$$f : F_{f \circ g}(A) \to F_{g \circ f}(B)$$

Dualement, g est une bijection de l'ensemble des fermés de B par la fermeture $g \circ f$ vers l'ensemble des fermés de A par la fermeture $f \circ g$:

$$g : F_{g \circ f}(B) \to F_{f \circ g}(A)$$

De plus, f et g sont des applications inverses :

$$f^{-1} = g$$

et

$$g^{-1} = f$$

Preuve 4.8 ($f : F_{f\circ g}(A) \to F_{g\circ f}(B)$ bijection et $f^{-1} = g$)

- $f : F_{f\circ g}(A) \to F_{g\circ f}(B)$ **injective**
  Supposons $a_1, a_2 \in F_{f\circ g}(A)$ tels que $a_1 \neq a_2$ et $f(a_1) = f(a_2)$.
  $f(a_1) = f(a_2) \Rightarrow g \circ f(a_1) = g \circ f(a_2) \Rightarrow a_1 = a_2$ car a_1 et a_2 sont des fermés.
- $f : F_{f\circ g}(A) \to F_{g\circ f}(B)$ **surjective**
  Supposons $b_1 \in F_{g\circ f}(B)$ tel que $\nexists\ a_1 \in F_{f\circ g}(A)$ avec $f(a_1) = b_1$.
  Soit $a_2 \in A$ tel que $a_2 = g(b_1)$. $g \circ f(a_2) = g \circ f \circ g(b_1) = g(b_1)$ car $g \circ f$ est une fermeture.
  $a_2 = g \circ f(a_2)$, donc $a_2 \in F_{f\circ g}(A)$.
- $f^{-1} = g$ **inverses**
  On a bien par définition $\forall a_1 \in F_{f\circ g}(A)$, $a_1 = g \circ f(a_1)$.

Les démonstrations pour $g : F_{g\circ f}(B) \to F_{f\circ g}(A)$ bijection et $g^{-1} = f$ sont duales.

□

Comme f et g forment des bijections duales entre les fermés de A par $f \circ g$, $F_{f\circ g}(A)$ et les fermés de B par $g \circ f$, $F_{g\circ f}(B)$, nous allons former des couples constitués d'un fermé dans A et de son correspondant dans B.
Notre objectif est de montrer que l'on peut ordonner les couples ainsi constitués en treillis (le treillis de Galois). Pour ce faire, il faut que les ensembles partiellement ordonnés aient en plus une structure de demi-treillis. C'est ce que nous considérerons désormais. Nous allons donc considérer une généralisation des concepts formels développés dans [Ganter and Wille, 1999].[33]

Définition 4.22 (Contexte formel généralisé, Concepts formels généralisés)
Soit deux demi-treillis $\mathcal{T}_\vee(A, \leq_A)$ et $\mathcal{T}_\vee(B, \leq_B)$ et une correspondance de Galois (f, g), $f : A \to B$, $g : B \to A$.
$\mathbb{C}\left(\mathcal{T}_\vee(A, \leq_A), \mathcal{T}_\vee(B, \leq_B), (f, g)\right)$ dénote un **contexte formel généralisé**.
On définit l'ensemble des **concepts formels généralisés** $\mathfrak{C}$ de $\mathbb{C}$ par l'ensemble des couples (F_A, F_B), $F_A \in F_{f\circ g}(A)$, $F_B \in F_{g\circ f}(B)$ et $F_B = f(F_A)$ (ce qui est équivalent à $F_A = g(F_B)$).

Montrons maintenant que les concepts formels généralisés peuvent être munis d'une relation d'ordre partiel basée sur $\leq_A$ ou $\leq_B$.

Définition 4.23
Soit un contexte formel généralisé $\mathbb{C}\left(\mathcal{T}_\vee(A, \leq_A), \mathcal{T}_\vee(B, \leq_B), (f, g)\right)$.
On définit $\leq_{\mathfrak{C}_A}$ une relation d'ordre partiel pour $\mathfrak{C}$ par :

$$(F_{A1}, F_{B1}) \leq_{\mathfrak{C}_A} (F_{A2}, F_{B2}) \text{ si et seulement si } F_{A1} \leq_A F_{A2}$$

et $\leq_{\mathfrak{C}_B}$ une relation d'ordre partiel pour $\mathfrak{C}$ par :

$$(F_{A1}, F_{B1}) \leq_{\mathfrak{C}_B} (F_{A2}, F_{B2}) \text{ si et seulement si } F_{B2} \leq_B F_{B1}$$

Propriété 4.6
Soit un contexte formel généralisé $\mathbb{C}\left(\mathcal{T}_\vee(A, \leq_A), \mathcal{T}_\vee(B, \leq_B), (f, g)\right)$.
$\forall (F_{A1}, F_{B1}), (F_{A2}, F_{B2}) \in \mathfrak{C}$:

$$(F_{A1}, F_{B1}) \leq_{\mathfrak{C}_A} (F_{A2}, F_{B2}) \Leftrightarrow (F_{A1}, F_{B1}) \leq_{\mathfrak{C}_B} (F_{A2}, F_{B2})$$

Preuve 4.9
Se déduit directement du fait que (F_{A1}, F_{B1}) et (F_{A2}, F_{B2}) soient fermés et que (f, g) soient une correspondance de Galois.

[33] Dans la thèse de Maille ([Maille, 1999]), les notions ci-dessous sont également développées. Le contexte formel généralisé est appelé contexte généralisé, le produit $\mathfrak{C}\times$, $\sqcap$ et la somme $\mathfrak{C}\oplus$, $\sqcup$

□

De la même manière que nous l'avons montré avec les treillis de Galois exprimés à l'aide d'une relation binaire, nous allons montrer que l'ensemble des concepts formels d'un contexte peuvent s'organiser en treillis si A et B sont eux-mêmes des demi-treillis. Notre utilisation des treillis est essentiellement dans le cadre de l'apprentissage artificiel, donc nécessitera une programmation pour une construction, même partielle des treillis. C'est pourquoi nous présenterons pour les treillis de Galois des définitions constructives, utilisant des opérateurs produit $\otimes$ construisant les supremums, plutôt que des définitions supposant l'existence de ceux-ci.

Proposons tout d'abord des opérateurs produit $\otimes$ et somme $\oplus$ pour les concepts formels, basés respectivement sur les opérateurs produit $\otimes_A$ et $\otimes_B$.

Définition 4.24 (produit $\otimes$ et somme $\oplus$ sur les concepts formels)
Soit un contexte formel généralisé $\mathbb{C}\left(\mathcal{T}_\vee(A, \leq_A, \otimes_A), \mathcal{T}_\vee(B, \leq_B, \otimes_B), (f, g)\right)$.
Soit l'opérateur binaire **produit** $\otimes$ sur $\mathfrak{C}$:

$$\otimes_{\mathfrak{C}} : \mathfrak{C} \times \mathfrak{C} \to \mathfrak{C}$$

et défini par :
Soit $(F_{A1}, F_{B1}), (F_{A2}, F_{B2}) \in \mathfrak{C}$,
$(F_{A1}, F_{B1}) \otimes_{\mathfrak{C}} (F_{A2}, F_{B2}) = (F_{A\otimes_{\mathfrak{C}}}, F_{B\otimes_{\mathfrak{C}}})$ avec :

$$F_{A\otimes_{\mathfrak{C}}} = F_{A1} \otimes_A F_{A2} \text{ et } F_{B\otimes_{\mathfrak{C}}} = f(F_{A\otimes_{\mathfrak{C}}})$$

Soit l'opérateur binaire **somme** $\oplus$ sur $\mathfrak{C}$:

$$\oplus_{\mathfrak{C}} : \mathfrak{C} \times \mathfrak{C} \to \mathfrak{C}$$

et défini par :
Soit $(F_{A1}, F_{B1}), (F_{A2}, F_{B2}) \in \mathfrak{C}$,
$(F_{A1}, F_{B1}) \oplus_{\mathfrak{C}} (F_{A2}, F_{B2}) = (F_{A\oplus_{\mathfrak{C}}}, F_{B\oplus_{\mathfrak{C}}})$ avec :

$$F_{B\oplus_{\mathfrak{C}}} = F_{B1} \otimes_B F_{B2} \text{ et } F_{A\oplus_{\mathfrak{C}}} = g(F_{B\oplus_{\mathfrak{C}}})$$

Preuve 4.10 ($\otimes_{\mathfrak{C}}$ et $\oplus_{\mathfrak{C}}$ opérateurs produit et somme)
Comme $\otimes_A$ est un opérateur produit, par définition, $\otimes_{\mathfrak{C}}$ a les mêmes propriétés de construction du supremum pour la partie gauche des couples $(F_A, F_B) \in \mathfrak{C}$.
Les propriétés équivalentes sur la partie droite des couples $(F_A, F_B) \in \mathfrak{C}$ proviennent du fait que l'on ait une bijection entre les éléments $F_A \in F_{f\circ g}(A)$ et $F_B \in F_{g\circ f}(B)$.
On procède de la même manière pour démontrer que $\oplus_{\mathfrak{C}}$ est une somme. On notera que c'est bien l'opérateur produit de B, $\otimes_B$ qui sert à construire l'opérateur somme $\oplus_{\mathfrak{C}}$.

□

La structure d'ordre partiel sur les concepts formels ainsi que les opérateurs produit $\otimes_{\mathfrak{C}}$ et somme $\oplus_{\mathfrak{C}}$, nous amènent à la définition suivante.

Définition 4.25 (Treillis de Galois)
Soit un contexte formel généralisé $\mathbb{C}\big(\mathcal{T}_\vee(A, \leq_A, \otimes_A), \mathcal{T}_\vee(B, \leq_B, \otimes_B), (f, g)\big)$.
Soit la relation d'ordre partiel et opérateurs suivants :

- $\leq_{\mathfrak{C}}$ sur $\mathfrak{C}$ défini par :
  $\forall (F_{A1}, F_{B1}), (F_{A2}, F_{B2}) \in \mathfrak{C}$
  $F_{A1} \leq_A F_{A2} \Leftrightarrow (F_{A1}, F_{B1}) \leq_{\mathfrak{C}} (F_{A2}, F_{B2})$
- $\otimes_{\mathfrak{C}} : \mathfrak{C} \times \mathfrak{C} \to \mathfrak{C}$ défini par
  $(F_{A1}, F_{B1}), (F_{A2}, F_{B2}) \in \mathfrak{C}$,
  $(F_{A1}, F_{B1}) \otimes_{\mathfrak{C}} (F_{A2}, F_{B2}) = \big(F_{A1} \otimes_A F_{A2}, f(F_{A1} \otimes_A F_{A2})\big)$
- $\oplus_{\mathfrak{C}} : \mathfrak{C} \times \mathfrak{C} \to \mathfrak{C}$ défini par
  $(F_{A1}, F_{B1}), (F_{A2}, F_{B2}) \in \mathfrak{C}$,
  $(F_{A1}, F_{B1}) \oplus_{\mathfrak{C}} (F_{A2}, F_{B2}) = \big(g(F_{B1} \otimes_B F_{B2}), F_{B1} \otimes_B F_{B2}\big)$

L'ensemble des concepts formels $\mathfrak{C}$ de $\mathbb{C}$ muni de $\leq_{\mathfrak{C}}$, $\otimes_{\mathfrak{C}}$ et $\oplus_{\mathfrak{C}}$ est un treillis complet noté : $\mathcal{T}(\mathfrak{C}, \leq_A, \otimes_A, \otimes_B)$. Celui-ci est nommé **treillis de Galois**.
Nous faisons l'abus d'écriture qui consiste à remplacer $\leq_{\mathfrak{C}}$ par $\leq_A$, $\otimes_{\mathfrak{C}}$ par $\otimes_A$ et $\oplus_{\mathfrak{C}}$ par $\otimes_B$, car dans la notation générale pour les treillis que nous avons donnée, la place des éléments dans la parenthèse définit bien les propriétés de chacun des éléments. C'est à dire respectivement, ensemble, ordre partiel, produit et somme.

Preuve 4.11 ($\mathcal{T}(\mathfrak{C}, \leq_A, \otimes_A, \otimes_B)$ est un treillis)
$\leq_{\mathfrak{C}}$ basé directement sur $\leq_A$ est un ordre partiel pour $\mathfrak{C}$.
$\otimes_{\mathfrak{C}} : \mathfrak{C} \times \mathfrak{C} \to \mathfrak{C}$ et $\oplus_{\mathfrak{C}} : \mathfrak{C} \times \mathfrak{C} \to \mathfrak{C}$ définissant bien des opérateurs respectivement produit et somme pour $(\mathfrak{C}, \leq_A)$, $\mathcal{T}(\mathfrak{C}, \leq_A, \otimes_A, \otimes_B)$ est un treillis.

□

4.5 Treillis de Galois pour la description de l'espace de généralisation d'un ensemble d'objets contraints par un langage

Nous avons premièrement montré une utilisation du Treillis de Galois pour représenter un ensemble d'objets reliés par une relation binaire à un ensemble d'attributs. Ceci nous a permis de donner un exemple montrant l'intérêt d'utiliser le Treillis de Galois comme espace de généralisation contraint par un langage sur un ensemble d'objets.
Ensuite, nous avons donné, notamment avec la notion de correspondance de Galois, les conditions nécessaires et suffisantes à la construction d'un Treillis de Galois à partir de deux ensembles ayant une structure de demi-treillis.
Finalement, nous allons utiliser ces dernières propriétés pour donner les conditions générales permettant de décrire un espace de généralisation d'un ensemble d'objets contraint par un langage de description. Le schéma général de la démarche est représenté figure 4.7 page suivante.

Premièrement, nous allons montrer comment formellement, relier la notion de treillis de Galois et d'espace de généralisation contraint par un langage sur un ensemble d'objets. Nous donnerons notamment les notions de relation de description et de langage de généralisation.
Ensuite, nous montrerons qu'il y a une cohérence sémantique entre la notion d'espace de généralisation d'un ensemble d'objets à l'aide d'un langage et les contraintes formelles que nous avons posées.

4.5.1 Langage de Généralisation, fonction de description et d'instanciation et correspondance de Galois

Nous avons montré que pour construire un Treillis de Galois, plusieurs éléments étaient nécessaires : deux ensembles partiellement ordonnés avec une structure de treillis et une correspondance de Galois. Notre objectif est de montrer que le treillis de Galois est un cadre formel général pour formaliser la généralisation contrainte par un langage.
Nous devons donc dans ce cadre, identifier les éléments pour la construction du treillis de Galois.

Treillis des Parties Nous avons vu partie 4.2 que l'ensemble des regroupements possibles pour un ensemble d'objets $\mathcal{O}bj$ est le treillis des parties. Nous gardons un cadre constructif et donc nous définissons les treillis et demi-treillis à l'aide d'un opérateur produit $\otimes$. Ici, nous pouvons choisir comme demi-treillis soit $(\mathcal{O}bj, \subseteq, \cap)$, soit $(\mathcal{O}bj, \supseteq, \cup)$. $\cap$ et $\cup$ pouvant servir d'opérateur produit et soit $\subseteq$ ou $\supseteq$ pouvant servir de relation d'ordre partiel. D'un point de vue algorithmique, l'ensemble du treillis des parties d'un ensemble E peut être construit à partir de E.

Langage de Généralisation, fonction de description et d'instanciation Un des deux ensembles partiellement ordonné sera donc le treillis des parties des objets. L'autre sera un ensemble d'éléments constituant un langage pour décrire les objets. Notre objectif final étant de généraliser un apprentissage, les capacités du langage que nous choisissons portent donc sur la capacité pour deux éléments du langage de produire un généralisé. Pour ce, nous définissons comme langage de généralisation un langage ayant cette possibilité.

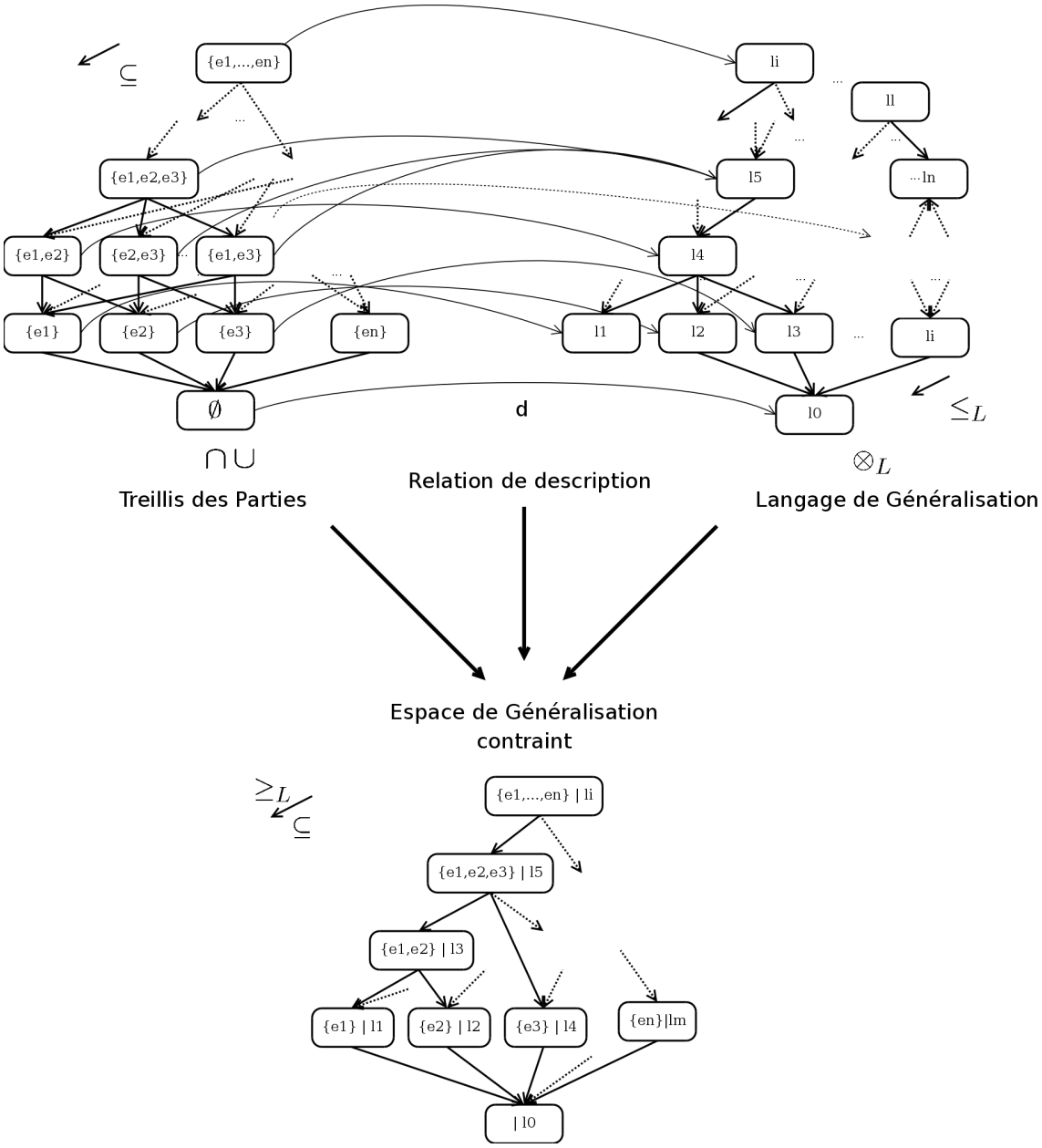


FIG. 4.7 – Schéma général de l'utilisation d'un treillis de Galois comme espace de généralisation pour un ensemble d'objets $\mathcal{O}bj$ contraint par un Langage de généralisation $(L, \leq_L, \otimes_L)$ et une relation de description d.

Définition 4.26 (Langage de description)
Soit un ensemble d'objets $\mathcal{O}bj$. Soit un ensemble L. S'il existe une fonction $d : \mathcal{O}bj \rightarrow L$, L est un **langage de description** pour $\mathcal{O}bj$ par la fonction de description d. $d(o), o \in \mathcal{O}bj$ est la **description** de l'objet O. On appellera les éléments de L des termes du langage L.

Définition 4.27 (fonction de description d)
Soit un ensemble d'objets $\mathcal{O}bj$ et un langage de description L pour $\mathcal{P}(\mathcal{O}bj)$ tel que $\mathcal{T}_\vee(L, \leq_L, \otimes_L)$ soit un demi-treillis.
Soit la fonction d définie de $\mathcal{P}(\mathcal{O}bj)$ dans L :

$$d : \mathcal{P}(\mathcal{O}bj) \rightarrow L \text{ et}$$

$$\forall O_1 \in \mathcal{P}(\mathcal{O}bj), d(O_1) = \bigotimes\nolimits_L \{d(o) \mid o \in O_1\}$$

d est alors appelée **fonction de description** de l'ensemble $\mathcal{O}bj$ par le langage de description L.

Remarque 4.3 *Pour représenter une fonction de description, compte-tenu de la définition que nous venons de donner, il est nécessaire de connaître explicitement la description $d(o)$ de chacun des objets $o \in \mathcal{O}bj$. Cependant, cette connaissance est suffisante pour calculer toute la fonction d pour les parties $\mathcal{O}bj$.*

Réciproquement, nous allons définir une fonction de L dans $\mathcal{P}(\mathcal{O}bj)$, associant à un élément du langage l les objets dont la description est moins générale que l.

Définition 4.28 (fonction d'instanciation i, objet couvert par un élément du langage)
Soit un ensemble d'objets $\mathcal{O}bj$ décrit par le langage de description L tel que $\mathcal{T}_\vee(L, \leq_L, \otimes_L)$ soit un demi-treillis et selon la fonction de description d.
Soit la fonction i, définie de L vers $\mathcal{P}(\mathcal{O}bj)$:

$$i : L \rightarrow \mathcal{P}(\mathcal{O}bj) \text{ définie par}$$

$$\forall l \in L, i(l) = \{o \in \mathcal{O}bj \mid d(o) \leq_L l\}$$

i est alors appelée **fonction d'instanciation** du langage de description L par l'ensemble $\mathcal{O}bj$.
Soit $o \in \mathcal{O}bj$ et $l \in L$ tels que $d(o) \leq l$, on dit que l'objet o est **couvert** par l.

Remarque 4.4 *La connaissance de la fonction de description $d(o)$ pour les objets $o \in \mathcal{O}bj$ est nécessaire est suffisante pour calculer la fonction i pour l'ensemble des termes du langage L.*

La définition suivante correspond exactement avec celle de *Learning Language*[34] défini dans [Liquière and Sallantin, 1998] et de *Description Language*[35] défini dans [Liquière, 2006].

Définition 4.29 (langage de généralisation, généralisé)
Soit un ensemble d'objets $\mathcal{O}bj$. Soit un langage de description L pour $\mathcal{P}(\mathcal{O}bj)$, tel que $\mathcal{T}_\vee(L, \leq_L, \otimes_L)$ soit un demi-treillis, soit la fonction de description d et la fonction d'instanciation i.
L est alors un **langage de généralisation** pour $\mathcal{O}bj$ par la relation de description d. La relation d'ordre $\leq_L$ est appelée relation de généralisation. Si $l_1 \leq_L l_2$ on dira l_2 est plus générale que l_1 et que l_1 est plus spécifique que l_2. Le supremum de l_1 et de l_2 appartient à L et est appelé le **généralisé** de l_1 et de l_2.

Dans le but de construire un treillis de Galois pour l'espace de généralisation d'un ensemble d'objets contraint par un langage, nous avons fait correpondre le treillis des parties avec un des deux ensembles de la définition de la correspondance de Galois, la notion de langage de généralisation va nous donner l'ensemble des autres éléments : l'autre demi-treillis ainsi que les fonctions duales.

Propriété 4.7
Soit un ensemble d'objets $\mathcal{O}bj$. Soit un langage de généralisation L pour $\mathcal{O}bj$ par la fonction d. d et i forment une correspondance de Galois entre $(\mathcal{P}(\mathcal{O}bj), \subseteq, \cap)$ et $(L, \geq_L, \otimes_L)$.

[34] langage d'apprentissage en français
[35] langage de description en français

Remarque 4.5 *On peut identifier sans conséquences $\leq_A$ à $\subseteq$ et $\leq_B$ à $\geq_L$ ou $\leq_A$ à $\supseteq$ et $\leq_B$ à $\leq_L$. La contrainte étant de respecter le fait que les fonctions d'instanciation et de description soient les fonctions adjointes d'une correspondance de Galois. Nous choisissons la première solution car il est plus aisé d'un point de vue sémantique et intelligibilité de considérer que les descriptions les plus grandes sont les plus générales. Si on représente classiquement les treillis avec les plus grands éléments « en haut » cela correspond, par conséquent, également à la façon de présenter les objets avec les ensembles de cardinal plus petit « en bas » pour avoir des ensembles avec de plus en plus d'éléments au fur et à mesure qu'on « monte » dans le treillis. reprendre la figure 4.5 page 83 pour se faire une idée.*

Preuve 4.12 (d et i forment une correspondance de Galois)

- $\forall O_1, O_2 \in \mathcal{P}(\mathcal{O}bj)$, $O_1 \subseteq O_2 \Rightarrow d(O_1) \leq_L d(O_2)$
  Soit $O_1, O_2 \in \mathcal{P}(\mathcal{O}bj)$ avec $O_1 \subseteq O_2$.
  Si $O_1 = O_2$ alors $d(O_1) = d(O_2)$.
  Si $O_1 \subset O_2$ alors soit $O_- = \{o \in \mathcal{O}bj \mid o \in O_2 \text{ et } o \notin O_1\}$.
  $d(O_2) = \bigotimes_L \{d(o) \mid o \in O_2\}$ et $d(O_1) = \bigotimes_L \{d(o) \mid o \in O_1\}$
  $\otimes_L$ étant associatif, on peut écrire $d(O_2) = d(O_1) \otimes d(O_-)$
  Or $d(O_1) \leq d(O_1) \otimes d(O_-)$. Donc $d(O_1) \leq_L d(O_2)$.
- $\forall l_1, l_2 \in L, l_1 \geq_L l_2 \Rightarrow i(l_1) \supseteq i(l_2)$
  Soit $l_1, l_2 \in L$ avec $l_1 \geq_L l_2$.
  $i(l_1) = \{o \in \mathcal{O}bj \mid d(o) \leq_L l_1\}$ et $i(l_2) = \{o \in \mathcal{O}bj \mid d(o) \leq_L l_2\}$
  Comme $l_1 \geq_L l_2$, si $d(o) \leq_L l_2$ alors $d(o) \leq_L l_1$.
  Par conséquent, $i(l_1) \supseteq i(l_2)$.
- $\forall O_1 \in \mathcal{P}(\mathcal{O}bj)$, $O_1 \subseteq i \circ d(O_1)$
  Soit $O_1 \in \mathcal{P}(\mathcal{O}bj)$
  $i \circ d(O_1) = \{o \in \mathcal{O}bj \mid d(o) \leq_L d(O_1)\}$
  Soit $o \in O_1$. $d(O_1) = \bigotimes_L \{d(o) \mid o \in O_1\}$ donc $d(o) \leq_L d(O_1)$.
  Donc finalement $o \in i \circ d(O_1)$.
- $\forall l_1 \in L$, $l_1 \geq_L d \circ i(l_1)$
  $i(l_1) = \{o \in \mathcal{O}bj \mid d(o) \leq_L d(O_1)\}$.
  l_1 est donc majorant de $\{d(o) \mid o \in i(l_1)\}$.
  Or $d \circ i(l_1) = \bigotimes_L \{d(o) \mid o \in i(l_1)\}$. Le produit d'un ensemble étant le plus petit de ses majorants, on a bien $l_1 \geq_L d \circ i(l_1)$.

□

La correspondance de Galois entre les deux demi-treillis $(\mathcal{P}(\mathcal{O}bj), \subseteq, \cap)$, $(L, \leq_L, \otimes_L)$ et les fonctions de description d et d'instanciation i forment alors un treillis de Galois. Que nous appelons espace de généralisation.

Propriété 4.8 (Espace de Généralisation)
Soit un ensemble d'objets $\mathcal{O}bj$, un langage de généralisation $(L, \leq_L, \otimes_L)$ décrivant $\mathcal{O}bj$ par la relation de description d et la relation d'instanciation i.
$\mathcal{T}\big((F_{d \circ i}(\mathcal{O}bj), F_{i \circ d}(L)), \subseteq, \otimes_L, \cap_{\mathcal{O}bj}\big)$ est un treillis de Galois nommé **espace de généralisation** de l'ensemble $\mathcal{O}bj$ contraint par le langage de généralisation $(L, \leq_L, \otimes_L)$ et la fonction de description d. Par abus d'écriture et pour simplifier les notations, nous noterons $\mathcal{TG}(\mathcal{O}bj, L)$ ce treillis. Cette notation implique implicitement que $\mathcal{O}bj$ est un ensemble, L un langage de généralisation et qu'il existe une fonction de description $d : \mathcal{O}bj \to L$ et d'instanciation $i : L \to \mathcal{O}bj$.

Remarque 4.6 *La notation utilisée ci-dessus inverse celle donnée par rapport à la définition 4.25 page 87. En effet, nous avons identifié le demi-treillis $\mathcal{T}_\vee(A, \leq_A, \otimes_A)$ au demi-treillis $\mathcal{T}_\vee(\mathcal{P}(\mathcal{O}bj), \subseteq, \cap)$ et le demi-treillis $\mathcal{T}_\vee(B, \leq_B, \otimes_B)$ au demi-treillis $\mathcal{T}_\vee(L, \geq_L, \otimes_L)$.*
L'opérateur produit pour les concepts était donc celui fourni par l'opérateur produit $\otimes_A$ et l'opérateur somme était fourni à partir de l'opérateur $\otimes_B$. Ici, nous avons inversé. C'est sans conséquences compte-tenu de la dualité mais peut-être source d'erreurs.

Remarque 4.7 *Compte-tenu des définitions que nous avons données dans ce paragraphe, l'ensemble $\mathcal{O}bj$, le langage $(L, \leq_L, \otimes_L)$ et la description $d(o)$ des éléments $o \in \mathcal{O}bj$ sont les seuls éléments nécessaires à la construction de $\mathcal{T}\big((F_{d \circ i}(\mathcal{O}bj), F_{i \circ d}(L), \subseteq_{\mathcal{O}bj}, \cap_{\mathcal{O}bj}, \otimes_L\big)$.*

Exemple 4.12 *Nous allons montrer comme exemple que la définition classique des treillis de Galois d'une relation binaire entre un ensemble d'objets $\mathcal{O}bj$ décrits par un ensemble d'attributs $\mathcal{A}tt$ est bien un espace de généralisation.*
Identifions les divers éléments :

- *$\mathcal{O}bj$ est toujours $\mathcal{O}bj$*
- *L est $\mathcal{P}(\mathcal{A}tt)$*
- *$\leq_L$ est $\subseteq$*
- *$\otimes_L$ est $\cap$*
- *$d : \mathcal{P}(\mathcal{O}bj) \to L$ est $d : \mathcal{P}(\mathcal{O}bj) \to \mathcal{P}(\mathcal{A}tt)$*
- *$i : L \to \mathcal{P}(\mathcal{O}bj)$ est $i : \mathcal{P}(\mathcal{A}tt) \to \mathcal{P}(\mathcal{O}bj)$*
- *La relation binaire décrivant chacun des éléments de $\mathcal{O}bj$ dans $\mathcal{A}tt$ est bien la seule fonction explicite à connaître.*
- *$(F_{d \circ i}(\mathcal{O}bj), F_{i \circ d}(L))$ sont bien les concepts formels du treillis de Galois de la relation binaire.*

4.5.2 Sémantique du treillis de Galois en tant qu'espace de généralisation

Dans ce paragraphe, nous allons montrer la cohérence entre les impératifs algébriques que nous avons posés dans les définitions de construction d'un treillis de Galois et son utilisation en tant qu'espace de généralisation pour un ensemble d'objets contraint par un langage. Ceci permettra également de résumer les éléments nécessaires afin de pouvoir formaliser ce type d'espace pour un ensembles d'objets.

Langage de description Nous avons proposé comme langage de description d'un ensemble $\mathcal{O}bj$, par un langage L, une fonction de chaque éléments de $\mathcal{O}bj$ vers un élément de L. La procédure est tout à fait classique, on associe à chaque élément une description dans un langage donné.

Fonction de description Nous avons imposé une contrainte supplémentaire par rapport au langage de description. Si on regroupe des objets entre eux, on doit toujours pouvoir décrire l'ensemble ainsi constitué à l'aide du langage de description. C'est pourquoi le langage de description porte sur $\mathcal{P}(\mathcal{O}bj)$ plutôt que sur $\mathcal{O}bj$. De plus, nous avons imposé en corollaire la contrainte qui est que la description d'un ensemble d'objets puisse être calculée à partir des descriptions de chacun des objets qui le compose. Ceci impose donc l'opérateur produit $\otimes_L$. On peut voir que la complexité de calcul de la description d'un ensemble d'objets sera fonction du produit $\otimes_L$ et du nombre d'objets.

Fonction d'instanciation La fonction d'instanciation est, rappelons-le, la recherche pour un terme l du langage L, de l'ensemble des objets dont la description est plus spécifique que l. C'est un fonction coûteuse en terme de complexité, car elle est fonction du nombre total d'objets à représenter, et non pas seulement du nombre d'objets dont la description est plus spécifique que l.

Remarque 4.8 *Nous avons présenté les fonctions adjointes d'une correspondance de Galois comme des fonctions et non comme des applications. C'est à dire qu'à un élément de A est associé un et un seul élément de B et réciproquement. Or, d'un point de vue sémantique, il serait acceptable d'une part que deux objets puissent avoir la même description et réciproquement qu'un même élément du langage puisse désigner deux ensembles d'objets différents. Pour se conformer au formalisme mathématique, on peut par exemple généraliser le procédé de clarification de contexte donné dans [Ganter and Wille, 1999], c'est à dire fusionner les objets ayant même description et éléments du langage correspondants au même objet.*

Langage de généralisation Nous avons employé le terme de langage de généralisation pour un ensemble d'objets $\mathcal{O}bj$ pour définir un langage L ayant une structure de demi-treillis et tel qu'il existe une fonction d'intanciation et de description entre $\mathcal{O}bj$ et L.
Ceci est justifié par le fait que L permet de décrire tout regroupement d'éléments de $\mathcal{O}bj$ entre eux. De plus, à partir de la description d'un regroupement, on effectue implicitement une généralisation.

En effet, la description engendrée par le regroupement de deux éléments n'a pas simplement regroupé ceux-ci, mais aussi implicitement l'ensemble des objets qui ne sont pas représentés dans $\mathcal{O}bj$ et qui peuvent être décrits à l'aide de L.
En règle général, dans le cadre de l'apprentissage artificiel, pour trouver une règle ou une classification à partir d'un ensemble d'exemples, on n'essaie pas d'induire sur l'ensemble des exemples, mais sur un échantillon. Ainsi, le terme de langage décrivant la généralisation de deux exemples est en fait également une généralisation sur les exemples non-vus.
Nous le verrons dans la partie 5, pour l'apprentissage par renforcement, l'utilisation sera entre autre, qu'une politique apprise pour la généralisation de plusieurs états de l'environnement sera également appliquée aux états de l'environnement juste découverts et pour lesquels par conséquent, un algorithme d'apprentissage par renforcement classique n'aurait pas eu d'a priori sur l'action à sélectionner.

d et i forment une correspondance de Galois Nous avons également montré que les fonctions d et i que nous avons définies forment une correspondance de Galois. Voyons l'implication sémantique d'une correspondance de Galois dans le cadre de la généralisation et montrons leur cohérence.

- $\forall O_1, O_2 \in \mathcal{P}(\mathcal{O}bj)$, $O_1 \subseteq O_2 \Rightarrow d(O_1) \leq_L d(O_2)$
  Cette implication impose que si un ensemble d'éléments en inclus un autre, alors l'élément de langage décrivant le plus grand des deux ensembles doit être plus général que l'autre.
- $\forall l_1, l_2 \in L, l_1 \geq_L l_2 \Rightarrow i(l_1) \supseteq i(l_2)$
  Inversement, pour deux termes de langage dont l'un est plus général que l'autre, le plus général des deux termes doit regrouper plus d'exemples.
- $\forall O_1 \in \mathcal{P}(\mathcal{O}bj)$, $O_1 \subseteq i \circ d(O_1)$
  Cette implication impose que la description généralisant un ensemble d'éléments O_1 doit potentiellement regrouper au moins les éléments de O_1 et éventuellement plus.
- $\forall l_1 \in L$, $l_1 \geq_L d \circ i(l_1)$
  Cette implication est sans doute la moins intuitive d'un point de vue sémantique. Elle indique que le terme généralisant l'ensemble des éléments qui peuvent être regroupés par un terme l du langage L, est plus spécifique (éventuellement égal) que le terme l. Pour comprendre en quoi elle est cohérente, on peut dire que la généralisation sur le langage d'un ensemble d'éléments $\mathcal{O}bj$, doit être un terme qui certes, couvre toutes les descriptions des éléments de $\mathcal{O}bj$, mais pas plus. Ainsi, s'il y a plusieurs généralisations possibles pour les descriptions d'un ensemble d'éléments $\mathcal{O}bj$, celle qu'il convient de produire est la moins générale d'entre elle.

4.5.3 Implémentation et construction des treillis de Galois

L'implémentation machine des treillis de concepts est un sujet de recherche en lui-même. De nombreux algorithmes existent pour ce faire (par exemple Chein [Chein, 1969], Bordat [Bordat, 1986], Godin [Godin et al., 1995], Nourine [Nourine and Raynaud, 1999], Norris [Norris, 1978] pour ne citer que des classiques). Nous renvoyons le lecteur aux références traitant de ce sujet. [Kuznetzov and Obiedkov, 2002] propose une comparaison des différents algorithmes de construction des treillis de Galois pour une relation binaire avec une étude de la complexité. Ce sujet de recherche est tojours actif, voir [Choi, 2006] par exemple.
Dans le cadre de nos travaux autour de l'apprentissage par renforcement, nous privilégions les approches de construction incrémentale du treillis (par exemple [Nourine and Raynaud, 1999], [Norris, 1978], [van der Merwe et al., 2004]). En effet, comme nous le verrons dans les chapitres suivants, l'ensemble des objets qui seront structurés par treillis seront l'ensemble des états de l'environnement dans lequel évolue l'agent. Nous partons du postulat que l'agent ne connaît pas l'ensemble des états de l'environnement au début de son apprentissage. Il faut donc pouvoir rajouter des objets, ici des états de l'environnement, au fur et à mesure de leur découverte par l'agent. De plus, nous ne partons pas du postulat que les objets sont décrits par un ensemble d'attributs, mais plus généralement par un langage de généralisation. Il nous faut donc un algorithme ne reposant pas exclusivement sur ce type de représentation.
Pour nos expériences impliquant la construcion de treillis (voir chapitre 6 page 125), nous utilisons donc une version incrémentale de l'algorithme de Norris ([Norris, 1978]) qui a ces deux

caractéristiques.

4.6 Treillis de Galois des Partitions

Nous avons vu dans la partie 3.3 page 54 la notion d'Homomorphisme de Processus de Décision Markovien comme opérateur algébrique pouvant servir à une minimisation d'un Processus de Décision Markovien tout en préservant sa dynamique.
Si cette notion est utile pour formaliser certaines formes de généralisation pour l'apprentissage par renforcement, elle ne donne pas de méthodes en soit pour trouver ces généralisations. Il est bien entendu impensable d'un point de vue combinatoire d'énumérer l'ensemble des applications ayant une chance de correspondre à un homomorphisme pour sélectionner ceux qui remplissent les conditions.
Nous allons ici proposer une structure algébrique nous permettant également de considérer la généralisation par partitionnement des états de l'environnement (l'utilisation pour l'apprentissage par renforcement sera faite dans le chapitre suivant). Nous allons adopter cependant un point de vue différent en donnant une structure permettant d'utiliser des propriété similaires à celles du treillis de Galois classique.
Dans la partie précédente, nous avons vu comment un langage de description L muni d'opérateur produit $\otimes_L$, d'une relation d'ordre $\geq_L$ pouvait limiter les classifications possibles d'un ensemble d'objets décrits grâce à L en utilisant la notion de treillis de Galois.

Nous allons dans cette partie, présenter une structure algébrique originale, le *treillis de Galois des partitions*, permettant d'appliquer les propriétés de généralisation, de découverte de règles et biais de langage pour un ensemble d'objets. Cependant, à la place de considérer le treillis des parties, c'est à dire les possibilités de regrouper des éléments entre eux, nous considérerons le treillis des partitions, c'est à dire l'ensemble des façons de partitionner un ensemble d'éléments.
Notre objectif est donc, de manière similaire à la démarche que nous avons adoptée pour le treillis de Galois classique, de présenter un espace produit par les fermés d'une correspondance de Galois.

La figure 4.8 page ci-contre donne un schéma général de notre démarche. Il nous faut alors identifier l'ensemble des éléments d'une correspondance de Galois :

- Les ensembles partiellement ordonnés $(A, \leq_A, \otimes_A)$ $(B, \leq_B, \otimes_B)$
- La correspondance de Galois $f : A \to B$ et $g : B \to A$
- Le treillis de Galois produit par les fermés des fermetures $f \circ g$ et $g \circ f$

Nous aurons alors premièrement à définir les notions autour du treillis des partitions, que nous identifierons à l'ensemble $(A, \leq_A, \otimes_A)$. Deuxièmement, nous aurons à définir la notion de partition respectant un langage de généralisation $(L, \leq_L, \otimes_L)$. Ceci nous permettra de décrire la fonction de description d'une partition $d_{\mathbb{P}}$, les ensembles d'éléments de L pouvant être instanciés et la fonction d'instanciation d'éléments du langage L. Ces éléments seront respectivement identifiés à la fonction f, à l'ensemble partiellement ordonné $(B, \leq_B, \otimes_B)$ et à la fonction g.
Nous proposerons finalement les propriétés de l'ensemble des fermés produits par la correspondance de Galois ainsi construite : l'ensemble des partitions respectant un langage, c'est à dire le treillis de Galois des partitions.

4.6.1 Partitions d'un ensemble

Définition 4.30 (couvertures $\mathbb{C}(E)$ d'un ensemble)
Soit un ensemble E. Une **couverture** de E est un ensemble $\mathcal{P}art$ tel que

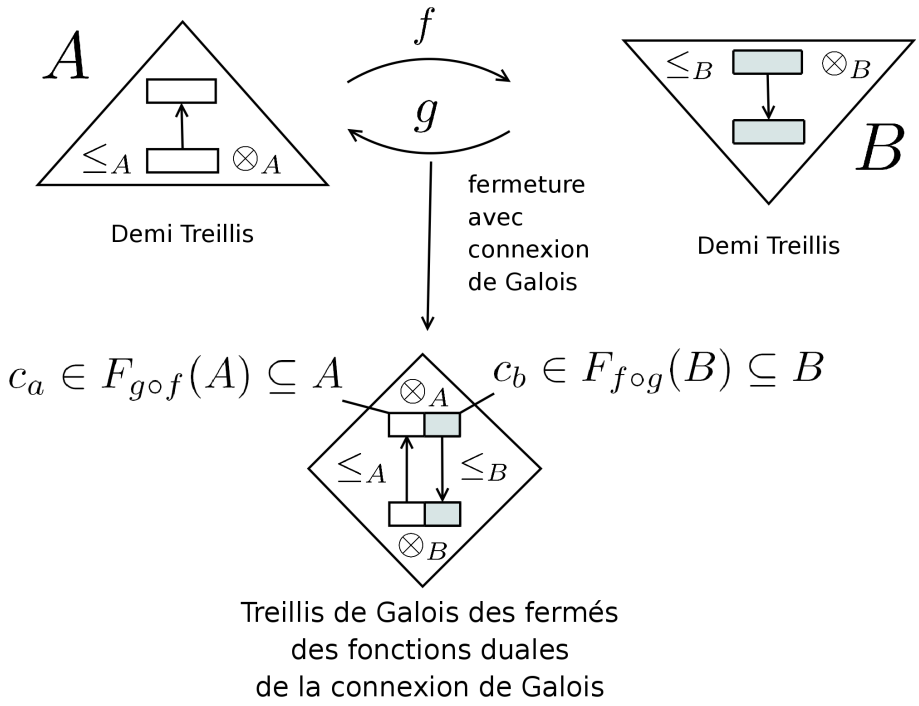
1. $\forall P \in \mathcal{P}art, P \neq \emptyset$
2. $\bigcup\{P \mid P \in \mathcal{P}art\} = E$

L'ensemble des couvertures de E est noté $\mathbb{C}(E)$.

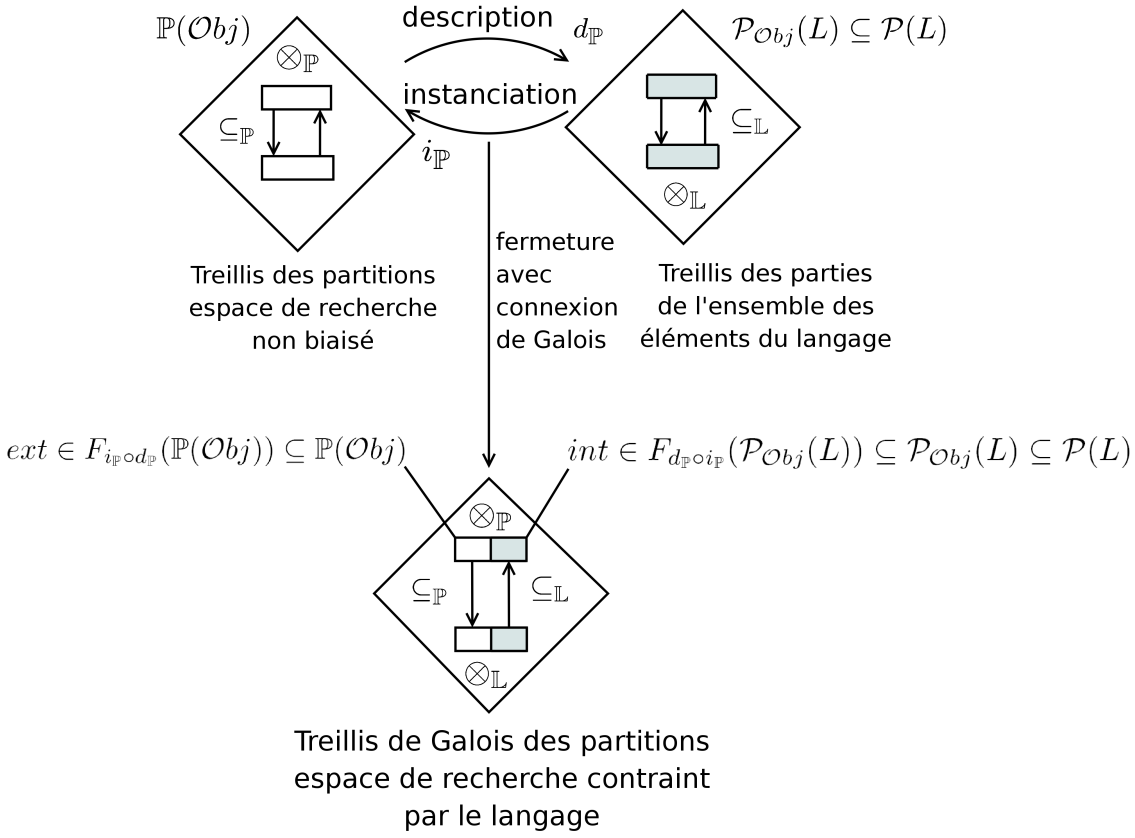
Définition 4.31 (partitions $\mathbb{P}(E)$ d'un ensemble)
Soit un ensemble E. Une **partition** de E est un ensemble $\mathcal{P}art$ tel que

1. $\mathcal{P}art$ est une couverture de E
2. $\forall P_1, P_2 \in \mathcal{P}art, P_1 \neq P_2,\ P_1 \cap P_2 = \emptyset$

L'ensemble des partitions de E est noté $\mathbb{P}(E)$.



(a): Eléments de contruction d'un treillis de Galois



(b): Eléments de contruction d'un treillis de Galois des partitions

FIG. 4.8 – Une instance spécifique du treillis de Galois : le treillis de Galois des partitions

Remarque 4.9 *On remarquera que tous les éléments d'une partition sont des parties de E. En effet, sinon leur union ne pourrait pas être E. Une partition est donc l'union de parties de E dont l'union est exactement E et qui sont deux à deux disjointes. Nous nous servirons de cette remarque ultérieurement.*

Exemple 4.13 *Soit un ensemble $E = \{e_1, e_2, e_3, e_4\}$.*
$\mathcal{P}art = \big\{\{e_1, e_2\}, \{e_3, e_4\}\big\}$ est une partition de E.
L'ensemble des partitions de E sont représentées figure 4.10 page 101.

Le nombre de partitions d'un ensemble est la nombre de façon de regrouper les éléments de celui-ci les uns avec les autres. Ce nombre est connu sous le nom de de *Nombre de Bell*.

Propriété 4.9 (nombre de partitions d'un ensemble, nombre de Bell)
Soit un ensemble E, $E \neq \emptyset$ tel que $|E| = n$. Le nombre de partitions de E est défini par récurence :

$$B_1 = 1$$
$$B_{(n+1)} = \sum_{k=0}^{n} C_n^k B_k$$

L'ensemble des nombres B_n, $n \in \mathbb{N}$ sont appelés nombres de Bell[36]

Exemple 4.14 $B_1 = 1, B_2 = 2, B_3 = 5, B_4 = 15, B_5 = 52, B_6 = 203, ..., B_{20} = 51724158235372$

4.6.2 Partitions et relation d'équivalence

Les résultats qui suivent sont des résultats classiques d'algèbre, mais dont nous nous servirons de façon importante au cours de nos travaux, c'est pourquoi nous les rappelons. Ils établissent la correspondance entre une relation d'équivalence et le partitionnement d'un ensemble.

Définition 4.32 (Relation d'équivalence)
Soit un ensemble E et une relation binaire R de E dans lui-même. R est une **relation d'équivalence** notée $\equiv_E$ ou $\equiv$ s'il n'y a pas d'ambiguité, si est seulement si :

1. $\equiv$ est un préordre ($\equiv$ est *réflexive* et *transitive*)
   - $\equiv$ est *réflexive* : $\forall e_1 \in E, e_1 \equiv e_1$
   - $\equiv$ est *transitive* : $\forall e_1, e_2, e_3 \in E, e_1 \equiv e_2$ et $e_2 \equiv e_3 \Rightarrow e_1 \equiv e_3$
2. $\equiv$ est *symétrique* : $\forall e_1, e_2 \in E, e_1 \equiv e_2 \Rightarrow e_2 \equiv e_1$

Définition 4.33 (Classe d'équivalence)
Soit E un ensemble, $\equiv$, une relation d'équivalence sur E et soit $e \in E$.
L'ensemble des éléments $e_1 \in E$ tels que $e \equiv e_1$ définit la classe d'équivalence de e sur E par la relation $\equiv$. Celle-ci est notée $[e]_\equiv$ ou $[e]$ s'il n'y a pas d'ambiguïté.

Deux théorèmes réciproques établissent le fait qu'à partir d'une partition d'un ensemble, on définit une relation d'équivalence satisfaisant cette partition et réciproquement, qu'une relation d'équivalence sur un ensemble, définit une unique partition sur celui-ci.

Théorème 4.1 *Soit un ensemble E et soit $\mathcal{P}art \in \mathbb{P}(E)$, une partition de E. La relation binaire sur E, $\equiv_{\mathcal{P}art}$ définie par :*

$$\forall e_1, e_2 \in E, e_1 \equiv_{\mathcal{P}art} e_2 \Leftrightarrow \exists! P \in \mathcal{P}art \text{ tel que } \{e_1\} \cup \{e_2\} \subseteq P$$

est une relation d'équivalence.
La relation $\equiv_{\mathcal{P}art}$ peut se traduire par e_1 et e_2 appartiennent au même ensemble P de $\mathcal{P}art$.
Les éléments de la partitions définissent les classes d'équivalence $C(e)$ des éléments $e \in E$ par la partition $\mathcal{P}art$.

[36] On définit généralement également $B_0 = 0$.

Preuve 4.13

- L'unicité de P pour chaque $e \in E$ vient du fait que l'intersection des éléments de $\mathcal{P}art$ soient disjoints deux à deux.

- Montrons que $\equiv_{\mathcal{P}art}$ est une relation d'équivalence.
  1. **réflexivité**
     $\forall e \in E$, il existe bien un unique $P \in \mathcal{P}art$ tel que $\{e\} \subseteq P$, car $\mathcal{P}art$ est une partition.
  2. **transitivité**
     Soit e_1, e_2 et $e_3 \in E$ tels que $e_1 \equiv_{\mathcal{P}art} e_2$ et $e_2 \equiv_{\mathcal{P}art} e_3$.
     $\exists! P_1 \in \mathcal{P}art$ avec $\{e_1\} \cup \{e_2\} \subseteq P_1$ et $\exists! P_2 \in \mathcal{P}art$ avec $\{e_2\} \cup \{e_3\} \subseteq P_2$. $\mathcal{P}art$ étant une partition, e_2, e_1, e_3 n'appartiennent qu'à un seul élément de $\mathcal{P}art$. Donc $P_1 = P_2$ est l'unique élément de $\mathcal{P}art$ tel que $\{e_1\} \cup \{e_3\} \subseteq P_1$.
  3. **symétrie**
     Immédiat par la symétrie de la relation sur les ensembles $\cup$.
- $\{e_1\} \cup \{e_2\} \subseteq P \Leftrightarrow e_1 \in P$ et $e_2 \in P$. Les éléments $P \in \mathcal{P}art$ regroupent des éléments équivalents entre eux et seulement ceux-ci car P est unique. Les éléments $P \in \mathcal{P}art$ sont donc bien les classes d'équivalence $C(e)$, $e \in E$.

□

Théorème 4.2 *Réciproquement, soit une relation d'équivalence $\equiv$ sur un ensemble E. Les classes d'équivalence $C(e)$, $e \in E$ définissent une partition $\mathcal{P}art_{\equiv} \in \mathbb{P}(E)$.*

Preuve 4.14

Montrons que $\mathcal{P}art_{\equiv}$ ainsi défini est bien une partition unique.

1. $\forall P \in \mathcal{P}art_{\equiv}, P \neq \emptyset$
   Comme $\equiv$ est réflexive, $\forall e \in E, e \equiv e$. Donc $\forall P \in \mathcal{P}art_{\equiv}, P \neq \emptyset$.
2. $\bigcup\{P \mid P \in \mathcal{P}art_{\equiv}\} = E$
   Le domaine de définition de $\equiv$ est E. Pour tout $e \in E$, il existe au moins une classe d'équivalence contenant e car $\equiv$ est réflexive. De plus, la relation ne concerne que les éléments de E. Donc $\bigcup\{P \mid P \in \mathcal{P}art_{\equiv}\} = E$.
3. $\forall P_1, P_2 \in \mathcal{P}art_{\equiv}, P_1 \cap P_2 = \emptyset$
   Soit $e \in P_1$ et $e \in P_2$. $\forall e_1 \in P_1$, $e \equiv e_1$, $\forall e_2 \in P_2$, $e \equiv e_2$, par transitivité, $e_1 \equiv e_2$. Donc $e_1 \in P_2$ et $e_2 \in P_1$. Donc finalement, $P_1 = P_2$.

□

Définition 4.34 (Quotient)

Soit un ensemble E, une relation d'équivalence $\equiv$ sur E et la partition $\mathcal{P}art_{\equiv} \in \mathbb{P}(E)$ constituée des classes d'équivalence de $\equiv$. $\mathcal{P}art_{\equiv}$ est alors appelé le **quotient** de E par la relation $\equiv$ et noté $E/\equiv$.

Définissons maintenant la notion de *langage d'équivalence* qui nous servira au chapitre suivant pour décrire deux états ou des actions ayant la même description.

Définition 4.35 (Langage d'équivalence)

Soit un ensemble d'objets $\mathcal{O}bj$. Soit un langage de description L sur $\mathcal{O}bj$. Soit une relation d'équivalence $\equiv_L$ sur L. L est alors un **langage d'équivalence** sur $\mathcal{O}bj$ noté $(L, \equiv_L)$.

Propriété 4.10

Soit un ensemble d'objets $\mathcal{O}bj$ et un langage d'équivalence $(L, \equiv_L)$ sur $\mathcal{O}bj$. $(L, \equiv_L)$ définit une relation d'équivalence entre les éléments de $\mathcal{O}bj$ notée également $\equiv_L$. $\equiv_L$ définit une partition de $\mathcal{O}bj$ notée $P_{\equiv_L}(\mathcal{O}bj)$.

Preuve 4.15

Immédiat avec le fait que le langage de description soit une fonction et le théorème 4.2 exprimant le fait qu'une relation d'équivalence sur un ensemble définit une partition de celui-ci.

□

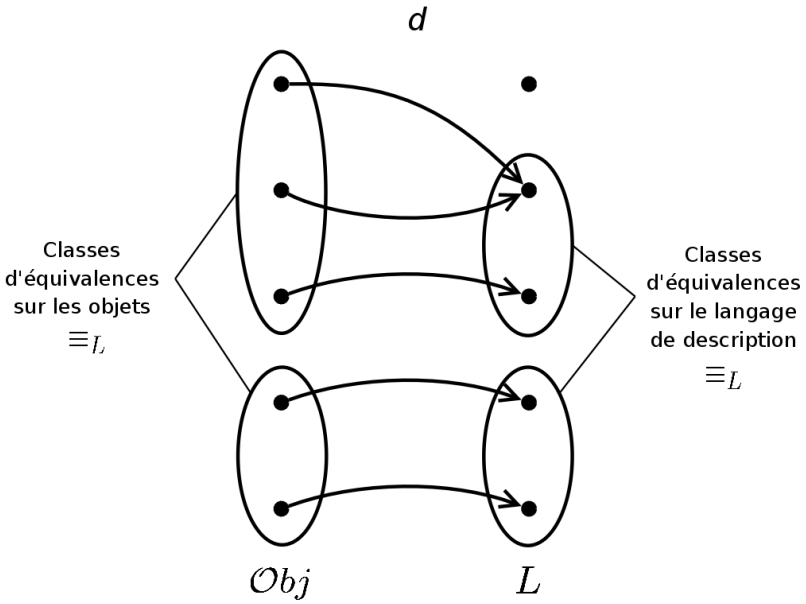


FIG. 4.9 – Exemple où l'on peut voir qu'un langage d'équivalence $(L, \equiv_L)$ sur un ensemble d'objets $\mathcal{O}bj$ implique une partition de $\mathcal{O}bj$

4.6.3 Treillis des partitions

Nous allons maintenant montrer que les partitions peuvent s'organiser en un treillis nommé treillis des partitions. Définissons d'abord la relation d'ordre partiel $\subseteq_{\mathbb{P}}$ ainsi que les opérateurs produit $\otimes_{\mathbb{P}}$ et somme $\oplus_{\mathbb{P}}$. Nous procédons à l'aide d'opérateurs plutôt que par la définition de *suppremum* $\vee$ et d'*infimum* $\wedge$, toujours pour la nécéssité d'avoir une méthode constructive.

Définition 4.36 ($\subseteq_{\mathbb{P}}$ ordre partiel sur les partitions)
Soit un ensemble E, l'ensemble $\mathbb{P}(E)$ des partitions de E la relation $\subseteq_{\mathbb{P}}$ définie par

$$\forall \mathcal{P}art_1, \mathcal{P}art_2 \in \mathbb{P}(E), \mathcal{P}art_1 \subseteq_{\mathbb{P}} \mathcal{P}art_2 \Leftrightarrow$$
$$\forall P_1 \in \mathcal{P}art_1, \exists P_2 \in \mathcal{P}art_2 \text{ tel que } P_1 \subseteq P_2$$

$\subseteq_{\mathbb{P}}$ est une relation d'ordre partielle pour $\mathbb{P}(E)$.
$\mathcal{P}art_1 \subseteq_{\mathbb{P}} \mathcal{P}art_2$ se lit $\mathcal{P}art_2$ est plus grossière que $\mathcal{P}art_1$ ou que $\mathcal{P}art_1$ est plus fine que $\mathcal{P}art_2$.

Exemple 4.15 *Soit un ensemble* $E = \{e_1, e_2, e_3, e_4\}$.
Soit $\mathcal{P}art_1 = \{\{e_1, e_2\}, \{e_3, e_4\}\}$ *et* $\mathcal{P}art_2 = \{\{e_1\}, \{e_2\}, \{e_3, e_4\}\}$ *deux partitions de* E. $\mathcal{P}art_1$ *est une partition plus grossière que* $\mathcal{P}art_2$ *:* $\mathcal{P}art_2 \subseteq_{\mathbb{P}} \mathcal{P}art_1$.

Preuve 4.16

- **reflexivité** $\mathcal{P}art_1 \subseteq_{\mathbb{P}} \mathcal{P}art_1$
  $\forall \mathcal{P}art_1 \in \mathbb{P}(E)$, tous les ensembles de $\mathcal{P}art_1$ sont bien inclus dans un ensemble de $\mathcal{P}art_1$ puisqu'ils sont inclus dans eux-mêmes.
- **transitivité**
  Soit $\mathcal{P}art_1, \mathcal{P}art_2, \mathcal{P}art_3 \in \mathbb{P}(E)$ tels que
  $\mathcal{P}art_1 \subseteq_{\mathbb{P}} \mathcal{P}art_2$ et $\mathcal{P}art_2 \subseteq_{\mathbb{P}} \mathcal{P}art_3$.
  $\forall P_1 \in \mathcal{P}art_1, \exists P_2 \in \mathcal{P}art_2 \mid P_1 \subseteq P_2$
  or, $\forall P_2 \in \mathcal{P}art_2, \exists P_3 \in \mathcal{P}art_3 \mid P_2 \subseteq P_3$
  donc $\forall P_1 \in \mathcal{P}art_1, \exists P_3 \in \mathcal{P}art_3 \mid P_1 \subseteq P_3$ donc $\mathcal{P}art_1 \subseteq_{\mathbb{P}} \mathcal{P}art_3$
- **anti-symétrie**
  Soit $\mathcal{P}art_1, \mathcal{P}art_2 \in \mathbb{P}(E)$ tels que
  $\mathcal{P}art_1 \subseteq_{\mathbb{P}} \mathcal{P}art_2$ et $\mathcal{P}art_2 \subseteq_{\mathbb{P}} \mathcal{P}art_1$
  Soit $P_1 \in \mathcal{P}art_1$. Il existe $P_2 \in \mathcal{P}art_2$ tel que $P_1 \subseteq P_2$. Or, il existe $P_3 \in \mathcal{P}art_1$ tel que

$P_2 \subseteq P_3$.
Comme tous les ensembles de la partition $\mathcal{P}art_1$ sont deux à deux disjoints, $P_1 = P_3$.
On a donc bien $\mathcal{P}art_1 = \mathcal{P}art_2$.

□

Remarque 4.10 *Notons que la relation d'ordre $\subseteq_{\mathbb{P}}$ est définie pour tout ensemble de parties de E et pas seulement pour les partitions de E. Pour le démontrer, la démonstration de l'antisymétrie doit néanmoins être remaniée, car elle utilise la propriété de partitionnement.*

Nous allons maintenant définir l'opérateur produit $\otimes_{\mathbb{P}}$ pour les partitions et basé sur la relation d'ordre partiel $\subseteq_{\mathbb{P}}$.

Définition 4.37 (Opérateur produit $\otimes_{\mathbb{P}}$)
Soit un ensemble E, l'ensemble partiellement ordonné de ses partitions $(\mathbb{P}(E), \subseteq_{\mathbb{P}})$ et deux partitions de E, $\mathcal{P}art_1$ et $\mathcal{P}art_2$. Soit R la relation binaire sur E définie par :

$$\forall e_i, e_j \in E, e_i R e_j \Leftrightarrow e_i \text{ et } e_j \text{ sont dans la même partie dans } \mathcal{P}art_1 \textbf{ ou } \text{dans } \mathcal{P}art_2.$$

Soit $R_{\otimes}$, la fermeture transitive de R. $R_{\otimes}$ définit la partition $\mathcal{P}art_1 \otimes_{\mathbb{P}} \mathcal{P}art_2$.
Le produit de deux partitions $\mathcal{P}art_1$ et $\mathcal{P}art_2$ est donc la partition formée de telle sorte que si deux éléments sont dans la même classe d'équivalence dans $\mathcal{P}art_1$ ou $\mathcal{P}art_2$ alors, ils sont dans la même classe d'équivalence dans la partition produit. De plus, la fermeture transitive assure que le résultat est bien une partition.[37]

Remarque 4.11 *Notons que l'opérateur $\otimes_{\mathbb{P}}$ peut en fait être appliqué à n'importe quelle famille (c'est à dire ensemble de parties) d'un ensemble, pas seulement aux partitions. Nous nous servirons par la suite de cet opérateur indistinctement sur des familles quelconques ou des partitions, en gardant la spécificité que le produit de deux partitions reste bien une partition.*

Preuve 4.17 ($R_{\otimes}$ est une relation d'équivalence)
- **réflexité**
  $\forall e_1 \in E$, e_1 appartient bien à la même partie que lui même.
- **symétrie**
  Soit $e_1, e_2 \in E$, on a bien si e_1 est dans le même ensemble que e_2, alors e_2 est dans le même ensemble que e_1.
- **transitivité**
  La transitivité est impliquée par la fermeture transitive de R.

□

Exemple 4.16 *Soit un ensemble $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Soit $\mathcal{P}art_1 = \big\{\{e_1, e_2\}, \{e_3, e_4\}, \{e_5, e_6\}\big\}$ et $\mathcal{P}art_2 = \big\{\{e_1\}, \{e_2\}, \{e_3\}, \{e_4, e_5\}, \{e_6\}\big\}$.*
$\mathcal{P}art_1 \otimes_{\mathbb{P}} \mathcal{P}art_2 = \big\{\{e_1, e_2\}, \{e_3, e_4, e_5, e_6\}\big\}$.
Notons que e_3 et e_6 ne sont pas dans la même partie ni dans $\mathcal{P}art_1$, ni dans $\mathcal{P}art_2$, mais qu'ils le sont dans la partition produit à cause de la fermeture transitive.

Preuve 4.18 ($\otimes_{\mathbb{P}}$ est un opérateur produit)
Soit $\mathcal{P}art_{\otimes_{\mathbb{P}}} = \mathcal{P}art_1 \otimes_{\mathbb{P}} \mathcal{P}art_2$.
- **domaine**
  Montrons que $\mathcal{P}art_{\otimes_{\mathbb{P}}}$ est une partition de E.
  E est non vide et $R_{\otimes}$ est une relation d'équivalence.

[37] Il existe de nombreux algorithmes pour calculer la fermeture transitive de graphes orientés, comme celui de *Floyd-Warshall*. Ils peuvent être utilisés pour calculer le produit de deux partitions.

- **majorant**
  Montrons que $\mathcal{P}art_{\otimes_{\mathbb{P}}} \geq_{\mathbb{P}} \mathcal{P}art_1$ et $\mathcal{P}art_{\otimes_{\mathbb{P}}} \geq_{\mathbb{P}} \mathcal{P}art_2$.
  Si $\forall e_1, e_2 \in E$, $(\exists! P_1 \in \mathcal{P}art_1$ tel que $\{e_1\} \cup \{e_2\} \subseteq P_1) \Rightarrow (\exists! P_\otimes \in Part_\otimes$ tel que $\{e_1\} \cup \{e_2\} \subseteq P_\otimes)$ alors $\forall P_1 \in \mathcal{P}art_1, \exists! P_\otimes \in \mathcal{P}art_{\otimes_{\mathbb{P}}}$ tel que $P_1 \subseteq P_\otimes$.
  Donc $\mathcal{P}art_{\otimes_{\mathbb{P}}} \geq_{\mathbb{P}} \mathcal{P}art_1$.
  Idem pour $\mathcal{P}art_{\otimes_{\mathbb{P}}} \geq_{\mathbb{P}} \mathcal{P}art_2$.
- **plus petit des majorants**
  Par définition, la fermeture transitive de R produit la plus petite relation transitive incluant R.

□

Dualement, définissons l'opérateur somme $\oplus_{\mathbb{P}}$ de deux partitions.

Définition 4.38 (Opérateur somme $\oplus_{\mathbb{P}}$)
Soit un ensemble E, l'ensemble partiellement ordonné de ses partitions $(\mathbb{P}(E), \subseteq_{\mathbb{P}})$ et deux partitions de E, $\mathcal{P}art_1$ et $\mathcal{P}art_2$. Soit $R_\oplus$ la relation binaire sur E définie par :

$$\forall e_i, e_j \in E, e_i R_\oplus e_j \Leftrightarrow e_i \text{ et } e_j \text{ sont dans la même partie dans } \mathcal{P}art_1 \textbf{ et } \text{dans } \mathcal{P}art_2.$$

$R_\oplus$ définit la partition $\mathcal{P}art_1 \oplus_{\mathbb{P}} \mathcal{P}art_2$.
La somme de deux partitions $\mathcal{P}art_1$ et $\mathcal{P}art_2$ est donc la partition formée de telle sorte que si deux éléments sont dans la même classe d'équivalence dans $\mathcal{P}art_1$ et $\mathcal{P}art_2$ alors, ils sont dans la même classe d'équivalence dans la partition somme.

Notons que par rapport à l'opérateur produit il n'est pas nécessaire de faire une fermeture transitive.

Preuve 4.19 ($R_\oplus$ est une relation d'équivalence)
- **réflexité**
  $\forall e_1 \in E$, e_1 appartient bien à la même partie que lui même.
- **symétrie**
  Soit $e_1, e_2 \in E$, on a bien si e_1 est dans le même ensemble que e_2, alors e_2 est dans le même ensemble que e_1.
- **transitivité**
  La transitivité est impliquée par le fait que l'appartenance à une classe d'équivalence est une relation d'équivalence, donc transitive.

□

Exemple 4.17 *Soit un ensemble* $E = \{e_1, e_2, e_3, e_4, e_5\}$. *Soit* $\mathcal{P}art_1 = \big\{\{e_1\}, \{e_2\}, \{e_3, e_4, e_5\}\big\}$ *et* $\mathcal{P}art_2 = \big\{\{e_1, e_2\}, \{e_3\}, \{e_4, e_5\}\big\}$.
$\mathcal{P}art_1 \oplus_{\mathbb{P}(E)} \mathcal{P}art_2 = \big\{\{e_1\}, \{e_2\}, \{e_3\}, \{e_4, e_5\}\big\}$

Preuve 4.20 ($\oplus_{\mathbb{P}}$ est un opérateur somme)
La démonstration est duale avec la démonstration de $\otimes_{\mathbb{P}}$ est un opérateur produit, mais comporte cependant quelques différences.
Soit $\mathcal{P}art_{\oplus_{\mathbb{P}}} = \mathcal{P}art_1 \oplus_{\mathbb{P}} \mathcal{P}art_2$.
- **domaine**
  Montrons que $\mathcal{P}art_{\oplus_{\mathbb{P}}}$ est une partition de E.
  E est non vide et $R_\oplus$ est une relation d'équivalence.
- **minorant**
  Montrons que $\mathcal{P}art_{\oplus_{\mathbb{P}}} \leq_{\mathbb{P}} \mathcal{P}art_1$ et $\mathcal{P}art_{\oplus_{\mathbb{P}}} \leq_{\mathbb{P}} \mathcal{P}art_2$.
  Par construction de la somme $\oplus_{\mathbb{P}}$, pour tout élément $P_1 \in \mathcal{P}art_1$ et $P_2 \in \mathcal{P}art_2$, il existe un élément $P_\oplus \in \mathcal{P}art_{\oplus_{\mathbb{P}}}$ qui soit tel que $P_\oplus \subseteq P_1$ et $P_\oplus \subseteq P_2$, car tout élément $P_1 \in \mathcal{P}art_1$ et $P_2 \in \mathcal{P}art_2$ génère un élément $P_\oplus \in \mathcal{P}art_{\oplus_{\mathbb{P}}}$ par intersection.

- **plus grand des minorants**
  Montrons que si $\exists \mathcal{P}art_3 \in \mathbb{P}(E)$ tel que $\mathcal{P}art_3 \leq_{\mathbb{P}} \mathcal{P}art_1$ et $\mathcal{P}art_3 \leq_{\mathbb{P}} \mathcal{P}art_2$ et $\mathcal{P}art_3 \geq_{\mathbb{P}} \mathcal{P}art_{\oplus_{\mathbb{P}}}$ alors $\mathcal{P}art_3 = \mathcal{P}art_{\oplus_{\mathbb{P}}}$.
  On a $\forall P_3 \in \mathcal{P}art_3$, $\exists P_1 \in \mathcal{P}art_1$ tel que $P_3 \subseteq P_1$ et $\forall P_3 \in \mathcal{P}art_3$, $\exists P_2 \in \mathcal{P}art_2$ tel que $P_3 \subseteq P_2$.
  Soit $e_1, e_2 \in E$ tels que e_1 et e_2 appartiennent à la même partition P_3 dans $\mathcal{P}art_3$. On a alors e_1 et e_2 appartiennent à la même partition P_1 dans $\mathcal{P}art_1$ et e_1 et e_2 appartiennent à la même partition P_2 dans $\mathcal{P}art_2$.
  Donc e_1 et e_2 appartiennent à la même partition $P_\oplus$ dans $\mathcal{P}art_\oplus$.
  Donc $\forall P_3 \in \mathcal{P}art_3, \exists P_\oplus \in \mathcal{P}art_\oplus$ tel que $P_3 \subseteq P_\oplus$.

□

Des opérateurs produit $\otimes_{\mathbb{P}(E)}$ et somme $\oplus_{\mathbb{P}(E)}$, découle immédiatamment la structure de treillis.

Définition 4.39 (Treillis des partitions)
Soit un ensemble E. $(\mathbb{P}(E), \subseteq_{\mathbb{P}(E)})$ muni des opérateurs $\otimes_{\mathbb{P}(E)}$ et $\oplus_{\mathbb{P}(E)}$ est un treillis $\mathcal{T}(\mathbb{P}(E), \subseteq_{\mathbb{P}(E)}, \otimes_{\mathbb{P}(E)}, \oplus_{\mathbb{P}(E)})$ noté $\mathcal{T}_{\mathbb{P}(E)}$ et appelé treillis des partitions de E.
$1_{\mathbb{P}(E)} = E$ et $O_{\mathbb{P}(E)}$ est l'ensemble composé de singletons avec chacun des éléments de E. Sa hauteur est de $|E|$. Son nombre d'éléments est égale au nombre de Bell de $|E|$.

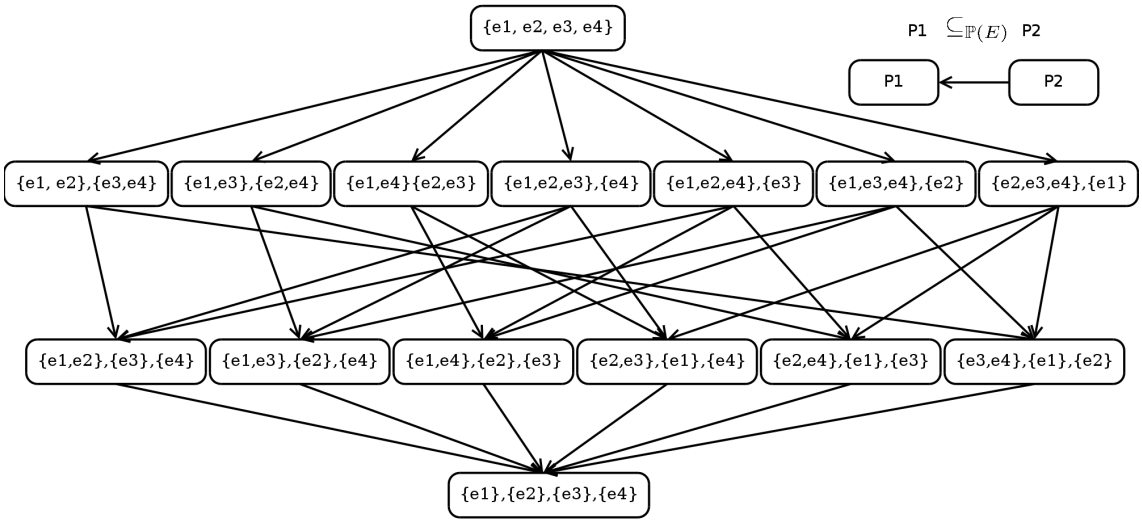


FIG. 4.10 – Treillis des partitions $\mathcal{T}_{\mathbb{P}(E)}$, avec $E = \{e_1, e_2, e_3, e_4\}$

Exemple 4.18 *La figure 4.10 montre le treillis des partitions d'un ensemble de quatre éléments. Il contient 15 éléments et est de hauteur 4.*

4.6.4 Langage de description des partitions

Nous venons, avec la structure de treillis des partitions, de décrire la structure pour l'ensemble des façon possibles de regrouper un ensemble d'objets. Nous allons voir comment contraindre cette structure à l'aide d'un langage de description de la même manière que nous l'avons fait pour le treillis des parties.
Rappelons d'abord les idées qui sous-tendent la structure mathématique que nous proposons.
Pour cela, il nous faut déterminer (se référer à la figure 4.8 page 95) :

- les ensembles partiellement ordonnés $(A, \leq_A, \otimes A)$ et $(B, \leq_B, \otimes B)$
- la fonction $f : A \to B$
- la fonction $g : B \to A$
- que les fonctions f et g forment bien une correspondance de Galois

• l'ensemble organisé en treillis des fermés de la correspondance de Galois $\mathcal{T}(\mathfrak{C}, \leq_A, \otimes_A, \otimes_B)$

Nous allons tout d'abord déterminer, puisque cela a été notre démarche et pour donner au lecteur une intuition du résultat, l'ensemble des fermés que nous comptons obtenir. Nous détaillerons ensuite les autres éléments permettant de donner la structure algébrique. Commençons par la notion de *partition respectant un langage de description.*

Définition 4.40 (Partie ou Partition respectant un langage de description)
Soit un ensemble d'objets $\mathcal{O}bj$, un langage de généralisation $(L, \leq_L, \otimes_L)$ décrivant $\mathcal{O}bj$ par la relation de description $d : \mathcal{P}(\mathcal{O}bj) \to L$ et la relation d'instanciation $i : L \to \mathcal{P}(\mathcal{O}bj)$.
$P \in \mathcal{P}(\mathcal{O}bj)$ est une partie **respectant** L ssi P est un fermé de la fonction de fermeture $i \circ d : \mathcal{P}(\mathcal{O}bj) \to \mathcal{P}(\mathcal{O}bj)$, c'est à dire $i \circ d(P) = P$.
$\{P_1, \ldots, P_n\}$ est une partition de $\mathcal{O}bj$ **respectant** L ssi $\forall j, P_j$ respecte L.
On notera $\mathbb{P}_L(\mathcal{O}bj)$ l'ensemble des partitions de $\mathcal{O}bj$ respectant L.

Les éléments du treillis de Galois dans le cadre d'un espace de généralisation peuvent se voir comme l'ensemble des parties d'un ensemble d'objets pouvant être décrites par un langage de généralisation donné et dualement, l'ensemble des éléments du langage de généralisation décrivant une partie de l'ensemble des objets. Ainsi, une partition respectant un langage est une partition dont l'ensemble des parties peut être décrite par un élément du langage de généralisation.

Exemple 4.19 *Reprenons l'exemple défini par le contexte formel suivant. Soit $\mathcal{O}bj$ = {Dame de Coeur, 7 de Coeur, 7 de Pique, Roi de Carreau}, L = ({7, Dame, Roi, Carreau, Coeur, Rouge, Noir, Pique}, $\subseteq, \cup$). les relations d'instanciation i et de description d sont définies par la matrice suivante.*

	7	*Dame*	*Roi*	*Carreau*	*Coeur*	*Rouge*	*Noir*	*Pique*
Dame de Coeur		×			×	×		
7 de Coeur	×				×	×		
7 de Pique	×						×	×
Roi de Carreau			×	×		×		

La partition {{Dame de Coeur, 7 de Coeur, Roi de Carreau}, {7 de Pique}} respecte L. En effet,
$i \circ d$({Dame de Coeur, 7 de Coeur, Roi de Carreau}) = i({Rouge}) = {Dame de Coeur, 7 de Coeur, Roi de Carreau}
et $i \circ d$({7 de Pique}) = i({7, Noir, Pique}) = {7 de Pique}
tandis que la partition {{Dame de Coeur, 7 de Coeur}, {Roi de Carreau, 7 de Pique}} ne respecte pas L. En effet
$i \circ d$({Roi de Carreau, 7 de Pique}) = i({$\emptyset$}) = {Dame de Coeur, 7 de Coeur, Roi de Carreau, 7 de Pique}
Cela peut se traduire par le fait que le langage L permet de regrouper la Dame de Coeur, le 7 de Coeur et le Roi de Carreau (par l'élément du langage L {Rouge}). Par contre, il n'existe pas d'élément du langage L permettant de regrouper Le Roi de Carreau et le 7 de Pique à l'exclusion d'autres éléments.

L'espace que nous cherchons à produire est donc l'ensemble des partitions respectant un langage de description.
Concernant l'identification des éléments pour la correspondance de Galois, l'ensemble partiellement ordonné $(A, \leq_A, \otimes A)$ sera donc le treillis des partitions $\mathcal{T}_{\mathbb{P}(E)}$.
La description d'une partition sera un ensemble d'éléments du langage de description L. Ainsi, l'ensemble partiellement ordonné $(B, \leq_B, \otimes B)$, que nous noterons $\mathcal{P}_{\mathcal{O}bj}(L)$ sera inclus dans le treillis des parties de L (nous préciserons ceci plus loin).

description d'une partition Voyons maintenant la définition de la fonction de description d'une partition.
La fonction de description d'une partition $d_{\mathbb{P}}$ sera identifiée à la fonction $f : A \to B$ de notre schéma. Dualement, la fonction d'instanciation d'une partition $i_{\mathbb{P}}$ sera identifiée à la fonction $g : B \to A$.
Comme nous avons déterminé que les fermés de la correspondance de Galois $i_{\mathbb{P}} \circ d_{\mathbb{P}}$ et $d_{\mathbb{P}} \circ i_{\mathbb{P}}$

devaient avoir pour extension une partition respectant le langage de description L, il faut que la description d'une partition ait pour instanciation une partition respectant le langage L (ceci est dû à l'idempotence d'une correspondance de Galois).

Définition 4.41 (langage de description de partitions)
Soit un ensemble d'objets $\mathcal{O}bj$ et un langage de généralisation L pour $\mathcal{O}bj$ défini par $\mathcal{T}_\vee(L, \leq_L, \otimes_L)$ ainsi que la fonction de description $d : \mathcal{P}(\mathcal{O}bj) \to L$ et la fonction d'instanciation $i : L \to \mathcal{P}(\mathcal{O}bj)$. La fonction de description d'une partition

$$d_{\mathbb{P}} : \mathbb{P}(\mathcal{O}bj) \to \mathcal{P}_{\mathcal{O}bj}(L)$$

$\forall \mathcal{P} \in \mathbb{P}(\mathcal{O}bj)$, $d_{\mathbb{P}}(\mathcal{P})$ est la partie de L formée par la description par la fonction d de chacune des parties de la plus petite partition respectant L et telle que $\mathcal{P} \subseteq_{\mathbb{P}} d_{\mathbb{P}}(\mathcal{P})$.

Données :
- Un ensemble d'objets $\mathcal{O}bj$
- Un langage de généralisation L pour $\mathcal{O}bj$ défini par $\mathcal{T}_\vee(L, \leq_L, \otimes_L)$
- La fonction de description $d : \mathcal{P}(\mathcal{O}bj) \to L$
- La fonction d'instanciation $i : L \to \mathcal{P}(\mathcal{O}bj)$
- Une partition $\mathcal{P} \in \mathbb{P}(\mathcal{O}bj)$. $\mathcal{P} = \{E_1, \ldots, E_n\}$

Résultat :
- $d_{\mathbb{P}}(\mathcal{P})$, la description de la partition $\mathcal{P}$

début
  $\mathcal{P}_{res} \leftarrow \{i \circ d(E_1), \ldots, i \circ d(E_n)\}$;
  tant que $\mathcal{P}_{res}$ *n'est pas une partition* **faire**
    $\mathcal{P}_{res} \leftarrow$ fermeture transitive($\mathcal{P}_{res}$);
    $\mathcal{P}_{res} \leftarrow \{i \circ d(E_1^{'}), \ldots, i \circ d(E_n^{'})\}$;
  fin
  $d_{\mathbb{P}}(\mathcal{P}) \leftarrow \{i(E_1^{'}), \ldots, i(E_n^{'})\}$;
  retourner $d_{\mathbb{P}}(\mathcal{P})$;
fin

Algorithme 4.1: Description d'une partition

L'algorithme 4.1 permet de construire une description correspondant à la définition 4.41. Précisons quelques éléments.

- Premièrement, il s'agit de fermer chacune des parties de la partition à décrire $\mathcal{P}$. Cela produit une couverture de la partition. En effet, d'un point de vue mathématique, la fermeture est extensive, donc chacune des parties peut avoir grandit. Du point de vue sémantique, on obtient le résultat des règles impliquées par le treillis de Galois du type « si ces éléments sont à regrouper, alors selon le langage, il faut également inclure ceux-la ».
- Si la couverture est une partition, cela implique que chacune des parties est un fermé par la relation de description $i \circ d$ et donc que la partition respecte le langage.
- Si des parties ont des éléments en commun, il faut alors les fermer transitivement afin d'obtenir une partition. Puis refaire la même procédure sur la partition obtenue jusqu'à l'obtention d'une partition respectant L.
- Finalement, chacun des éléments de la partition obtenue est décrit avec la fonction de description d.

Exemple 4.20 *Soit* $\mathcal{P} = \big\{\{$*Dame de Coeur, Roi de Carreau*$\}, \{$*7 de Coeur*$\}, \{$*7 de Pique*$\}\big\}$
$\mathcal{P}_{res} = \big\{i \circ d(\{$*Dame de Coeur, Roi de Carreau*$\}), i \circ d(\{$*7 de Coeur*$\}), i \circ d(\{$*7 de Pique*$\})\big\}$
$\mathcal{P}_{res} = \big\{\{$*Dame de Coeur, Roi de Carreau, 7 de Coeur*$\}, \{$*7 de Coeur*$\}, \{$*7 de Pique*$\}\big\}$
fermeture transitive de $\mathcal{P}_{res} = \big\{\{$*Dame de Coeur, Roi de Carreau, 7 de Coeur*$\}, \{$*7 de Pique*$\}\big\}$
$\mathcal{P}_{res_1} = \big\{i \circ d(\{$*Dame de Coeur, Roi de Carreau, 7 de Coeur*$\}), i \circ d(\{$*7 de Pique*$\})\big\}$
$\mathcal{P}_{res_1} = \big\{\{$*Dame de Coeur, Roi de Carreau, 7 de Coeur*$\}, \{$*7 de Pique*$\}\big\}$
$\mathcal{P}_{res_1}$ *est une partition donc*

$d_{\mathbb{P}}(\mathcal{P}) = \big\{i(\{\textit{Dame de Coeur, Roi de Carreau, 7 de Coeur}\}), i(\{\textit{7 de Pique}\})\big\}$
$d_{\mathbb{P}}(\mathcal{P}) = \big\{\{\textit{Rouge}\}, \{\textit{7, Pique, Noir}\}\big\}$

instanciation d'éléments du langage L Voyons maintenant dualement la fonction $i_{\mathbb{P}}$ qui servira de fonction adjointe pour la correspondance de Galois : la fonction d'instanciation d'éléments du langage L.

La fonction d'instanciation $i_{\mathbb{P}}$ doit, à partir d'un ensemble d'éléments du langage L, produire une partition respectant le langage L. Rappelons que $\mathcal{P}_{\mathcal{O}bj}(L) \subseteq \mathcal{P}(L)$, nous préciserons ceci au regard de la définition.

Définition 4.42 (fonction d'instanciation d'une partition $i_{\mathbb{P}}$)
Soit un ensemble d'objets $\mathcal{O}bj$ et un langage de généralisation L pour $\mathcal{O}bj$ défini par $\mathcal{T}_{\vee}(L, \leq_L, \otimes_L)$, la fonction de description $d : \mathcal{P}(\mathcal{O}bj) \to L$ et la fonction d'instanciation $i : L \to \mathcal{P}(\mathcal{O}bj)$.
Soit la fonction $i_{\mathbb{P}}$ définie par :

$$i_{\mathbb{P}} : \mathcal{P}_{\mathcal{O}bj}(L) \to \mathbb{P}(\mathcal{O}bj)$$

$$\forall \mathcal{L} \in \mathcal{P}_{\mathcal{O}bj}(L), \mathcal{L} = \{L_1, \ldots, L_j\}, i_{\mathbb{P}}(\mathcal{L}) = \big\{i(L_1), \ldots, i(L_j)\big\}$$

L'instanciation d'une partie $\mathcal{L}$ d'éléments du langage L est donc la partition composée de l'instanciation par i de chacun des éléments de $\mathcal{L}$.
D'un point de vue sémantique, l'instanciation d'un ensemble d'éléments du langage peut se traduire par la recherche de la partition qui regroupe les éléments couverts par un même éléments du langage L_j et qui séparent ceux qui sont couverts par deux éléments du langage différents, L_j, L_k, $j \neq k$.
Analysons maintenant les problèmes liés au domaine de définition de $i_{\mathbb{P}}$. Si on considère comme domaine de définition toutes les parties de L, $\mathcal{P}(L)$, on se trouve confronter à quelques difficultés mathématiques.
En effet, le problème est que le résultat n'est pas nécessairement une partition.

- Des éléments de $\mathcal{O}bj$ peuvent ne pas être présents suite à l'instanciation. On pourra toujours pallier à ce problème en rajoutant les éléments de $\mathcal{O}bj$ manquant sous forme de singletons.
- Les parties de $\mathcal{O}bj$ obtenues peuvent ne pas être disjointes. Reprenons l'interprétation sémantique proposée ci-dessus. On considère un élément e de $\mathcal{O}bj$ appartenant à deux parties E_j et E_k de $\mathcal{O}bj$ issues de l'instanciation. On est en train d'essayer de trouver une partition telle que e soit regroupé avec tous les éléments de E_j et avec tous les éléments de E_k, mais de telle sorte que E_j et E_k soient disjointes. Une telle partition n'existe tout simplement pas. C'est pourquoi nous allons exclure du domaine de définition les parties de L dont l'instanciation que nous venons de décrire produit une famille de $\mathcal{O}bj$ dont les éléments ne sont pas disjoints deux à deux. Nous notons $\mathcal{P}_{\mathcal{O}bj}(L)$ un tel ensemble.

Définition 4.43 ($\mathcal{P}_{\mathcal{O}bj}(L)$)
Soit un ensemble d'objets $\mathcal{O}bj$ et un langage de généralisation L pour $\mathcal{O}bj$ défini par $\mathcal{T}_{\vee}(L, \leq_L, \otimes_L)$ ainsi que $i_{\mathbb{P}}$, la fonction d'instanciation d'éléments de L.
$\mathcal{P}_{\mathcal{O}bj}(L) \subseteq \mathcal{P}(L)$ est l'ensemble des parties de L dont l'instanciation par la fonction $i_{\mathbb{P}}$ est une partition de $\mathcal{O}bj$.

Exemple 4.21 *Soit* $\mathcal{L}_1 = \big\{\{\textit{7}\}, \{\textit{Dame}\}, \{\textit{Carreau}\}\big\}$
$i_{\mathbb{P}}(\mathcal{L}_1) = \big\{i(\{\textit{7}\}), i(\{\textit{Dame}\}), i(\{\textit{Carreau}\})\big\} \otimes_{\mathbb{P}} \big\{\{\textit{Dame de Coeur}\}, \{\textit{Roi de Carreau}\}, \{\textit{7 de Coeur}\}, \{\textit{7 de Pique}\}\big\}$
$i_{\mathbb{P}}(\mathcal{L}_1) = \big\{\{\textit{7 de Coeur, 7 de Pique}\}, \{\textit{Dame de Coeur}\}, \{\textit{Roi de Carreau}\}\big\} \otimes_{\mathbb{P}} \big\{\{\textit{Dame de Coeur}\}, \{\textit{Roi de Carreau}\}, \{\textit{7 de Coeur}\}, \{\textit{7 de Pique}\}\big\}$
$i_{\mathbb{P}}(\mathcal{L}_1) = \big\{\{\textit{7 de Coeur, 7 de Pique}\}, \{\textit{Dame de Coeur}\}, \{\textit{Roi de Carreau}\}\big\}$
$\mathcal{L}_2 = \big\{\{\textit{7}\}, \{\textit{Rouge}\}\big\}$
$i_{\mathbb{P}}(\mathcal{L}_2)$ *n'est pas défini. En effet*
$\big\{i(\{\textit{7}\}), i(\{\textit{Rouge}\})\big\} = \big\{\{\textit{7 de Coeur, 7 de Pique}\}, \{\textit{7 de Coeur, Dame de Coeur, Roi de Carreau}\}\big\}$
Les parties ne sont pas disjointes (le 7 de Coeur est en commun).

Définition 4.44 ($\subseteq_{\mathbb{L}}$)
Soit un ensemble d'objets $\mathcal{O}bj$ et un langage de généralisation L pour $\mathcal{O}bj$ défini par $\mathcal{T}_{\vee}(L, \leq_L, \otimes_L)$ ainsi que la fonction de description $d : \mathcal{P}(\mathcal{O}bj) \rightarrow L$ et la fonction d'instanciation $i : L \rightarrow \mathcal{P}(\mathcal{O}bj)$. Soit $\mathcal{L}_1$, $\mathcal{L}_1 \in \mathcal{P}_{\mathcal{O}bj}(L)$.

$$\mathcal{L}_1 \subseteq_{\mathbb{L}} \mathcal{L}_2 \Leftrightarrow \forall L_2 \in \mathcal{L}_2 \exists L_1 \in \mathcal{L}_1 \text{ tel que } L_2 \leq_L L_1$$

$\subseteq_{\mathbb{L}}$ est une relation d'ordre partiel pour $\mathcal{P}(L)$ et donc pour $\mathcal{P}_{\mathcal{O}bj}(L)$.
On dira que $\mathcal{L}_1$ est plus général que $\mathcal{L}_2$ ou que $\mathcal{L}_2$ est plus spécifique que $\mathcal{L}_1$.

La preuve que $\subseteq_{\mathbb{L}}$ est une relation d'ordre partiel est similaire à la preuve 4.16 pour la relation $\subseteq_{\mathbb{P}}$.

Exemple 4.22 *Soit* $\mathcal{L}_1 = \big\{\{7,\ Pique,\ Noir\}, \{Dame,\ Coeur,\ Rouge\}, \{7,\ Coeur,\ Rouge\}, \{Roi,\ Carreau,\ Rouge\}\big\}$ *et* $\mathcal{L}_2 = \big\{\{7,\ Pique,\ Noir\}, \{Rouge\}\big\}$.
On a $\mathcal{L}_1 \subseteq_{\mathbb{L}} \mathcal{L}_2$.

Définition 4.45 ($\otimes_{\mathbb{L}}$)
Soit un ensemble d'objets $\mathcal{O}bj$ et un langage de généralisation L pour $\mathcal{O}bj$ défini par $\mathcal{T}_{\vee}(L, \leq_L, \otimes_L)$.
Soit $\mathcal{L}, \mathcal{L}' \in \mathcal{P}_{\mathcal{O}bj}(L)$. $\mathcal{L} = \{L_1, \ldots, L_n\}, \mathcal{L}' = \{L'_1, \ldots, L'_m\}$.
L'opérateur produit $\otimes_{\mathbb{L}} : \mathcal{P}_{\mathcal{O}bj}(L) \times \mathcal{P}_{\mathcal{O}bj}(L) \rightarrow \mathcal{P}_{\mathcal{O}bj}(L)$ est définit par :

$$\mathcal{L} \otimes_{\mathbb{L}} \mathcal{L}' = \bigcup \big\{ d \circ i(L_i) \otimes_L d \circ i(L'_j) \text{ tels que } L_i \in \mathcal{L}, L'_j \in \mathcal{L}' \text{ et } i(L_i) \cap i(L'_j) \neq \emptyset \big\}$$

Pour faire le produit de deux ensembles $\mathcal{L}_1$ et $\mathcal{L}_2$ d'éléments d'un langage L, il faut :
- fermer chacun des éléments de $\mathcal{L}_1$ et de $\mathcal{L}_2$.
- chacun des éléments de $\mathcal{O}bj$ est couvert par un est un seul élément de $\mathcal{L}_1$ et de $\mathcal{L}_2$. Pour chacun des éléments de $\mathcal{O}bj$, on trouve donc le généralisé de l'élément de $\mathcal{L}_1$ et de $\mathcal{L}_2$ correspondant.
- finalement, on supprime les doublons avec l'opérateur d'union.

Le résultat est bien un ensemble d'éléments de L dont l'instanciation est une fermeture (voir la preuve ci-dessous).

Preuve 4.21 ($\otimes_{\mathbb{L}}$ est un opérateur produit)
Soit $\mathcal{L}_1 \in \mathcal{P}_{\mathcal{O}bj}(L)$, $\mathcal{L}_1 = \{L_{11}, \ldots, L_{1n}\}$ et $\mathcal{L}_2 \in \mathcal{P}_{\mathcal{O}bj}(L)$, $\mathcal{L}_2 = \{L_{21}, \ldots, L_{1m}\}$
$\mathcal{L}'_1 = \{d \circ i(L_{11}), \ldots, d \circ i(L_{1n})\}, \mathcal{L}'_2 = \{d \circ i(L_{21}), \ldots, d \circ i(L_{2m})\}$
$i_{\mathbb{P}}(\mathcal{L}_1) = i_{\mathbb{P}}(\mathcal{L}'_1)$ car $\forall L_{1j} \in \mathcal{L}_1, i \circ d \circ i(\mathcal{L}_1) = i(\mathcal{L}_1)$
de même, $i_{\mathbb{P}}(\mathcal{L}_2) = i_{\mathbb{P}}(\mathcal{L}'_2)$.
Toutes les parties $P_{1j} = i(L_{1j})$ et $P_{2k} = i(L_{2k})$ sont les extensions d'un concept du treillis de Galois $\mathcal{T}((F_{d \circ i}(\mathcal{O}bj), F_{i \circ d}(L)), \subseteq, \cap, \otimes_L)$ dont l'intension est respectivement $d \circ i(L_{1j})$ et $d \circ i(L_{2k})$.
Par conséquent, il est dual d'appliquer l'opérateur $\otimes_{\mathbb{L}}$ sur chacun des couples $d \circ i(L_{1j}), d \circ i(L_{2k})$ dont les instanciations ont au moins un élément en commun et d'appliquer l'opérateur $\cap$ sur les P_{1j}, P_{2k} ayant au moins un élément en commun.
Finalement, il est dual d'appliquer l'opérateur produit $\otimes_{\mathbb{L}}$ sur $\mathcal{L}_1$ et $\mathcal{L}_2$ et d'appliquer l'opérateur $\otimes_{\mathbb{P}}$ sur les partitions $i_{\mathbb{P}}(\mathcal{L}_1)$ et $i_{\mathbb{P}}(\mathcal{L}_2)$. Comme $\otimes_{\mathbb{P}}$ est un opérateur produit, $\otimes_{\mathbb{L}}$ est également un opérateur produit.

□

Exemple 4.23 *Prenons l'exemple suivant (nous rajoutons un élément à l'exemple précédent pour que l'exemple ne soit pas trivial) par le contexte formel suivant. Soit* $\mathcal{O}bj = \{Dame\ de\ Coeur,\ 7\ de\ Coeur,\ 7\ de\ Pique,\ Roi\ de\ Carreau,\ 2\ de\ Carreau\}$, $L = (\{2,\ 7,\ Dame,\ Roi,\ Chiffre,\ Carreau,\ Coeur,\ Rouge,\ Noir,\ Pique\}, \subseteq, \cap)$. *les relations d'instanciation i et de description d sont définies par la matrice suivante.*

	7	*2*	*Dame*	*Roi*	*Chiffre*	*Ca*	*Co*	*Pi*	*Tr*	*Rouge*	*Noir*
Dame de Coeur			×				×			×	
7 de Coeur	×				×		×			×	
2 de Coeur		×			×		×			×	
Roi de Pique				×				×			×
Roi de Trèfle				×					×		×

Soit $\mathcal{L} = \big\{\{Coeur\}, \{Pique\}, \{Trèfle\}\big\}, \mathcal{L}' = \big\{\{Chiffre\},\{Dame\},\{Roi\}\big\}$
$i(\{Coeur\}) = \{Dame\ de\ Coeur,\ 7\ de\ Coeur,\ 2\ de\ Coeur\}$
$d \circ i(\{Coeur\}) = \{Coeur,\ Rouge\}$
$i(\{Pique\}) = \{Roi\ de\ Pique\}$
$d \circ i(\{Pique\}) = \{Noir,\ Roi,\ Pique\}$
$i(\{Trèfle\}) = \{Roi\ de\ Trèfle\}$
$d \circ i(\{Trèfle\}) = \{Noir,\ Roi,\ Trèfle\}$
$i(\{Chiffre\}) = \{7\ de\ Coeur,\ 2\ de\ Coeur\}$
$d \circ i(\{Chiffre\}) = \{Chiffre,\ Coeur,\ Rouge\}$
$i(\{Dame\}) = \{Dame\ de\ Coeur\}$
$d \circ i(\{Dame\}) = \{Dame,\ Rouge,\ Coeur\}$
$i(\{Roi\}) = \{Roi\ de\ Pique,\ Roi\ de\ Trèfle\}$
$d \circ i(\{Roi\}) = \{Roi,\ Noir\}$
$\mathcal{L} \otimes_{\mathbb{L}} \mathcal{L}' = \bigcup\big\{d \circ i(\{Coeur\}) \otimes_L d \circ i(\{Chiffre\}), d \circ i(\{Coeur\}) \otimes_L d \circ i(\{Dame\}), d \circ i(\{Pique\}) \otimes_L d \circ i(\{Roi\}), d \circ i(\{Trèfle\}) \otimes_L d \circ i(\{Roi\})\big\}$
$= \bigcup\big\{\{Coeur,\ Rouge\} \otimes_L \{Chiffre,\ Coeur,\ Rouge\}, \{Coeur,\ Rouge\} \otimes_L \{Dame,\ Coeur,\ Rouge\}, \{Pique,\ Roi,\ Noir\} \otimes_L \{Roi,\ Noir\}, \{Trèfle,\ Roi,\ Noir\} \otimes_L \{Roi,\ Noir\}\big\}$
$= \bigcup\big\{\{Coeur,\ Rouge\}, \{Coeur,\ Rouge\}, \{Roi,\ Noir\}, \{Roi,\ Noir\}\big\}$
$\mathcal{L} \otimes_{\mathbb{L}} \mathcal{L}' = \big\{\{Coeur,\ Rouge\}, \{Roi,\ Noir\}\big\}$

Propriété 4.11
Soit un ensemble d'objets $\mathcal{O}bj$ et un langage de généralisation L pour $\mathcal{O}bj$ défini par $\mathcal{T}_\vee(L, \leq_L, \otimes_L)$ ainsi que la fonction de description $d : \mathcal{P}(\mathcal{O}bj) \to L$ et la fonction d'instanciation $i : L \to \mathcal{P}(\mathcal{O}bj)$. Soit $\mathbb{P}(\mathcal{O}bj)$, l'ensemble des partitions de $\mathcal{O}bj$, $\mathcal{P}_{\mathcal{O}bj}(L)$, l'ensemble des parties de L dont l'instanciation est une partition, $d_{\mathbb{P}} : \mathbb{P}(\mathcal{O}bj) \to \mathcal{P}_{\mathcal{O}bj}(L)$, la fonction de descrition des partitions et $i_{\mathbb{P}} : \mathcal{P}_{\mathcal{O}bj}(L) \to \mathbb{P}(\mathcal{O}bj)$ la fonction d'instanciation d'une partie de L et $\otimes_{\mathbb{L}}$ l'opérateur de généralisation sur $\mathcal{P}_{\mathcal{O}bj}(L)$.
$d_{\mathbb{P}}$ et $i_{\mathbb{P}}$ forment une **correspondance de Galois** pour les demi-treillis $\mathcal{T}_\vee(\mathbb{P}(\mathcal{O}bj), \subseteq_{\mathbb{P}}, \oplus_{\mathbb{P}})$ et $\mathcal{T}_\vee(\mathcal{P}_{\mathcal{O}bj}(L), \supseteq_{\mathbb{P}}, \otimes_{\mathbb{L}})$.

Preuve 4.22

- $\forall \mathcal{P}_1, \mathcal{P}_2 \in \mathbb{P}(\mathcal{O}bj), \mathcal{P}_1 \subseteq_{\mathbb{P}} \mathcal{P}_2 \Rightarrow d_{\mathbb{P}}(\mathcal{P}_1) \subseteq_{\mathbb{P}} d_{\mathbb{P}}(\mathcal{P}_2)$
- $\forall \mathcal{L}_1, \mathcal{L}_2 \in \mathcal{P}_{\mathcal{O}bj}(L), \mathcal{L}_1 \supseteq_{\mathbb{P}} \mathcal{L}_2 \Rightarrow i_{\mathbb{P}}(\mathcal{L}_1) \supseteq_{\mathbb{P}} i_{\mathbb{P}}(\mathcal{L}_2)$
- $\forall \mathcal{P}_1 \in \mathbb{P}(\mathcal{O}bj), \mathcal{P}_1 \subseteq_{\mathbb{P}} i_{\mathbb{P}} \circ d_{\mathbb{P}}(\mathcal{P}_1)$
- $\forall \mathcal{L}_1 \in \mathcal{P}_{\mathcal{O}bj}(L), \mathcal{L}_1 \supseteq_{\mathbb{P}} d_{\mathbb{P}} \circ i_{\mathbb{P}}(\mathcal{L}_1)$

□

4.6.5 Treillis de Galois des partitions

Nous avons présenté l'ensemble des éléments permettant de constituer un treillis de Galois (définition 4.25 page 87) : deux ensembles ayant une structure de demi-treillis et deux fonctions duales formant une correspondance de Galois entre ceux-ci.
De la même manière que pour le treillis de Galois classique, nous pouvons produire le treillis de Galois des partitions.

Définition 4.46 (treillis de Galois des partitions)
Soit un ensemble d'objets $\mathcal{O}bj$ et un langage de généralisation L pour $\mathcal{O}bj$ défini par $\mathcal{T}_\vee(L, \leq_L, \otimes_L)$ ainsi que la fonction de description $d : \mathcal{P}(\mathcal{O}bj) \to L$ et la fonction d'instanciation $i : L \to \mathcal{P}(\mathcal{O}bj)$. Soit $\mathbb{P}(\mathcal{O}bj)$, l'ensemble des partitions de $\mathcal{O}bj$, $\mathcal{P}_{\mathcal{O}bj}(L)$, l'ensemble des parties de L dont l'instanciation est une partition, $d_{\mathbb{P}} : \mathbb{P}(\mathcal{O}bj) \to \mathcal{P}_{\mathcal{O}bj}(L)$, la fonction de descrition des partitions et $i_{\mathbb{P}} : \mathcal{P}_{\mathcal{O}bj}(L) \to \mathbb{P}(\mathcal{O}bj)$ la fonction d'instanciation d'une partie de L et $\otimes_{\mathbb{L}}$ l'opérateur de généralisation sur $\mathcal{P}_{\mathcal{O}bj}(L)$.

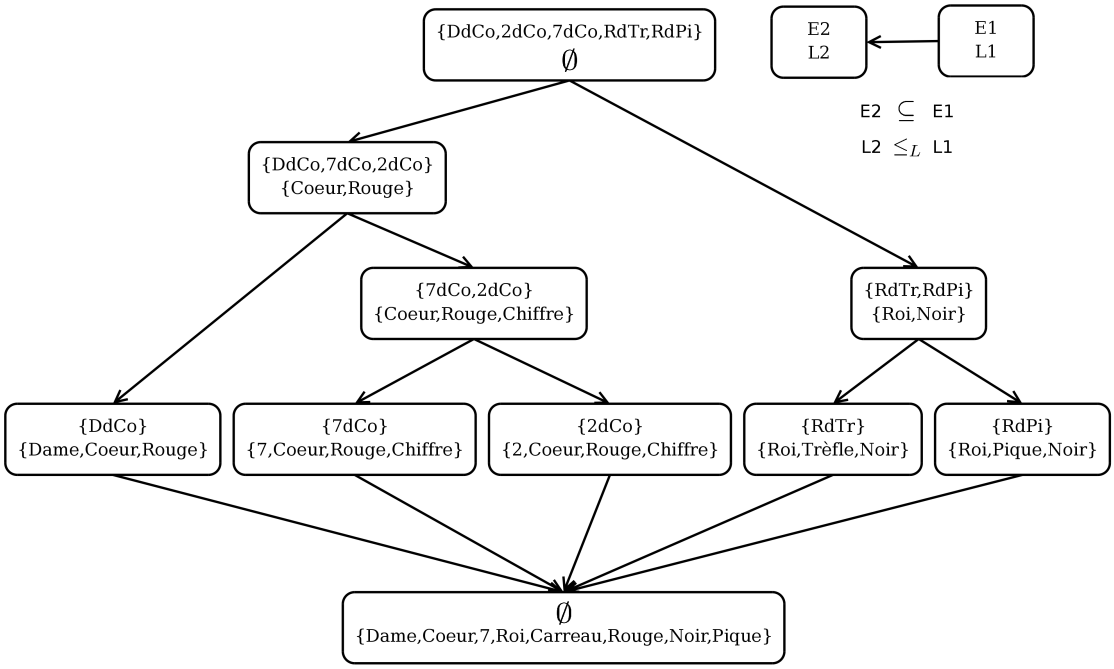
$\mathbb{C}\big(\mathcal{T}_\vee(\mathbb{P},\subseteq_\mathbb{P},\oplus_\mathbb{P}),\mathcal{T}_\vee(\mathcal{P}_{\mathcal{O}bj}(L),\supseteq_\mathbb{L},\otimes_\mathbb{L}),(d_\mathbb{P},i_\mathbb{P})\big)$ est un contexte formel généralisé. $\mathcal{T}\big((F_{i_\mathbb{P}\circ d_\mathbb{P}}(\mathbb{P}(\mathcal{O}bj)),F_{d_\mathbb{P}\circ i_\mathbb{P}}(\mathcal{P}_{\mathcal{O}bj}(L))),\subseteq_\mathbb{P},\otimes_\mathbb{L},\oplus_\mathbb{P}\big)$ est un treillis de Galois nommé **treillis de Galois des partitions**. Par abus d'écriture et pour simplifier les notations, nous noterons $\mathcal{TG}_\mathbb{P}(\mathcal{O}bj,L)$ ce treillis. Cette notation implique implicitement que $\mathcal{O}bj$ est un ensemble, L un langage de généralisation et qu'il existe une fonction de description $d:\mathcal{O}bj\rightarrow L$ et d'instanciation $i:L\rightarrow\mathcal{O}bj$.

Remarque 4.12 *Nous avons le même problème que pour le treillis de Galois classique quand au sens des relations d'ordre partiel $\subseteq_\mathbb{L}$ pour $\mathcal{P}_{\mathcal{O}bj}(L)$. Nous choisissons ici pour des questions de sémantique une réponse similaire. Comme l'opérateur $\otimes_\mathbb{L}$ produit un généralisé, donc un élément plus grand, il faut que la relation d'ordre partiel correspondante soit $\supseteq_\mathbb{L}$ pour conserver le fait qu'il y ait une correspondance de Galois entre $\mathbb{P}(\mathcal{O}bj)$ et $\mathcal{P}_{\mathcal{O}bj}(L)$.*

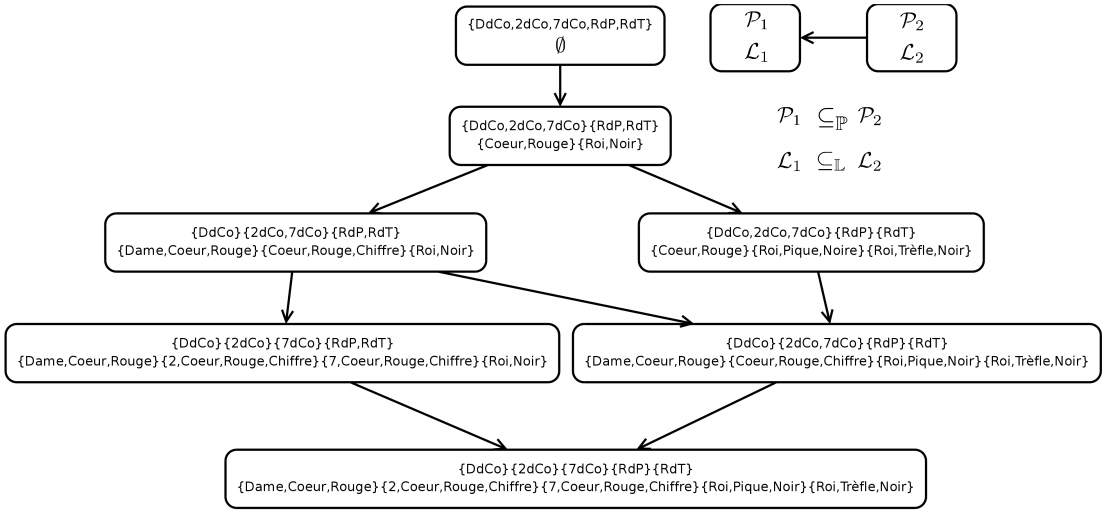
Exemple 4.24 *La figure 4.11 page suivante montre le treillis de Galois et le treillis de Galois des partitions associés au contexte formel de l'exemple 4.23 page 105.*

Dans ce chapitre, nous avons présenté les notions algébriques nécessaires pour définir les *treillis*. Ensuite, nous avons présenté le *treillis de Galois d'une relation binaire* ou *treillis de concepts*. Cette structure est une méthode classique permettant de trouver les propriétés associées à un ensemble d'objets décrits par un langage par attributs, champ de recherche connu sous le nom d'analyse formelle de concepts.
Nous avons ensuite généralisé cette notion par le *treillis de concepts généralisé*, permettant d'obtenir des propriétés similaires, mais pour n'importe quel ensemble d'objets décrits à l'aide d'un langage de généralisation, c'est à dire un langage dont on sait généraliser deux éléments.
Finalement, nous avons proposé le *treillis de Galois des partitions*, une structure algébrique basée sur la même démarche, mais s'appliquant aux partitions d'un ensemble d'objets plutôt qu'aux parties. Nous disposons donc d'un espace de généralisation similaire à ceux utilisés pour l'apprentissage artificiel classique, mais dans le cadre de partition.

Nous avons déjà montré que la généralisation de politiques pour l'apprentissage par renforcement pouvait se voir comme un partitionnement de l'ensemble des états de l'environnement. Ainsi, dans le chapitre suivant, nous allons voir comment le treillis de Galois des partitions nous servira d'espace de généralisation contraint pour l'ensemble des politiques et nous permettra d'utiliser des méthodes de recherche de règles et de généralisation similaires à celles obtenues à l'aide d'un treillis de Galois classique, mais appliquées à des politiques.



(a): Treillis de Galois



(b): Treillis de Galois des partitions

FIG. 4.11 – Treillis de Galois et treillis de Galois des partitions associés au contexte formel de l'exemple 4.23 page 105

5 Généralisation en apprentissage par renforcement biaisé par des langages de description

La problématique générale de nos travaux porte sur la généralisation des politiques pour l'apprentissage par renforcement. Nous avons présenté chapitre 3 différentes façons existantes d'aborder cette question : la généralisation par approximation de fonction, l'apprentissage par renforcement relationnel, ainsi que la généralisation algébrique ; nos travaux présentant des similarités principalement avec ces deux dernières familles de méthode. Nous en retiendrons notamment que la généralisation pour l'apprentissage par renforcement, *du point de vue du modèle*, peut être formalisée comme un partitionnement de l'ensemble des états par une relation d'équivalence basée sur le respect de la dynamique de l'environnement.
De façon similaire, nous allons montrer dans ce chapitre, que *du point de vue de l'agent*, la généralisation peut être formalisée comme un partitionnement de l'ensemble des descriptions des états de l'envronnement :

1. impliqué par une relation d'équivalence entre états, dépendant de la fonction de qualité $Q(e, a)$ apprise par renforcement.
2. en utilisant les langages de description des états et des actions comme biais de langage pour contraindre les partitionnements possibles et généraliser la politique ainsi apprise.

Nous avons précisé dans le chapitre précédent, les propriétés des structures algébriques d'espace de généralisation d'un ensemble d'objets contraint par un langage de généralisation : le *treillis de Galois*, ainsi que l'espace des partitions d'un ensemble d'objets contraint par un langage de généralisation sur cet ensemble : le *treillis de Galois des partitions*.
Nous allons donc montrer dans ce chapitre comment appliquer ces techniques à la généralisation de politiques apprises par renforcement. Pour ce faire, nous utiliserons un *langage de généralisation* pour décrire l'ensemble des états et un *langage d'équivalence* pour décrire l'ensemble des actions d'un Processus de Décision de Markov. Nous introduirons la notion de **Processus de Décision de Markov décrit**, reformulant ainsi le problème d'apprentissage par renforcement tel que nous l'avons vu chapitre 2.

D'autre part, le formalisme de Processus de Décision de Markov classique pour l'apprentissage par renforcement ne fait pas de différence entre les états du modèle de l'environnement et ce que perçoit l'agent de ceux-ci. L'agent a directement accès aux actions et aux états du modèle, pas à une perception de ceux-ci. Les Processus de Décision de Markov Partiellement Observables[38] permettent déjà cette distinction, mais en incluant une probabilité sur la perception des états, et sans modifier le fait que les agent n'ont pas directement accès aux actions du modèle. Nous verrons que le formalisme de PDM décrits que nous présentons permet également de respecter la notion d'agent en interaction et percevant son environnement, mais sans inclure l'aspect stochastique des PDMPO.

Outre les capacités de généralisation de l'apprentissage envisageable à l'aide des PDM décrits, nous verrons également les influences que cette reformulation entraîne sur la recherche et la présentation des solutions dans le cadre de l'apprentissage par renforcement. Nous montrerons la possibilité d'avoir des **solutions plus intelligibles** que par l'utilisation classique des politiques. De plus, nous verrons la relation entre **connaissances préalables** concernant une tâche d'apprentissage par le biais du langage, apprentissage de cette tâche par renforcement et **sélection de concepts pertinents** suite à l'apprentissage.

Nous verrons, partie 5.1, la notion de **Processus de Décision de Markov décrit**. Partie 5.2, nous verrons les conséquences de ce formalisme pour la généralisation des politiques. Nous montrerons la pertinence de leur appliquer les notions de treillis de Galois et de treillis de Galois

[38] PDMPO, Partially Observable Markov Decision Processes (POMDP) en anglais

des partitions. Ensuite, nous verrons partie 5.3 les relations en terme de recherche et d'existence de politique optimale entre les Processus de Décision de Markov et les Processus de Décision de Markov décrits. Partie 5.4, nous proposerons un algorithme et une expérimentation utilisant ces différents concepts. Finalement, nous donnerons des conclusions et des ouvertures partie 5.5.
Dans le chapitre suivant, nous aborderons les conséquences algorithmiques de cette vision en présentant une famille d'algorithmes basée sur celle-ci : les algorithmes de *Q-Concept-Learning*.

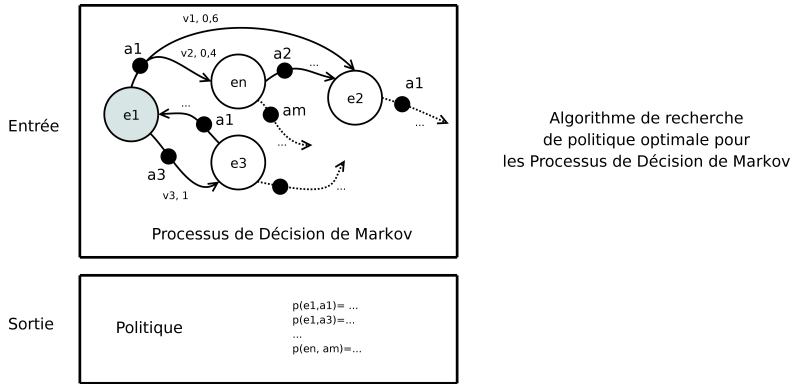
5.1 Processus de Décision de Markov décrit

Le passage du domaine de contrôle optimal à l'apprentissage par renforcement, c'est à dire le passage d'un problème mathématique à un problème d'apprentissage artificiel avec une conception orientée agent, a suivi l'évolution que l'on retrouve dans d'autres domaines (résumée figure 5.1 page ci-contre) ; c'est à dire le passage de la résolution d'un problème dont les données sont parfaitement connues, et dont on recherche les solutions exactes, à une recherche heuristique puis finalement à une recherche incrémentale de solutions approchées, en n'ayant qu'une partie des éléments du problème à résoudre.
Ainsi, les algorithmes de programmation dynamique produisant une politique optimale pour un PDM par résolution d'un système d'équations ont été suivis par une approximation statistique avec les algorithmes d'apprentissage par renforcement. Finalement, et c'est dans cette approche que nous nous situons, les données du problème ne sont connues que par des observations « extérieures » du système.

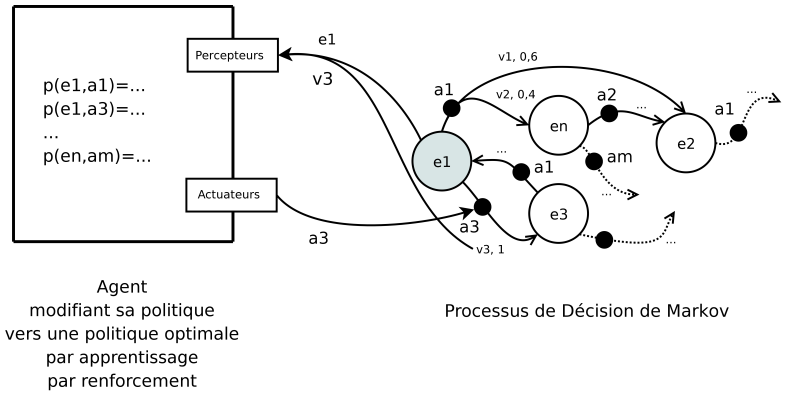
Cette évolution n'a pas été un simple changement de formalisme, elle a également rajouté certaines contraintes relatives au domaine d'application ainsi que des paradigmes propres. Nous allons donc proposer la notion de *Processus de Décision de Markov décrit*. Outre la meilleure compatibilité du ce formalisme à notre propos, une des évolutions majeures de l'apprentissage par renforcement sera nécessairement de s'affranchir des Processus de Décision Markovien pour s'ouvrir vers des classes de problème plus larges où l'hypothèse markovienne est au moins partiellement levée.
Par exemple, dans le cadre d'un apprentissage collaboratif avec d'autres agents, même si la mécanique du système est markoviennne, le changement de comportement des autres agents dû à leur apprentissage en fait un environnement non-markovien. De manière plus générale, la notion d'*agents*, c'est à dire d'entités dont le fonctionnement interne est par nature inaccessible, est en contradiction avec l'hypothèse de Markov. Il est clair que l'apprentissage par renforcement a quand même vocation à s'attaquer à ce genre de sujets.
Des travaux utilisant d'autres formalismes que les PDM sont d'ailleurs déjà en cours ainsi que des travaux concernant des environnements tels que celui que nous venons de citer.
Néanmoins, la levée de l'hypothèse markovienne étant lourde de conséquences sur le plan de la convergence des algorithmes, nous proposons le formalisme de Processus de Décision de Markov décrit qui tout en étant orienté « agent », permet néanmoins de préserver l'hypothèse markovienne.

Présentons dans un premier temps la modification de formalisme des PDM consistant à imposer un langage de description pour les états et les actions, puis nous détaillerons les conséquences du point de vue de l'apprentissage par renforcement, notamment de la généralisation des politiques.

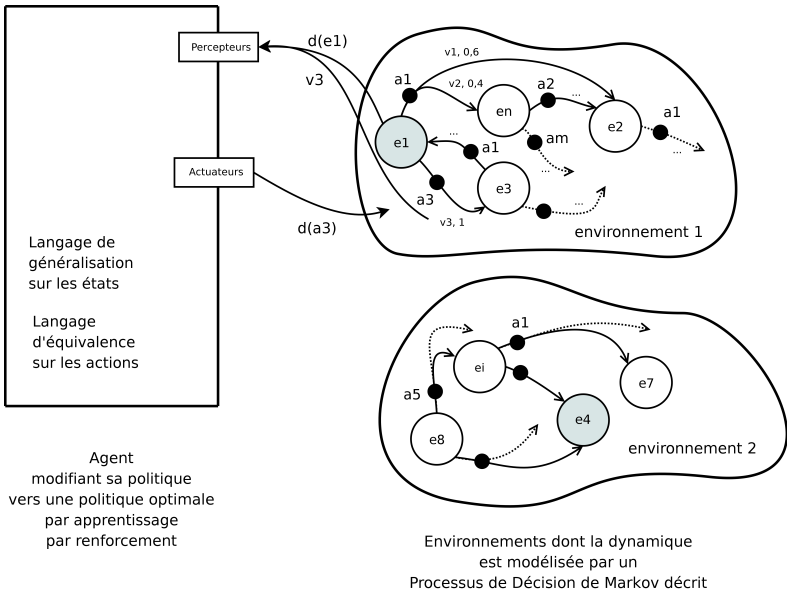
Formalisme Nous avons présenté, définition 2.5 page 23, la notion de Processus de Décision Markovien. Cependant, nous avons précisé, lors de la présentation du problème d'apprentissage par renforcement (voir définition 2.1 page 15, figure 2.1 page 16 et remarque 2.1 page 17), que l'agent n'aurait pas accès directement à la dynamique de l'environnement, mais par l'intermédiaire d'une fonction d'observation. Notre objectif étant l'utilisation des structures de treillis de Galois décrites dans le chapitre 4 pour proposer un espace de généralisations des politiques, nous allons faire le lien en précisant cette fonction d'observation en utilisant les langages de description et de généralisation décrits respectivement définitions 4.26 page 90 et 4.29 page 90. Définissons formellement la notion de PDM décrit.



(a): Représentation classique d'un algorithme de recherche de politique optimale pour un PDM, par exemple l'agorithme de programmation dynamique.



(b): Représentation classique d'un algorithme d'apprentissage par renforcement dont l'environnement est modélisé par un PDM, par exemple l'algorithme de *Q-Learning*. L'agent a néanmoins accès directement aux états et aux actions de l'environnement.



(c): Représentation *agent* d'un problème d'apprentissage par renforcement dont l'environnement est modélisé par un PDM. L'agent n'a accès qu'aux perceptions des états de l'environnement. Il peut réutiliser une politique apprise sur des environnements similaires en regard des langages de description utilisés.

FIG. 5.1 – De la résolution exacte à la conception « agent » de l'apprentissage par renforcement pour un environnement modélisé par Processus de Décision Markovien

Définition 5.1 (Processus de Décision de Markov décrit)
Un Processus de Décision de Markov décrit est un tuple $\langle E, L_E, A, L_A, \psi, T, \mathcal{R} \rangle$ tel que :

- $E = \{e_1, e_2, \ldots, e_n\}$ est un ensemble d'états
- L_E est un langage de description pour E
- $A = \{a_1, a_2, \ldots, a_n\}$ est un ensemble d'actions
- L_A est un langage de description pour A
- $\psi \subseteq E \times A$ est l'ensemble des couples $(état, action)$ admissibles tels que l'action $a \in A(e)$
- $T : \psi \times E \longrightarrow [0, 1]$ est l'ensemble des probabilités de transition de l'état s vers l'état e' en ayant effectué l'action a.
- $\mathcal{R} : \psi \longrightarrow \mathbb{R}$ est la fonction de récompense affectant une valeur numérique à chacune des transitions.

L'agent n'a donc accès qu'aux descriptions des états et des actions. L'ensemble des descriptions des couples $(état, action)$ admissibles est noté $\psi_d \subseteq L_E \times L_A$. Ainsi, du point de vue de la tâche d'apprentissage, l'agent utilisera pour évaluer la qualité de la politique, non pas la fonction $Q : \psi \rightarrow \mathbb{R}$, mais la fonction $Q_d : \psi_d \rightarrow \mathbb{R}$.

Notons que les langages de description L_E et L_A peuvent être des langages de généralisation ou d'équivalence. Dans notre cadre, L_E sera un langage de généralisation et L_A un langage d'équivalence. Le problème ainsi formulé correspond au schéma de la figure (c) page précédente.

Nous allons maintenant voir les possibilités concernant la généralisation des politiques pour l'apprentissage par renforcement qu'offre ce formalisme.

5.2 Généralisation de politiques pour les PDM décrits

L'objectif est ici de montrer comment le *treillis de Galois des partitions* proposé dans le chapitre précédent permet d'établir un espace de recherche biaisé par un langage pour les politiques à la manière d'un apprentissage par induction classique.
Dans la vision originale des PDM, tous les couples $(états, actions)$ sont considérés comme indépendants. Inversement, on peut considérer que ces couples ont quelque chose en commun, ce qui est impliqué par l'utilisation des langages de description pour les actions L_A et les états L_E. En présentant une politique optimale comme une application des états vers leurs actions optimales, il devient possible d'utiliser les méthodes habituelles d'apprentissage artificiel pour décrire ces couples $(état, action\ optimale)$ et ainsi en induire des ressemblances impliquant des règles générales concernant les politiques optimales. Ces règles peuvent alors être utilisées pour des états peu, voire pas, visités.
Le détail de cette utilisation une question complexe puisque cela rentre dans la catégorie de l'action et du raisonnement avec des informations incertaines. Nous donnerons des éléments concernant ce problème au chapitre suivant. Dans ce chapitre, nous allons considérer que l'agent a terminer son apprentissage, c'est à dire que l'apprentissage a convergé vers une politique optimale.

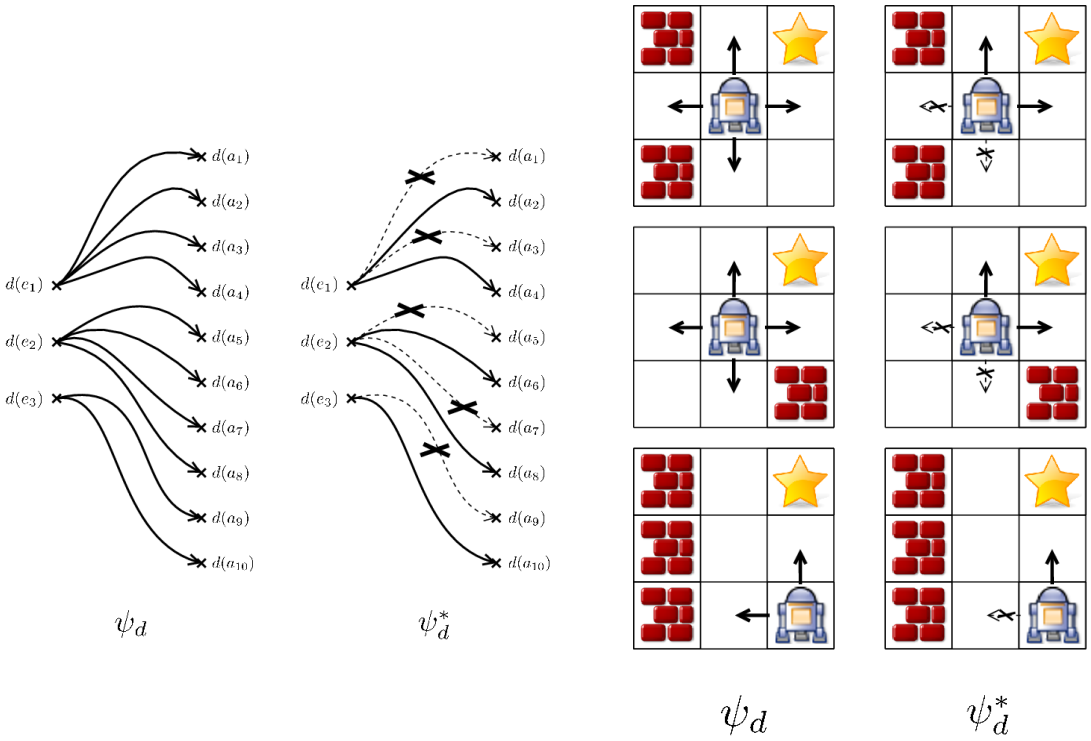
Nous allons d'abord montrer que l'apprentissage par renforcement peut se définir comme la sélection des actions optimales parmi les actions possibles pour chaque état.
Nous établirons alors que le langage d'équivalence, utilisé pour décrire les actions, appliqué aux actions optimales permet de définir une relation d'équivalence, donc un partitionnement des états de l'environnement. Le problème de l'apprentissage par renforcement sera alors un problème de recherche de regroupement des états en fonction de leurs actions optimales.
Ainsi, l'ensemble des partitions des états, en relation avec un langage de généralisation correspondra alors précisément à la notion de *treillis de Galois des partitions*.

5.2.1 Apprentissage par renforcement comme sélection des actions optimales

L'objectif d'un apprentissage par renforcement tel que nous l'avons décrit définition 2.1 page 15 est de trouver une politique optimale pour un agent dans un environnement donné. Nous avons vu que la connaissance de la fonction $Q(e, a)^*$ impliquait la connaissance de l'ensemble des politiques optimales. Ainsi, généralement, la confusion est faite entre la recherche de la fonction $Q(e, a)^*$ et la recherche de politique optimale, la fonction $Q(e, a)^*$ faisant office d'encodage implicite de la

politique.

Rappelons qu'une politique optimale est une politique qui sélectionne les actions de telle façon que seules les actions optimales aient une probabilité non-nulle d'être sélectionnées (propriété 2.4 page 30). Ainsi, si on considère que l'agent au départ de son apprentissage a la connaissance de ψ (le domaine de définition de $Q(e,a)$) alors on peut considérer que l'apprentissage par renforcement consiste à opérer une sélection d'un sous-ensemble $\psi^* \subseteq \psi$ tel que pour tout couple $(e,a) \in \psi^*$, $a \in A^*(e)$. La figure 5.2 propose un exemple graphique de cette notion.



(a): vue fonctionnelle

(b): vue graphique sur l'exemple du monde-grille

FIG. 5.2 – Apprentissage par renforcement vu comme une sélection d'un sous-ensemble $\psi_d^* \subseteq \psi_d$. ψ_d est l'application de l'ensemble des descriptions des états vers l'ensemble des descriptions des actions admissibles pour chacun des états. Une fois l'apprentissage effectué, c'est à dire ψ_d^* trouvée, les associations entre descriptions d'état et descriptions d'action non-optimale sont supprimées.

5.2.2 Apprentissage par renforcement comme partionnement des états de l'environnement

Voyons maintenant comment relier le formalisme de treillis de Galois des partitions décrite dans le chapitre précédent avec l'apprentissage par renforcement.
Comme nous l'avons présenté dans le chapitre précédent, on peut munir A d'un langage d'équivalence[39] $(L_A, \equiv_{L_A})$. Ainsi, deux actions peuvent être considérées comme équivalentes si elles ont la même description. Avec l'égalité ensembliste, nous pouvons étendre cette équivalence à un ensemble d'actions.

Définition 5.2 (Relation d'équivalence entre ensemble d'actions $\equiv_{L_A}$)
Soit $(L_A, \equiv_{L_A})$ un langage d'équivalence. Soit deux ensembles d'actions décrites avec L_A : $A_1, A_2 \in A$.
$A_1 \equiv_{L_A} A_2$ si et seulement si

$$\forall a_1 \in A_1,\ \exists a_2 \in A_2 \text{ tel que } d(a_1) \equiv_{L_A} d(a_2) \text{ et réciproquement}$$
$$\forall a_2 \in A_2,\ \exists a_1 \in A_1 \text{ tel que } d(a_1) \equiv_{L_A} d(a_2)$$

[39] Voir définition 4.35 page 97

Par extension, cette équivalence entre ensemble d'actions appliquée aux actions optimales nous permet également de définir une relation d'équivalence entre états. Deux états étant équivalents si l'ensemble de leurs actions optimales sont équivalentes.

Définition 5.3 ($\equiv_*$, $\mathcal{P}^*$)
Soit un PDM décrit $\langle E, L_E, A, L_A, \psi, P, R\rangle$ tel que $(L_A, \equiv_{L_A})$ soit un langage d'équivalence. Soit deux états $e_1, e_2 \in E$.
$e_1 \equiv_* e_2$ si et seulement si $A^*(e_1) \equiv_{L_A} A^*(e_2)$.
La partition de E engendrée par la relation d'équivalence $\equiv_*$ est notée $\mathcal{P}^*$.

Ainsi, à l'aide de la notion d'équivalence de description entre les actions et la connaissance des actions optimales pour chaque état, nous avons montré que la recherche d'une politique optimale pour un PDM décrit par apprentissage par renforcement pouvait s'exprimer comme la recherche du partitionnement $\mathcal{P}^*$ de l'ensemble des états. La figure 5.3 page suivante donne un exemple de cette notion.

L'espace de recherche des politiques optimales peut donc être considéré comme l'ensemble des partitions de E, $\mathbb{P}(E)$; muni des opérateurs $\otimes_{\mathbb{P}(E)}$ et $\oplus_{\mathbb{P}(E)}$, celui-ci correspond au treillis des partitions de E $\mathcal{T}(\mathbb{P}(E), \subseteq_{\mathbb{P}}, \otimes_{\mathbb{P}}, \oplus_{\mathbb{P}})$.
Insistons sur le fait que par définition, pour un langage d'équivalence sur les actions $(L_A, \equiv_{L_A})$ donné, $\mathcal{P}^*$ est unique et définit ainsi l'ensemble des politiques optimales pour la tâche.

5.3 Solutions d'un PDM décrit

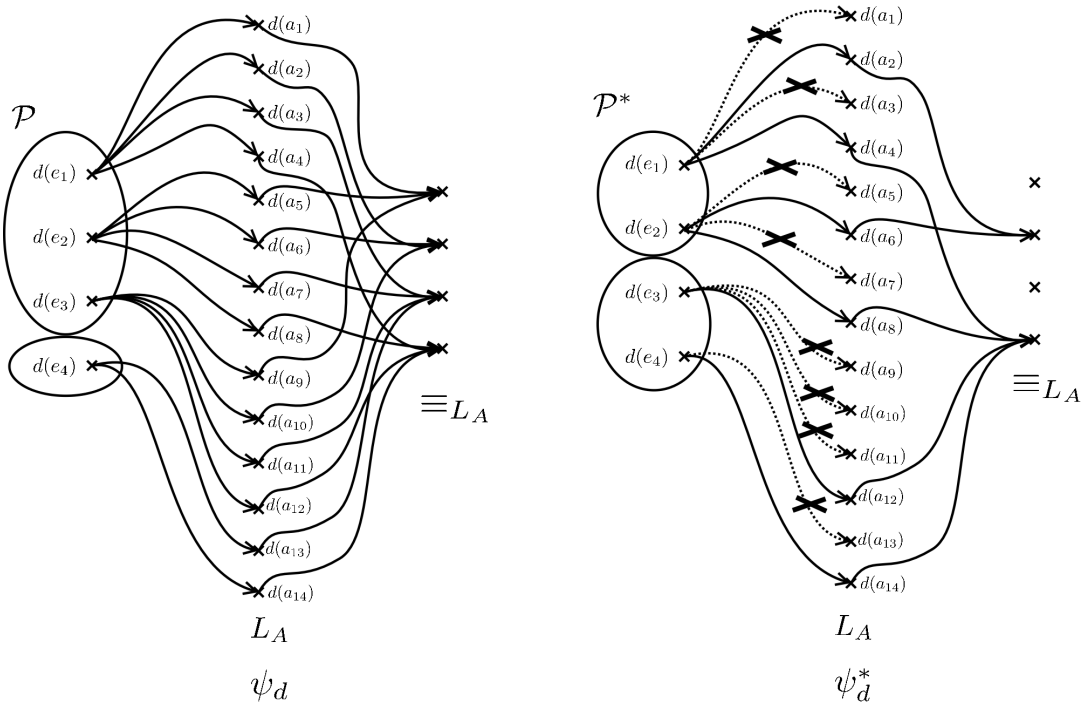
Nous venons de définir les PDM décrits par des langages de description sur les états et les actions. De plus, nous avons proposé de représenter l'ensemble des politiques optimales pour un PDM décrit comme une partition des états de l'environnement. Ceci permet d'envisager le treillis des partitions comme espace de recherche ainsi que le treillis des partitions de Galois comme biais. Voyons les conséquences de ces changements par rapport aux PDM classiques sur l'existence de solutions.
Si l'existence de solutions optimales est assurée pour les PDM classiques, c'est à dire en utilisant l'ensemble des états E et des actions A pour l'apprentissage, elle ne l'est plus pour les PDM décrits par des langages de description, c'est à dire en utilisant E_{L_E} et A_{L_A}.
En effet, du point de vue sémantique, deux états e_1 et e_2 différents et ayant des actions, y compris des actions optimales, différentes peuvent avoir des descriptions équivalentes, $d(e_1) \equiv_{L_E} d(e_2)$. L'apprentissage sur les états e_1 et e_2 sera alors confondus, empêchant de trouver une action optimale pour $d(e_1)$. Ceci est notamment dû au fait qu'il peut ne pas y avoir d'action optimale commune entre e_1 et e_2 malgrè le fait que $d(e_1) \equiv_{L_E} d(e_2)$.
Imaginons dans notre exemple un agent ne sachant pas faire la différence devant une récompense et un gouffre. Sans moyen de différencier ces deux états, il n'y a effectivement pas de descriptions d'action optimale en commun.

Nous avons montré que les solutions des PDM pouvaient être formulées sous la forme de partition de l'ensemble des états E ; ceux-ci étant regroupés entre eux en fonction de relations d'équivalences à partir des propriétés des actions. Nous avons, de plus, vu la notion de partition respectant un langage de description. A partir de ces notions, nous allons déterminer trois cas possibles discriminants quant à la caractérisation de solutions dans un PDM décrit à l'aide de langages de descriptions.

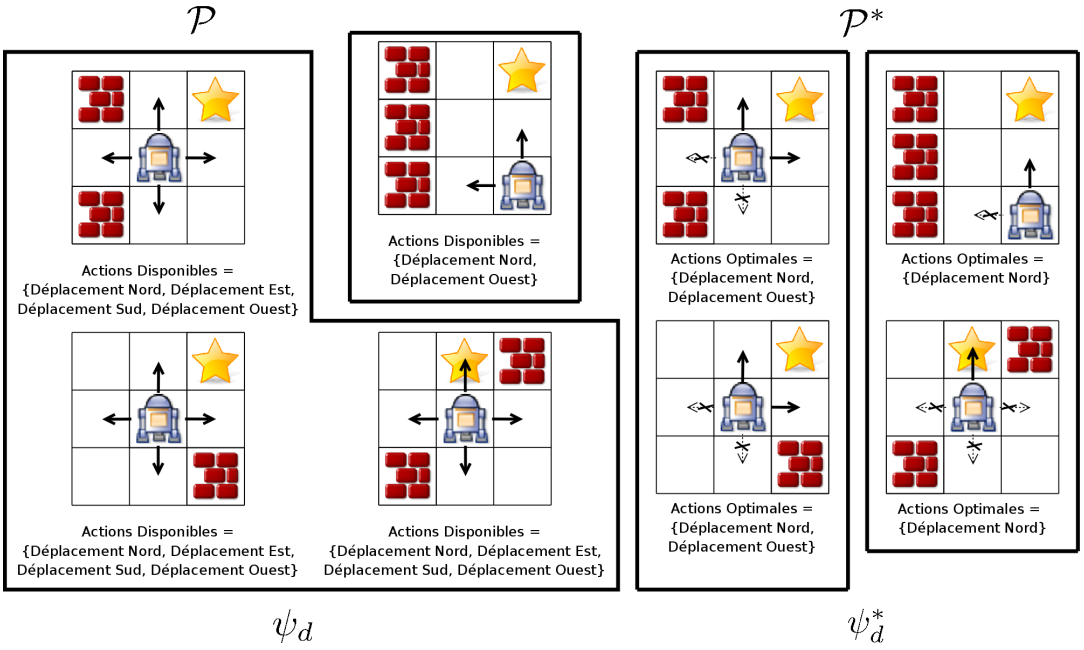
Propriété 5.1
[Caractérisation de l'existence de solutions optimales d'un PDM décrit] Soit un PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$ et $\mathcal{P}^*$, la partition de l'ensemble des états E, recherchée comme généralisation des politiques optimales. Il existe trois cas discriminant l'existence de solutions :

- Cas 1 (Figure 5.4 page 116) : $\mathcal{P}^*$ respecte L_E[40]
- Cas 2 (Figure 5.5 page 117) : $\mathcal{P}^*$ ne respecte pas L_E et il existe une partition $Part_1 \subset \mathcal{P}^*$ qui respecte L_E[41]

[40] repecte est défini définition 4.40 page 102
[41] $\subseteq$ au sens de la définition 4.36 page 98



(a): vue fonctionnelle



(b): vue graphique sur l'exemple du monde-grille

FIG. 5.3 – Relation d'équivalence $\mathcal{P}^*$ entre les états de l'environnement impliquée par le langage d'équivalence sur les actions $(L_A, \equiv_{L_A})$ et par ψ_d^*. Au début de l'apprentissage (ψ_d), toutes les actions sont supposées optimales, les descriptions d'état ayant les mêmes descriptions d'action sont donc regroupées selon la partition $\mathcal{P}$. Après apprentissage (ψ_d^*), les descriptions d'état ayant même descriptions d'action optimale sont regroupés. La parition engendrée est $\mathcal{P}^*$.

- Cas 3 (Figure 5.6 page 118) : Il n'existe pas de partition $Part_1 \subseteq \mathcal{P}^*$ qui respecte L_E

$\mathcal{P}^*$ respecte L_E Premièrement, la partition cible $\mathcal{P}^*$ respecte le langage de description sur les états L_E. Formellement :

$$\forall P \in \mathcal{P}^*, P'' = P$$

Le langage permet d'exprimer l'ensemble des solutions qui aurait été trouvé si on avait cherché directement les politiques optimales en utilisant le PDM $M\langle E, A, \psi, P, R\rangle$, sans langage de description.
Cela ne veut d'ailleurs pas dire que certains états de l'environnement n'ont pas de description équivalente, mais que le langage de description des états permet de différencier tous les états qui doivent l'être dans le cadre de la recherche d'une politique optimale.

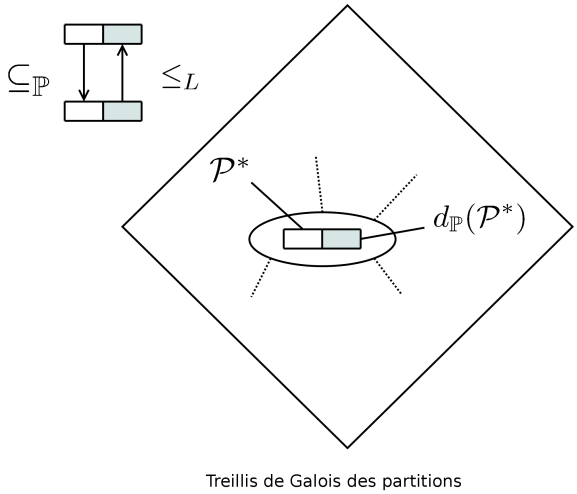


FIG. 5.4 – Cas 1 : $\mathcal{P}^*$ respecte le langage de description L_E

Il existe une partition $Part_1 \subset \mathcal{P}^*$ qui respecte L_E Deuxièmement, la partition cible $\mathcal{P}^*$ ne respecte pas le langage de description des états L_E, mais permet de décrire au moins une partition $Part_1 \subset \mathcal{P}^*$.
La conséquence est que des solutions optimales, du point de vue de la dynamique de l'environnement peuvent être trouvées en utilisant le langage L_E. En effet, le fait que la partition cible, qui regroupe les états de l'environnement équivalents par leurs actions, ne puisse pas être exprimée dans le langage L_E, n'exclue pas qu'une politique optimale puisse être trouvée en utilisant L_E. Cela exclue cependant le fait de pouvoir exprimer de la façon la plus générale possible l'ensemble des politiques optimales.

Il n'existe pas de partition $Part_1 \subseteq \mathcal{P}^*$ qui respecte L_E Troisièmement, il n'existe pas de partition $Part_1 \subseteq \mathcal{P}^*$ qui respecte L_E ce qui a comme corollaire qu'aucune partition représentant au moins une politique optimale ne respecte le langage de description des états L_E. Autrement dit, toutes les partitions exprimables par L_E sont trop générales (au sens d'inclusion des partitions) pour exprimer une politique optimale.
Dans ce cas, le comportement des agents apprenants peut être très varié. En fonction des algorithmes d'apprentissage, il peut y avoir convergence vers une politique sous-optimale, mais néanmoins intéressante (c'est à dire meilleur qu'une politique aléatoire et dont l'erreur est bornée). Il peut aussi y avoir un comportement complètement divergent, entraînant des politiques tout à fait erratiques du point de vue de la distance à la politique optimale en terme de récompense.
Ce problème est également à rapprocher fortement des travaux sur les *Processus de Décision Markoviens Partiellement Observables*[42], notamment la thèse de McCallum ([McCallum, 1996b])

[42]Partially Observable Markov Decision Processes, POMDP en anglais

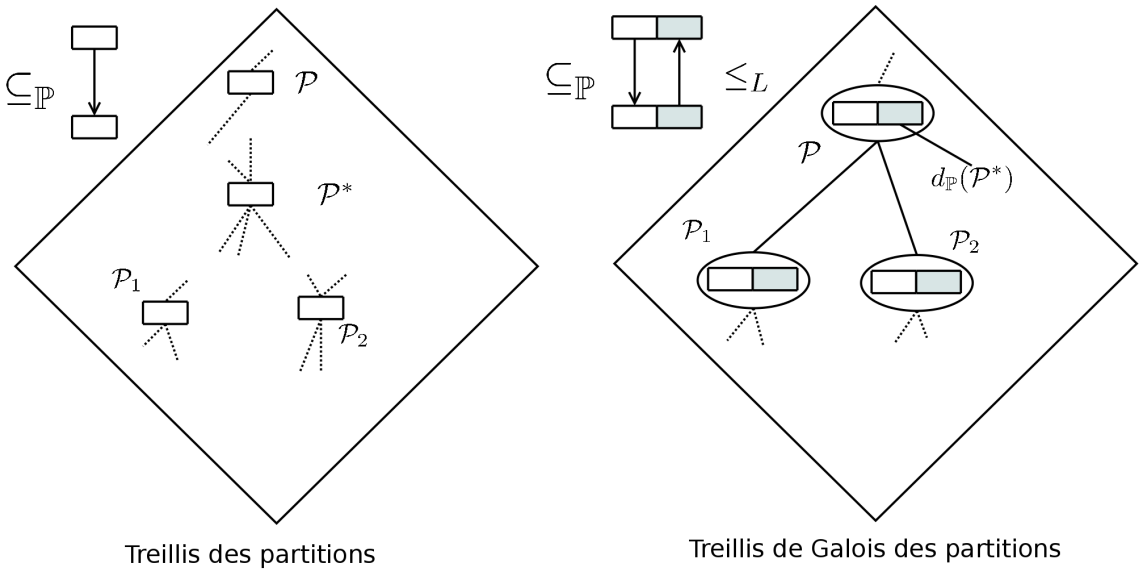


FIG. 5.5 – Cas 2 : Il existe une partition $\mathcal{P}_1 \subset \mathcal{P}^*$ qui respecte L_E

utilisant une mémoire pour tenter de lever l'ambiguïté entre des états confondus du point de vue de la perception que peut en avoir l'agent, cette ambiguïté ne permettant pas de trouver une politique optimale.

La cause de ce problème est l'impossibilité pour l'agent de différencier des états de l'environnement qui devraient l'être puisque la politique optimale est différente pour chacun d'eux. Cette confusion peut avoir deux causes :
Premièrement et classiquement, l'agent n'a pas les moyens au niveau de ses percepteurs, de percevoir différemment les états. Dans ce cas, nous sommes dans le problème sus-cité des POMDP que nous ne traiterons pas ici.
Deuxièmement, c'est le langage de description utilisé par l'agent qui ne permet pas de différencier les états de l'environnement qui devraient l'être. Si tel est le cas et que l'agent possède un mécanisme pour préciser son langage, le problème peut potentiellement être levé.

5.3.1 Cas particulier

Pour la suite de nos travaux, nous allons exclure le cas où l'agent n'a pas les moyens de percevoir différemment des états qui devraient l'être par rapport à une politique optimale.
Si on n'a aucune information sur la partition cible, ce qui est généralement le cas pour imposer cette condition, on peut proposer que tous les états de l'environnement d'un agent soient différentiables par les percepteurs de l'agent.

Propriété 5.2
Soit un PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$ tel que pour tout e_1, $e_2 \in E$, $e_1 \neq e_2 \Rightarrow d_{L_E}(e_1) \neq d_{L_E}(e_1)$.
Il existe alors au moins une partition de E_{L_E} respectant L_E et telle que E_{L_E} soit l'expression d'une politique optimale.

Preuve 5.1
La partition de cardinal maximal de E_{L_E}, c'est à dire la partition de E_{L_E} ne contenant que des singletons respecte le langage L_E car si

$$\forall e_1, e_2 \in E, e_1 \neq e_2 \Rightarrow d_{L_E}(e_1) \neq d_{L_E}(e_1)$$

alors

$$\forall e \in E, d_{L_E}(e)'' = d_{L_E}(e)$$

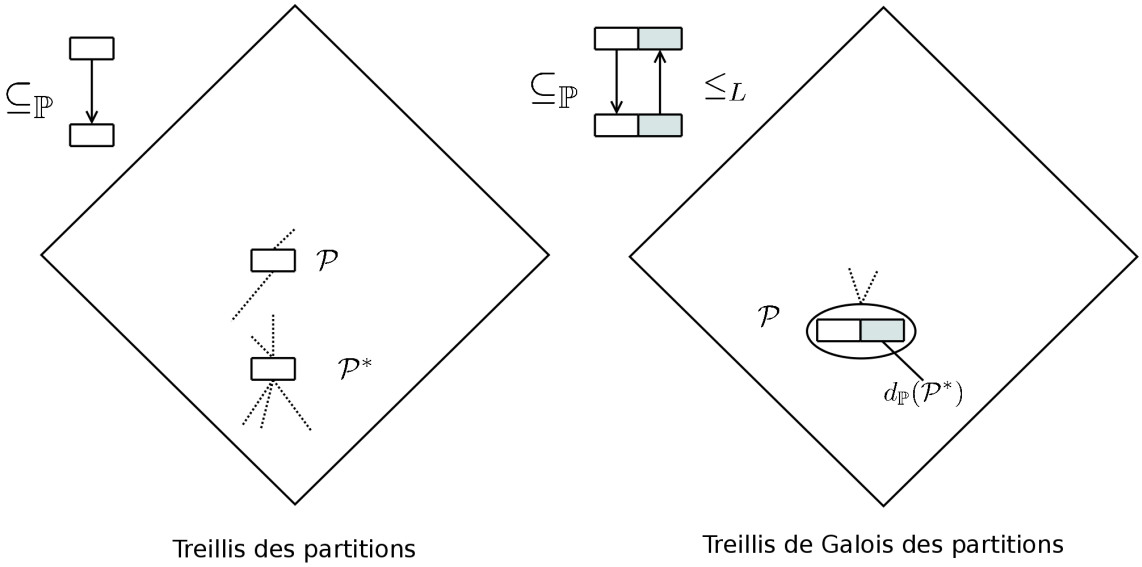


FIG. 5.6 – Il n'existe pas de partition $\mathcal{P}_1 \subseteq \mathcal{P}^*$ qui respecte L_E

Ceci implique qu'il existe au moins la partition de E de cardinal maximum pour exprimer une solution optimale. Remarquons que si par ailleurs, il n'existe qu'un moyen de généraliser la description de deux états, c'est à dire que :

$$\forall e_1, e_2 \in E, d_{L_E}(e_1) \otimes d_{L_E}(e_2) = \bigotimes_{E_{L_E}}$$

Alors l'agent est dans le même cadre qu'un PDM classique complètement observable. On est donc dans le cas classique où l'on peut appliquer les algorithmes habituels d'apprentissage par renforcement.

□

5.4 Algorithme en généralisation

Les notions que nous venons de développer sont plus destinées à poser un cadre formel original pour la généralisation des politiques pour l'apprentissage par renforcement qu'à être utilisées pour une application directe. Néanmoins, nous allons proposer un algorithme montrant une utilisation possible de ce cadre et une application de celui-ci sur un exemple académique.

Nous avons montré les rapports possibles entre une partition des états de l'environnement et un langage de généralisation pour les états de l'environnement. En particulier, après apprentissage, et en utilisant un langage d'équivalence L_A pour les descriptions des actions, nous avons montré les rapports possibles entre $\mathcal{P}^*$ et le treillis de Galois des partitions $\mathcal{TG}_{\mathbb{P}}(E, L_E)$.
L'algorithme que nous présentons se place dans le cadre du premier et du deuxième cas de ces rapports, c'est à dire qu'il existe au moins une partition $\mathcal{P}art \in \mathbb{P}$ telle que $\mathcal{P}art \subseteq \mathcal{P}^*$. Pour garantir ceci, nous posons le fait que tous les états de l'environnement aient une description différente (voir partie 5.3.1).
Par contre, nous envisageons le fait que la partition $\mathcal{P}^*$ puisse ne pas être une extension d'un concept du treillis de Galois des partitions $\mathcal{TG}_{\mathbb{P}}(E, L_E)$, c'est à dire $\mathcal{P}^* \subset i_{\mathbb{P}} \circ d_{\mathbb{P}}(\mathcal{P}^*)$.

Définition 5.4 ($\mathbb{P}_{GEN}(\mathcal{P}_1, L)$)
Soit un ensemble $\mathcal{O}bj$, un langage de généralisation L pour $\mathcal{O}bj$ et $\mathcal{P}_1 \in \mathbb{P}(\mathcal{O}bj)$. On définit $\mathbb{P}_{GEN}(\mathcal{P}_1, L)$, l'ensemble des partitions de $\mathcal{O}bj$ qui sont des généralisations conformes à $\mathcal{P}_1$ et expressibles dans le langage L par :

$$\mathbb{P}_{GEN}(\mathcal{P}_1, L) = \{\mathcal{P} \in \mathbb{P}(\mathcal{O}bj) \text{ telle que } \mathcal{P} \text{ respecte } L \text{ et } \mathcal{P} \subseteq \mathcal{P}_1\}$$

L'ensemble $\mathbb{P}_{GEN}(\mathcal{P}_1, L) \subseteq \mathbb{P}(\mathcal{O}bj)$ est donc l'ensemble des partitions qui sont l'extension d'un concept du treillis de Galois des partitions $\mathcal{TG}_{\mathbb{P}}(\mathcal{O}bj, L)$ et moins générales que $\mathcal{P}_1$.

En revenant dans le cadre de l'apprentissage par renforcement, l'ensemble $\mathbb{P}_{GEN}(\mathcal{P}^*, L_E)$ est donc l'ensemble des regroupements d'états exprimables avec le langage L_E et qui ont la même politique optimale conformément à un apprentissage par renforcement. Listons des propriétés de $\mathbb{P}_{GEN}(\mathcal{P}^*, L_E)$ dont nous nous servirons :

- $\mathbb{P}_{GEN}(\mathcal{P}^*, L_E) \neq \emptyset$. Ceci est dû au cadre que nous avons imposé de descriptions différentes pour chacun des états de l'environnement. En effet, la partition composée de singletons appartient nécessairement à $\mathbb{P}_{GEN}(\mathcal{P}^*, L_E)$.
- si $\mathcal{P}^*$ est l'extension d'un concept du treillis de Galois des partitions $\mathcal{TG}_{\mathbb{P}}(E, L_E)$ alors $\mathcal{P}^*$ est le majorant de $\mathbb{P}_{GEN}(\mathcal{P}^*, L_E)$. Ceci est impliqué par la nature d'un treillis.

Démarche algorithmique L'objectif de la généralisation des politiques pour l'apprentissage par renforcement est évidemment de construire l'intention la plus générale possible en accord avec l'apprentissage.
L'algorithme 5.1 que nous présentons produira donc un des éléments maximaux de $\mathbb{P}_{GEN}(\mathcal{P}^*, L_E)$, idéalement $\mathcal{P}^*$.
La démarche de notre algorithme, démarche en généralisation, est de partir de la partition composée de singletons et d'une relation d'équivalence entre états (basée sur l'équivalence des actions optimales) et de regrouper autant que possible les états équivalents en s'assurant que les parties produites respectent L_E.

Comme il n'est pas sûr que la partition impliquée par la relation d'équivalence respecte L_E, il faut que la méthode engendre un des éléments maximaux de $\mathbb{P}_{GEN}(\mathcal{P}^*, L_E)$. Pour ce faire, nous privilégions de regrouper les états ayant même valeur pour leur fonction de qualité pour leurs actions optimales. Dans le cas général, il est complètement fortuit que deux couples *(états,actions)* aient la même valeur de qualité $Q(e, a)$. Cependant la fonction $Q(e, a)$ intègre entre autre une distance au but dans certains contextes, tels que les problèmes avec un objectif à atteindre, comme celui de notre exemple. C'est pourquoi, à choisir, nous préférons privilégier de regrouper ces états. On pourrait cependant envisager d'autres priorités de regroupement, en considérant de minimiser la taille de la partition par exemple. Ces pistes n'ont pas été explorées.

Notons que cet algorithme peut être appliqué n'importe quand au cours de l'apprentissage, si on considère comme nous l'avons présenté partie 5.3 l'apprentissage par renforcement comme un partitionnement de l'ensemble des états, le résultat de l'algorithme est alors l'hypothèse courante concernant la partition des états de l'environnement et par conséquent l'hypothèse courante concernant la généralisation des politiques.

Éléments de preuve pour l'algorithme

Tous les états sont considérés successivement en utilisant une structure FIFO[43] (variable `Etats` repère 1). Dans le pire des cas, chaque cas est ajouté comme singleton à la partition résultat $\mathcal{P}$ (repère 6). Ceci assure que le résultat final soit une couverture de E. Les repères 2 et 5 assurent que les éléments soient ajoutés dans seulement un sous-ensemble, assurant que le résultat est une partition de E.
Repère 3, l'algorithme construit un sous-ensemble de E à partir d'un état, selon la relation d'équivalence $\equiv_*$ et en utilisant l'opérateur de fermeture $i \circ d$. Ceci assure que le sous-ensemble produit respecte le langage L_E. Ensuite, repère 4, nous vérifions que la fermeture n'a ajouté que des éléments équivalents. Ceci assure que la partition résultat $\mathcal{P}$ est inférieure (au sens de $\subseteq_{\mathbb{P}}$) à la partition $\mathcal{P}^*$.

Expérimentation

Nous avons réalisé une expérimentation de notre algorithme sur un problème académique pour illustrer notre méthode et valider l'approche théorique par une implémentation. L'exemple consiste en un monde-grille de taille 3×3 contenant un agent, une récompense et un mur. Toutes les

[43] First In - First Out, premier arrivé – premier sorti en français

Données : • PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$
• Un langage de généralisation $(L_E, \leq_{L_E}, \otimes_{L_E})$ pour E
• L'estimation courante de $Q(e, a)$

Résultat : • Une partition $\mathcal{P}$, basée sur $\equiv_*$, respectant L_E, telle que $\mathcal{P}$ soit la plus générale possible, en regroupant prioritairement les états dont la valeur de l'action optimale est identique.

début
  $\mathcal{P} \leftarrow \emptyset$;
1 $Etats \leftarrow$ trier E selon les valeurs décroissantes de $Q(e, a^*)$;
  tant que $Etats \neq \emptyset$ **faire**
    $e \leftarrow$ retirer($Etats$);
    $Equivalents \leftarrow e_1 \in Etats \mid e_1 \equiv_* s$; $NouveauSousEnsemble \leftarrow \{e\}$;
    tant que $Equivalents \neq \emptyset$ **faire**
2 $e_2 \leftarrow$ retirer($Equivalents$);
3 $ajout \leftarrow i \circ d(NouveauSousEnsemble \cup \{e_2\}) - (NouveauSousEnsemble \cup \{e_2\})$;
4 **si** $\forall e_3 \in ajout$, $e_3 \equiv_* e$ **alors**
$NouveauSousEnsemble \leftarrow NouveauSousEnsemble \cup \{e_2\} \cup ajout$;
5 retirerDe($Equivalents, ajout \cup \{e_2\}$);
retirerDe($Etats, ajout \cup \{e_2\}$);
fin
fin
6 $\mathcal{P} \leftarrow \mathcal{P} \cup \{NouveauSousEnsemble\}$;
fin
retourner $\mathcal{P}$;
fin

Algorithme 5.1: Partitionnement de l'ensemble E selon $\equiv_*$ en respectant le langage de généralisation $(L_E, \leq_{L_E}, \otimes_{L_E})$

autres cases sont vides. Les murs peuvent être déplacés, s'il n'y en a pas d'autres derrière, avec l'action « pousser ». La tâche à apprendre est de se déplacer aussi vite que possible jusqu'à la case « récompense ». Chaque épisode dure jusqu'à ce que l'agent atteigne la case « récompense » ou la détruise en poussant un mur dessus. Les caractéristiques pour contruire le PDM correspondant sont les suivantes :

- L'agent reçoit une récompense de +1 s'il atteint la case récompense. Toutes les autres actions reçoivent -0,1. Il y a 504 états différents ($9 \times 8 \times 7$, en fonction de la position de l'agent, du mur et de la récompense. Les états finaux ne sont pas comptés). Chacun des états comporte 4 actions parmi {« se déplacer Nord », « se déplacer Est », « se déplacer Sud », « se déplacer Ouest », « pousser Nord », « pousser Est », « pousser Sud », « pousser Ouest »}.
- La fonction $|\psi = E \times A| = 504 \times 4 = 2016$.
- Les conséquences des actions sont déterministes et l'environnement est stationnaire.
- Le langage $(L_E, \leq_{L_E}, \otimes_{L_E})$ utilisé pour décrire les états est un language par attribut-valeurs. La description d'un état est composée d'un attribut par case que l'agent peut percevoir (voir figure 5.7 page suivante).
- L'ordre partiel $\leq_{L_E}$ est défini par :
  $d(e_1) \leq_{L_E} d(e_2) \iff$ tous les attributs de $d(e_1)$ sont inférieurs ou égaux aux attributs correpondant dans $d(e_2)$ selon le demi-treillis fourni figure 5.7.
- Le produit $\otimes_{L_E}$ de deux descriptions $d(e_1)$ et $d(e_2)$ est construit par la généralisation de chaque attribut de $d(e_1)$ et de $d(e_2)$.
- La sélection des actions et l'apprentissage a été faite par un *Q-Learning* classique (algorithme 2.5 page 38).

Notre algorithme permet l'extraction d'éléments de description pertinents relativement à la tâche considérée. La figure 5.8 montre un extrait de la description d'éléments de la partition obtenue après convergence de l'apprentissage. Quantitativement, il reste 92 descriptions pour décrire la

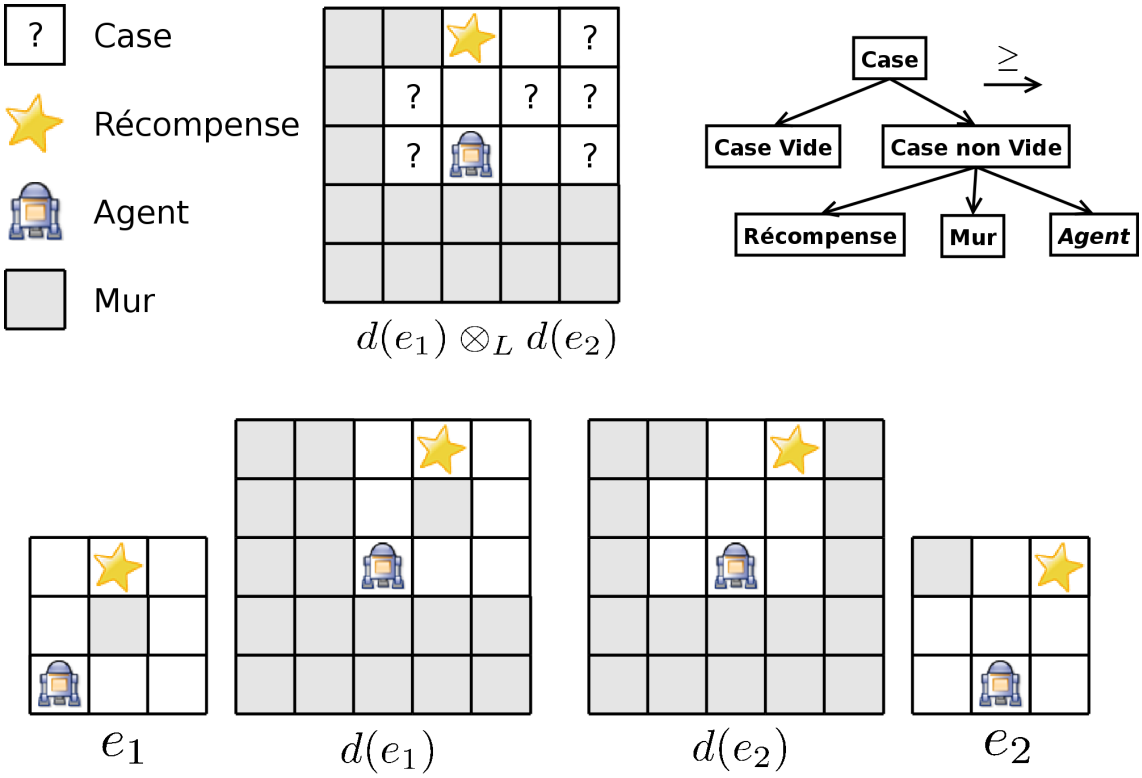


FIG. 5.7 – Un exemple graphique de deux états de l'environnement, leur description et le produit de leurs descriptions

tâche au lieu de 504 initialement.
On retrouve des éléments attendus comme « être à une case au sud de la récompense » implique l'action optimale « se déplacer au Nord », peu importe les autres cases. On retrouve aussi des motifs moins intuitifs comme l'exemple 4 de la figure 5.8.
Notons que certains ensembles d'états ayant les mêmes actions optimales sont néanmoins séparés (exemple 1 et 2 de la figure 5.8). Ceci est dû au fait qu'il n'existe pas d'élément du langage L_E permettant de regrouper ces ensembles d'états sans que la description engendrée en inclue d'autres n'ayant pas le même ensemble d'actions optimales.

5.5 Conclusions

Application du treillis de Galois des partitions Nous avons vu chapitre 4, qu'un langage de généralisation sur un ensemble d'objets $\mathcal{O}bj$ pouvait définir à la fois un biais de langage et un espace de généralisation pour cet ensemble avec l'utilisation du *treillis de Galois des partitions*. Ainsi, nous avons directement appliquer ces méthodes sur l'ensemble E, des états de l'environnement, avec un langage de généralisation L_E sur celui-ci.

Apprentissage de comportement et langage de description Le fait d'envisager la généralisation d'un apprentissage par renforcement amène à considérer deux éléments. Premièrement, lors d'une même tâche, les états et les actions ne sont pas complètement indépendants les uns des autres. Il y a donc des rapprochements possibles entre les états, voire les actions, permettant éventuellement d'apprendre plus vite, ou de réutiliser un comportement appris pour des états inconnus. Deuxièmement, la tâche en elle même peut faire partie d'un domaine sur lequel il existe déjà des solutions, des concepts pertinents,... c'est à dire des connaissances *a priori* utilisables. Il faut donc des mécanismes pour les intégrer. De tels mécanismes ont déjà été envisagés concernant la généralisation pour l'apprentissage par renforcement.
Dans le cadre de l'apprentissage par renforcement relationnel, l'environnement y est décrit par

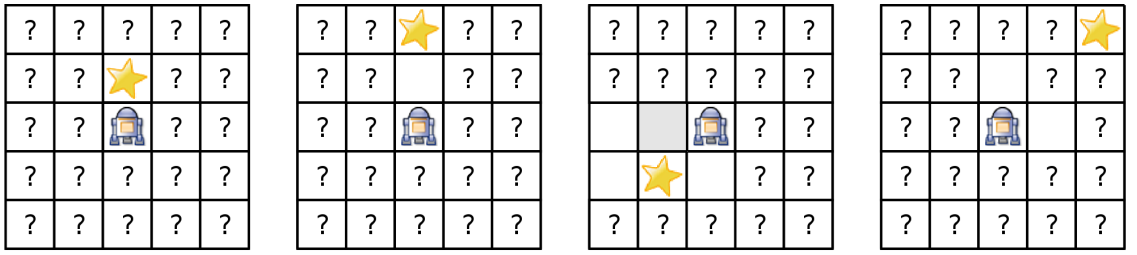


FIG. 5.8 – Descriptions et actions optimales de quatre sous-ensembles d'états parmi les 92 produits en partitionnant l'ensemble des 504 états de l'environnement après convergence de l'apprentissage par renforcement

un mécanisme qui pourrait être traduit dans le formalisme de PDM décrit. Le mécanisme de généralisation s'appuie principalement sur le langage de description constitué de termes fondés en logique comme indiqué partie 3.4. De plus, l'utilisation d'une base de connaissance prévoit d'enrichir les descriptions des états en y rajoutant des littéraux. La connaissance *a priori* est donc apportée par le biais du langage de description.
Pour ce qui concerne la généralisation par approximation de fonction (partie 3.2), le choix des motifs paramétrant les fonctions peut modifier radicalement la convergence des algorithmes d'apprentissage. La généralisation est donc là aussi biaisée par le langage de description utilisé comme connaissance *a priori*.

Solution intelligible Un des critères important à prendre en considération pour la résolution de problème dans le cadre de l'apprentissage par machine est la possibilité de produire des résultats intelligibles pour un être humain. Dans le cadre de l'apprentissage par renforcement, la présentation d'une politique sous la forme basique de la fonction de probabilité $\pi : \psi \rightarrow [0, 1]$ est de ce fait, très peu utilisée.
Le critère d'intelligibilité du résultat divise les algorithmes en deux catégories : d'une part, les algorithmes par approximation de fonction qui représente la fonction $Q(e, a)$ ou la politique $\pi(e, a)$ sous forme fonctionnelle, et d'autre part, ceux conservant une représentation symbolique des états de l'environnement et des actions, comme l'apprentissage par renforcement relationnel. Par exemple, le résultat produit par un réseau de neurones est difficilement explicable seulement avec la connaissance des ses paramètres, on ne peut que constater ses performances.
Nos travaux se placent dans le second cas. Ainsi, une politique exprimée sous la forme d'une association entre un groupe d'états et un groupe d'actions chacun désigné par un élément de langage est plus intelligible qu'une liste de probabilités dans la mesure où le nombre d'associations n'est pas trop élevé.
Le langage de description employé pour décrire les états de l'environnement peut avoir, même si ce n'est pas obligatoire, un rapport sémantique avec le langage naturel. On facilite ainsi la compréhension du résultat.
Nos méthodes permettent de présenter des solutions optimales sous la forme de règles les plus générales possibles, le nombre dépendant de la complexité de la tâche.

Réutilisation et modification du résultat L'utilisation possible d'un langage ayant un rapport avec un langage naturel pour la description des actions et des états de l'environnement facilite la réutilisation d'une politique exprimée sous cette forme pour d'autres utilisations. De plus, il est envisageable de faire des modifications directes sur la politique obtenue.

Généralisation L'apprentissage par renforcement permet à l'agent d'apprendre un comportement optimal pour une tâche donnée ; cependant, un comportement appris sur une tâche contient nécessairement des éléments d'apprentissage sur des tâches connexes. La question corollaire est

donc : « comment et sous quelles formes utiliser ce comportement appris pour des tâches présentant des similarités ? »
Comme on le voit, la question de l'apprentissage par renforcement peut se voir comme deux apprentissages distincts mais pouvant se servir l'un de l'autre. D'une part, on a un apprentissage d'un comportement sur une tâche donnée, d'autre part, un problème d'apprentissage machine extrayant des règles, des regroupements, des classes,..., d'un ensemble donné. La méthode présentée propose une approche traitant les deux questions simultanément.

Enrichissement du langage de description des états On remarquera un corollaire important du deuxième cas pour l'existence de solution en fonction du langage L_E : s'il existe une partition $Part_1 \subset \mathcal{P}^*$ qui respecte L_E.
Ce cas implique que le langage est suffisamment précis pour exprimer l'ensemble des politiques optimales, mais que des états qui devraient être regroupés du point de vue de la relation d'équivalence entre états ne peuvent l'être sans en inclure d'autre non équivalents.
Formellement, on a :

$$\exists d_{L_E}(e_1), d_{L_E}(e_2) \in E_{L_E} \text{ tels que } d_{L_E}(e_1) =_{L_E} d_{L_E}(e_2)$$

et

$$d_{L_E}(e_1) \otimes d_{L_E}(e_2))'' \neq d_{L_E}(e_1) \otimes d_{L_E}(e_2)$$

Ceci implique qu'un concept qui devrait pouvoir être exprimable pour décrire correctement la tâche impliquée par le PDM ne l'est pas avec le langage L_E. Cette propriété peut servir à enrichir le langage de description L_E.
Les mécanismes d'enrichissement du langage n'ont pas été étudiés de manière approfondie dans le cadre de nos travaux. On peut néanmoins avancer deux pistes de solutions.
Premièrement, malgré le fait que l'apprentissage soit non supervisé, l'agent a des possibilités d'interaction avec un humain, il peut exhiber les états en question et demander l'introduction par l'utilisateur d'un nouvel élément du langage regroupant spécifiquement ceux-ci.
Deuxièmement, on peut avoir un mécanisme de génération automatique de descriptions.
Il se pose dans les deux cas, le problème mathématique, potentiellement complexe, de l'insertion du nouvel élément dans le langage existant tout en maintenant sa cohérence, notamment ses propriétés algébriques, c'est à dire de l'opérateur produit, d'équivalence et de la relation d'ordre partiel.

Même s'il y a nécessité d'une investigation plus profonde, la découverte d'éléments du langage importants, c'est à dire l'extraction de motifs discriminants du langage dans le cadre de l'apprentissage par renforcement est une des avancées importantes de nos méthodes.

Nous allons dans le chapitre suivant voir une autre utilisation des treillis de Galois pour l'apprentissage par renforcement. Nous avons dans ce chapitre étudié une utilisation *a posteriori* après apprentissage, nous allons présenter dans le chapitre suivant une démarche dynamique, intégrant les mécanismes de treillis de Galois pour représenter la fonction $Q(e, a)$ au cours de l'apprentissage.

6 Q-Concept Learning : Utilisation de la structuration par treillis de Galois pour l'apprentissage par renforcement

Nous avons vu dans le chapitre précédent comment le treillis des partitions de Galois pouvait être une structure permettant de généraliser une politique en utilisant comme biais un langage de description sur les états de l'environnement. Le problème principal de cette méthode vient du fait qu'il faut attendre la fin de la convergence de l'algorithme d'apprentissage par renforcement pour pouvoir l'appliquer. Ceci est en contradiction avec le cadre général de l'apprentissage par renforcement qui suppose un apprentissage « en ligne ».

Dans ce chapitre, nous allons donc proposer un certain nombre de notions dont l'objectif est d'appliquer les méthodes issues des treillis de Galois à l'apprentissage par renforcement tout en conservant le caractère « en ligne » de l'apprentissage. Nous regroupons ces notions sous le terme **Q-Concept Learning**.

Nous présenterons dans un premier temps, partie 6.1, les principes qui sous-tendent nos méthodes, nous redéfinirons, partie 6.1.1, la fonction d'apprentissage pour l'apprentissage par renforcement pour que celle-ci soit représentée à l'aide d'une structure inspirée d'un treillis de Galois formé par l'ensemble des états de l'environnement et un langage de généralisation sur ceux-ci. C'est la fonction $Q(concept, action)$. Nous analyserons partie 6.1.2, des éléments de convergence pour la fonction ainsi revisitée. Nous proposerons alors, partie 6.2, la notion de *concepts pertinents* comme objectif de l'apprentissage. Nous donnerons également une méthode heuristique pour la recherche de ces concepts pertinents avec la notion de *concepts améliorant une politique*. Ensuite, nous montrerons, partie 6.3 comment utiliser la fonction $Q(c, a)$ pour la génération d'une politique, c'est à dire pour la sélection des actions. Nous proposerons partie 6.4 une expérimentation de nos méthodes. Nous continuerons, partie 6.5 en détaillant une autre piste heuristique pour la recherche des concepts pertinents que nous relierons avec des travaux similaires. Nous conclurons finalement partie 6.6.

6.1 Principes du Q-Concept Learning

Appliquer un même apprentissage à plusieurs couples $(état, action)$ Une des façons de concevoir la généralisation des politiques pour l'apprentissage par renforcement est la suivante : considérons deux états de l'environnement e_1 et e_2 équivalents, dans le sens où il revient au même pour un agent de se retrouver dans e_1 que dans e_2, que ce soit du point de vue de la récompense potentielle que des actions admissibles. On a alors intérêt effectivement à utiliser l'apprentissage issu du fait de se retrouver en e_1 sur l'état e_2 et réciproquement.
Rappelons que l'apprentissage par renforcement, tel que nous l'avons décrit, procède de l'apprentissage de la qualité de chacune des actions de A pour chacun des états de E par échantillonnages successifs à partir de valeurs de récompenses estimées et de valeurs réellement obtenues. Par exemple, pour l'algorithme de *Q-Learning*, à chacune des actions effectuées par l'agent, l'estimation de l'intérêt à long terme du nouvel état rencontré combinée à la récompense obtenue est prise comme base d'échantillon et il y a combinaison entre l'ancienne valeur estimée et la nouvelle estimation constatée (voir partie 2.4.3).
Ainsi, appliquer conjointement l'apprentissage sur deux états de l'environnement e_1 et e_2 revient à appliquer les règles de mise à jour de l'action a sélectionnée dans l'état e_1 à la même action a dans l'état e_2 et réciproquement (« même » voulant dire ici ayant une description équivalente). Les valeurs de $Q(e_1, a)$ et $Q(e_2, a)$ convergeront donc plus rapidement vers la valeur $Q^*(e_{1,2}, a)$, puisque bénéficiant de mises à jours potentiellement plus fréquentes. Ceci est à rapprocher des notions d'équivalence entre états appelées *optimal value equivalence* et *action-sequence equivalence* dans [Givan et al., 2003].

Apprentissage *en ligne* Le problème vient précisément du fait que l'on ne sait pas, au début de l'apprentissage, si l'action a dans l'état e_1 et l'action a dans l'état e_2 sont équivalentes. Cette information n'est disponible de façon certaine qu'à la fin de l'apprentissage (c'est d'ailleurs sur cette propriété que nous avons développé l'algorithme 5.1 page 120).
Le processus que nous allons proposer maintenant tente, dans l'esprit de l'apprentissage par renforcement, de trouver conjointement les équivalences entre les états et d'utiliser la connaissance courante sur ces équivalences pour accélérer l'apprentissage.

Le treillis de Galois comme biais d'apprentissage Comme nous ne pouvons pas envisager l'ensemble des équivalences possibles entre actions des états de l'environnement, il nous faut alors considérer un biais permettant de limiter le nombre des généralisations considérées. Nous avons introduit au chapitre 4, la notion de *treillis de Galois*, en tant qu'espace de généralisation, c'est à dire en tant qu'ensemble des généralisations possibles d'un ensemble d'objets compte tenu d'un langage de généralisation. C'est donc cette structure que nous proposons d'utiliser comme biais de généralisation.
Définissons L_E comme le langage de généralisation utilisé pour décrire l'ensemble E des états de l'environnement. Nous allons ainsi, pour un pas d'apprentissage donné, appliquer la connaissance nouvellement acquise (c'est à dire la valeur de la récompense effectivement obtenue combinée avec une estimation des récompenses à venir) classiquement à l'état envisagé comme dans tout algorithme d'apprentissage par renforcement, mais également à l'ensemble de ses généralisations possibles selon L_E. C'est le principe de base de nos algorithmes de *Q-Concept Learning*.

6.1.1 Quelle fonction de qualité considérons-nous ?

Voyons maintenant ce qu'implique le principe de base exprimé ci-dessus sur la fonction considérée pour l'apprentissage par renforcement. La fonction classiquement estimée est la fonction de $Q(e, a)$, ayant pour domaine de définition ψ, l'ensemble des couples $(état, action)$ admissibles. Nous venons d'énoncer que les éléments du langage L_E allaient être utilisés pour la généralisation des états de l'environnement. Il nous faut alors repréciser la fonction de qualité que nous envisagerons. Définissons dans un premier temps la notion d'action définie pour un élément de langage $l \in L_E$.

Définition 6.1 (action définie pour un ensemble d'états)
Soit une tâche d'apprentissage par renforcement définie par un PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$, $(L_E, \leq_{L_E}, \otimes_{L_E})$ étant un langage de généralisation pour E et $(L_A, \equiv_{L_A})$ un langage d'équivalence pour A. Soit $E_1 \subseteq E$ un ensemble d'états et $a \in A$ une action. On dit que l'action a est **définie** pour E_1 si pour tous les états $e \in E_1$, on a $(e, a) \in \psi$.
Soit $l \in L_E$. Par abus de langage, on dit que a est **définie** pour l si a est définie pour $i(l)$.
Soit $\mathcal{P} \in \mathbb{P}_E$ une partition de E. On note $\psi_{\mathcal{P}} \subseteq \psi$ l'ensemble des couples $(e, a) \in \psi$ tels que $\forall (e, a) \in \psi$, a est définie pour $[e]_{\equiv_{\mathcal{P}}}$.

Remarque 6.1 *Comme souvent dans le formalisme d'apprentissage par renforcement, il y a un sous-entendu concernant une certaine relation d'équivalence entre actions dans le sens ou deux actions pour deux états différents peuvent avoir la même étiquette. Dans notre formalisme, ceci est explicite par le fait que L_A soit un langage d'équivalence. Ainsi, deux actions seront équivalentes si elles ont la même étiquette.*
Pour le reste de notre propos, nous allons effectivement considérer que l'équivalence entre les actions se fait sur leur étiquette. Ainsi, nous pouvons faire l'abus d'écriture qui consiste à considérer $A = L_A$.

Pour ce qui est de la fonction de qualité estimée au cours de l'apprentissage, ceci revient à considérer la fonction $Q(l, a) : L_E \times A \to \mathbb{R}$, plutôt que la fonction $Q(e, a) : E \times A \to \mathbb{R}$.
Ainsi, compte tenu de ce domaine de définition et de la structure des treillis de Galois, utiliser l'apprentissage d'une action dans un état $e \in E$ pour tous les éléments de langage qui couvrent e revient à estimer la valeur attendue à long terme de chacune des actions pour chacun des concepts formels du treillis de Galois formé à partir de l'ensemble E des états de l'environnement contraint par le langage de généralisation L_E.

Définition 6.2 (fonction d'apprentissage pour le *Q-Concept Learning*)
Soit une tâche d'apprentissage par renforcement formalisée par un PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$, avec $(L_E, \leq_{L_E}, \otimes_{L_E})$ étant un langage de généralisation pour E et tel que $L_A = A$. Soit le treillis de Galois $\mathcal{TG}(E, L_E)$ et $\mathfrak{C}_{(E,L_E)}$ l'ensemble de ses concepts formels.
Par extension de la définition 6.1, on dira qu'une action $a \in A$ est définie pour un concept $c = (E_i, l) \in \mathfrak{C}_{(E,L_E)}$ si a est définie pour l. On dira alors que le couple (c, a) est **admissible**. L'ensemble des couples (c, a) admissibles est noté $\psi_{\mathcal{TG}(E,L_E)}$. Finalement, la fonction considérée pour les algorithmes de *Q-Concept Learning* est $Q : \psi_{\mathcal{TG}(E,L_E)} \longrightarrow \mathbb{R}$.

Notation 6.1
$c_e \in \mathfrak{C}_{(E,L_E)}$ désigne le concept dont l'extention est le singleton $\{e\}, e \in E$.

Comme nous avons spécifié que tous les états étaient observables, c_e existe pour tous les états de l'environnement.

Implémentation Dans un premier temps, nous envisagerons effectivement la construction du treillis de Galois $\mathcal{TG}(E, L_E)$. Nous associerons à chacun des concepts de $\mathfrak{C}_{(E,L_E)}$ l'ensemble des actions définies pour chacun des éléments de L_E. Ceci revient bien à considérer $L_E \times A$ comme domaine de définition car chaque concept de $\mathfrak{C}_{(E,L_E)}$ est un couple composé d'un élément $l \in L_E$ d'une part et de l'ensemble des éléments de E couverts par l d'autre part. A chacun de ces concepts, nous associerons également la valeur de la fonction $Q(c, a)$. Nous bénéficierons bien d'une structure permettant de représenter $Q(c, a)$ et de plus, en conservant la structure de treillis, des opérations courantes quant à l'utilisation de $Q(c, a)$ seront facilitées : révision de la valeur pour un couple (c, a) donné et recherche de la valeur pour un tel couple. La figure 6.1 donne un exemple graphique de cette implémentation.

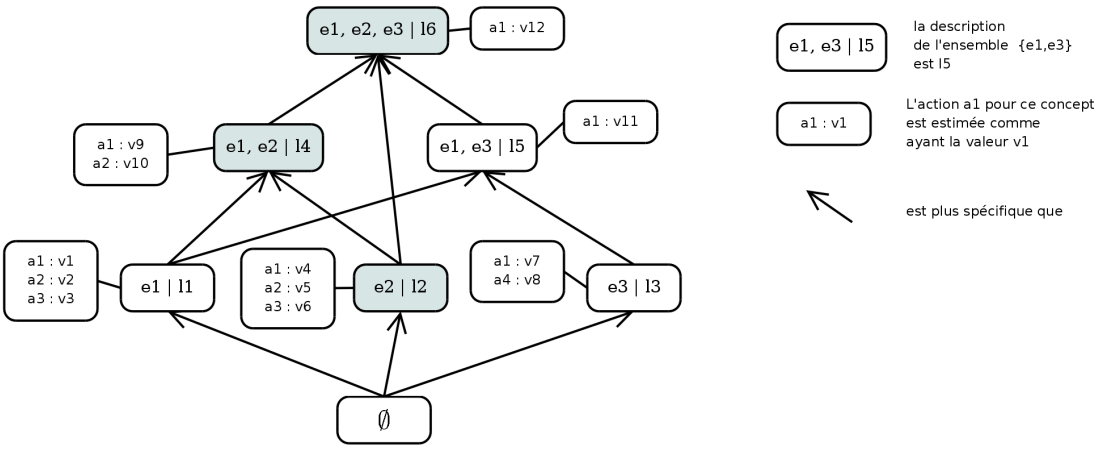


FIG. 6.1 – Schéma de l'implémentation de la fonction $Q(c, a)$. Pour chacune des actions admissibles pour un concept donné, la valeur de sa qualité estimée est stockée.

Apprentissage appliqué à l'ensemble des états potentiellement équivalents Remarquons que la fonction proposée inclut la fonction classique $Q(e, a)$ pour l'apprentissage par renforcement. En effet, on a bien pour chaque élément $e_i \in E$, un concept n'ayant que e_i comme extension, car chacun des états est différentiable des autres par le langage L_E. Les actions admissibles pour celui-ci sont alors exactement les actions admissibles pour l'état e_i.

Finalement, cette structure va permettre de pratiquer les mises à jour classiques d'un apprentissage par renforcement, mais en plus, chaque mise à jour effectuée pour un état e_i pourra être appliquée à l'ensemble des états potentiellement équivalents à e_i. C'est cette procédure que décrit

l'algorithme 6.1 et qui est l'idée clé du *Q-Concept Learning*.
Nous avons proposé cet algorithme comme variante du *Q-Learning*, mais il est facilement adaptable aux autres algorithmes d'apprentissage par renforcement comme *Sarsa* en adaptant la fonction de mise à jour.

Données : • PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$ avec un état initial E_0 et un état final E_T
• Un langage de généralisation $(L_E, \leq_{L_E}, \otimes_{L_E})$ pour E
• $0 \leq \gamma \leq 1$, le paramètre de dégressivité
• $0 < \alpha < 1$ le paramètre de remise en cause de la connaissance
Résultat : • $\pi^*(e,a) \in \Pi^*$, une politique optimale pour M
début
  Initialisation de la politique
  Initialiser $\hat{Q}(c,a) = v_0 \in \mathbb{R}$, $\forall (c,a) \in \psi_{\mathcal{TG}_{(E,L_E)}}$;
  pour *Chaque épisode* **faire**
    $e \leftarrow E_0$;
    tant que $e_t \neq E_T$ **faire**
1   a = choisir une action en utilisant $\hat{Q}(c,a)$;
      appliquer a;
      $r \leftarrow$ recevoir une récompense;
      $e' \leftarrow$ recevoir un nouvel état;
      pour $c \in \mathfrak{C}_{(E,L_E)}$ *tels que* $e \in ext(c)$ **faire**
2       $\hat{Q}(c,a) \leftarrow (1-\alpha)\hat{Q}(c,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(e',a))$
      fin
      $e \leftarrow e'$;
    fin
  fin
fin

Algorithme 6.1: *Q-Concept learning* : appliquer un apprentissage effectué sur un état à toutes les généralisations permises par un langage de description

6.1.2 Convergence de la valeur de $\hat{Q}(c,a)$

Pour certaines actions, la formule de mise à jour (ligne 2 de l'algorithme 6.1) fait converger la valeur des couples $\hat{Q}(c,a)$ vers une valeur $Q^*(c,a)$ de la même manière qu'un algorithme par renforcement classique. Cependant, ce n'est pas le cas pour toutes les actions et pour tous les concepts. Détaillons les différents cas.

Concepts dont l'extention est réduite à un état Remarquons tout d'abord que pour les concepts dont l'extention est réduite à un état, l'algorithme se comporte exactement comme un algorithme de *Q-Learning* classique. En effet, dans ce cas, la formule de mise à jour devient :

$$\hat{Q}(c_e,a) \leftarrow (1-\alpha)\hat{Q}(c_e,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(e',a))$$

Ainsi, pour tous les couples $(e,a) \in \psi$, la valeur $\hat{Q}(c_e,a)$ convergera vers la valeur $Q^*(e,a)$. Rappelons que la méthode ne converge pas complètement en utilisant une *moyenne exponentiellement pondérée par récence*, comme dans notre cas, car nous utilisons un facteur α constant.

Couples (c,a) dont la valeur $\hat{Q}(c,a)$ converge Considérons $\psi_{converge} \subseteq \psi_{\mathcal{TG}}$ tel que

$$\psi_{converge} = \left\{(c,a) \in \psi_{\mathcal{TG}} \text{ tel que } \forall e_i, e_j \in ext(c), Q^*(c_{e_i},a) = Q^*(c_{e_j},a)\right\}$$

C'est l'ensemble des couples $(concept, action)$ pour lesquels les états inclus dans l'extention de *concept* ont même valeur en suivant une politique optimale.

La formule de mise à jour :

$$\hat{Q}(c,a) \leftarrow (1-\alpha)\hat{Q}(c,a) + \alpha(r + \gamma \underset{a'}{max}\hat{Q}(e',a))$$

fait alors converger $\hat{Q}(c_e,a)$ vers $v^* = Q^*(e,a)$ pour tous les états inclus dans l'extention de *concept*. Par conséquent, elle fait également converger $\hat{Q}(c,a)$ vers v^* pour tous les couples $(c,a) \in \psi_{converge}$.

Couples (c,a) dont la valeur $\hat{Q}(c,a)$ ne converge pas Considérons un couple (c,a) tel que pour au moins deux états e_1 et $e_2 \in ext(c)$ on ait $Q^*(e_1,a) \neq Q^*(e_2,a)$.
Dans ce cas, la valeur de $\hat{Q}(c,a)$ ne convergera pas vers une valeur fixe. Celle-ci oscillera autour d'une valeur telle que $Min\ Q^*(c_{e_i},a) < \hat{Q}(c,a) < Max\ Q^*(c_{e_j},a)$, $e_i, e_j \in ext(c)$.
En effet, soit e_{min} l'état appartenant à l'extention de c tel que $Q^*(c_{e_{min}},a) = Min\ Q^*(c_{e_i},a), e_i \in ext(c)\}$ et réciproquement e_{max}, tel que $Q^*(c_{e_{max}}) = Max\ Q^*(c_{e_i},a) e_i \in ext(c)\}$. Après convergence de toutes les valeurs de $\hat{Q}(c_e,a), e \in E$ vers $Q^*(c_e,a)$, la mise à jour

$$\hat{Q}(c,a) \leftarrow (1-\alpha)\hat{Q}(c,a) + \alpha(r + \gamma \underset{a'}{max}\hat{Q}(e',a))$$

rapprochera $\hat{Q}(c,a)$ de $Q^*(c_{e_{min}},a)$ à chaque fois que l'agent effectuera une mise à jour pour le concept $c_{e_{min}}$. Réciproquement, $\hat{Q}(c,a)$ se rapprochera de $Q^*(c_{e_{max}},a)$ à chaque fois que l'agent effectuera une mise à jour pour le concept $c_{e_{max}}$.
Pour résumer, la suite ayant pour abscisse le nombre des mises à jour et pour ordonnée la succession des valeurs de $\hat{Q}(c,a)$ est une fonction des paramètres α, γ, du modèle de l'environnement P, de la fonction de récompense R et de la politique π suivie par l'agent.
De plus, il existe un rang à partir duquel celle-ci est bornée par $Min\ Q^*(c_{e_i},a)$ et $Max\ Q^*(c_{e_i},a)$.

Voyons maintenant comment caractériser les concepts en considérant ces propriétés de convergence, la structure de treillis et les politiques optimales.

6.2 Pertinence d'un concept - concepts améliorant une politique

Nous venons de proposer le cadre général du *Q-Concept Learning* : l'estimation de la fonction de qualité pour chacune des actions pour chacune des généralisations permises par le langage de description des états de l'environnement. Ligne 1, de notre algorithme, nous avons proposé que l'agent utilise la fonction $\hat{Q}(c,a)$ afin de déterminer l'action qu'il va sélectionner. Habituellement, le seul compromis envisagé pour le choix des actions dans le cadre de l'apprentissage par renforcement est le compromis exploration-exploitation. Ici, il y aura un deuxième compromis : quel niveau de généralisation utiliser ? En effet, l'état courant, c'est à dire l'état dans lequel l'agent doit sélectionner une action peut être considéré avec différents niveaux de généralisation. Le niveau le plus spécialisé est le concept dont l'extension est réduite à l'état lui-même, le niveau le plus général est le concept dont l'extension est l'ensemble des états.
Nous allons maintenant donner un premier élément pour déterminer le bon niveau de généralisation afin de gérer ce compromis. Nous traiterons la question de la sélection des actions dans la partie suivante.

Nous verrons dans cette partie comment catégoriser les concepts du treillis ainsi construit. Nous verrons que l'on peut les découper en trois catégories : les concepts non pertinents, les concept pertinents et parmi ces derniers, les concepts pertinents maximaux. Nous montrerons en quoi il est intéressant de repérer les concepts pertinents maximaux. Finalement, nous proposerons une modification de l'équation de mise à jour de l'algorithme 6.1 page précédente permettant de les trouver plus rapidement. Nous nous baserons pour cela sur la propriété de concepts améliorant une politique.

6.2.1 Concepts pertinents

Comme nous l'avons exprimé plus haut, un des critères que devra utiliser l'agent pour sélectionner son action est le niveau de généralisation avec lequel il devra considérer l'état de l'environnement qu'il observe. Le paradigme est alors le suivant : plus un concept est spécifique

(éventuellement avec une extention réduite à un état), moins la valeur des actions définies sur celui-ci auront été mises à jour, donc moins l'estimation de la fonction de qualité sera précise. Celle-ci peut même, dans le pire des cas, ne jamais avoir été mise à jour.
Inversement, si un concept est trop général, son extention regroupera des états qui n'ont pas d'actions optimales en commun. La considération de ce concept sera alors inutile pour ce qui est notre objet principal, c'est à dire la sélection d'actions optimales.
Pour formaliser ce paradigme et donner effectivement des méthodes pour le prendre en compte, nous allons proposer les notions de **concepts pertinents** et de **concepts pertinents maximaux**. Nous relierons ces éléments à la partition $\mathcal{P}^*$ décrite au chapitre 5.

Concepts pertinents Parmi l'ensemble des concepts qui généralisent un état observé par l'agent, nous allons définir les concepts intéressants du point de vue d'une politique optimale, c'est à dire intéressants du point de vue de la sélection des actions. Ces concepts que nous appelons concepts pertinents sont ceux qui permettent de généraliser des états ayant des actions optimales équivalentes. De plus, la valeur après convergence de celles-ci devra être identique.

Définition 6.3 (pertinence d'un concept)
Soit une tâche d'apprentissage par renforcement définie par un PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$, $(L_E, \leq_{L_E}, \otimes_{L_E})$ étant un langage de généralisation pour E et $(L_A, \equiv_{L_A})$ un langage d'équivalence pour A. Soit un concept $c \in \mathfrak{C}_{(E,L_E)}$.
c est **pertinent** s'il existe au moins une action a définie pour c et telle que $\forall e_i \in ext(c)$, $a \in A^*(e_i)$ et telle que $Q^*(e_i, a) = Q^*(c, a)$, c'est à dire que a doit être optimale pour tous les états de l'extension de c et avoir la même valeur de qualité en suivant une politique optimale. Cette valeur unique est notée $Q^*(c, a)$
L'ensemble des actions a_i définies pour c telle que a_i soit optimale pour toutes les extensions de c en suivant une politique optimale est notée $A^*(c)$.

Par conséquent, nous avons la propriété suivante :

Propriété 6.1
$\forall c, c_i \in \mathfrak{C}_{(E,L_E)}$, tels que $c_i \leq c$, $\forall a \in A^*(c)$, on a $a \in A^*(c_i)$ et $Q^*(c, a) = Q^*(c_i, a)$.

Si une action a est optimale avec une valeur $Q^*(e, a)$ unique pour tous les états e appartenant à l'extension de c, comme $ext(c) \supseteq ext(c_i)$, il en va de même pour c_i.

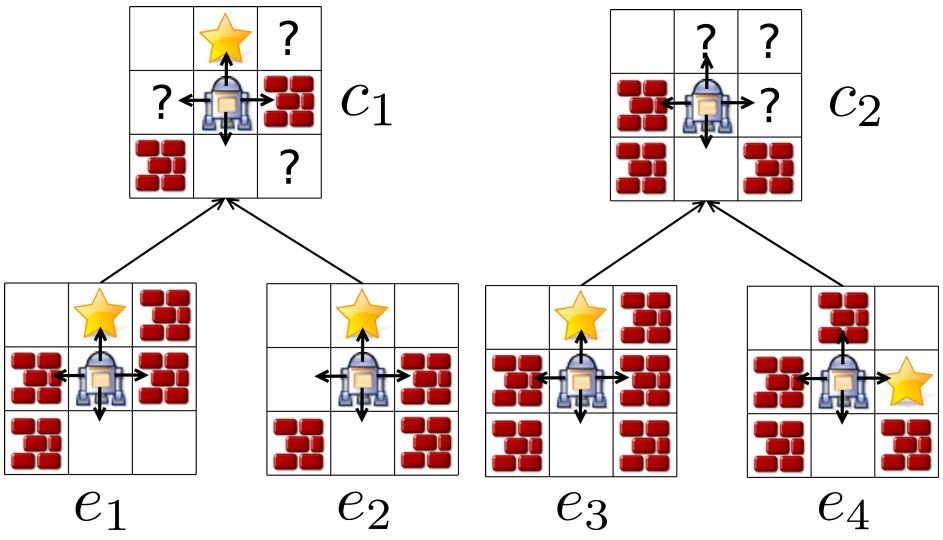
Notons que pour un concept pertinent, toutes les extensions de celui-ci sont telles que $\forall e_i, e_j \in ext(c)$, $e_i \equiv_* e_j$[44]. Des états peuvent être équivalents selon $\equiv_*$ bien que leurs actions optimales ne partagent pas la même valeur.
On dira que ces concepts sont semi-pertinents.

Définition 6.4 (concept semi-pertinents)
Soit une tâche d'apprentissage par renforcement définie par un PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$, $(L_E, \leq_{L_E}, \otimes_{L_E})$ étant un langage de généralisation pour E et $(L_A, \equiv_{L_A})$ un langage d'équivalence pour A. Soit un concept $c \in \mathfrak{C}_{(E,L_E)}$.
c est **semi-pertinent** s'il existe au moins une action a définie pour c et telle que $\forall e_i \in ext(c)$, $a \in A^*(e_i)$ et telle que $\exists e_i, e_j \in ext(c)$ tels que $Q^*(e_i, a) \neq Q^*(e_j, a)$, c'est à dire que a doit être optimale pour tous les états de l'extension de c mais que pour au moins deux états de l'extension, la valeur de qualité en suivant une politique optimale est différente.

Exemple 6.1 *La figure 6.2 page suivante propose trois cas concernant la pertinence des concepts : un concept pertinent, un concept non pertinent et un concept semi-pertinent. Par exemple, l'action d'aller sur la case avec l'étoile engendre une récompense observée $r = 1$, $r = 0$ pour les autres actions, avec un facteur de dégressivité $\gamma = 0,9$.*

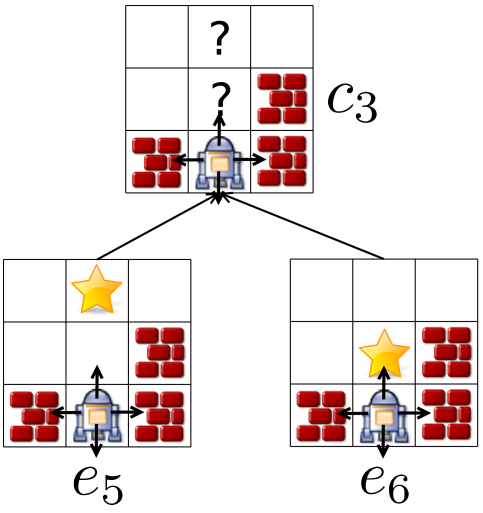
- *Le concept c_1 est pertinent puisque $Q^*(e_1,$ « déplacement Nord » $) = Q^*(e_2,$ « déplacement Nord » $) = 1$.*

[44] Deux états sont équivalents par $\equiv_*$ si leurs actions optimales ont même étiquette (définition 5.3 page 114)



(a): Exemple de concept pertinent

(b): Exemple de concept non pertinent



(c): Exemple de concept semi-pertinent.

FIG. 6.2 – Concepts pertinents : trois cas de figure. Premièrement, figure (a), le concept c_1 est pertinent car il est la généralisation de deux états qui ont le même ensemble d'actions optimales, celles-ci partageant la même valeur. Par contre, figure (b), le concept c_2 n'est pas pertinent. Figure (c), les états e_5 et e_6 ont même ensemble d'actions optimales, cependant, le concept c_3 n'est que semi-pertinent car les valeurs des actions ne sont pas les mêmes.

- *Le concept c_2 n'est pas pertinent car e_3 et e_4 n'ont pas les mêmes actions optimales.*
- *c_3 est semi-pertinent car e_5 et e_6 ont même ensemble d'actions optimales ($\{$ « déplacement Nord » $\}$) mais $Q^*(e_5,$ « déplacement Nord » $) = 0,9$ et $Q^*(e_6,$ « déplacement Nord » $) = 1$.*

Établissons maintenant une propriété importante qui nous servira à caractériser les politiques optimales dans notre cadre : pour tout concept non pertinent, il existe un concept pertinent qui lui est inférieur et dont la valeur des actions optimales sont supérieures.

Propriété 6.2
$\forall c \in \mathfrak{C}_{(E,L_E)}$ tel que c soit non pertinent, $\exists c_{pertinent} \in \mathfrak{C}_{(E,L_E)}$ tel que :

1. $c_{pertinent} < c$
2. $c_{pertinent}$ soit pertinent
3. si c a au moins une action définie alors $\underset{a'}{max} Q^*(c_{pertinent}, a) > \underset{a''}{max} Q^*(c, a)$

Preuve 6.1
c est non pertinent, on a alors trois cas :

1. c n'a aucune action définie
2. $\exists a$ définie pour c et telle que $a \in A^*(c)$ et telle que $\exists e_1 \in ext(c)$ tels que $a \notin A^*(e_1)$
3. c est semi-pertinent

1. Soit $e_1 \in ext(c)$. Le concept c_{e_1} ayant $\{e_1\}$ pour extension est tel que $c_{e_1} < c$. De plus, c_{e_1} est pertinent.
2. Idem que pour 1. De plus Soit $a_1 \in A^*(c_{e_1})$ on a bien $Q^*(c_{e_1}, a_1) > Q^*(c, a)$. Sinon, on aurait $a \in A^*(c_{e_1})$.
3. idem que pour 1. De plus, soit $a \in A^*(c)$, $e_i = \underset{e \in ext(c)}{arg\ max} Q(e, a)$ et $e_j \in ext(c)$ tel que $Q^*(e_j, a) < Q^*(e_i, a)$.

□

Concepts pertinents maximaux Définissons parmi l'ensemble des concepts pertinents les concepts pertinents les plus généraux.

Définition 6.5 (concepts pertinents maximal)
Soit une tâche d'apprentissage par renforcement définie par un PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$, $(L_E, \leq_{L_E}, \otimes_{L_E})$ étant un langage de généralisation pour E et $(L_A, \equiv_{L_A})$ un langage d'équivalence pour A. Soit un concept $c \in \mathfrak{C}_{(E,L_E)}$.
c est un **concept pertinent maximal** ssi c est pertinent et $\forall c_i \in \mathfrak{C}_{(E,L_E)}$, $c \leq_{L_E} c_i \Rightarrow c_i$ n'est pas pertinent.

Exemple 6.2 *La figure 6.3 page ci-contre présente un exemple de concept pertinent maximal. Les concepts $e_1, e_2, e_3, e_4, c_1, c_2, c_3$ sont pertinents, mais seuls les concepts e_1 et c_3 sont pertinent maximaux. En effet, l'ensemble des états couverts par ceux-ci ont même ensemble d'actions optimales, et il n'existe pas de concept plus général qui soit pertinent.*

L'intérêt des concepts pertinents maximaux est double. D'un point de vue statistique, les valeurs des actions définies pour ces concepts sont plus fréquemment mises à jour que d'autres, puisque leur extention inclut de nombreux états.
D'un point de vue sémantique, leur intention est la plus générale possible relativement à une politique optimale pour la tâche en cours. Leurs intentions sont donc des éléments du langage de description des états L_E les plus généraux possibles ayant une pertinence concernant la prise de décision.
Cela nous renvoie à ce que nous avons développé dans le chapitre précédent aux parties 5.2 et 5.3 concernant la généralisation des politiques pour l'apprentissage par renforcement. Nous pouvons donc établir une relation entre les notions de concepts pertinents et la partition $\mathcal{P}^*$.

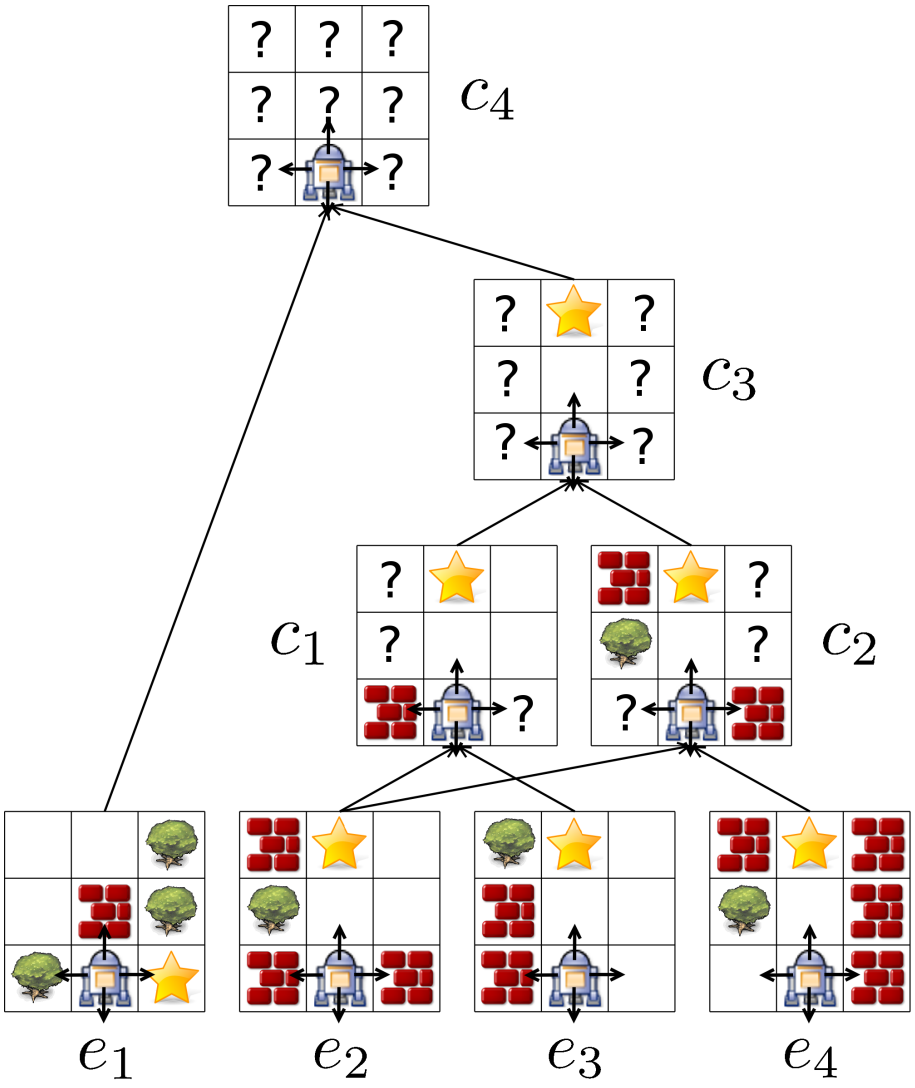


FIG. 6.3 – Seuls les concepts e_1 et e_3 sont pertinents maximaux pour cet exemple.

Relation entre concepts pertinents et $\mathcal{P}^*$ Rappelons que $\mathcal{P}^*$ est la partition des états de l'environnement impliquée par la relation d'équivalence $\equiv_*$. Nous pouvons reprendre les trois cas que nous avons développés partie 5.2 page 112.

- Cas 1 : $\mathcal{P}^*$ respecte L_E.
  Dans ce cas, $\mathcal{P}^*$ est constituée par l'ensemble des concepts pertinents maximaux est $\mathcal{P}^*$.
- Cas 2 : $\mathcal{P}^*$ ne respecte pas L_E et il existe une partition $Part_1 \subset \mathcal{P}^*$ qui respecte L_E.
  Alors chacune de ces partitions est composée d'extension de concepts pertinents maximaux.
- Cas 3 : Il n'existe pas de partition $Part_1 \subset \mathcal{P}^*$ qui respecte L_E.
  Nous avons exclu cette situation (voir le dernier cas de la propriété 5.1 page 114). En effet, nous avons admis qu'un agent pouvait distinguer l'ensemble des états de l'environnement. Ainsi, la plus petite partition $\mathcal{P}_0$ respecte L_E.

Exemple 6.3 *Figure 6.3 page précédente, si on restreint l'ensemble des états de l'environnement à $E = \{e_1, e_2, e_3, e_4\}$, l'ensemble des concepts pertinents maximaux est $\{e_1, c_3\}$. $\{int(e_1), int(c_3)\} = \big\{\{e_1\}, \{e_2, e_3, e_4\}\big\}$ forment bien une partition des états de l'environnement selon les classes d'équivalences définies par la relation $\equiv_*$. On est donc dans le cas 1 que nous venons de présenter. Figure 6.2 (c) page 131, il n'existe pas de concept pertinent incluant dans leur extention les états e_3 et e_6, malgré le fait que ces deux états aient le même ensemble d'actions optimales. Cela est dû au fait qu'il n'existe pas, dans le langage de description que nous nous sommes donné pour représenter les états de l'environnement ainsi que leurs généralisations, d'élément du langage plus général que les descriptions des états e_3 et e_6 qui ne soit pas plus général que la description de l'état e_5. On est donc dans le cas 2.*

6.2.2 Recherche des concepts pertinents

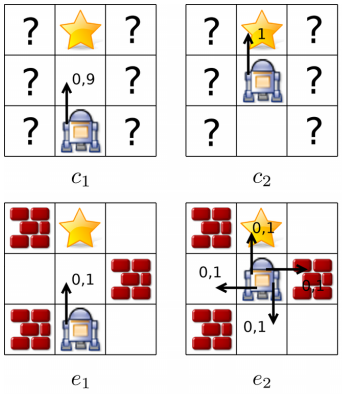
Ligne 2, algorithme 6.1 page 128, nous avons proposé le fait que la valeur de l'action a sélectionnée dans l'état e soit mise à jour pour chacun des concepts couvrant e par la formule classique de *Q-learning* en utilisant $\underset{a'}{max} Q(e', a)$ pour estimation de la récompense cumulée.

$$\hat{Q}(c, a) \leftarrow (1 - \alpha)\hat{Q}(c, a) + \alpha(r + \gamma \underset{a'}{max}\hat{Q}(e', a)) \tag{6.1}$$

Voyons dans un premier temps en quoi cette formule de mise à jour, bien que permettant effectivement une convergence, pose problème en pratique. Dans un deuxième temps, nous proposerons la notion de concept améliorant la politique. Finalement, nous verrons comment cette notion induira une formule de mise à jour répondant aux problèmes pratiques mentionnés.

Problèmes posés par l'estimation $\underset{a'}{max}\hat{Q}(e', a)$ Nous avons indiqué que les concepts pouvaient être regroupés en quatre catégories : 1) les concepts non pertinents 2) parmi ceux-ci, les concepts semi-pertinents 3) les concepts pertinents 4) parmis ceux-ci, les concepts pertinents maximaux. La succession de la mise à jour 6.1 permet effectivement de faire converger $\hat{Q}(c, a)$ vers $Q^*(c, a)$ pour les concepts pertinents. En effet, par définition, les concepts pertinents sont ceux dont les actions optimales sont équivalentes avec même valeur pour une politique optimale. Pour les concepts non pertinents, la valeur ne converge pas nécessairement, cela dépend de la politique suivie.
L'objectif de la généralisation n'est pas seulement de trouver les concepts pertinents maximaux après convergence comme nous l'avons fait dans le chapitre précédent. Nous voulons en plus ici utiliser en ligne, c'est à dire au fur et à mesure de l'apprentissage, la généralisation des politiques. Si l'agent vient de sélectionner l'action a dans l'état e, en quoi l'utilisation de la mise à jour 6.1 sur un concept $c, e \in ext(c)$ va-t-elle contre ce principe ?
La valeur $\underset{a'}{max}\hat{Q}(e', a)$ est la valeur de l'action ayant la meilleure récompense à long terme quand l'agent vient de sélectionner l'action a dans l'état e et se retrouve dans l'état e'. Le problème est que cette valeur n'est relative qu'à l'état e', et n'est le produit d'aucune généralisation. Dans le pire des cas, elle n'a même jamais été évaluée et correspond à la valeur d'initialisation.
L'exemple suivant va montrer en quoi au cours de l'apprentissage, cette mise à jour peut provoquer une **dégradation de la connaissance** plutôt qu'une amélioration.

Exemple 6.4 *Admettons figure 6.4 page suivante, que l'agent ait sélectionné l'action a « déplacement Nord » dans l'état e_1 et se retrouve donc dans l'état e_2. Pour le concept c_1, la*



Mise à jour de $\hat{Q}(c_1, a)$ avec a=« Déplacement Nord » en utilisant l'état e_2.

$$\hat{Q}(c_1, a) \leftarrow (1-\alpha)\hat{Q}(c_1, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(e_2, a))$$
$$\hat{Q}(c_1, a) \leftarrow (1-0,9) \times 0,9 + 0,9 \times (0 + 0,9 \times 0,1)$$
$$\hat{Q}(c_1, a) \leftarrow 0,171$$

Mise à jour de $\hat{Q}(c_1, a)$ en utilisant le concept c_2.

$$\hat{Q}(c_1, a) \leftarrow (1-\alpha)\hat{Q}(c_1, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(c_2, a))$$
$$\hat{Q}(c_1, a) \leftarrow (1-0,9) \times 0,9 + 0,9 \times (0 + 0,9 \times 1)$$
$$\hat{Q}(c_1, a) \leftarrow 0,9$$

FIG. 6.4 – Illustration du problème posé par la mise à jour de la fonction $\hat{Q}(c, a)$ avec la valeur de la meilleur action pour l'état suivant plutôt qu'avec la valeur de la meilleur action d'un concept pertinent. Ici, c'est la première fois que l'agent se retrouve dans les états e_1 et e_2. C'est donc la valeur d'initialisation qui est utilisée comme valeur pour toutes les actions.

valeur de l'action a, $\hat{Q}(c_1, a) = 0,9$ *a déjà convergé vers la valeur* $Q^*(c_1, a)$*, suite à de nombreux exemples rencontrés. La récompense r observée est alors* $r = 0$ *et* $\gamma = \alpha = 0,9$*. L'estimation des récompenses à venir à partir de l'état* e_2 *est alors à sa valeur d'initialisation* $e_2 = 0,1$ *par exemple. La mise à jour*

$$\hat{Q}(c_1, a) \leftarrow (1-\alpha)\hat{Q}(c_1, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(e_2, a))$$

a alors pour effet de dégrader la connaissance. En effet, la valeur est alors modifiée comme suit :

$$\hat{Q}(c_1, a) \leftarrow (1-0,9) \times 0,9 + 0,9 \times (0 + 0,9 \times 0,1)$$
$$\hat{Q}(c_1, a) \leftarrow 0,1 \times 0,9 + 0,9 \times 0,09$$
$$\hat{Q}(c_1, a) \leftarrow 0,09 + 0,081$$
$$\hat{Q}(c_1, a) \leftarrow 0,171$$

Bien sûr, au fur et à mesure des interactions suivantes, la valeur de $Q(e_2, a)$ *convergera vers* $Q^*(e_2, a) = Q^*(c_1, a) = 0,9$ *et de même pour* $Q(c_1, a)$ *grâce à la correction apportée à* $Q(e_2, a_2)$*. Cependant, c'est précisément ce que nous voulons éviter grâce à la généralisation : explorer l'ensemble des couples* $(état, action)$ *infiniment souvent. Nous voudrions au contraire que le concept* c_2 *dont l'action optimale* a_2*, « Déplacement Nord » serve pour mettre à jour la valeur* $Q(c_1, a)$ *et confirme ainsi la valeur de* $Q(c_1, a)$ *par la mise à jour suivante :*

$$\hat{Q}(c_1, a) \leftarrow (1-\alpha)\hat{Q}(c_1, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(c_2, a))$$
$$\hat{Q}(c_1, a) \leftarrow (1-0,9) \times 0,9 + 0,9 \times (0 + 0,9 \times 1)$$
$$\hat{Q}(c_1, a) \leftarrow 0,1 \times 0,9 + 0,9 \times 0,9$$
$$\hat{Q}(c_1, a) \leftarrow 0,09 + 0,81$$
$$\hat{Q}(c_1, a) \leftarrow 0,9$$

Pour l'exemple que nous venons de donner, la sélection du concept c_1 est évidente, d'autant plus que nous connaissons déjà la réponse. Cependant, la recherche du concept adéquat avec une connaissance incomplète pose de nombreuses difficultés. Nous allons maintenant proposer notre résolution de la question.

Nous avons, dans la partie précédente, indiqué que les concepts pouvaient se découper en quatre ensembles (concepts pertinents, concepts pertinents maximaux, concepts semi-pertinents et concepts non pertinents) après convergence de la fonction $Q(c, a)$ vers $Q^*(c, a)$. Nous allons

exposer comment différencier ces types concepts au cours de l'apprentissage avec la notion de concepts améliorant une politique. Nous utiliserons alors l'heuristique qui consiste à considérer que la fonction courante $\hat{Q}(c,a)$ possède la propriété de correctement séparer ces ensembles pour utiliser les meilleurs concepts comme valeur de mise à jour, c'est à dire les concepts pertinents maximaux.
Comme il s'agit d'une heuristique, plutôt que d'un résultat exact, nous allons montrer les problèmes engendrés et montrer qu'il n'y a pas remise en cause du processus général de convergence.
Nous proposerons finalement l'algorithme intégrant tous ces éléments.

Séparation des types de concepts avec la fonction $\hat{Q}(c,a)$ Nous allons développer maintenant l'idée suivante : les différents types de concepts ont un comportement différent au cours de l'apprentissage. Ce comportement différent va permettre de repérer les concepts pertinents maximaux. Premièrement, montrons cette capacité de séparation en utilisant $Q^*(e,a)$ comme valeur pour la mise à jour. Nous faisons dans un premier temps l'hypothèse que nous connaissons la fonction $Q^*(e,a)$
Montrons tout d'abord que l'on peut distinguer les concepts pertinents maximaux parmi les concepts pertinents. Étudions le scénario suivant.
Nous disposons de la fonction $Q^*(c,a)$. Prenons quatre concepts pertinents c_1, c_2, c_3 et c_4 tels que $c_1 > c_3$, $c_1 > c_2$ et a, une action admissible et optimale pour c_1, c_2 et c_3 telle qu'après avoir sélectionné l'action a, l'agent se retrouve dans un état couvert par c_4. Admettons que c_1, c_2 et c_3 ne couvrent que des états qui n'ont pas été observés. Ainsi, toutes les actions admissibles ont une valeur initiale $\hat{Q}(c_1,a) = \hat{Q}(c_2,a) = \hat{Q}(c_3,a) = v_0 = 0,1$, par exemple.
Comme c_1, c_2 et c_3 sont pertinents, $c_1 > c_2$, $c_1 > c_3$, et a est optimale, on a $Q^*(c_1,a) = Q^*(c_2,a) = Q^*(c_3,a)$. Par exemple, $Q^*(c_1,a) = 0,81$.
L'agent observe donc un état couvert par c_2 et sélectionne l'action a puis observe un état couvert par c_4.
Appliquons une mise à jour au couple (c_2,a) :

$$\hat{Q}(c_2,a) \leftarrow (1-\alpha)\hat{Q}(c_2,a) + \alpha\big(r + \gamma \max_{a'} Q^*(c_4,a)\big)$$
$$\hat{Q}(c_2,a) \leftarrow (1-0,9) \times 0,1 + 0,9 \times (0 + 0,9 \times 0,9)$$
$$\hat{Q}(c_2,a) \leftarrow 0,1 \times 0,1 + 0,9 \times 0,81$$
$$\hat{Q}(c_2,a) \leftarrow 0,01 + 0,729$$
$$\hat{Q}(c_2,a) \leftarrow 0,739$$

Comme $c_1 > c_2$, toute mise à jour appliquée à c_2 l'est aussi pour c_1 :

$$\hat{Q}(c_1,a) \leftarrow (1-\alpha)\hat{Q}(c_1,a) + \alpha\big(r + \gamma \max_{a'} Q^*(c_4,a)\big)$$
$$\hat{Q}(c_1,a) \leftarrow (1-0,9) \times 0,1 + 0,9 \times (0 + 0,9 \times 0,9)$$
$$\hat{Q}(c_1,a) \leftarrow 0,1 \times 0,1 + 0,9 \times 0,81$$
$$\hat{Q}(c_1,a) \leftarrow 0,01 + 0,729$$
$$\hat{Q}(c_1,a) \leftarrow 0,739$$

L'agent observe ensuite un état couvert par c_3 et sélectionne l'action a et obsreve un état couvert par c_4. On applique donc la mise à jour pour le couple (c_3,a) :

$$\hat{Q}(c_3,a) \leftarrow (1-\alpha)\hat{Q}(c_3,a) + \alpha\big(r + \gamma \max_{a'} Q^*(c_4,a)\big)$$
$$\hat{Q}(c_3,a) \leftarrow (1-0,9) \times 0,1 + 0,9 \times (0 + 0,9 \times 0,9)$$
$$\hat{Q}(c_3,a) \leftarrow 0,1 \times 0,1 + 0,9 \times 0,81$$
$$\hat{Q}(c_3,a) \leftarrow 0,01 + 0,729$$
$$\hat{Q}(c_3,a) \leftarrow 0,739$$

Comme $c_1 > c_3$, toute mise à jour appliquée à c_3 l'est aussi pour c_1 :

$$\hat{Q}(c_1, a) \leftarrow (1-\alpha)\hat{Q}(c_1, a) + \alpha\big(r + \gamma \max_{a'} Q^*(c_4, a)\big)$$
$$\hat{Q}(c_1, a) \leftarrow (1-0,9) \times 0,739 + 0,9 \times (0 + 0,9 \times 0,9)$$
$$\hat{Q}(c_1, a) \leftarrow 0,1 \times 0,739 + 0,9 \times 0,81$$
$$\hat{Q}(c_1, a) \leftarrow 0,0739 + 0,739$$
$$\hat{Q}(c_1, a) \leftarrow 0,8029$$

Comme on peut le constater, la valeur de $\hat{Q}(c_1, a)$ est plus proche de $Q^*(c_1, a)$ que ne l'est $\hat{Q}(c_2, a)$ ou $\hat{Q}(c_3, a)$. On pourrait le traduire grossièrement par le fait que chaque mise à jour faite à l'aide de la fonction $Q^*(c, a)$ rapproche $\hat{Q}(c, a)$ de la fonction $Q^*(c, a)$. Ainsi, les concepts les pertinents les plus généraux convergent plus vite vers $Q^*(c, a)$ puisqu'ils sont plus souvent mis à jour.

Notons que ce n'est pas vrai pour les concepts non pertinents, c'est à dire plus généraux que les concepts pertinents maximaux. Mieux, on sait que les concepts plus généraux que les concepts pertinents maximaux auront des valeurs pour leurs actions optimales inférieures ou égales à celles des concepts pertinents maximaux. Ceci est formalisé par le théorème suivant.

Théorème 6.1 *$\forall c \in \mathfrak{C}$ tel que c soit non pertinent et que c ait au moins une action définie, alors $\exists c_p \in \mathfrak{C}$ tel que c_p soit pertinent et tel que pour $a \in \mathcal{A}^*(c_p)$, $Q^*(c_p, a) \geq \max_{a'} Q^*(c, a)$.*

Preuve 6.2
Soit a_{max}, e_{max} tels que $Q^*(e_{max}, a_{max}) = \max_{\substack{e \in ext(c) \\ a \in disp(c)}} Q^*(e, a)$.
Comme c est non pertinent, $\exists e_1 \in ext(c)$ tel que $Q^*(e_1, a_{max}) < Q^*(e_{max}, a_{max})$. Par conséquent, $Q^*_{max}(c, a_{max}) < Q^*(e_{max}, a_{max})$.
$c_{e_{max}}$ est pertinent, $a_{max} \in \mathcal{A}^*(c_{e_{max}})$ et $Q^*(c_{e_{max}}, a_{max}) > Q^*(c, a_{max})$.

□

Le problème reste évidemment que nous ne connaissons pas la fonction $Q^*(c, a)$. Il nous faut alors développer une méthode heuristique nous permettant de distinguer les types de concepts en ligne de manière similaire. C'est ce que nous allons maintenant présenter avec les concepts améliorant une politique.

6.2.3 Concepts améliorant une politique

Parmi toutes les pistes existantes pour caractériser les concepts au cours de l'apprentissage, nous allons maintenant développer la méthode que nous avons implémentée et expérimentée. Nous présenterons néanmoins partie 6.5 d'autres approches qui ne nous ont pas permis d'obtenir des résultats satisfaisants, mais qui restent importantes pour la compréhension des phénomènes. De plus, elles pourraient servir de support à de futurs algorithmes après la résolution de certaines difficultés.
Nous allons présenter maintenant le critère d'**améliorations des politiques** pour caractériser les concepts. Une démarche similaire, bien que moins générale, peut-être trouvée dans [Jong and Stone, 2005] et rappelée dans [Stone, 2007]. L'idée de base est la suivante : un concept est la généralisation de plusieurs états. Un concept améliore une politique si en utilisant ce concept plutôt que ses spécialisations, pour une même session d'apprentissage, on obtient de meilleurs résultats. Plus formellement :

Définition 6.6 (concept améliorant une politique)
Soit une tâche d'apprentissage par renforcement définie par un PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$, $(L_E, \leq_{L_E}, \otimes_{L_E})$ étant un langage de généralisation pour E et $(L_A, \equiv_{L_A})$ un langage d'équivalence pour A. Soit un algorithme d'apprentissage par renforcement $\mathcal{A}lgo$ estimant la fonction $Q : \psi \longrightarrow \mathbb{R}$.

$Q_{\mathcal{P}}(e,a)$ est la fonction de qualité obtenue par apprentissage telle que tout état $e \in E$ ait été partout remplacé par $[e]_{\equiv_{\mathcal{P}}}$ dans l'algorithme $\mathcal{A}lgo$.
Une partition $\mathcal{P}_1$ est meilleure par rapport à l'amélioration de la politique à l'étape t en utilisant l'algorithme $\mathcal{A}lgo$ qu'une partition $\mathcal{P}_2$ si pour tout couple $(e,a) \in \psi$, $Q_{\pi_{Q_{\mathcal{P}_1}}}(e,a) \geq Q_{\pi_{Q_{\mathcal{P}_2}}}(e,a)$[45].
Soit un concept $c \in \mathfrak{C}$. Soit $\mathcal{P}_c$ la partition définie par :

$$\begin{cases} \text{Si } e \in ext(c) \text{ alors } ext(c) \in \mathcal{P} \\ \text{Sinon } \{e\} \in \mathcal{P} \end{cases}$$

$\mathcal{P}_c$ améliore la politique à l'étape t en utilisant l'algorithme $\mathcal{A}lgo$, si $\mathcal{P}_c$ est meilleure par rapport à l'améliroation de la politique à l'étape t en utilisant l'algorithme $\mathcal{A}lgo$ que $\mathcal{P}_0$.

Pour résumer, un concept améliore une politique pour un algorithme donné pour une session d'apprentissage donnée, s'il permet d'obtenir une estimation de $Q(e,a)$ produisant une meilleure politique que si les états avaient été considérés séparément.
On notera que l'on n'estime pas ici la qualité de convergence de $Q(e,a)$ vers $Q^*(e,a)$, mais directement la politique issue de l'estimation $Q(e,a)$.

Concept améliorant une politique et concepts pertinents Il est facile de constater qu'après convergence d'un apprentissage par renforcement, tous les concepts pertinents sont des concepts améliorant toutes les politiques. En effet, pour toutes les actions optimales, la valeur des concepts pertinents (maximaux ou non) est $Q^*(c,a)$, qui produit une politique optimale si on la suit de façon gloutonne.
Nous soutenons ici que les concepts améliorant une politique sont une bonne approximation des concepts pertinents à un moment donné de l'apprentissage.

Connaissance non monotone Notons que les concepts améliorant une politique constituent une connaissance non monotone. En effet, ils sont à un moment donné de l'apprentissage, la meilleure généralisation, considérant l'objectif à atteindre, c'est à dire l'accumulation des récompenses. Cependant, il est nécessaire que le statut de concept améliorant une politique puisse être une propriété évolutive. En effet, cette notion est dépendante de l'ordre des états de l'environnement rencontrés par l'agent, donc de la dynamique du système ainsi que de la politique de l'agent elle-même. Cependant, il existe un point de convergence que sont les concepts pertinents.

Recherche des concepts améliorant une politique Pour appliquer directement la notion de concepts qui améliorent une politique, il faudrait effectuer des sessions d'apprentissage avec chacun des concepts et comparer les différentes fonctions $Q(e,a)$ obtenues en réévaluant la politique qu'elles engendrent. Ce mécanisme est évidement beaucoup trop désavantageux en terme de temps d'apprentissage. Plutôt que d'appliquer telle quelle cette notion, nous allons, dans la philosophie de l'apprentissage par renforcement (voir partie 2.3.2 et [Sutton and Barto, 1998]), procéder simultanément à l'évaluation de la politique et à son utilisation. C'est à dire que nous allons considérer que la politique $\hat{Q}(e,a)$ courante est une bonne évaluation d'elle-même, c'est à dire de la fonction $Q_{\pi_{\hat{Q}(e,a)}}(e,a)$. Ainsi, c'est par rapport à la politique engendrée par l'évalutation courante de $Q(e,a)$, c'est à dire $\hat{Q}(e,a)$, que les concepts seront classés comme améliorant ou non la politique.

De plus, plutôt que de s'intéresser à toutes les actions définies pour un concept donné, nous ne considérerons que les actions optimales de celui-ci. La propriété $\hat{Q}(c_1,a) \geq \hat{Q}(c_2,a)$ ne sera testée que pour les actions optimales.

6.2.4 Algorithmes

Nous allons présenter dans cette partie les algorithmes découlant des notions que nous venons de présenter : la fonction renvoyant si un concept améliore la politique courante ou non et la procédure de mise à jour de la fonction $\hat{Q}(c,a)$. Nous verrons également un exemple indiquant

[45] Rappelons la notation 2.6 page 33 : π_Q est la politique engendrée par la sélection dans chaque état, de l'action avec la valeur maximum, selon Q. Définition 2.7 page 26 : Q_π est la fonction de qualité engendrée par la politique π.

pourquoi nous différencierons dans la fonction $\hat{Q}(c,a)$ la partie « récompense immédiate » de la partie « récompense différée ».

Algorithme concept améliorant une politique Nous présentons algorithme 6.2 page suivante, la fonction permettant de calculer si un concept améliore la politique ou non. Il reprend les éléments que nous avons déjà mentionnés : il s'agit de rechercher, pour un concept donné, si ses meilleures actions donnent de meilleurs résultats que pour l'ensemble des concepts qui lui sont inférieurs. Comme nous le verrons plus bas, le fait d'utiliser la fonction $\hat{Q}(c,a)$ comme évaluation d'elle même pose un biais problématique venant du fait que la valeur $\hat{Q}(c,a)$ cumul l'estimation de la récompense immédiate ainsi que l'estimation de la récompense à long terme. Ainsi, des couples $\hat{Q}(c,a)$ peuvent converger vers des valeurs supérieures à $Q^*(c,a)$. Nous allons donc effectivement séparer les deux éléments. Ainsi, nous allons diviser la fonction

$$\hat{Q}(c,a) \rightarrow \mathbb{R}$$

en une fonction

$$\hat{Q}_{r_t}(c,a) \rightarrow \mathbb{R}$$

stockant l'estimation de la valeur de la récompense immédiate et une fonction

$$\hat{Q}_{r_{t+1}}(c,a) \rightarrow \mathbb{R}$$

stockant l'estimation de la valeur de la récompense différée.

Techniquement, ce n'est pas compliqué, il suffit d'associer deux valeurs plutôt qu'une seule pour chacune des actions définies pour chacun de concepts. Il est également aisé de retrouver $\hat{Q}(c,a)$ à partir de $\hat{Q}_{r_t}(c,a)$ et $\hat{Q}_{r_{t+1}}(c,a)$ puisque :

$$\hat{Q}(c,a) = \hat{Q}_{r_t}(c,a) + \gamma \hat{Q}_{r_{t+1}}(c,a)$$

Finalement, pour être considéré comme concept améliorant la politique courante, il faudra qu'un concept c réponde aux deux propriétés suivantes :

$$\forall a_j \in arg\ \max_{a'} \hat{Q}(c,a), \forall c_i \in Voisins\_Inférieurs(c)$$
$$\hat{Q}(c,a_j) \geq \hat{Q}(c_i,a_j), \tag{6.2}$$

et

$$\hat{Q}_{r_t}(c,a_j) \geq \hat{Q}_{r_t}(c_i,a_j) \text{ ou } \hat{Q}_{r_{t+1}}(c,a_j) \geq \hat{Q}_{r_{t+1}}(c_i,a_j) \tag{6.3}$$

Autrement dit, il faudra qu'il y ait amélioration globale de la politique (équation 6.2) et que cette amélioration provienne au moins d'une amélioration concernant la récompense immédiate ou de la récompense différée (équation 6.3).

Il faut noter qu'une recherche locale de cette propriété (simple recherche pour les voisins inférieurs) ne suffit pas. En effet, si la fonction $\hat{Q}(c,a)$ converge vers une valeur fixe pour les concepts pertinents, ce n'est pas le cas pour les concepts plus généraux. Il n'y a donc aucune raison pour que le cas suivant ne se produise pas : soit deux concepts n'améliorant pas la politique c_1, c_2 avec $c_2 < c_1$ et tels que pour toute action optimale $a_i \in A^*(c_1)$, $a_j \in A^*(c_2)$, $Q(c_1,a_i) \geq Q(c_2,a_j)$. Il se peut donc que c_1 soit promu comme améliorant la politique de manière infondée.
Ainsi, il faut faire une recherche récursive pour tous les voisins inférieurs (les concepts dont les extensions sont réduites à des singletons constituant la fin de la récursion).

Voyons maintenant l'algorithme de mise à jour de la fonction $\hat{Q}(c,a)$ utilisant la notion de concept améliorant une politique.

Mise à jour de la fonction $\hat{Q}(c,a)$ Supposons que l'agent se trouve au temps t dans l'état e_t et sélectionne l'action a puis se retrouve dans l'état e_{t+1} en ayant reçu la récompense r_t. Nous avons proposé algorithme 6.1 page 128 que l'approximation de la fonction de qualité $\hat{Q}$ soit mise à jour non seulement pour le couple (e_t,a), mais également pour l'ensemble des couples $(concept,a)$,

Données : • PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$ avec un état initial E_0 et un état final E_T
• Un langage de généralisation $(L_E, \leq_{L_E}, \otimes_{L_E})$ pour E
• $0 \leq \gamma \leq 1$, le paramètre de dégressivité
• $0 < \alpha < 1$ le paramètre de remise en cause de la connaissance
• Un concept $c \in \mathfrak{C}$
• $\hat{Q}(c,a)$ l'estimation courante de la fonction de qualité $Q : \psi_{\mathcal{TG}_{(E,L_E)}} \longrightarrow \mathbb{R}$ décomposée en $\hat{Q}_{r_t}(c,a)$ et $\hat{Q}_{r_{t+1}}(c,a)$

Résultat : • `Vrai` si c améliore la politique courante
• `Faux` sinon

début
  si $ext(c)$ *est un singleton* **alors retourner** `Vrai`;
  si c *n'a pas d'action définie* **alors retourner** `Faux`;
  retourner $\forall c_i \in$ Voisins Inferieurs(c), améliorePolitique(c_i) **et**
  $\forall a_j \in arg\ \max_{a'} \hat{Q}(c,a)$, $\forall c_i \in$ Voisins Inférieurs(c), $\hat{Q}(c,a_j) \geq \hat{Q}(c_i,a_j)$ **et**
  $(\hat{Q}_{r_t}(c,a_j) \geq \hat{Q}_{r_t}(c_i,a_j)$ **ou** $\hat{Q}_{r_{t+1}}(c,a_j) \geq \hat{Q}_{r_{t+1}}(c_i,a_j))$;
fin

Algorithme 6.2: Fonction *améliorePolitique(Concept c)* retournant `Vrai` si le concept c améliore la politique au sens donné définition 6.6

pour les concepts tels que $e_t \in ext(concept)$. Nous avons également montré partie 6.2.2 page 134 que sélectionner comme récompense à long terme la valeur $\max_{a'} \hat{Q}(e_{t+1},a)$ posait problème car celle-ci n'est le produit d'aucune généralisation. La question est donc : « que choisir comme valeur espérée à long terme ? »

Nous proposons ainsi un algorithme de mise à jour pour la fonction $\hat{Q}(c,a)$ reposant sur les principes suivants pour le choix du concept c_{t+1} qui fournira la valeur espérée à long terme $\hat{Q}(c_{t+1},a)$ pour la mise à jour de $\hat{Q}(c_{maj},a)$, tel que $e_t \in ext(c_{maj})$:

1. c_{t+1} doit être tel que $e_{t+1} \in ext(c_{t+1})$. C'est à dire que le concept choisi doit couvrir e_{t+1}.
2. c_{t+1} doit être le plus général possible.
3. c_{t+1} doit améliorer la politique.
4. Respect du langage : Si $e_{t+1} \in ext(c_{maj})$ alors $c_{t+1} = c_{maj}$ et si $e_{t+1} \notin ext(c_{maj})$, alors il faut que $c_{maj} \not\subseteq c_{t+1}$. Rappelons que $e_t \in ext(c_{maj})$. Il s'agit donc de spécifier que si e_{t+1} est aussi couvert par c_{maj}, c'est à dire que le concept c_{maj} confond les états de départ et d'arrivée de l'agent, alors celui-ci doit également servir pour sa propre mise à jour. Inversement, si c_{maj} ne couvre pas e_{t+1} alors il faut que le concept choisi comme généralisation de e_{t+1} conserve la séparation entre $ext(c_{maj})$ et e_{t+1}. Cette condition limite certaines généralisations abusives dues au fait que l'on approxime $Q_{\pi_{\hat{Q}}}(c,a)$ par $\hat{Q}(c,a)$.

Les algorithmes 6.3 page suivante et 6.4 page 142 montrent la procédure de mise à jour utilisant ces principes.

- Le principe 1 est respecté ligne 2 ($i(c) \geq_{L_E} d(e_{t+1})$).
- Le principe 2 est respecté car on prend le concept le plus général (Max) ligne 2.
- Le principe 3 est respecté ligne 2 par l'utilisation de la fonction *améliorePolitique(c)*.
- Le principe 4 est respecté d'une part ligne. 1, nous passons outre la condition d'amélioration de la politique, le concept d'arrivé étant confondu avec le concept de départ. D'autre part ligne 2, nous imposons la condition $c_1 \not\geq_{L_E} c$.

Si tous les principes sont respectés, nous choisissons un concept aléatoirement parmi ceux qui conviennent pour ne pas créer de biais.

Maintenant que l'ensemble de la procédure de mise à jour est exposée, voyons un exemple montrant pourquoi il faut séparer la fonction $\hat{Q}(c,a)$ en $\hat{Q}_{r_t}(c,a)$ et $\hat{Q}_{r_{t+1}}(c,a)$.

Données : • PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$ avec un état initial E_0 et un état final E_T
• Un langage de généralisation $(L_E, \leq_{L_E}, \otimes_{L_E})$ pour E
• e_{t+1} l'état observé
• c un concept à préserver
• $\hat{Q}(c,a)$ l'estimation courante de la fonction de qualité $Q : \psi_{\mathcal{TG}_{(E,L_E)}} \longrightarrow \mathbb{R}$

Résultat : • Un concept c_a tel que 1) soit $c_a = c$, 2) soit c_a est un des concepts les plus généraux améliorant la politique courante tout en préservant c.

début
  si $c \neq \emptyset$ **alors**
1 **si** $int(c) \geq_{L_E} d(e_{t+1})$ **alors** $c_a \leftarrow c$;
    sinon
      $c_a \leftarrow$ Choisir un concept aléatoirement parmi
2 $Max\{c',$ tel que $int(c') \geq_{L_E} d(e_{t+1})$ **et non** $c' \geq_{L_E} c$ **et** améliorePolitique$(c')\}$;
    fin
  fin
  sinon
    $c_a \leftarrow$ Choisir un concept aléatoirement parmi
3 $Max\{c',$ tel que $int(c') \geq_{L_E} d(e_{t+1})$ **et** améliorePolitique$(c')\}$;
    fin
  retourner c_a;
fin

Algorithme 6.3: Fonction *Concept concept_Améliore_Et_Préserve*(*Etat* e_{t+1}, *Concept* c) de sélection d'un concept améliorant la politique courante tout en préservant un concept donné

Problème de l'utilisation de $\hat{Q}(c,a)$ à la place de $Q_{\pi_{\hat{Q}(c,a)}}(c,a)$ Le fait de considérer que $\hat{Q}(c,a)$ est une bonne approximation de $Q_{\pi_{\hat{Q}}}(c,a)$, pour rechercher les concepts améliorant une politique (algorithme 6.2 page précédente), implique par le mécanisme que nous allons décrire que des couples $(concept, actions)$ peuvent converger vers une valeur $\hat{Q}(c,a)$ supérieure à $Q^*(c,a)$. Considérons l'exemple suivant pour situer le problème.

Exemple 6.5 *Les états de l'environnement sont tirés de la figure 6.5 page suivante. Supposons qu'au cours des épisodes précédents, l'agent ait estimé la sélection de l'action « Déplacement Nord » dans l'état* e_1 *à* 1 *:* $\hat{Q}(e_1,$ *« Déplacement Nord »*$) = 1$. *C'est sa valeur exacte (*$Q^*(e_1,$ *« Déplacement Nord »*$) = 1$*). L'état suivant est terminal, c'est à dire que l'épisode prend fin.*
L'agent se retrouve au cours d'un épisode suivant dans l'état e_2*, sélectionne l'action « Déplacement Nord » et se retrouve en* e_1.
Le concept c_1 *est tel que* $e_2 \in ext(c_1)$ *et sera donc mis à jour. Admettons que* $Q^*(c_1,$ *« Déplacement Nord »*$) = 1$*, puisque* $Q^*(c_1,$ *« Déplacement Nord »*$)$ *a été mis à jour à chaque fois que* $Q^*(e_1,$ *« Déplacement Nord »*$)$ *a été mis à jour.* $Q^*(e_1,$ *« Déplacement Nord »*$)$ *ou* $Q^*(c_1,$ *« Déplacement Nord »*$)$ *peuvent être indisctinctement choisis comme concept améliorant la politique.* $Q^*(c_1,$ *« Déplacement Nord »*$)$ *étant plus général, c'est néanmoins celui-ci qui sera choisi. La formule de mise à jour est alors :*

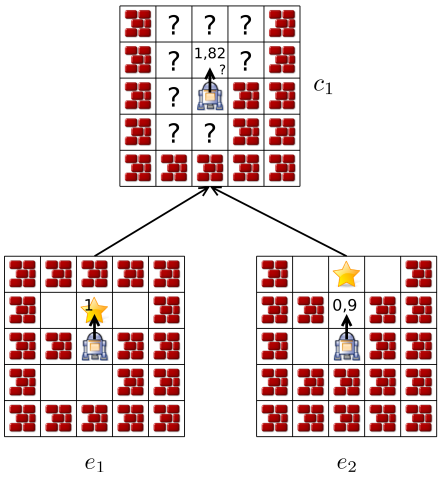
$$\begin{aligned}\hat{Q}(c_1, \text{« } DN \text{ »}) &\leftarrow (1-\alpha)\hat{Q}(c_1, \text{« } DN \text{ »}) + \alpha(r_t + \gamma \max_{a'} \hat{Q}(c_1, \text{« } DN \text{ »})) \\ &\leftarrow 0,1 \times 1 + 0,9(0 + 0,9 \times 1) \\ &\leftarrow 0,91\end{aligned}$$

Ensuite, l'agent se retrouve dans l'état $e1$ *et va de nouveau sélectionner l'action « Déplacement Nord ». Il se trouve que le concept* $c1$ *couvre également* $e1$ *et sera de nouveau mis à jour. Cette*

Données : • PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$ avec un état initial E_0 et un état final E_T
• Un langage de généralisation $(L_E, \leq_{L_E}, \otimes_{L_E})$ pour E
• $0 \leq \gamma \leq 1$, le paramètre de dégressivité
• $0 < \alpha < 1$ le paramètre de remise en cause de la connaissance
• Un concept $c_1 \in \mathfrak{C}$ tel que $e_t \in ext(c)$
• e_{t+1} l'état observé
• r_t la récompense observée
• a_t l'action précédemment sélectionnée
• $\hat{Q}(c, a)$ l'estimation courante de la fonction de qualité $Q : \psi_{\mathcal{TG}_{(E,L_E)}} \longrightarrow \mathbb{R}$

Résultat : • La fonction $\hat{Q}$ mise à jour
début
concept choisi$\leftarrow$ *concept_Améliore_Et_Préserve*(e_{t+1}, c);
$\hat{Q}(c, a_t) \leftarrow (1-\alpha)\hat{Q}(c, a_t) + \alpha\big(r_t + \gamma \max_{a'}\hat{Q}(\text{concept choisi}, a)\big)$;
fin

Algorithme 6.4: Procédure $MAJQ(Concept\ c_1, Etat\ e_{t+1}, Réel\ r_t, Action\ a_t)$ de mise à jour de la fonction $\hat{Q}(c, a)$ utilisant la notion de concept améliorant la politique courante



1) L'agent se retrouve en e_2, choisit l'action a=« Déplacement Nord » et met à jour l'action pour le concept c_1 car $e_2 \in ext(c_1)$. $\max_{a'}\hat{Q}(c_1, a)$ est choisi car il est le concept le plus général améliorant la politique pour l'état e_1.
$\hat{Q}(c_1, a) \leftarrow (1-\alpha)\hat{Q}(c_1, a) + \alpha(r + \gamma \max_{a'}\hat{Q}(c_1, a))$
$\hat{Q}(c_1, a) \leftarrow (1-0,9) \times 1 + 0,9 \times (0 + 0,9 \times 1)$
$\hat{Q}(c_1, a) \leftarrow 0,91$
2) L'agent se retrouve en e_1, choisit l'action a et met à jour l'action pour le concept c_1 car $e_1 \in ext(c_1)$ $\max_{a'}\hat{Q}(c_{e_1}, a)$ est choisi car il est le concept le plus général améliorant la politique pour l'état e_1.
$\hat{Q}(c_1, a) \leftarrow (1-\alpha)\hat{Q}(c_1, a) + \alpha(r + \gamma \max_{a'}\hat{Q}(c_{e_1}, a))$
$\hat{Q}(c_1, a) \leftarrow (1-0,9) \times 0,91 + 0,9 \times (1 + 0,9 \times 1)$
$\hat{Q}(c_1, a) \leftarrow 1,82$

FIG. 6.5 – Exemple de mise à jour faisant en sorte qu'une action pour un concept ait une valeur $\hat{Q}(c, a)$ plus élevée que pour tous ses voisins inférieurs de façon inapropriée.

fois, c'est $Q^*(e_1$, *« Déplacement Nord »*$) = 1$ *qui sera choisi comme concept alémiorant la politique* $(0,91 < 1)$*. L'équation de mise à jour sera donc la suivante :*

$$\begin{aligned}\hat{Q}(c_1, \text{« } DN \text{ »}) &\leftarrow (1-\alpha)\hat{Q}(c_1, \text{« } DN \text{ »}) + \alpha(r_t + \gamma \max_{a'}\hat{Q}(e_1, \text{« } DN \text{ »}))\\ &\leftarrow 0,1 \times 0,91 + 0,9(1 + 0,9 \times 1)\\ &\leftarrow 1,801\end{aligned}$$

On a alors $\hat{Q}(c_1$, *« Déplacement Nord »*$) > Q^*(e_1$, *« Déplacement Nord »*$)$ *et* $\hat{Q}(c_1$, *« Déplacement Nord »*$) > Q^*(e_2$, *« Déplacement Nord »*$)$.

Que se passe-t-il sur l'exemple que nous venons de voir ? Le concept c_1 couvrant à la fois un état où l'agent reçoit une récompense et un état menant à un état recevant une récompense, la valeur de c_1, « Déplacement Nord » inclut à la fois la récompense et la promesse de récompense. Alors que l'agent n'aura effectivement pas les deux. Remarquons que c'est bien ce concept qui sera choisi pour une mise à jour par la fonction améliore politique (algorithme 6.4) car on a bien $\hat{Q}(c_1$, « Déplacement Nord »$) > \hat{Q}(e_1$, « Déplacement Nord »$)$ et $\hat{Q}(c_1$, « Déplacement Nord »$) >$

$\hat{Q}(e_2, \text{« Déplacement Nord »})$.

Est-ce vraiment problématique ? La question se pose en effet, puisque du point de vue de la sélection des actions, l'action optimale est préservée, le problème vient de sa valeur. Ceci reste bien un problème car c'est précisément cette valeur erronée qui va être utilisée de proche en proche pour évaluer les actions des états menant dans un des états couverts par le concept. Ainsi, cette valeur erronée va se diffuser par le mécanisme de mise à jour au fur et à mesure de l'apprentissage.

Pour résoudre la question, nous pourrions considérer qu'il faut que les actions soient non seulement les mêmes (au sens de même étiquette), mais également qu'elles aient même valeur (ce qui revient à la notion de bissimilarité de [Givan et al., 2003]). Cependant, il est difficile en cours d'apprentissage de différencier ce cas de celui d'un concept dont la valeur de l'action optimale a déjà convergé vers une valeur a^* alors que pour les états qu'il généralise, la valeur de la même action n'a pas encore convergé et est inférieure. Il faut pouvoir estimer le « degrés de convergence » d'une fonction dont on connaît peu de choses, ce qui reste une question ouverte dans nos travaux.

La surévaluation venant du fait que l'on cumule dans la fonction $\hat{Q}(c, a)$ (comme on le fait classiquement en apprentissage par renforcement) la récompense immédiate et espérée à venir, Comme nous l'avons montré précédemment, nous avons tout simplement séparer ces deux fonctions. Ainsi, dans l'exemple que nous venons de proposer, nous avons $\hat{Q}(c_1, \text{« Déplacement Nord »}) > \hat{Q}(e_1, \text{« Déplacement Nord »})$ et $\hat{Q}(c_1, \text{« Déplacement Nord »}) > \hat{Q}(e_2, \text{« Déplacement Nord »})$. Cependant, nous avons également :

- $Q^*_{r_t}(e1, \text{« Déplacement Nord »}) = 1$
- $Q^*_{r_{t+1}}(e1, \text{« Déplacement Nord »}) = 0$
- $Q^*_{r_t}(e2, \text{« Déplacement Nord »}) = 0$
- $Q^*_{r_{t+1}}(e1, \text{« Déplacement Nord »}) = 0,9$
- $\hat{Q}_{r_t}(c_1, \text{« Déplacement Nord »})$ ne converge pas mais $\hat{Q}_{r_t}(c_1, \text{« Déplacement Nord »}) < 1$, la valeur dépend de la distribution des états[46].
- $\hat{Q}_{r_{t+1}}(c_1, \text{« Déplacement Nord »})$ ne converge pas mais $\hat{Q}_{r_{t+1}}(c_1, \text{« Déplacement Nord »}) < 0,9$, la valeur dépend de la distribution des états.

Finalement, le concept c_1 ne sera considéré comme améliorant la politique ni dans l'état e_1, ni dans l'état e_2.

Maintenant que nous avons établi la fonction $Q : \psi_{\mathcal{TG}_{(E,L_E)}} \longrightarrow \mathbb{R}$ permettant à l'agent de stocker la récompense estimée à long terme pour chacun des états de l'environnement ainsi que pour chacune des généralisations permises par le langage de description L_E, que nous avons indiqué comment celle-ci était mise à jour en fonction des observations successives des états de l'environnement ainsi que des récompenses, il nous reste à proposer une méthode permettant à l'agent d'utiliser $Q(c, a)$ *en ligne* pour la sélection des actions.

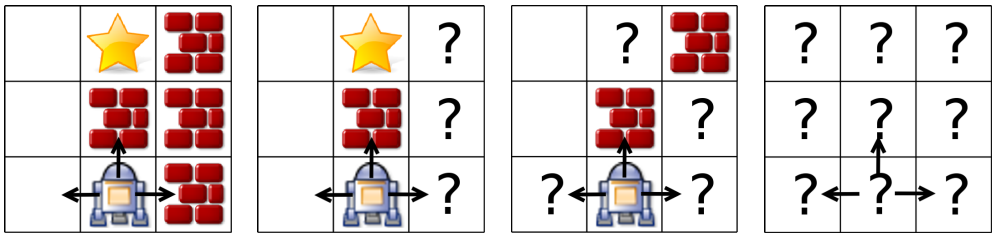
6.3 Sélection des actions

Nous allons dans cette partie finaliser l'algorithme d'apprentissage par renforcement proposant une fonction de sélection des actions basée sur la fonction $Q : \psi_{\mathcal{TG}_{(E,L_E)}}$ et prenant en compte ses capacités de généralisation. Notons que cette sélection se fait *en ligne*, c'est à dire que l'on peut conjointement approximer la fonction $\hat{Q}(c, a)$ et utiliser cette approximation. Il n'y a donc pas de séparation entre une phase d'apprentissage et une phase d'utilisation de l'apprentissage[47].

Pour l'apprentissage par renforcement classique, une fois les principaux paramètres d'apprentissage réglés : la balancement entre exploration et exploitation (par exemple avec le paramètre α de l'équation 2.15 page 37 de *Q-Learning*) et le rythme de révision de sa connaissance (par exemple avec le paramètre γ), la sélection d'action se fait de façon gloutonne par rapport à la valeur de qualité estimée des actions (voir paragraphe 2.4.5 page 41).
Compte tenu de la structure que nous venons de décrire, l'agent doit gérer un nouvel élément : le niveau de généralisation. En effet, chaque état e peut-être interprété par l'agent de façon plus

[46] voir partie 6.1.2
[47] voir partie 2.4.4

ou moins générale. En fait, il peut-être vu comme l'un parmi tous les éléments du langage qui couvrent e.



(a): Description de l'état (b): Généralisation 1 (c): Généralisation 2 (d): Généralisation la plus générale

FIG. 6.6 – Exemple de la description d'un état (figure (a)) et de trois généralisations possibles de celui-ci. Intuitivement, la généralisation 2 (figure (c)) ainsi que la généralisation la plus générale (figure (d)) ne permettent pas à l'agent de sélectionner une action optimale, car elles masquent une information importante (la présence de la case récompense).

Exemple 6.6 *Reprenons l'exemple sur lequel nous avons effectué l'expérimentation dans le chapitre précédent : partie 5.4 page 119. La figure 6.6 représente pour un état de l'environnement e_1 observé 4 généralisations (description supérieure ou égale à la description de l'état) parmi l'ensemble des généralisations possibles pour l'état e_1. La figure (a) montre la généralisation de l'état la plus spécifique (la description de l'état elle-même) et la figure (d) la généralisation la plus générale. Les figure (b) et (c) représentent des généralisations intermédiaires. Intuitivement, on estimerait que seules les généralisations (a) et (b) permettent de sélectionner optimalement une action (« déplacement ouest »), alors que les généralisations (c) et (d) ne le permettent pas car il manque des éléments importants pour la décision, notamment la localisation de la récompense. De plus, la généralisation (b) est plus intéressante que la généralisation (a) car elle est applicable à d'autres états de l'environnement.*

Le paradigme introduit par notre méthode est le suivant : les concepts les plus généraux sont plus visités que les concepts plus spécifiques (en particulier les concepts dont l'extension se réduit à un seul état comme pour l'apprentissage par renforcement classique). Les valeurs des actions pour ceux-ci sont donc globalement mieux connues, puisque bénéficiant d'un échantillonnage plus fréquent.
Cependant, certains concepts sont trop généraux par rapport à la tâche en cours, en particulier, ceux qui regroupent des états ayant des actions optimales différentes. Il faut donc trouver un compromis entre une certaine généralisation permettant un transfert de l'apprentissage d'états fréquemment visités vers des états peu visités et ne pas surgénéraliser.

En fait, nous avons déjà traité ce compromis dans la partie précédente. Nous allons donc simplement pour la sélection des actions utiliser la même méthode que nous avons présentée pour la sélection des concepts pour la mise à jour de $\hat{Q}(c,a)$, c'est à dire la sélection des concepts *améliorant la politique courante*. En effet, la notion de concept améliorant une politique permet, relativement à la politique courante, de trouver les concepts les plus généraux permettant de maximiser la succession des récompenses. Bien entendu, cela se fait toujours au prix de l'heuristique considérant que $\hat{Q}(c,a)$ est une bonne approximation de $Q_{\pi_{\hat{Q}}}(c,a)$.

L'algorithme 6.5 page ci-contre indique la procédure de sélection d'une action pour un état donné.

Proposons maintenant une validation expérimentale de notre algorithme appliquant les méthodes de *Q-Concept Learning*.

Données : • PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$ avec un état initial E_0 et un état final E_T
• Un langage de généralisation $(L_E, \leq_{L_E}, \otimes_{L_E})$ pour E
• e_t l'état observé
• $\hat{Q}(c,a)$ l'estimation courante de la fonction de qualité $Q : \psi_{\mathcal{TG}_{(E,L_E)}} \longrightarrow \mathbb{R}$

Résultat : • Une action a_t telle que $e_t \times a_t \in \psi$ et (c, a_t), $e_t \in ext(c)$ soit un des concepts les plus généraux améliorant la politique courante.

début

concept choisi $\leftarrow$ *concept_Améliore_Et_Préserve*$(e_t, \emptyset)$;
retourner aléatoirement une des actions parmi $\{arg\ \underset{a}{max}\hat{Q}(concept\ choisi, a)\}$;

fin

Algorithme 6.5: Fonction *Action selectionAction*(*Etat* e_t) de sélection d'une action avec la fonction $\hat{Q}(c,a)$ en utilisant la notion de concept améliorant la politique courante

6.4 Expérimentation

Nous allons maintenant proposer une validation expérimentale de notre algorithme. Celle-ci a pour but de prouver la possibilité de l'implémenter ainsi que de montrer son exécution sur un cas simple. Pour ce faire, nous allons l'appliquer sur un problème similaire à celui que nous avons proposé chapitre 5. La fonction $Q(c,a)$ est stockée sous forme de treillis (voir schéma 6.1 page 127). Celui-ci est construit de façon incrémentale. L'agent ne connaît donc pas au départ d'un apprentissage, l'ensemble des états de l'environnement ainsi que l'ensemble des actions possibles. La fonction $Q(c,a)$ n'inclut donc à un moment donné de l'exécution que le treillis de Galois $\mathcal{TG}(E_t, L_E)$, E_t étant l'ensemble des états rencontrés par l'agent.

Tout d'abord, nous montrerons comment dans un cas simple, mais possédant de nombreux états de l'environnement différents, notre algorithme fait en sorte que l'agent généralise sa politique au cours de son apprentissage, permettant ainsi d'obtenir de bons résultats, même sur des états inconnus, là où un apprentissage par renforcement classique doit tout réapprendre. Deuxièmement nous introduirons la présence d'erreur possible par généralisation abusive. Nous verrons alors comment notre algorithme se comporte. Finalement, nous montrerons que bien que notre algorithme soit utilisable sans une différentiation d'une phase d'exploration et d'une phase d'exploitation, il est intéressant de présenter à l'agent quelques exemples significatifs afin d'accélérer l'apprentissage des généralisations intéressantes pour la politique.

6.4.1 Algorithme

Pour toutes les expériences suivantes, la fonction d'amélioration des politiques 6.4 page 142, la procédure de mise à jour de la fonction $Q(c,a)$ 6.4 page 142 ainsi que la fonction de sélection des actions 6.5 ont été utilisées pour l'algorithme de *Q-Concept Learning* 6.6 page suivante.

Notons qu'il existe des différences par rapport à l'algorithme 6.1 page 128 que nous avons introduit en premier.

Premièrement, nous avons y avons inclu les notions de concept améliorant la politique développée tout au long du chapitre.

Deuxièmement, nous utilisons des pseudos méthodes de traces d'éligibilité adaptées à notre contexte. On se référera à [Sutton and Barto, 1998] pour le détail des traces d'éligibilité.
Pour expliquer rapidement, il s'agit d'appliquer une expérience non pas simplement au dernier couple $(état, action)$ sélectionné, mais à l'ensemble des couples $(état, action)$ sélectionnés depuis le début de l'épisode (ligne 4 de l'algorithme 6.6). Sémantiquement, cela revient à attribuer une récompense non pas à l'action qui vient d'être sélectionnée, mais bien à l'ensemble des actions précédentes de l'épisode. Classiquement, on dégrade cette récompense transmise en fonction de

Données : • PDM décrit $M\langle E, L_E, A, L_A, \psi, P, R\rangle$ avec un état initial E_0 et un état final E_T
• Un langage de généralisation $(L_E, \leq_{L_E}, \otimes_{L_E})$ pour E
• $0 \leq \gamma \leq 1$, le paramètre de dégressivité temporelle
• $0 < \alpha < 1$ le paramètre de remise en cause de la connaissance
• $0 < \epsilon < 1$ le paramètre d'exploration aléatoire

Résultat : • $\pi^*(e, a) \in \Pi^*$, une politique optimale pour M

début

Initialisation de la politique
Initialiser $\hat{Q}_{r_t}(c, a) = v_{r_t 0} \in \mathbb{R}$, $\forall (c, a) \in \psi_{\mathcal{TG}_{(E,L_E)}}$;
Initialiser $\hat{Q}_{r_{t+1}}(c, a) = v_{r_{t+1} 0} \in \mathbb{R}$, $\forall (c, a) \in \psi_{\mathcal{TG}_{(E,L_E)}}$;
pour *Chaque épisode* **faire**
  $file\_couples\_(état, action) \leftarrow \emptyset$;
  $e_t \leftarrow E_0$;
  tant que $e_t \neq E_T$ **faire**
    si $tirage\_Aléatoire() > \epsilon$ **alors**
      $a_t \leftarrow ActionDisponibleAléatoire(e_t)$;
1     $file\_couples\_(état, action) \leftarrow \emptyset$;
    fin
    sinon
      $a_t \leftarrow selectionAction(e_t)$;
    fin
    appliquer a_t;
    $r_t \leftarrow$ recevoir une récompense;
    $e_{t+1} \leftarrow$ recevoir un nouvel état;
2   **pour** $c_j \in \mathfrak{C}_{(E,L_E)}$ *tels que* $e_t \in ext(c_j)$ **faire**
      $\hat{Q}_{r_t}(c_j, a_t) \leftarrow (1 - \alpha)\hat{Q}_{r_t}(c_j, a_t) + \alpha.r_t$;
      $concept\ choisi \leftarrow concept\_Améliore\_Et\_Préserve(e_{t+1}, c_j)$;
      $\hat{Q}_{r_{t+1}}(c_j, a_t) \leftarrow (1 - \alpha)\hat{Q}_{r_{t+1}}(c_j, a_t) + \alpha.\max_{a'}\hat{Q}(concept\ choisi, a)$;
    fin
3   **si** $(e_{t+1}, *) \in file\_couples\_(état, action)$ **alors**
      $file\_couples\_(état, action) \leftarrow \emptyset$;
    fin
4   **pour** $(e_i, a_i) \in file\_couples\_(état, action)$ **faire**
      pour $c_j \in \mathfrak{C}_{(E,L_E)}$ *tels que* $e_i \in ext(c_j)$ **faire**
        $concept\ choisi \leftarrow concept\_Améliore\_Et\_Préserve(e_i, c_j)$;
        $\hat{Q}_{r_{t+1}}(c_j, a_i) \leftarrow (1 - \alpha)\hat{Q}_{r_{t+1}}(c_j, a_i) + \alpha.\max_{a'}\hat{Q}(concept\ choisi, a)$;
      fin
    fin
    ajouter en tête (e_t, a_t) à $file\_couples\_(état, action)$;
    $e_t \leftarrow e_{t+1}$;
  fin
fin
fin

Algorithme 6.6: *Q-Concept learning* avec pseudo traces d'éligibilités

l'éloignement dans le temps.
Cependant, il est néfaste dans notre cas d'attribuer telle quelle la récompense à l'ensemble des couples $(état, action)$ précédents. En effet, les concepts les plus généraux bénéficieraient de nombreuses fois de la même récompense, empirant les effets présentés partie 6.2.4. Nous allons donc utiliser une méthode inspirée et plutôt que de distribuer la récompense, nous allons indiquer aux couples (état,action) précédents (ainsi qu'à leurs généralisations) de se remettre à jour si l'action avec une valeur maximale pour l'état suivant a été modifiée. Par conséquent :

- Nous ne répercutons les changements d'une mise à jour que si c'est effectivement la valeur de l'action optimale qui a été mise à jour pour l'état suivant (ligne 1 de l'algorithme).
- Nous ne mettons à jour que la partie « estimation de la récompense différée » de la fonction $Q(c, a)$, c'est à dire $Q_{r_{t+1}}(c, a)$ pour les états où l'on ne fait que répercuter l'expérience courante.
- Par contre, pour le couple $(état, action)$ qui vient d'être sélectionné, les deux parties $Q_{r_t}(c, a)$ et $Q_{r_{t+1}}(c, a)$ sont mises à jour (ligne 2 de l'algorithme). Ceci revient à la mise à jour classique développée algorithme 6.4.
- Finalement, l'environnement étant markovien, c'est à dire indépendant du chemin suivi, lorsque l'agent repasse deux fois au même endroit, toute la séquence de couples $(état, action)$ précédente est supprimée (ligne 3 de l'algorithme).

6.4.2 Détails techniques

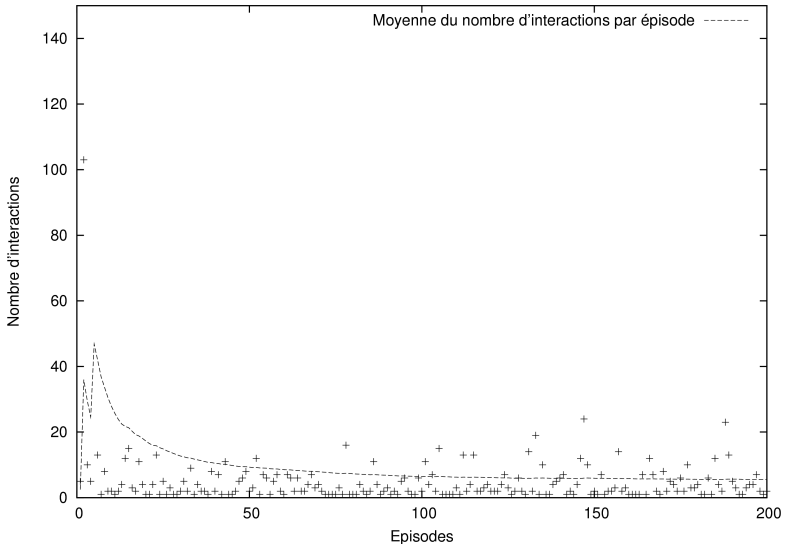
- L'environnement dans lequel évolue l'agent est le même que celui décrit partie 5.4 page 118. A chaque épisode, un des états est sélectionné aléatoirement. Un épisode se termine lorsque l'agent arrive sur la case « récompense ».
- L'algorithme est programmé dans le langage de programmation Java, exécuté dans l'environnement d'exécution Java JRE 1.6.0.10 (voir [ref]).
- L'algorithme est utilisé dans le cadre du programme MadKit version 3.1b5 (voir [ref]). Celui-ci sert également d'outils de visualisation des déplacements de l'agent, notamment en utilisant le module TurtleKit ([ref]).
- Les graphiques présentés sont produits par sauvegardes dans un fichier des différentes valeurs puis traités à l'aide du logiciel Gnuplot ([ref]).
- Nous comparons notre algorithme à un algorithme classique d'apprentissage par renforcement (*Q-Learning*, [Sutton and Barto, 1998]), ainsi qu'à une marche aléatoire.

On pourra noter que le langage de description ne contient pas des éléments de langage cruciaux comme l'orientation de la récompense. Pour avoir une solution du problème minimale, il faudrait pouvoir exprimer « aller dans la direction où se trouve la récompense », ce qui nécessite un langage structurel. Ceci serait parfaitement possible. Il faudrait alors définir l'opérateur de généralisation, par exemple par abstraction de variables comme dans [Dzeroski et al., 2001] ou bien un opérateur de « moins générale des généralisation » (*lgg*).

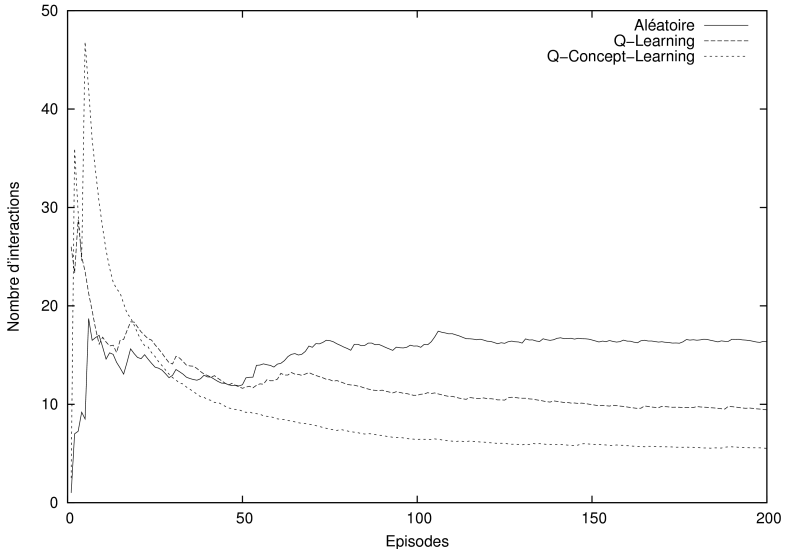
Notre objectif n'est pas ici de résoudre le problème posé, mais plutôt de montrer les possibilités de notre algorithme sur un cas d'école de l'apprentissage par renforcement.

6.4.3 Expérience 1

Dans la première expérience, il y aura positionné de façon aléatoire : l'agent, un mur et une récompense sur une grille de 9 cases. L'agent pourra dans chacune des cases sélectionner une action parmi { *« Déplacement Nord »*,*« Déplacement Ouest »*,*« Déplacement Sud »*,*« Déplacement Est »*,*« Pousser Nord »*,*« Pousser Ouest »*,*« Pousser Sud »*,*« Pousser Est »* }. Les actions *« Déplacement »* sont disponibles si la cases correspondant à l'orientation ne contient pas de mur. Sinon, c'est une action *« Pousser »* qui sera disponible.
Il y a 576 états de l'environnement avec 4 actions pour 504 d'entre eux (les autres étant terminaux). On a donc $|E| = 576$, $|\psi| = 2016$. Le treillis des parties de E est de taille $|\mathcal{P}(E)| = 2^{576}$.



(a): Durée des épisodes et moyenne de la durée des épisodes pour le *Q-Concept Learning*



(b): Comparaison de la moyenne de la durée des épisodes pour une marche aléatoire, le *Q-Learning* et le *Q-Concept Learning*

FIG. 6.7 – Durée des épisodes pour 200 épisodes avec une grille de 9 cases contenant un agent, un mur et une récompense

Comparaison de la durée des épisodes Figure 6.7 (b) page ci-contre, nous voyons la comparaison de la moyenne de la durée des sessions entre une marche aléatoire (pour référence) un algorithme de *Q-Learning* classique et notre algorithme de *Q-Concept-Learning*.
Les graphiques montrent les performances de 200 épisodes tirés aléatoirement (néanmoins, les mêmes épisodes ont été utilisés et dans le même ordre pour les trois algorithmes).

Nous voyons que l'algorithme de *Q-Concept-Learning* apprend plus vite. En effet, la durée des épisodes diminue rapidement.
En fait, on peut voir plusieurs phases durant l'apprentissage (figure 6.7 (a) page précédente). Durant les premiers épisodes, on voit un épisode hors normes pour lequel le temps d'apprentissage est très long. En fait, durant cet épisode, l'agent a une connaissance très parcellaire car il n'a trouvé que peu de récompenses. Beaucoup de concepts très généraux ont eu la valeur de leurs actions modifiée par celles-ci alors que des contres exemples (c'est à dire des états également couverts, mais avec une valeur plus faible) n'ont pas été rencontrés. L'agent essaie d'appliquer les actions qui l'ont mené la première fois sur la case récompense et en fait « tourne en rond ». L'exploration aléatoire le fait généralement sortir de ce comportement.
De toute façon, comme nous avons choisi une valeur positive comme valeur d'initialisation pour la fonction $Q(c, a)$, l'agent sort nécessairement de ce comportement. En effet, la valeur des actions pour les concepts choisis baisse à chaque interaction étant donné que la récompense effectivement constatée est nulle. Il se trouve donc nécessairement un pas où la valeur d'initialisation devient supérieure aux valeur des actions pour les actions infructueuses, amenant l'agent à changer d'action.

Des séquences où l'agent « tourne en rond » se reproduisent régulièrement mais de façon totalement hors normes comme au début. C'est la deuxième phase. Autrement, l'agent sélectionne presque systématiquement les actions optimales (ce qui fait chuter rapidement la moyenne de la durée des sessions). On remarquera que plus tard dans l'apprentissage, on retrouve encore des épisodes où l'agent « tourne en rond ». Il s'agit de moments où des états relativement différents sont introduits. Le fonctionnement de l'algorithme amène à faire une sorte de tri parmi les concepts généralisant les états avec des généralisations peu ou pas connues.

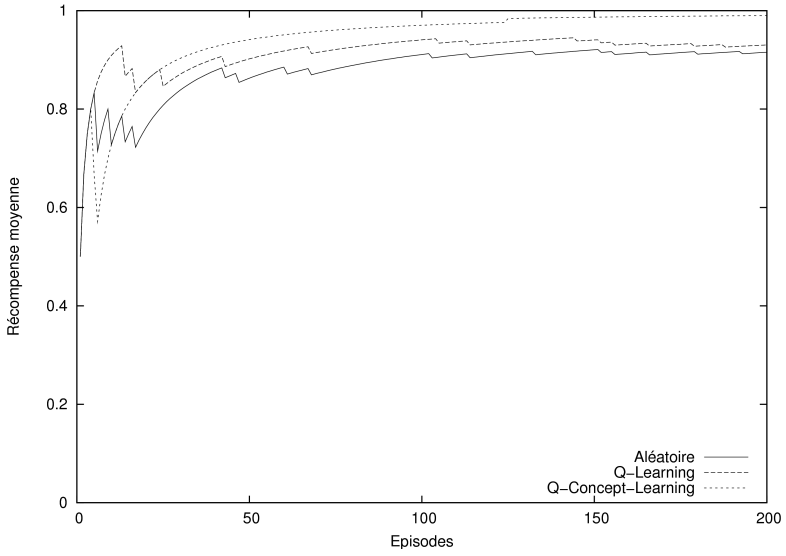


FIG. 6.8 – Comparaison de la moyenne des récompenses par épisode pour une marche aléatoire, le *Q-Learning* et le *Q-Concept-Learning*

Comparaison des récompenses Algorithme 6.8, nous voyons une différence importante dans les moyennes des récompenses obtenues par épisode. En effet, en l'absence de connaissance de l'environnement, le *Q-Learning* est de fait exploratoire, surtout en utilisant des valeurs

d'initialisation supérieures à 0. Ainsi, il arrive fréquemment que l'agent mette fin à l'épisode en détruisant la récompense en poussant un mur dessus.
Dans le cas de notre algorithme, à part au tout début de l'apprentissage et pour les actions exploratoires dues à la politique $\epsilon - gloutonne$, ceci n'arrive jamais. En effet, les actions sélectionnées sont les actions applicables dans des états ou la récompense différée est effective. L'action de *« Pousser »* n'amène jamais de récompense immédiate et est toujours inférieure à au moins une action de déplacement si on considère la récompense à long terme. Ainsi, l'agent termine l'épisode quasi systématiquement en ayant obtenu la récompense.

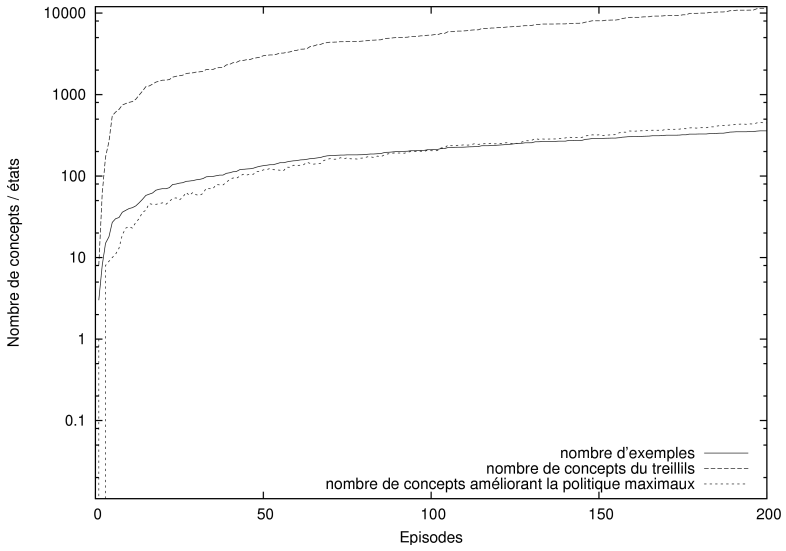


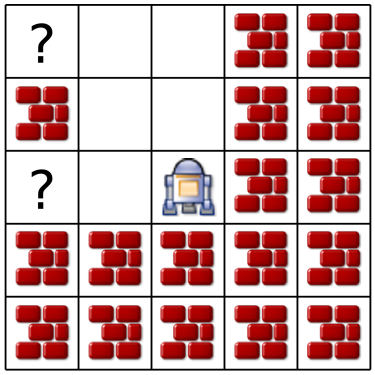
FIG. 6.9 – Taille du treillis, nombre d'exemples et nombre de concepts améliorants maximaux (échelle logarithmique)

Taille de la fonction $Q(c, a)$ Concernant la taille du treillis, c'est à dire la taille de la fonction $Q(c, a)$, il est visible sur la figure 6.9 que le nombre de concepts croît de façon relativement rapide (l'échelle du graphique est logarithmique). Rappelons que la taille du treillis de Galois est égale à $min(2^{|E|}, |L|)$. Ici, la structure du langage est assez lâche, amenant à un grand nombre de concepts (11753 au bout de 200 épisodes sur 405 états de l'environnement observés). Cependant, nous n'avons pas choisi ce langage pour sa commodité par rapport à la tâche, mais inversement car il correspond à une description assez basique de l'environnement. Nous voulons montrer par là la possibilité d'émergence de concepts intéressants, malgré ce type de description.

En revanche, le nombre de concepts effectivement utilisés pour générer la politique, c'est à dire les concepts les plus généraux améliorant la politique sont du même ordre que le nombre d'états de l'environnement rencontrés (au bout de 200 épisodes, 405 états de l'environnement rencontrés pour 433 concepts améliorant la politiques maximaux). Ainsi, on peut dire que si on décide d'arrêter l'apprentissage à un moment donné, on peut décrire les états de l'environnement à l'aide de ces concepts, ce qui ne prend pas plus en mémoire. Non seulement ceux-ci assurent une politique meilleure qu'en considérant les états séparément, mais en plus, ils permettent également d'avoir un comportement sur les exemples non vus.

Exemples de concepts trouvés La figure 6.10 page ci-contre propose 5 exemples de concepts parmi les 433 concepts améliorants maximaux. Voici quelques exemples parmi les 433 concepts améliorants maximaux au bout de 200 épisodes.

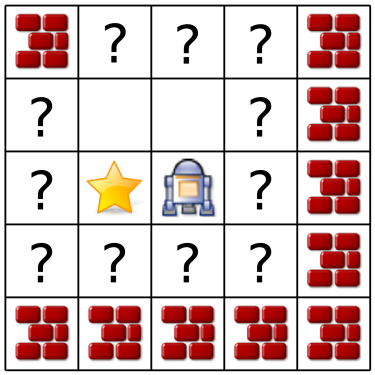
- Le concept c_2 est un des concepts parmi les plus généraux. Son extension contient 14 états de l'environnement. La valeur de son action optimale (*« Déplacement Ouest »*) est exactement



Etats : 2

c_1

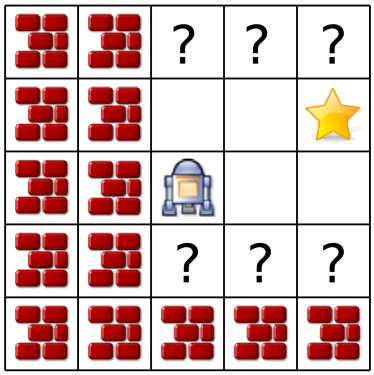
D Nord : 0,419 (?) D Ouest : 0,763 (?)
P Sud : 0,173 (?) P Est : 0,173 (?)



Etats : 14

c_2

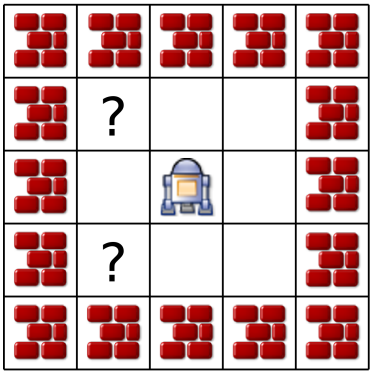
D Nord : 0,809 (0,81) D Ouest : 1(1)



Etats : 3

c_3

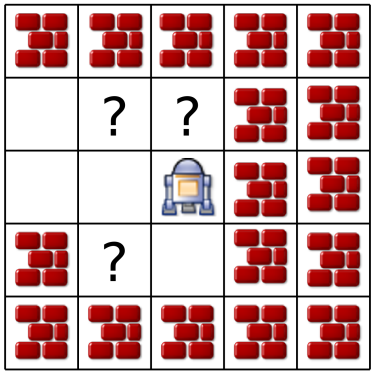
D Nord : 0,809 (0,81) P Ouest : 0,635 (0,729)
D Est : 0,76 (0,81)



Etats : 2

c_4

D Nord : 0,83 (?) D Ouest : 0,878 (0,9)
D Sud : 0,735 (?) D Est : 0,774 (0,729)



Etats : 3

c_5

D Nord : 0,919 (?) D Ouest : 0,902 (?)
D Sud : 0,743 (?) P Est : 0,19 (?)

FIG. 6.10 – Sélection de 5 concepts améliorants maximaux parmi les 433 existants avec les valeurs de leurs actions disponibles. Les valeurs entre parenthèses sont les valeurs pour une politique optimale. Un (?) signifie que l'action n'est pas pertinente. *Etats* : n correspond à la taille de l'extention du concept.

connue et la valeur de son autre action disponible est connue à 10^{-2} près. C'est un concept pertinent et donc son intention pourrait faire partie de la partition $\mathcal{P}^*$. De même, le concept c_3 est un concept pertinent. Les valeurs de ses actions sont un peu moins bien approximées car il regroupe moins d'états dans son extension.

- Le concept c_1 est un concept non pertinent. En effet, aucune de ses actions ne convergera vers une valeur fixe. Néanmoins, ses actions trouvées comme optimales le sont effectivement. En revanche, le concept c_5 est non pertinent et en plus, il n'a pas d'actions optimales. Il est encore considéré comme améliorant la politique courante par manque d'exploration. C'est le genre de concept qui fait que l'agent « tourne en rond » de temps à autre.
- Le concept c_4 est intéressant car il est pertinent, mais la récompense ne fait pas partie de sa description.

Voyons maintenant une deuxième expérience dans laquelle l'agent peut faire des généralisations « piégeantes ».

6.4.4 Expérience 2

Dans la première expérience, l'agent pouvait faire des généralisations non appropriées, voire abusives l'amenant à sélectionner des actions sous-optimales. Ici, nous rajoutons la possibilité de généralisations néfastes. En effet, certaines cases constituent des malus. Chaque sélection d'une action amenant dans une telle case rapporte une *« récompense »* de -1. Le malus disparaît s'il est détruit par un mur ou si l'agent de déplace sur sa case.
Le malus est une sorte de piège pour l'agent. En effet, une action *« Déplacement »* n'était pas disponible quand l'agent avait un mur dans une direction donnée. Ainsi, la généralisation entre un mur et une récompense ne donnait pas lieu à un concept avec une action admissible. En revanche, ici, un *« malus »* et une *« récompense »* deviennent généralisables en *« case »* , vers laquelle un *« Déplacement »* est possible.

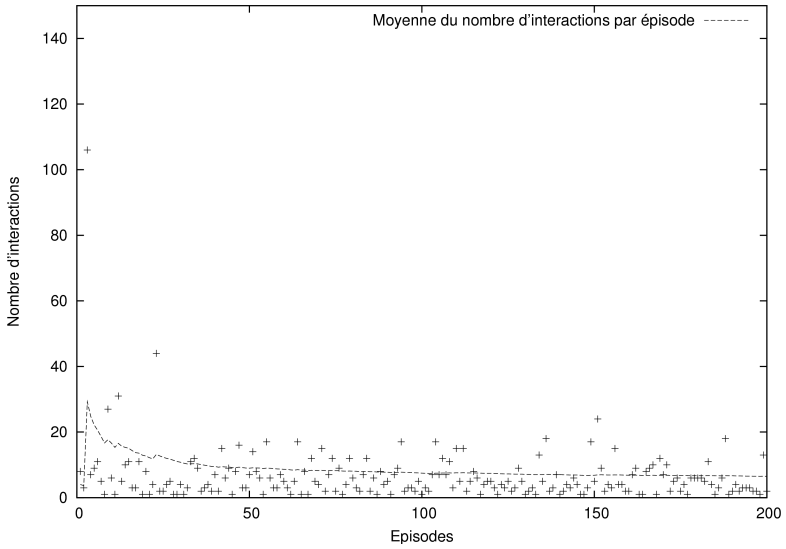
Nous reproduisons donc l'expérience précédente en positionnant de façon aléatoire : l'agent, un mur, une récompense, un malus, sur une grille de 9 cases.

Il y a 4104 états de l'environnement avec 4 actions pour 3528 d'entre eux (les autres étant terminaux). On a donc $|E| = 4104$, $|\psi| = 16416$. Le treillis des parties de E est de taille $|\mathcal{P}(E)| = 2^{4104}$.

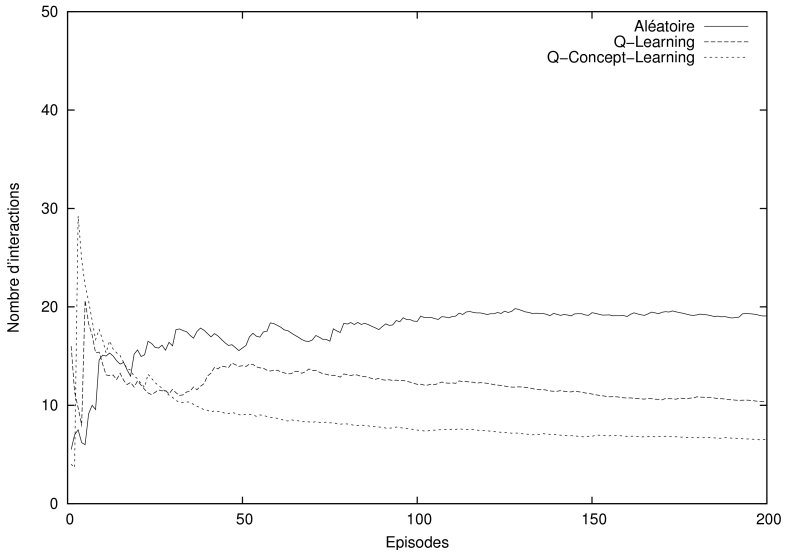
Des constats similaires Les commentaires que nous avons faits pour la première expérience restent valables pour celle-ci. Le nombre d'interactions par épisode reste inférieur pour l'algorithme de *Q-Concept-Learning* (figure 6.11 (b) page suivante). La différence entre les récompenses (figure 6.12 page 154) est accentuée étant donné qu'il y a maintenant des récompenses négatives.

De même, on retrouve les phases pour l'algorithme de *Q-Concept-Learning* qui « tourne en rond » (figure 6.11 (a) page suivante). La taille du treillis ainsi que le rapport entre le nombre d'états et de concepts améliorants maximaux reste également identique (figure 6.13 page 155).

Nous avons proposé cette expérimentation pour montrer les généralisations abusives qui pouvaient être faites compte tenu du langage de description des états envisagé. En effet, dans l'expérience précédente, l'agent ne pouvait pas sélectionner l'action « Déplacement » quand l'action amenait sur une case contenant un mur. Ceci amenait au fait que le concept généralisant deux état semblables mais intervertissant une récompense et un mur ne pouvait pas être considéré comme améliorant, étant donné que leurs actions étaient différentes. Inversement, ici, un malus et une récompense ne modifient pas les actions disponibles. Ainsi, la généralisation de deux états inversant une récompense et d'un malus produit un concept pouvant être considéré par l'agent. Pire, si l'état avec la récompense est rencontré en premier, quand l'agent ira pour la première fois dans celui avec un malus au même endroit, celui-ci sélectionnera prioritairement l'action menant vers le malus. Inversement, si l'action amenant au malus est rencontrée en premier, quand l'agent rencontrera le même état avec une récompense à la place, une autre action sera préférée.



(a): Durée des épisodes et moyenne de la durée des épisodes pour le *Q-Concept Learning*



(b): Comparaison de la moyenne de la durée des épisodes pour une marche aléatoire, le *Q-Learning* et le *Q-Concept Learning*

FIG. 6.11 – Durée des épisodes pour 200 épisodes avec une grille de 9 cases contenant un agent, une récompense et un malus

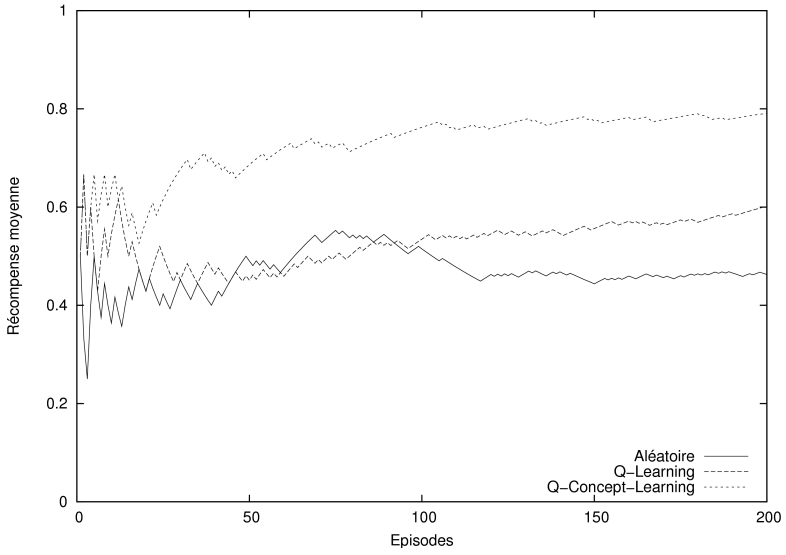


FIG. 6.12 – Comparaison de la moyenne des récompenses par épisodes pour une marche aléatoire, le *Q-Learning* et le *Q-Concept-Learning*

Spécificité de certains concepts Étudions quelques uns des 431 concepts améliorant maximaux présents dans le treillis (figure 6.14 page 156). On retrouve bien évidemment des concepts comme le concept c_2, assez généraux et donc regroupant de nombreux états dans son extension. C'est d'ailleurs un concept pertinent. On retrouve également des concepts non pertinents, mais dont les actions trouvées comme optimales sont effectivement optimales comme pour le concept c_2. Plus intéressant, on retrouve des concepts comme le concept c_3 qui indiquent que l'agent a appris à contourner le malus pour aller chercher la récompense. Le fait que l'agent ait déjà sélectionné l'action « Déplacement Ouest » pour ce concept montre que pour l'un ou l'autre des états concernés, il a auparavant essayé d'aller vers la case « malus ». Finalement, concept c_4, on retrouve une généralisation abusive. La récompense comme le malus peuvent être au nord ou au sud. Dans le cas présenté, l'agent a du rencontrer l'état avec un malus au sud et une récompense au nord. La prochaine fois qu'il rencontrera le même état avec l'inversion du malus et du bonus, il devrait sélectionner *« Déplacement Nord »* , ce qui l'amènera vers le malus. Notons d'ailleurs que si ce concept n'est pas pertinent maximal, il reste pertinent car l'action et *« Déplacement Est »* et *« Pousser Ouest »* convergent vers une valeur fixe si l'agent suit après une politique optimale.

6.4.5 Conclusions concernant l'expérimentation

Comme c'était l'objectif, l'algorithme proposé est plus efficace tant sur le plan de la vitesse de réalisation de la tâche que sur le montant de la récompense obtenue. Ceci est évidemment dû au fait que l'algorithme de *Q-Learning* ne procède à aucune généralisation de l'apprentissage contrairement à notre algorithme de *Q-Concept Leaning*.
Ainsi, tout nouvel état amène l'algorithme de *Q-Learning* à avoir un comportement exploratoire et quasiment aléatoire.

Sessions anormalement longues Il reste comme on peut s'en apercevoir sur les figures 6.7 (a) page 148 et 6.11 (a) page précédente, des épisodes où les sessions sont assez longues pour notre algorithme, même après un nombre d'états rencontrés assez grand. Trois éléments expliquent ceci.

1. Premièrement, nous avons utilisé une politique $\epsilon - gloutonne$ sans aucune modification au cours de l'apprentissage. Ainsi, il reste toujours une part aléatoire dans le comportement de l'agent.
2. Deuxièmement, certains nouveaux états correspondent à de nombreux concepts améliorant la politique, ceux-ci ont potentiellement des actions optimales contradictoires. Il faut donc que l'agent « fasse le tri » parmi ces concepts en expérimentant.

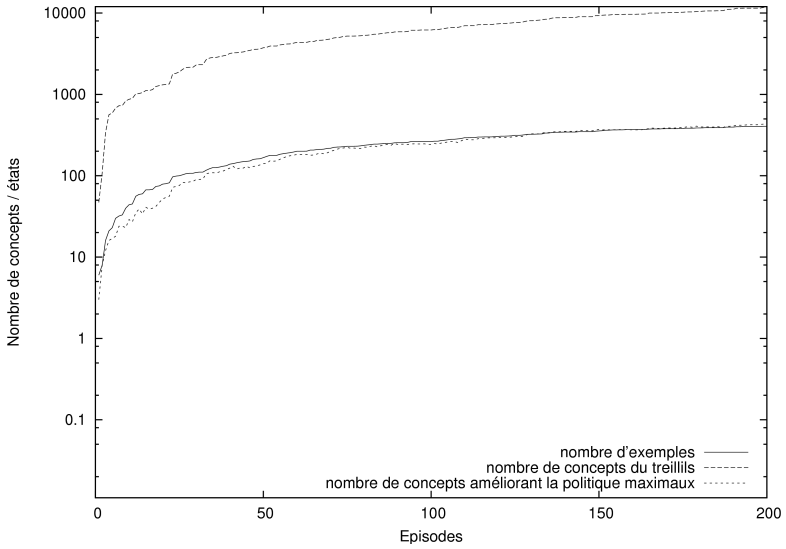


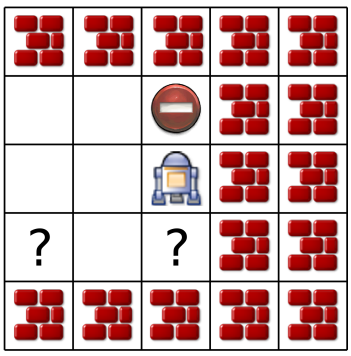
FIG. 6.13 – Taille du treillis, nombre d'exemples et nombre de concepts améliorants maximaux (échelle logarithmique)

3. Troisièmement, il subsiste un problème que nous avons peu abordé : les valeurs d'initialisation des actions pour les nouveaux concepts. Un concept est créé par la généralisation d'un nouvel exemple (état de l'environnement) inséré et d'un concept préexistant. La valeur d'initialisation des actions disponibles pour ce nouveau concept est celle des actions du concept préexistant. Ceci est relativement naturel mais a un effet pervers. En effet, lors de la sélection d'une action pour un état donné, l'agent sélectionne l'action parmi les concepts pertinents maximaux qui lui rapportent potentiellement le plus. Il n'est pas tenu compte du fait de savoir si le concept considéré est très bien connu ou s'il vient d'être créé. Or, les nouveaux états ont presque systématiquement quelque chose en commun dans leur description avec des concepts ayant une grande valeur pour leurs actions optimales (égale à 1 par exemple). Ainsi, pour un nouvel état, sera souvent créé un concept dont l'intention sera la généralisation de la description du nouvel état et de l'intention d'un concept ayant une grande valeur pour son action optimale. Ce nouveau concept n'aura pas de concepts inférieurs venant casser son statuts de concept améliorant la politique courante.

Nous pensons que les sessions anormalement longues viennent principalement de ce troisième phénomène.
Ainsi, il faudra probablement prendre en compte la fiabilité du concept en plus du fait de savoir s'il améliore la politique courante. On peut par exemple envisager la solution suivante (que nous n'avons pas expérimentée) : parmi les concepts maximaux améliorant la politique courante, nous pourrions rajouter une pondération selon le nombre d'états inclus dans l'extension des concepts concernés. Ainsi, un concept aurait d'autant plus de chance d'être sélectionné parmi les concepts maximaux améliorant que son extension a un cardinal important.

Taille de la fonction $Q(c, a)$ Notre algorithme n'est en l'état applicable que sur des cas académiques. En effet, comme on peut le constater figure 6.13, la taille de la fonction $Q(c, a)$ devient très importante assez rapidement, dû au nombre de concepts du treillis de Galois. Pour une approche plus applicable, nous pourrions améliorer notre algorithme du point de vue de l'efficacité en limitant la taille du treillis.

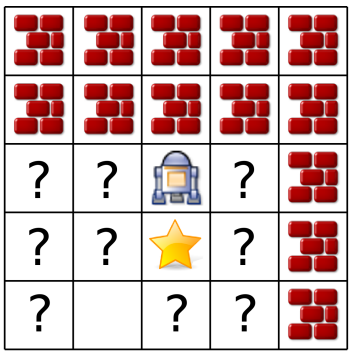
Ceci peut être fait premièrement en limitant la taille du langage L_E. En effet, si on limite le nombre de termes du langage cela limite obligatoirement le nombre de concepts.
Ceci peut-être fait de façon relativement simple si L_E dispose d'une « échelle de précision », c'est à dire si l'on dispose d'un ordre partiel sur les éléments du langage. Par exemple, si le langage est



Etats : 2

c_1

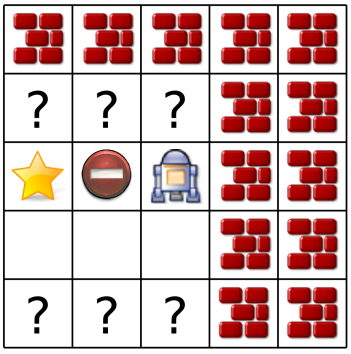
D Nord : -0,31 (?) D Ouest : 0,819 (?)
D Sud : 0,9919 (?) P Est : 0,821 (?)



Etats : 8

c_2

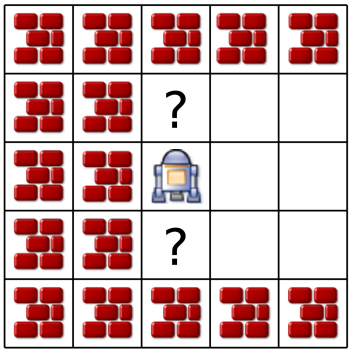
P Nord : 0,173 (0,9) D Ouest : 0,813 (?)
D Sud : 1 (1)



Etats : 2

c_3

D Sud : 0,674 (0,729) D Ouest :-0,137 (-0,1)
P Est : 0,19 (0,656)



Etats : 2

c_4

D Nord : 1 (?) P Ouest : 0,893 (0,9)
D Sud : -0,063 (?) D Est : 0,81 (0,81)

FIG. 6.14 – Sélection de 4 concepts améliorants maximaux parmi les 431 existants avec les valeurs de leurs actions. Les valeurs entre parenthèses sont les valeurs pour une politique optimale. Un (?) signifie que l'action n'est pas pertinente. *Etats* : n correspond à la taille de l'extention du concept.

de type « attribut-valeur », il sera possible de considérer ou non certains attributs. Si les états sont décrits par des graphes et que la généralisation de plusieurs états est constituée par les chemins de longueur n en commun dans les graphes, la variation de n permet également de produire une telle échelle de précision variable. Il n'est cependant pas évident que cette limitation ne soit pas nuisible à l'apprentissage. Cela revient en effet à fusionner artificiellement certains états, amenant potentiellement à empêcher les concepts pertinents d'être exprimés dans L_E.

Deuxièmement nous pouvons réduire la taille du treillis stocké. Ainsi, nous pouvons décider que le treillis ne doit comporter au maximum que x concepts. Pour réduire la taille du treillis si la valeur est dépassée, on peut fusionner des états qui sont couverts par les mêmes concepts améliorants. Ceci repose alors le problème de la non-monotonie de la connaissance. En effet, peut-être que les concepts fusionnés l'ont été par erreurs. Il faut alors un mécanisme permettant de le détecter puis de « revenir en arrière ».

L'heuristique que nous venons de proposer n'est qu'une parmi de nombreuses autres que nous avons envisagées afin de sélectionner des actions et d'interpréter la fonction $Q(c, a)$ en cours d'apprentissage. Toutes celles-ci n'ont pas abouti et nous n'avons pas la prétention d'avoir fourni la meilleure possible. Nous allons dans la partie suivante évoquer d'autres pistes envisagées sans permettre en l'état de fournir un algorithme implémenté avec des résultats concluants.

6.5 Autre piste : notion de stabilité

Nous allons maintenant évoquer une piste que nous avons suivie durant nos travaux (voir [Ricordeau, 2003, Ricordeau, 2004]). Celle-ci n'a pas complètement abouti étant donné qu'elle n'a pas produit d'algorithme général et applicable. Néanmoins, par les méthodes envisagées et les questions soulevées, elle nous semble primordiale pour la poursuite des travaux. La solution que nous avons proposée ci-dessus est basée sur la notion d'amélioration de la fonction de qualité pour un concept à un moment donné de l'apprentissage. Une autre approche est de considérer qu'un concept est fiable si son utilisation donne régulièrement le résultat attendu. Comme nous allons le voir, tout le problème consiste à déterminer les critères permettant de définir si la fonction de qualité produit le résultat attendu. Nous dirons que c'est une approche par **stabilité du résultat**. Nous verrons que les difficultés de trouver de tels critères sont liées à la pratique d'un apprentissage statistique.

6.5.1 Objectif de l'apprentissage

Notons tout d'abord que le cadre de convergence reste le même que celui que nous avons proposé partie 6.2.1. Ainsi, après convergence des algorithmes, c'est à dire après un apprentissage suffisant, pour une tâche d'apprentissage donnée, la fonction $Q(concept, action)$ structurée sous forme de treillis de Galois devra être divisée en quatre parties : concepts pertinents, pertinents maximaux, semi-pertinents et non pertinents. La recherche des concepts pertinents maximaux reste l'aboutissement de l'apprentissage dans le sens où ils permettent de produire une politique optimale exprimée avec une généralisation maximale dans les langages de description des états et des actions proposés.

L'idée de base de l'approche par stabilité du résultat est qu'il y a une forte similarité entre la valeur des actions pour les concepts pertinents et ce qui est effectivent observé ; inversement, pour les couples $(concept, action)$ non stables, ce retour est plus erratique. Voyons d'abord un exemple simple type bandit manchot pour comprendre le phénomène.

Considérons le cas le plus simple, c'est à dire trois états e_1, e_2 et e_3 avec deux actions a_1 et a_2 disponibles dans chacun des états. Le système est déterministe, l'épisode se termine dès qu'une action est sélectionnée. De plus, dans l'état e_1 et e_2, l'action a_1 amène l'agent à observer la récompense 1, alors que l'action a_2 amène l'agent à observer la récompense 0. Pour l'état e_3, c'est l'inverse : a_1 rapporte 0 et a_2 rapporte 1.

Si l'extension d'un concept regroupe e_1 et e_2, celui-ci pourra être considéré comme stable.

En effet, $Q(e_1, a_1) = Q(e_2, a_1)$ (d'ailleurs, on a également $Q(e_1, a_2) = Q(e_2, a_2)$). Par contre, un concept ayant $\{e_1, e_3\}$ comme extension ne pourra pas être qualifié de stable. En effet, $Q(e_1, a_1) \neq Q(e_3, a_1)$ et $Q(e_1, a_2) \neq Q(e_3, a_2)$.

La notion de stabilité est à rapprocher de la notion de concept pertinent dans le sens où les concepts pertinents ont mêmes valeurs pour leurs actions optimales en suivant une politique optimale.

Dans le cas que nous venons de citer, la notion de stabilité est très simple. Ceci est dû au contexte d'un environnement déterministe et sans récompense différée. Dès que l'on passe dans un environnement stochastique (ou au moins que l'on n'exclut pas le fait qu'il puisse l'être) et avec récompense différée, tout se complique. Il y a en effet plusieurs paramètres à prendre en compte :

1. **La stochasticité**, c'est à dire le fait que les actions ne mènent pas toujours dans les mêmes états avec la même récompense, mais selon une certaine distribution. Ainsi, simplement vérifier que le résultat est le même à chaque fois n'est pas une approche valide, puisque dans un environnement stochastique, une même action dans un même état peut produire des résultats différents. Pour dire qu'une action stochastique donne le même résultat pour deux états e_1 et e_2, il faudra considérer un test statistique qui permette d'indiquer que le résultat des actions suit la même distribution dans e_1 et e_2.
2. **L'absence de connaissance sur la distribution des états** (alors que celle-ci est nécessaire pour la plupart des tests statistiques). De fait, la distribution réelle des états de l'environnement est facteur de deux paramètres non maîtrisables : la dynamique de l'environnement d'une part, et la façon de produire une politique par l'agent d'autre part. Pire, ce dernier facteur est lui-même dépendant d'un facteur aléatoire (par exemple le paramètre ϵ pour les politiques ϵ-gloutonnes) et de l'algorithme d'apprentissage, puisque la politique suivie dépend généralement de la fonction $\hat{Q}(c, a)$. Finalement, on ne connaît même pas le nombre d'états de l'environnement. Il vaut donc mieux considérer que l'on ne connaît rien sur la distribution des états.

Ainsi, il devient complexe de faire la différence entre :

- un concept dont la valeur des actions sera par nature erratique. C'est le cas pour les concepts non pertinents. La valeur de leurs actions ne se stabilise jamais et dépend de la distribution des états (voir partie 6.1.2). Cependant, au cours de l'apprentissage, par défaut d'exploration, ce type de concept peut paraître stable.
- un concept pertinent, mais dans un environnement stochastique.
- un concept pertinent mais dont la valeur des actions n'a pas encore convergé. N'oublions pas que la valeur d'une action dans un état donné dépend d'une part de la récompense immédiate et d'autre part, de la suite des récompenses à venir (avec un facteur de dégressivité). La stabilité de la distribution des récompenses immédiates peut être assez aisément vérifiée, mais la fiabilité de la suite des récompenses à venir est beaucoup plus difficile à analyser.

Un premier constat que nous pouvons faire est que là encore, la nature de stabilité d'un concept doit être non monotone. En effet, la stabilité comme l'instabilité peuvent être le résultat d'une carence dans l'exploration.

Deuxièmement, nous pouvons identifier trois manières de vérifier la stabilité d'un concept :

1. **Disposer de renseignements de deuxième ordre**. Dans ce cas, on a des éléments sur la nature de l'environnement. Parmi les exemples donnant des indications concernant la stabilité on a : le fait de savoir si un environnement est ou non déterministe, que la variance des récompenses est inférieure à un certain réel, que la tâche est découpée en épisodes, le nombre d'états de l'environnement,... Évidemment, on peut considérer qu'on connaît ces éléments, ou bien on peut faire des suppositions dessus. Dans cette logique, l'agent peut lui-même se référer à des critères de deuxième ordre, mais qu'il a lui-même estimés. Par exemple, il peut avoir constaté que les récompenses oscillent entre 0 et 1, ce qui donne une indication en soit.
2. **Comparer la stabilité d'un concept sur la durée**. Pour un concept donné, on peut vérifier si le rapport entre ce qui est attendu par l'agent et ce qui est effectivement obtenu,

est stable, que ce soit pour les récompenses reçues ou pour les récompenses espérées à venir. Plus généralement, si la variance des valeurs se réduit au fur et à mesure des interactions, que la succession des valeurs « a l'air de » suivre une suite croissante et convergente, on peut soupçonner qu'il s'agit d'un concept pertinent. Si au contraire, il n'y a aucune régularité, on peut estimer que le concept est trop général. Rappelons encore qu'il faut prendre en compte que cette connaissance est non monotone. Un concept que l'on croyait stable peut très bien n'être stable que par manque d'exploration des états suivants. Cette notion est non triviale, voir par exemple partie 6.1.2, où de nouveaux exemples inconnus dégradent une connaissance parfaite sur un concept pourtant pertinent. On dispose pour cette méthode de la suite des valeurs (ainsi que de leur nombre), ce qui permet certains calculs statistiques. Il faut néanmoins en général que ceux-ci puissent être faits de manière incrémentale. Il est en effet inenvisageable de garder l'ensemble des valeurs.

3. **Comparer entre eux les concepts**. Les mesures présentées ci-dessus ne sont pas absolues, puisqu'elles dépendent, comme nous l'avons déjà expliqué, de paramètres non connus et non maîtrisable ainsi que du fait que la connaissance à un moment donné est partielle. En revanche, nous pouvons utiliser la relation d'ordre entre les concepts (comme nous l'avons fait dans la partie précédente). En effet, la variance peut-être comparée à celle des concepts inférieurs ou supérieurs, donnant des indications quant à la stabilité du concept étudié.

Si tous les éléments permettant de déterminer la notion de stabilité des concepts sont présents, en l'état actuel, nous ne sommes pas en mesure de proposer un algorithme les utilisant. En effet, il reste encore un travail permettant, en fonction de ces critères, de pouvoir distinguer avec une connaissance incomplète, à l'instar de ce que nous avons présenté algorithme 6.6, les concepts stables maximaux.

Si on utilise des critères pas assez restrictifs, les concepts non pertinents car trop généraux vont passer ces critères, faussant, au moins au début mais avec des conséquences importantes, l'apprentissage de la fonction de qualité.
De l'autre côté, en utilisant des critères trop permissifs, les concepts qui devraient être stables, au regard du fait qu'ils sont pertinents maximaux sont estimés comme non stables compte tenu de la situation que nous avons développée partie 6.1.2.
Nous n'avons pas encore réussi à exhiber des critères permettant d'équilibrer la notion de stabilité tout en assurant une convergence asymptotique dans un temps raisonnable. Évoquons également une autre difficulté rencontrée, non pas pour estimer la fonction $Q(c, a)$, mais pour la sélection des actions en utilisant la notion de concepts fiables

6.5.2 Stabilité et décision

L'objectif de l'agent n'est pas, bien entendu, de trouver les concepts stables ou instables. Ce n'est qu'un moyen pour permettre à l'agent de sélectionner les actions rapportant le maximum de récompenses à long terme. Ainsi, même si l'on disposait de critères permettant de différencier les couples $(concept, action)$ stables, sauf à considérer qu'on les connaît tous (par exemple une fois que la fonction $Q(c, a)$ a convergé), il reste un problème par rapport à la sélection des actions. En effet, si l'on privilégie systématiquement la sélection des actions dont le résultat est stable, des actions sous-optimales avec un résultat connu seront privilégiées. Par exemple, si l'agent peut rester sur place et que cela lui rapporte toujours le même malus, cette action a un résultat parfaitement prévisible, même s'il est mauvais. Il faudra donc arbitrer dans la fonction de sélection des actions en fonction de $Q(c, a)$ entre la stabilité et le gain potentiel en explorant. C'est une difficulté à laquelle nous n'avons en l'état pas non plus trouvé de solution satisfaisante.

6.5.3 Travaux approchants

D'autres travaux ont posé des questions similaires (voir également fin de la partie 3) :

- Dans [Reynolds, 2002], il s'agit d'avoir une fonction $Q(e, a)$ qui soit à échelle variable. C'est à dire qu'on se place dans un cadre où les états et les actions sont des ensembles continus. On divise alors l'ensemble des états en parties homogènes du point de vue de l'action optimale. La question de la division est vraiment similaire à notre problème. Cependant, les outils utilisés (comme le nombre de passages par un couple (état,action) donné) sont trop naïfs pour pouvoir être utilisés dans un cadre avec généralisation comme le nôtre.

- Les travaux de [McCallum, 1996b] posent également le même problème. En effet, il s'agit là de lever des ambiguïtés entre des états qui sont similaires du point de vue de l'observation en utilisant un système de mémoire. Notre problème se pose alors sous la forme : « Quand l'agent doit-il considérer qu'une observation correspond en réalité à deux états qu'il faut désambiguïser ? » Il s'agit bien de la même question car un concept peut être conçu comme une fusion entre plusieurs états, ceux-ci étant considérés comme un et un seul. Dans ces travaux, le test utilisé est le *Student-T test*. Le problème de celui-ci est qu'il repose sur l'assomption que les données suivent une distribution gaussienne. C'est certainement d'ailleurs, selon les dires même de son auteur, qu'il a obtenu des résultats limités en l'utilisant.
- Dans [Munos and Moore, 2002] il s'agit de travailler sur le cas continu. C'est à dire que les états et les actions ne sont pas des ensembles discrets, mais continus. Comme il est nécessaire de discrétiser à un moment donné, la question du découpage des régions se pose. Les questions sont alors les mêmes que celles que nous nous posons. Comment estimer qu'un ensemble d'états et un autre doivent être regroupés ? Il s'agit dans le cadre de ce travail de découper l'espace par différentes méthodes comme des grilles à résolution variable. Pour déterminer quels sont les ensembles d'états qui doivent être regroupés, l'auteur utilise les notions d'influence et de variance. Ces deux notions ne sont calculables qu'avec la connaissance de la dynamique de l'environnement, ce que nous avons exclu.

Nous venons de proposer la notion de stabilité permettant d'analyser la fonction $Q(c, a)$ dans le cadre du *Q-Concept learning*. Cependant, les implémentations effectuées jusque là se sont heurtées aux difficultés théoriques posées.

Pourtant, il nous semble que cette voie est importante. Premièrement, elle se confronte précisément à un problème théorique important : comment juger « en ligne » de la qualité d'approximation de la fonction $\hat{Q}(e, a)$? Les autres travaux, notamment [Reynolds, 2002, Munos and Moore, 2002, McCallum, 1996b] ayant été confrontés au même problème ne proposent pas non plus de résultats généralisables. Nous révoquerons ce point dans les perspectives développées dans la conclusion générale, mais il nous semble qu'il faille introduire la possibilité pour l'agent de faire des suppositions, éventuellement modifiables du deuxième ordre quant à l'environnement. Deuxièmement, comme nous l'évoquerons également dans la prochaine partie, il nous semble qu'une des perspectives de l'apprentissage par renforcement est de s'affranchir au moins partiellement de la contrainte markovienne de l'environnement. Que resterait-il donc à apprendre dans un monde non markovien ? Précisément, les concepts stables peuvent être une réponse à cette question.

6.6 Conclusion

Nous avons proposé dans ce chapitre une utilisation de la structure de treillis de Galois formé par l'ensemble des états de l'environnement et le langage de description des états pour représenter la fonction $Q(e, a)$. Nous avons montré que celle-ci devenait alors $Q(c, a)$. Plutôt que de valuer les actions dans chacun des états, nous avons proposé algorithme 6.1 page 128 que soient estimées les valeur de qualité des actions pour chacune des généralisations permises par le langage de description des états de l'environnement.
Ensuite, nous avons présenté la notion de concepts pertinents, permettant d'analyser les concepts de la fonction $Q(c, a)$. La recherche des concepts pertinents maximaux a été montrée comme étant l'objectif de l'apprentissage dans le sens où ils sont fortement corrélés à la partition $\mathcal{P}^*$ proposée partie 5.3.
Afin d'utiliser ces notions en ligne (contrairement à ce que nous avons fait avec l'algorithme 5.1 dans le chapitre précédent), nous avons proposé la notion de concepts améliorant la politique, comme étant une estimation à un moment donné de l'apprentissage des concepts pertinents maximaux.
Finalement, nous avons proposé l'algorithme par renforcement 6.6 page 146, regroupant l'ensemble de ces notions ainsi qu'une fonction de sélection des actions ayant comme paramètre la fonction $Q(c, a)$. Nous avons proposé partie 6.4 une expérimentation sur un cas académique. Finalement, nous avons évoqué une autre heuristique non aboutie partie 6.5 mise en relation avec des travaux

approchants.

Le prochain chapitre proposera une conclusion générale, qui contiendra, outre un récapitulatif global, des réflexions et perspectives ouvertes par nos travaux.

7 Conclusion

7.1 Résumé

Rappelons tout d'abord brièvement notre cadre de travail, nos objectifs, notre démarche ainsi que nos réalisations.

Nous avons placé nos travaux dans le champ particulier de l'apprentissage machine (chapitre 1) qu'est l'apprentissage par renforcement, c'est à dire un cadre général permettant à un agent de percevoir son environnement et d'agir sur celui-ci. Une des perceptions de l'agent est un signal de renforcement, une valeur numérique lui indiquant la qualité de l'état dans lequel il se trouve. L'apprentissage par renforcement est constitué par la famille d'algorithmes permettant à l'agent d'améliorer son comportement (ou politique) au sens où il aura tendance à plus sélectionner les actions lui rapportant une meilleure récompense à long terme (chapitre 2).

Les algorithmes de base de l'apprentissage par renforcement ne généralisent pas l'apprentissage. En effet, un comportement appris sur un ensemble d'états de l'environnement ne pourra pas être utilisé par l'agent sur un autre environnement similaire. Nous avons vu chapitre 3 que de nombreux travaux avaient été envisagés pour aborder cette question. Citons la plus utilisée des techniques, la généralisation par *approximation de fonction*. D'autres cadres plus algébriques ont été abordés, reposant sur l'*abstraction algébrique* des états. Nous avons également vu que cette question avait fait l'objet de ponts avec d'autres pans de l'apprentissage artificiel comme l'*apprentissage par renforcement relationnel*, permettant de faire le lien avec un domaine déjà très étudié de l'apprentissage machine. Nous avons également indiqué comment cette question était corrélée à celle de l'apprentissage par renforcement dans un *environnement continu*, d'une représentation de l'environnement avec une *échelle variable* ou même des *environnements partiellement observables*.

Pour notre part, notre démarche est de permettre la généralisation des politiques en préservant un certain nombre de principes de l'apprentissage par renforcement :

- a*pprentissage en ligne*, c'est à dire pas de différenciation entre une phase d'apprentissage et une phase d'utilisation de l'apprentissage.
- *connaissance minimale de l'environnement*, nous ne voulons notamment pas avoir accès à la dynamique de l'environnement.

En plus de la possibilité de réutiliser une politique apprise sur un environnement similaire, nous voulons pouvoir rajouter un élément qui nous semble essentiel en apprentissage artificiel : l'*interprétabilité du résultat*.

Nous avons choisi d'aborder cette question en utilisant un champ dont l'étude progresse en apprentissage artificiel, c'est à dire la structure algébrique des treillis de Galois (chapitre 4). Celle-ci permet en effet, en utilisant un langage de description, avec une capacité de généralisation, pour un ensemble d'objets, de considérer l'ensemble des généralisations admises pour le langage. Ainsi, un apprentissage sur un ensemble d'états permet d'induire un comportement sur l'ensemble des états de l'environnement ayant une similarité par le langage de description utilisé.
Nous avons de plus, adapté la notion de treillis de Galois en proposant le *treillis de Galois des partitions*. Cette structure contraint par un langage de description, non pas le treillis des parties d'un ensemble d'objets, mais plutôt le treillis des partitions de ceux-ci.

Nous avons donc, chapitre 5, montré comment formaliser l'apprentissage d'une tâche par apprentissage par renforcement comme un partitionnement des états de l'environnement selon leurs actions optimales. Ainsi, le treillis de Galois des partitions de l'ensemble des états de l'environnement constitue l'espace de généralisation des politiques. De plus, nous avons montré qu'une partition, que nous avons nommée partition $\mathcal{P}^*$, était la partition unique, pour une tâche donnée, pouvant être considérée comme l'objectif de l'apprentissage. La partition $\mathcal{P}^*$ des états de l'environnement est donc une réponse à la généralisation de l'apprentissage d'une politique et

présentée dans un langage interprétable.

Néanmoins, $\mathcal{P}^*$ peut ne pas être expressible dans le langage de description des états donné. Il convient alors premièrement de considérer l'une des partitions les plus générales expressibles dans le langage et inférieure à $\mathcal{P}^*$ comme objectif de l'apprentissage. Deuxièmement, ceci indique une certaine inadéquation entre le langage de description des états et la tâche à accomplir, ce qui peut amener à la possibilité de proposer l'enrichissement du langage par de nouveaux termes.
Cette approche a été validée par une implémentation et un test sur un problème académique. Celle-ci nécessite néanmoins une phase d'apprentissage préalable.

Dans le chapitre 6 nous avons eu a contrario une approche permettant un apprentissage « en ligne ». Nous avons donc redéfini la fonction d'apprentissage $Q(e, a)$ de l'apprentissage par renforcement classique en fonction $Q(c, a)$, organisant la fonction de qualité sous forme de treillis de Galois des états de l'environnement. Nous avons corrélé ceci avec les éléments établis dans le chapitre précédent en montrant que certains concepts dits *pertinents* étaient les composants de la partition $\mathcal{P}^*$, ou des partitions expressibles inférieures. Nous avons nommé *Q-Concept-Learning*, les algorithmes utilisant une fonction de la forme $Q(c, a)$ comme fonction de qualité.
Ainsi, nous avons proposé et expérimenté une heuristique visant à approximer les concepts pertinents en utilisant la notion de *concepts améliorant la politique courante*.
Finalement dans ce chapitre ainsi que dans la présente conclusion, sont évoquées des pistes suivies n'ayant pour l'instant pas abouties ainsi que des ouvertures et perspectives.

Ainsi, nous pouvons dire que notre travail se situe dans le rapprochement entre l'apprentissage statistique constitué par l'apprentissage par renforcement et un apprentissage plus en lien avec une interprétation sémantique.

7.2 Ouvertures et perspectives

Nous allons maintenant proposer quelques remarques qui sont des réflexions, des ouvertures ou des pistes possibles pour de futurs travaux. Elles sont issues des inévitables voies que l'on n'explore pas par manque de temps ou de compétences, mais qui semblent essentielles à la poursuite de la recherche dans le domaine. Dans la partie précédente, nous avons précisé quelques pistes possibles d'amélioration de notre algorithme, notamment quant à la gestion de l'incertitude dans notre cas précis. Nous n'y reviendrons donc pas. Nous allons aborder les éléments suivants : l'utilisation du treillis de Galois en tant qu'espace de recherche, la relativisation voire l'abandon de la formalisation markovienne, l'apprentissage en prenant en compte des données de deuxième ordre, les relations avec l'apprentissage par renforcement relationnel et l'extraction de la dynamique de l'environnement.

Améliorations algorithmiques Dans l'algorithme que nous avons présenté partie 6, le treillis de Galois formé par l'ensemble des états de l'environnement et par le langage de description des états est construit de façon complète[48].
Dans la partie 5, nous avons étudié un algorithme basé sur l'utilisation des treillis de Galois des partitions. Pour autant, nous n'avons pas construit intégralement cette structure dans nos algorithmes (cela serait d'ailleurs probablement infaisable compte tenu de sa taille). Nous avons donc utilisé le treillis de Galois des partitions comme un espace de recherche possédant des propriétés algébriques particulières. Ainsi, plutôt que de stocker l'intégralité de la fonction $Q(c, a)$, c'est à dire l'ensemble des généralisations admises par le langage de description des états, il serait certainement plus judicieux de ne stocker les valeurs que pour un sous-ensemble des couples $(concept, action)$.
Nous voyons dans cette perspective deux types de concepts qui devraient être construits : premièrement ceux qui sont nécessaires du point de vue de l'apprentissage lui-même et deuxièmement du point de vue algébrique :

- **Nécessité pour l'apprentissage** Si l'on ne conserve que certains concepts, du point de vue

[48] Bien que de façon incrémentale, ce qui implique que l'agent n'est pas obligé de connaître l'ensemble des états de l'environnement.

de l'apprentissage, la question est « lesquels conserve-t-on » ? Il paraît au moins évident qu'il faut garder ceux qui sont au centre de la politique courante, c'est à dire ceux possédant la propriété principale que l'on utilise.
Par exemple, si on utilise l'algorithme présenté dans la partie précédente, il faut garder les concepts les plus généraux améliorant la politique courante. De fait, tous les autres concepts, ne sont pas utiles. En effet, les concepts plus généraux (c'est à dire trop généraux), ne sont pas pertinents pour la tâche courante et les concepts moins généraux que les concepts maximaux sont redondants. Par ailleurs, les valeurs des actions pour ces derniers sont normalement moins bien approximées.
Cependant, les concepts améliorant la politique courante n'ont pas cette propriété intrinsèquement, ils l'ont compte tenu de l'état actuel de la fonction $Q(c, a)$, donc de l'apprentissage. La connaissance impliquée par la fonction $Q(c, a)$ est non-monotone par nature, étant donné que lorsque cette connaissance est stabilisée, cela implique que l'agent a déjà convergé vers une politique optimale. Ainsi, il convient de ne pas garder strictement les concepts améliorant la politique courante, mais certainement des concepts un peu plus généraux et un peu plus spécifiques, au moins afin qu'ils puissent servir de point de comparaison. La question difficile est de gérer le niveau de spécialisation et de généralisation pour conserver des propriétés de convergence satisfaisantes.

- **Nécessité algébrique** Si on ne conserve que certains concepts du treillis, on peut noter que la relation d'ordre partiel entre eux est néanmoins conservée. Ainsi, les opérations d'accès à une valeur $Q(c, a)$ donnée ne changent pas. Compte tenu du fait que l'on conserverait moins de concepts que dans le treillis complet, ces opérations sont même a priori plus rapides. En revanche, il se pose le problème de la construction incrémentale des nouveaux concepts. En effet, les algorithmes incrémentaux sont basés sur la comparaison du nouvel exemple (ici de la description du nouvel état) avec les concepts déjà existants dans le treillis. En corolaire, nous avons indiqué que les concepts à préserver relativement à la tâche d'apprentissage pouvaient être modifiés au cours de l'apprentissage. Il faut donc être en mesure d'enlever ou reconstruire un concept supprimé si l'évolution de la connaissance l'exige. Ainsi, il faut s'assurer de conserver l'ensemble des concepts permettant de reconstruire ces éléments si besoin est.

  Finalement, les treillis de Galois ont des propriétés que nous n'avons pas exploitées lors de nos travaux. Une utilisation possible de données structurées à l'aide d'un treillis de Galois est en effet de pouvoir extraire un certain nombre de règles sur le langage, comme « si on voit l'élément A sur un objet, alors on verra également nécessairement l'élément B ». Il faudrait également que la suppression de concepts inutiles dans notre contexte ne nuise pas à la recherche de telles règles. Des recherches existent dans la communauté des treillis de Galois pour savoir quels sont les concepts et/ou les exemples nécessaires et suffisants pour la construction du treillis. Elles constituent certainement une des pistes à étudier.

Abandon de la caractéristique markovienne pour le formalisme Nous avons abordé plusieurs fois au cours de nos travaux le fait que les prétentions de l'apprentissage par renforcement étaient importantes si on regarde le cadre proposé : un agent (éventuellement parmi d'autres), sans modèle de son environnement, simplement capable de percevoir celui-ci ainsi qu'un renforcement numérique. Cet agent peut agir sur son environnement et peut modifier son comportement en fonction de ses expériences, c'est à dire de ses observations successives des couples $(\textit{états}, \textit{récompense})$. C'est presque le cadre minimal pour pouvoir parler d'agent autonome. Cependant, ce cadre très général et avec beaucoup de prétentions est limité dès que l'on passe à la formalisation mathématique. En effet, la formalisation la plus communément admise pour les états de l'environnement est celle des Processus de Décision Markovien. Certaines extensions ont été faites, notamment pour intégrer la notion de multi-agents, mais certaines contraintes « abusives » restent néanmoins présentes. Notamment, on peut citer le fait que l'environnement, bien que généralement stochastique, est systématiquement statique (c'est à dire que sa dynamique n'évolue pas au cours du temps). C'est en effet une contrainte forte car de nombreux domaines dans lesquels pourraient intervenir des agents apprenants sont par nature dynamiques, notamment lorsque d'autres agents sont en interaction. Par définition, un autre agent apprenant (et pas forcément coopérant) modifie de fait la dynamique de l'environnement, car il change son propre comportement. Les perspectives

très généralistes de l'apprentissage par renforcement s'en trouvent donc réduites de façon drastique.

Des environnements dynamiques pourraient éventuellement être ramenés à un environnement statique avec une contrainte de temps, mais cela violerait de toute façon la condition de pouvoir passer potentiellement infiniment souvent par l'ensemble des états et des actions, ce qui est une condition nécessaire de convergence des algorithmes d'apprentissage par renforcement.
Pourtant, intuitivement, les algorithmes d'apprentissage par renforcement semblent adaptés pour de tels environnements. En effet, ils incluent déjà la capacité de faire un compromis entre les connaissances précédemment acquises et les connaissances nouvellement acquises (le paramètre α) de la formule 2.15 page 37 du *Q-Learning*. De plus, il s'agit d'un apprentissage généralement « en ligne », c'est à dire que l'agent utilise sa connaissance partielle de l'environnement pour agir sur celui-ci et continuer à apprendre dans le même temps. Il n'y a pas séparation entre apprentissage et utilisation de l'apprentissage.
D'ailleurs, des algorithmes comme *Dyna-Q* (cité dans [Sutton and Barto, 1998]) prennent en compte la possibilité d'avoir un environnement dynamique.

Néanmoins, il se pose au moins un problème important : la vitesse de modification de la dynamique de l'environnement. En effet, si l'environnement change tout le temps sur tout, au fond, il n'y a rien à apprendre, puisque toute connaissance devient rapidement obsolète. Sans évoquer ce cas extrême, il doit y avoir une corrélation entre la nature et la vitesse des changements d'une part et la capacité des algorithmes à s'adapter à ces changements d'autre part.
Il manque donc un formalisme mathématique pour intégrer aux Processus de Décision Markovien cette composante qui les rendrait fondamentalement non-markoviens. Nous pensons cependant qu'il existe des degrés dans l'abandon de la caractéristique markovienne qui permettraient de ne pas mettre à mal tous les théorèmes de convergence des algorithmes d'apprentissage par renforcement. Cela pose également la question d'améliorer les connaissances concernant les conditions d'arrêts et de l'estimation de l'apprentissage.

Apprentissage prenant en compte des données de deuxième ordre De nombreux problèmes modélisables en apprentissage par renforcement sont solvables mathématiquement, mais en faisant des suppositions de la connaissance de l'environnement inenvisageables pour le cadre agent ou en faisant des simplifications qui réduisent énormément les champs d'application. Par exemple, le problème du compromis exploration-exploitation est parfaitement solvable mathématiquement ([Sutton and Barto, 1998]). En effet, si l'on restreint le problème à celui du bandit manchot, en connaissant parfaitement le modèle de l'environnement et en se restreignant à un environnement sans délais entre la récompense et l'action, des solutions exactes sont connues. De même, avec une connaissance parfaite de l'environnement, une tâche modélisée par un Processus de Décision Markovien peut théoriquement être résolue par programmation dynamique.

Il y a sûrement des intermédiaires entre la connaissance parfaite de l'environnement et la méconnaissance absolue de celui-ci. Comme premier élément, nous pouvons noter que certaines indications du deuxième ordre permettent des simplifications énormes pour certains problèmes :

- Si on dispose de l'information que l'environnement est déterministe. Si on intègre cet élément dans nos algorithmes, cela permet par exemple, d'éliminer d'office certains concepts (ceux dont la récompense immédiate donne des résultats différents).
- Si on dispose de la récompense minimale ou maximale. Cet élément donne des indications importantes quant à la poursuite de l'exploration. Si on dispose d'une action pour un état donné (ou un ensemble d'états donné pour les concepts) ayant une valeur telle qu'elle soit égale à la récompense maximale, on peut orienter l'exploration.
- Si on connaît le nombre d'actions donnant une récompense particulière. Par exemple dans le cas d'un problème à cases, la connaissance du nombre de récompenses disponibles implique des possibilités de restreindre l'exploration si les actions menant à la récompense ont toutes été trouvées.

Il devrait donc être possible dans un premier temps de paramétrer les algorithmes d'apprentissage par renforcement pour qu'ils prennent en compte ces éléments.

Dans un deuxième temps, on pourrait envisager que ces paramètres portant sur des éléments du deuxième ordre puissent être directement intégrés à l'apprentissage, plutôt qu'en tant que paramètres externes. Ainsi, l'agent pourrait apprendre en ayant des a priori sur certains éléments de deuxième ordre concernant son environnement et en les modifiant en fonction des expériences.

On peut également noter que les théories de paramétrage des algorithmes d'apprentissage par renforcement (paramètres α et γ) sont peu nombreuses ([Even-Dar and Mansour, 2003] par exemple) et se restreignent généralement à des classes de problèmes bien particulières. Comme l'ajustement de ces paramètres est clairement influencé par ces éléments du deuxième ordre que nous venons d'évoquer, peut-être serait-il bon que les algorithmes puissent influer sur ces paramètres en cours d'apprentissage. Tout le problème est évidemment d'intégrer ces éléments tout en gardant une convergence des algorithmes vers une politique optimale.

Apprentissage par renforcement relationel Une des avancées importantes depuis la formalisation de l'apprentissage par renforcement comme famille algorithmique à part entière est l'apport de l'apprentissage par renforcement relationnel. Le premier apport est de lier formellement l'apprentissage par renforcement à un domaine déjà largement étudié et possédant par conséquent de nombreuses techniques et théorèmes. La possibilité d'utiliser des bases de connaissances est une bonne illustration de ceci. Deuxièmement, cela permet d'apporter des capacités descriptives plus riches pour la description des états de l'environnement, notamment, avec l'utilisation de langage du deuxième ordre.

L'apprentissage par renforcement relationnel n'était pas fondamentalement nécessaire pour faire cet apport, mais il permet de le faire dans de bonnes conditions de formalisation. Un apport important est également de faire la liaison entre un apprentissage statistique (l'apprentissage par renforcement classique) et un raisonnement automatique (Notons que c'est également le propos de nos travaux). Comme corolaire important, nous pouvons dire que cela permet aussi un lien avec la planification[49]. Nous avons donc une famille d'algorithmes qui fait en sorte qu'un agent améliore son comportement en sélectionnant de façon très réactive ses actions dans un environnement donné (l'apprentissage par renforcement classique) et qui permet également potentiellement à partir de ces éléments de planifier une partie de son comportement. Belle perspective !

Comme nous avons proposé des algorithmes assez généraux, par l'utilisation de propriétés algébriques, l'utilisation de langages propositionnels est également adaptée à nos algorithmes (les problèmes éventuels viennent des complexités des opérateurs de généralisation). De plus, nous avons une vision très générale de la généralisation. Or, dans l'apprentissage par renforcement relationnel, la généralisation proposée est la généralisation par liaison de variables, qui pour être puissante, n'est pas toujours adaptée.

Pour les raisons évoquées, comparer et réconcilier du point de vue théorique les algorithmes d'apprentissage par renforcement relationnel et nos propres travaux semble une piste importante.

Extraction de la dynamique de l'environnement Dans les algorithmes que nous avons proposés, notamment l'algorithme 5.1 page 120, nous extrayons à partir d'une politique de l'environnement apprise par renforcement, des généralisations permettant d'exprimer le comportement à l'aide du langage de description sur les états et les actions. On peut donc dire que c'est là aussi une forme du passage d'un apprentissage statistique vers un apprentissage symbolique.
Implicitement, cet algorithme trouve donc des éléments de la dynamique de l'environnement. Un des objectifs suivants devrait être de rendre explicite cette recherche de dynamique de l'environnement et de l'exploiter. Cette piste de réflexion est également à mettre en relation avec les remarques précédentes concernant l'apprentissage par renforcement relationnel.

[49] Ce lien a déjà été fait avec l'algorithme *Dyna-Q* dans [Sutton and Barto, 1998], mais avec des potentialités beaucoup plus faibles.

8 Annexe : Elements d'algèbre

Nous proposons dans cette partie des définitions pour les notions d'algèbre utilisée dans nos travaux. Ces définitions sont en annexe pour deux raisons. Soit elles sont classiques et on pourra alors les utiliser comme aide-mémoire ou référence. Soit parce qu'il existe plusieurs définitions différentes pour une même notion et nous précisons alors celle que nous utilisons.

Définition 8.1 (Relation binaire)
Soit deux ensemble E et F.
Une **relations binaire** R est un sous ensemble du produit cartésien $E \times F$: $R \subseteq E \times F$, constitué par les couples (x, y), tels que $x \in E$ et $y \in F$ et qui vérifient une certaine propriété P.
Pour exprimer le fait que le couple (x, y) vérifie la propriété P, on peut écrire xRy. On lira x est en relation avec y.
E et F sont alors nommés respectivement ensembles **initial** et **final** de R.
On appelle **domaine** $D(R)$ de la relation R, ou **domaine de définition** de R, le sous-ensemble de E tel que :

$$D(R) = \{a \in E \text{ tels que } \exists b \in F, (a, b) \in R\}$$

On appelle **ensemble des images** $\rho(R)$ de la relation R, le sous-ensemble de F tel que :

$$\rho(R) = \{b \in F \text{ tels que } \exists a \in E, (a, b) \in R\}$$

Définition 8.2 (Application)
Soit E et F, deux ensembles. Soit $G \subseteq E \times F$, un sous ensemble du produit cartésien de E et F tel que pour tout élément $x \in E$, il existe au moins un élément y de F tel que le couple (x, y) appartienne à G.
Le triplet (E, F, G) définit une **application** a de E vers F et notée :

$$a : E \to F$$

$a(x)$, $x \in E$, dénote l'ensemble des $y \in F$ tels que $(x, y) \in G$.
$a(E) \subseteq F$ dénote l'ensemble des images $\rho(G)$ de la relation définie par G.

Définition 8.3 (fonction)
Soit E et F, deux ensembles f, une application de E dans F définie par le triplet (E, F, G). f est une **fonction** si et seulement si pour tout élément $x \in E$, il existe un et un seul y tel que le couple (x, y) appartienne à G.
$f(x)$, $x \in E$ dénote l'unique élément $y \in F$ tel que $(x, y) \in G$

Propriété 8.1
Soit F et G deux ensembles. Toute application $a : E \to F$ est équivalente à une fonction $f_a : E \to \mathcal{P}(F)$.

Preuve 8.1
Il suffit de considérer $f_a(x) = a(x)$.

□

Définition 8.4 (fonction stochastique)
Soit E et F, deux ensembles. Soit a, une application de E dans F définie par le triplet (E, F, G).(x_1, y) dénote l'ensemble couples $(x, y) \in G$ tels que $x_1 = x$. Soit une fonction de probabilités $f_p : G \to [0, 1]$, telle que :

$$\forall x_1 \in E, \sum_{(x_1,y)\in G} f_p((x_1, y)) = 1$$

f est une **fonction stochastique** définie par le triplet (E, F, f_p)

La définition de fonction stochastique est à rapprocher de la définition de politique pour les Processus de Décision Markovien (définition 2.6 page 23).

Définition 8.5 (opérateur)
Soit un ensemble E. Un **opérateur** de E, $\mathcal{O}p$ est une fonction de n produits cartésiens de E dans lui-même :

$$\mathcal{O}p : E \times E \times \cdots \times E \to E$$

Si l'opérateur $\mathcal{O}p$ est une fonction de $E \times E$ dans E :

$$\mathcal{O}p : E \times E \to E$$

on dit qu'il s'agit d'un **opérateur binaire**. Ses deux opérandes sont alors souvent placées de part et d'autre de son symbole.

Exemple 8.1 *L'opérateur produit $\otimes$ définit définition 4.7.*

Références

[Tad, 2004] (2004). *Proceedings of the ICML'04 workshop on Relational Reinforcement Learning*. Dpt of Computer Science, Oregon State University, Corvallis ; Dpt of Electrical and Computer Engineering, Purdue University ; Dpt of Computer Science, Katholicke Universiteit Leuven, Heverlee, Belgium.

[Dri, 2005] (2005). *Proceedings of the ICML 2005 Workshop on Rich Representations for Reinforcement Learning*.

[Asadi and Huber, 1994] Asadi, M. and Huber, M. (1994). State space reduction for hierarchical reinforcement learning. In *Proceedings of the 17th International FLAIRS Conference*. Dpt of Computer Science ang Engineering, University of Texas, Austin.

[Asadi and Huber, 1995] Asadi, M. and Huber, M. (1995). Accelerating action dependent hierarchical reinforcement learning through autonomous subgoal discovery. In *Proceedings of the ICML'05 Workshop on Rich Representations for Reinforcement Learning*.

[Auzias, 2001] Auzias, C. (2001). *Un Paris révolutionnaire. Émeutes, subversions, colères*. Dagorno edition.

[Baird, 1995] Baird, L. (1995). Residual algorithms : Reinforcement learning with function approximation. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann.

[Barto and Mahadevan, 2003] Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierachical reinforcement learning. Number 13, pages 41–77.

[Bellman, 1957] Bellman, R. E. (1957). *Dynamic Programming*.

[Bernstein et al., 2002] Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4) :819–840.

[Bertsekas, 1976] Bertsekas, D. (1976).

[Biard, 1997] Biard, J. (1997). *Guillaume d'Ockam, logique et philosophie*.

[Birkhoff, 1973] Birkhoff, G. (1973). *Lattice Theory, third edition*. Coll. Publ. Amer. Math. Soc.

[Blockeel and Raedt, 1998] Blockeel, H. and Raedt, L. D. (1998). Top-down induction of logical decision trees. *Artificial Intelligence*, 101(1-2) :285–297.

[Boone, 1997] Boone, G. (1997). Minimum-time control of the acrobot. In *1997 International Conference on Robotics and Automation*, pages 3281–3287.

[Bordat, 1986] Bordat, J.-P. (1986). Calcul pratique du treillis de galois d'une correspondance. Number 96, pages 31–47.

[Boyan and Moore, 1995] Boyan, J. A. and Moore, A. W. (1995). Generalization in reinfocement learning : Safely approximating the value function. In Press, M., editor, *Advances in Neural Information Processing Systems*. MIT Press.

[Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Cole Statistics/Probability.

[Chein, 1969] Chein, M. (1969). Algorithme de recherche des sous matrices premières d'une matrice. volume 1, pages 21–25.

[Choi, 2006] Choi, V. (2006). Faster algorithms for constructing a concept (galois) lattice. Number abs/cs/0602069.

[Cornuéjols and Miclet, 2002] Cornuéjols, A. and Miclet, L. (2002). *Apprentissage Artificiel*.

[Davies, 1997] Davies, S. (1997). *Multidimensional Triangulation and Interpolation for Reinforcement Learning*, volume 9. The MIT Press.

[DeJong and Spong, 1994] DeJong, G. and Spong, M. (1994). Swinging up the acrobot : an example of intelligent control. In *American Control Conference, 1994*, volume 2, pages 2158–2162.

[Dietterich, 2000] Dietterich, T. G. (2000). State abstraction in maxq hierachical reinforcement learning. *Artificial Intelligence Research*, (13) :227–303.

[Diuk et al., 2006] Diuk, C., Strehl, A. L., and Littman, M. L. (2006). A hierarchical approach to efficient reinforcement learning in deterministic domains. In Press, A., editor, *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 313 – 319.

[Ducrou and Eklund, 2007] Ducrou, J. and Eklund, P. (2007). Searchsleuth : The conceptual neighbourhood of an web query. In *CLA 2007*, pages 249–259. School of Information Systems and Technology University of Wollongong.

[Duquenne, 2007] Duquenne, V. (2007). What can lattices do for teaching math. ? In *CLA 2007*, pages 72–83. CNRS-ECP6, Université Pierre et Marie Curie.

[Dutech et al., 2001] Dutech, A., Buffet, O., and Charpillet, F. (2001). Multi-agent systems by incremental gradient reinforcement learning. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*. LORIA/INRIA-Lorraine, Vandoeuvre-lès-Nancy.

[Dutech et al., 2003] Dutech, A., Buffet, O., and Charpillet, F. (2003). Apprentissage par renforcement pour la conception de systèmes multi-agents réactifs. In *JFSMA 2003*.

[Dzeroski et al., 1998] Dzeroski, S., Raedt, L. D., and Blockeel, H. (1998). Relational reinforcement learning. In Inc., M. K. P., editor, *ICML'98*, pages 136 – 143. Dpt of Intelligent Systems, Jozef Stefan Institute Ljubljana, Slovenia ; Dpt of Computer Science, Katholicke Universiteit Leuven, Heverlee, Belgium.

[Dzeroski et al., 2001] Dzeroski, S., Raedt, L. D., and Driessens, K. (2001). Relational reinforcement learning. *Machine Learning*, (43) :7–52.

[E.Taylor and Stone, 2007] E.Taylor, M. and Stone, P. (2007). Representation transfer for reinforcement learning. In *AAAI 2007 Fall Symposium on Computational Approaches to Representation Change during Learning and Development*.

[Even-Dar and Mansour, 2003] Even-Dar, E. and Mansour, Y. (2003). Learning rates for q-learning. volume 5, pages 1–25.

[Ganter and Wille, 1999] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis : Mathematical Foundations*. Springer.

[Garcia and Ndiaye, 1997] Garcia, F. and Ndiaye, S. M. (1997). Apprentissage par renforcement en horizon fini ii : Analyse par la méthode de l'ode. volume 1.

[Givan et al., 2003] Givan, R., Dean, T., and Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2) :163–223.

[Godin et al., 1995] Godin, R., Missaoui, R., and Alaoui, H. (1995). Incremental concept formation algorithms based on galois (concept) lattices. In *Computational Intelligence*, volume 11, pages 246–267.

[Gordon, 2001] Gordon, G. J. (2001). Reinforcement learning with function approximation converges to a region. In *In Advances in Neural Information Processing Systems*, pages 1040–1046. The MIT Press.

[Jong and Stone, 2005] Jong, N. K. and Stone, P. (2005). State abstraction discovery from irrelevant state variables. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 752–757. University of Texas at Austin.

[Kaelbling and Littman, 1996] Kaelbling, L. P. and Littman, M. L. (1996). Reinforcement learning : A survey. *Journal of Artificial Intelligence Research*, 4 :237–285.

[Kersting, 2006] Kersting, K. (2006). *An Inductive Logic Programming Approach to Statistical Relational Learning*, volume 148 of *Frontiers in Artificial Intelligence and its Applications series (Dissertations)*. IOS Press.

[Kramer, 1996] Kramer, S. (1996). Structural regression trees. In *AAAI/IAAI, Vol. 1*, pages 812–819.

[Kuznetzov and Obiedkov, 2002] Kuznetzov, S. O. and Obiedkov, S. (2002). *Comparing performance of algorithms for generating concept lattices*, volume 14, pages 189–216.

[Lavrac and Dzeroski, 1993] Lavrac, N. and Dzeroski, S. (1993). *Inductive Logic Programming : Techniques and Applications*. Routledge, New York, NY, 10001.

[Liquière, 2006] Liquière, M. (2006). *Some Links Between Formal Concept Analysis and Graph Mining*. à paraître.

[Liquière and Sallantin, 1998] Liquière, M. and Sallantin, J. (1998). Structural machine learning with galois lattice and graphs. In Ed, M. K., editor, *ICML 1998*, pages 305–313. Lirmm, Montpellier, Morgan Kaufmann Ed.

[Lovejoy, 1991] Lovejoy, W. S. (1991). *A survey of algorithmic methods for partially observed Markov decision processes*, volume 28, pages 47–66.

[Maille, 1999] Maille, N. (1999). *Modèle logico-algébrique pour la fusion symbolique et l'analyse formelle*. PhD thesis.

[McCallum, 1996a] McCallum, A. (1996a). Hidden state and reinforcement learning with instance-based state identification. 26 :464–474.

[McCallum, 1996b] McCallum, A. (1996b). *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis.

[Mehta and Tadepalli, 2005] Mehta, N. and Tadepalli, P. (2005). Multi-agent shared hierarchy reinforcement learning. In *Proceedings of the ICML'05 Workshop on Rich Representations for Reinforcement Learning*.

[Minsky, 1961] Minsky, M. (1961). Steps toward artificial intelligence. In *Proceedings of the Institute of Radio Engineers*, pages 8–30. Dpt of Mathematics, MIT.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

[Muggleton, 1992] Muggleton, S. (1992). *Inductive Logic Programming*. The A.P.I.C. Series. Stephen Muggleton.

[Muggleton et al., 1994] Muggleton, S., Luc, and Raedt, D. (1994). Inductive logic programming : Theory and methods. *Journal of Logic Programming*, 19 :629–679.

[Munos, 1997a] Munos, R. (1997a). A convergent reinforcement learning algorithm in the continuous case based on a finite difference method. In *International Joint Conference on Artificial Intelligence IJCAI 1997*. CEMAGREF, LISC, Antony.

[Munos, 1997b] Munos, R. (1997b). *L'apprentissage par Renforcement Etude du cas continu*. PhD thesis.

[Munos, 2001] Munos, R. (2001). Efficient ressource allocation for markov decision processes. CMAP, Ecole Polytechnique, Palaiseau.

[Munos, 2003] Munos, R. (2003). Error bounds for approximate policy iteration. In *International Conference on Machine Learning ICML 2003*, pages 560–567. Centre de Mathématiques Appliquées, Ecole Polytechnique, Palaiseau, France.

[Munos and Moore, 2002] Munos, R. and Moore, A. W. (2002). Variable resolution discretization in optimal control. *Machine Learning*, 49 :291–323.

[Nguifo, 1993] Nguifo, E. M. (1993). *Concevoir une Abstraction à Partir de Ressemblances*. PhD thesis.

[Norris, 1978] Norris, E. (1978). An algorithm for computing the maximal rectangles in a binary relation. volume 23, pages 243–250.

[Nourine and Raynaud, 1999] Nourine, L. and Raynaud, O. (1999). A fast algorithm for building lattices. volume 71, pages 199–204.

[Papavassiliou and Russell, 1999] Papavassiliou, V. A. and Russell, S. (1999). Convergence of reinforcement learning with general function approximators. In *In Proceedings of the seventeenth international joint conference on artificial intelligence*, pages 97–4. Computer Science Division, University of California, Berkeley, Morgan Kaufmann.

[Parr, 1998] Parr, R. (1998). *Hierarchical control and learning for markov decision processes*. PhD thesis.

[Parr and Russel, 1997] Parr, R. and Russel, S. (1997). Reinforcement learning with hierarchies of machine. In *NIPS 1997*. Computer Science Division, University of California, Berkeley.

[Plotkin, 1970] Plotkin, G. (1970). A note on inductive generalisation. *Machine Intelligence*, 5.

[Plotkin, 1971] Plotkin, G. (1971). A further note on inductive generalisation. *Machine Intelligence*, 6.

[Precup et al., 2001] Precup, D., Sutton, R. S., and Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. In Kaufmann, M., editor, *ICML '01 : Proceedings of the Eighteenth International Conference on Machine Learning*, pages 417–424. School of Computer Science, McGill University, Montreal ; AT and T Labs Research, Florham Park.

[Raedt and Blockeel, 1997] Raedt, L. D. and Blockeel, H. (1997). Using logical decision trees for clustering. In *In Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 133–140. Springer-Verlag.

[Ravindran and Barto, 2001] Ravindran, B. and Barto, A. G. (2001). Symmetries and model minimization in markov decision processes. Technical report.

[Ravindran and Barto, 2002] Ravindran, B. and Barto, A. G. (2002). Model minimization in hierachical reinforcement learning. In Springer-Verlag, editor, *SARA 2002*, pages 196–211. Dpt of Computer Science, University of Massachussetts, Amherst.

[Ravindran and Barto, 2003a] Ravindran, B. and Barto, A. G. (2003a). An algebraic approach to abstraction in reinforcement learning. In University, Y., editor, *WALS03*, pages 109–114. Dpt of Computer Science, University of Massachussetts, Amherst.

[Ravindran and Barto, 2003b] Ravindran, B. and Barto, A. G. (2003b). Smdp homomorphisms : An algebraic approach to abstraction in semi markov decision processes. In *IJCAI 2003*, pages 1011–1016. Dpt of Computer Science, University of Massachussetts, Amherst.

[Ravindran and Barto, 2004] Ravindran, B. and Barto, A. G. (2004). Approximate homomorphisms : A framework for non-exact minimization in markov decision processes. In *Proceedings of the Fifth International Conference on Knowledge Based Computer Systems*. Dpt of Computer Science and Engineering, Indian Institute of Technology, Madras Chennai ; Dpt of Computer Science, University of Massachussetts, Amherst.

[Reynolds, 2000] Reynolds, S. I. (2000). Adaptive resolution model-free reinforcement learning : Decision boundary partitioning. In Kaufmann, M., editor, *ICML'2000*, pages 783–790. School of Computer Science, University of Burmingham.

[Reynolds, 2001] Reynolds, S. I. (2001). Adaptive representation methods for reinforcement learning. In Verlag, S., editor, *AI'2001*, pages 345–348. School of Computer Science, University of Burmingham.

[Reynolds, 2002] Reynolds, S. I. (2002). *Reinforcement Learning with Exploration*. PhD thesis.

[Ricordeau, 2003] Ricordeau, M. (2003). Q-concept-learning : Generalization with concept lattice representation in reinforcement learning. In Society, I. C., editor, *International Conference on Tools with Artificial Intelligence, ICTAI 03*, pages 316–323. Lirmm, Montpellier.

[Ricordeau, 2004] Ricordeau, M. (2004). Q-concept-learning : Généralisation à l'aide de treillis de galois dans l'apprentissage par renforcement. In *Proceedings of 14ème Congrés Francophone Reconnaissance des Formes et Intelligence Artificielle, RFIA 2004*, pages 385–394. Lirmm, Montpellier.

[Ricordeau and Liquière, 2006] Ricordeau, M. and Liquière, M. (2006). Algebraic results and bottom-up algorithm for policies generalization in reinforcement learning using concepts lattices. In *International Conference on Hybrid Systems and Applications, ICHSA 2006*. Lirmm, Montpellier.

[Ricordeau and Liquière, 2007] Ricordeau, M. and Liquière, M. (2007). Policies generalization in reinforcement learning using galois partitions lattices. In *Proceedings of Fifth International Conference on Concept Lattices and Their Applications*, pages 282–293. Lirmm, Montpellier.

[Russel and Norvig, 2003] Russel, S. and Norvig, P. (2003). *Artificial Intelligence A Modern Approach*. Pearson international edition edition.

[Sébastien, 2007] Sébastien, F. (2007). Camelis : Organizing and browsing a personal photo collection with a logical information system. In *CLA 2007*, pages 108–119. Irisa / Universitée de Rennes 1.

[Shani et al., 2005] Shani, G., Brafman, R. I., and Shimony, S. E. (2005). Model-based online learning of pomdps. In Heidelberg, S. B. ., editor, *Machine Learning : ECML 2005*, volume 3720 of *Lecture Notes in Computer Science*, pages 353–364. Ben-Gurion University, Beer-Sheva, Israel.

[Singh et al., 2004] Singh, S. P., James, M. R., and Rudary, M. R. (2004). Predictive state representations : A new theory for modeling dynamical systems. In Press, A., editor, *AUAI '04 : Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 512–519. Dpt of Computer Science and Engineering, University of Michigan, Ann Arbor.

[Stone, 2007] Stone, P. (2007). Learning and multiagent reasoning for autonomous agents. In *The 20th International Joint Conference on Artificial Intelligence*, pages 13–30.

[Stone et al., 2001] Stone, P., Littman, M. L., Singh, S., and Kearns, M. (2001). Attac-2000 : An adaptive autonomous bidding agent. Number 15, pages 189–206.

[Sutton, 1996] Sutton, R. S. (1996). Generalization in reinforcement learning : Successful examples using sparse coarse coding. In Press, M., editor, *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044. Dpt of Computer Science, University of Massachussetts, Amherst, David S. Touretzky ; Michael C. Mozer ; Michael E. Hasselmo.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning : An Introduction*. MIT press.

[Sutton et al., 1999] Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps : A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2) :181–211.

[Tadepalli et al., 2004] Tadepalli, P., Givan, R., and Driessens, K. (2004). Relational reinforcement learning : an overview. In *Proceedings of the ICML'04 workshop on Relational Reinforcement Learning*. Dpt of Computer Science, Oregon State University, Corvallis ; Dpt of Electrical and Computer Engineering, Purdue University ; Dpt of Computer Science, Katholicke Universiteit Leuven, Heverlee, Belgium.

[Tesauro, 1992] Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8 :257–277.

[Tsitsiklis and van Roy, 1997] Tsitsiklis, J. N. and van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5).

[van der Merwe et al., 2004] van der Merwe, D., Obiedkov, S., and Kourie, D. (2004). Add intent : A new incremental algorithm to build concept lattice. In *ICFCA04*. Dpt of Computer Science, University of Pretoria, Pretoria ; Institute für Algebra, Technische Universität, Dresden Germany.

[Watkins and Dayan, 1992] Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8 :279–292.

[Zenou, 2004] Zenou, E. (2004). *Localisation Topologique, Amers Visuels et Treillis de Galois*. PhD thesis.

Résumé en français
Nos travaux envisagent la question de la généralisation des politiques pour l'apprentissage par renforcement. Or, le problème de la généralisation à partir d'un ensemble d'exemples ou de la recherche de régularités sont des sujets déjà intensivement traités en apprentissage artificiel. Ainsi, nous nous proposons d'utiliser des techniques de généralisations contraintes par un biais de langage pour la généralisation des politiques. Celles-ci sont principalement basées sur les propriétés des treillis de Galois.
Premièrement, nous proposons un cadre algébrique général, formalisant la généralisation des politiques sous l'angle du partitionnement de l'ensemble des états de l'environnement observés par l'agent. Habituellement utilisées sur le treillis des parties, nous proposerons d'utiliser les techniques de treillis de Galois sur les treillis des partitions. Ceci nous permet de proposer un algorithme produisant, après un apprentissage par renforcement, des concepts intéressants du point de vue de la tâche. Ceux-ci peuvent être utilisés pour décrire la politique ou servir de motifs pour d'autres tâches dans un environnement similaire.
Enfin, nous proposerons une reformalisation de la tâche d'apprentissage ainsi qu'une méthode algorithmique associée que nous appellerons Q-Concept-Learning, consistant à appliquer un apprentissage sur l'ensemble des généralisations admises par le biais de langage utilisé. Dans ce contexte, nous discuterons et avancerons des solutions permettant à l'agent de générer des politiques « en ligne ». Les méthodes sont implémentées et expérimentées sur des problèmes académiques.

Titre en anglais
Generalization of Policies for Reinforcement Learning using Galois Lattices

Résumé en anglais
This work considers the generalization of the policies for reinforcement learning. The problem of generalizing from a set of examples or researching features are widely explored subjects in artificial intelligence. Thus, we propose to use generalization methods based on a language bias to generalize policies. Those are principally based on Galois lattices properties.
Firstly, we propose a general algebraical framework formalizing the generalization of policies. During its learning, the agent observes a set of the environment's states. We formalize the generalization of policies as a partitioning of this set.
Usually used on the powerset, we propose to use the Galois lattices methods on the partition lattice. This allows us to propose an algorithm which produces, after a reinforcement learning, interesting concepts from the task point of view. These can be used to describe the policy or as features for other similar tasks. Finally, we propose a new way to formalized a reinforcement learning task and an associated algorithmic method called Q-Concept Learning, consisting in applying a learning step on the all set of the generalizations available considering the used language bias. In this context, we will discuss about solutions that allow the agent to generate in line policies. Our methods are implemented ans tested on academical problems.

Discipline : Informatique

Mots clés : Apprentissage par Renforcement, Espaces de généralisation, Treillis de Galois