UNIVERSITÉ PARIS XIII - SORBONNE PARIS NORD

École Doctorale Galilée

UMR7030 - Laboratoire d'Informatique de Paris Nord

Nukkai, Paris, France

Thèse présentée pour obtenir le grade universitaire de Docteur

Discipline : Informatique

Malik KAZI AOUAL

Apprentissage multiannotateur et explications communes en logique du premier ordre

Multiannotator supervised learning and common explanations in first order logic

Sous la direction de Céline ROUVEIROL et Henry SOLDANO

Soutenance le 20/11/2024 devant le jury composé de :

Gauvain BOURGNE	Rapporteur
Christel VRAIN	Rapporteure
Véronique VENTOS	Examinatrice
Élisa FROMONT	Examinatrice
Nathalie PERNELLE	Présidente du jury
Céline ROUVEIROL	Directrice de thèse
Henry SOLDANO	Directeur de thèse

Remerciements

Dans un premier temps, je tiens à remercier les personnes qui ont gracieusement accepté de juger ce travail et d'assister à la soutenance. Tout d'abord, je remercie chaleureusement Nathalie Pernelle, professeure à l'Université Sorbonne Paris-Nord, qui a brillamment présidé mon jury de thèse.

Je tiens à remercier tout particulièrement les personnes qui ont accepté d'endosser le rôle de rapporteur, Christel Vrain, professeure à l'Université d'Orléans, et Gauvain Bourgne, professeur associé à Sorbonne Université, qui ont rédigé des commentaires très précieux pour améliorer ce travail. Je remercie également Elisa Fromont, professeure à l'Université de Rennes 1, qui m'a suivi depuis le début de ma thèse et qui a vu ce travail évoluer, jusqu'à assister à sa soutenance en tant qu'examinatrice.

Maintenant, je m'adresse à mes directeurs de thèse, qui m'ont accompagné au jour le jour et qui ont démontré un soutien sans commune mesure, sans qui je n'aurais jamais pu mener à bien ce travail. Je remercie tout d'abord ma directrice de thèse, Céline Rouveirol, professeure à l'Université Sorbonne Paris-Nord, qui m'a grandement aidé durant cette thèse, qui m'a permis de tenir dans les moments difficiles, qui m'a accordé sa confiance en toutes circonstances et qui m'a fait comprendre beaucoup de choses en logique. Je remercie également mon co-directeur de thèse, Henry Soldano, maître de conférences émérite à l'Université Sorbonne Paris Cité, pour m'avoir pris sous son aile dès mon stage de M2, pour m'avoir inculqué la rigueur scientifique et, surtout, pour ses commentaires toujours pertinents et constructifs sur mes présentations. Vous deux, je ne saurais jamais assez vous remercier pour tout ce que vous avez fait pour moi.

Je remercie également l'équipe NukkAI pour l'opportunité qu'ils m'ont offerte de travailler sur des sujets passionnants et pour la liberté qu'ils accordent aux chercheurs. Merci à Véronique Ventos, responsable de la recherche à NukkAI, pour m'avoir accueilli depuis le tout début dans l'entreprise et même avant, lorsqu'elle était encore à l'Université, merci pour ton énergie et ta bonne humeur contagieuse. Merci à Jean-Baptiste Fantun, directeur général de NukkAI, pour m'avoir permis de travailler sur des sujets qui me passionnent et pour avoir su trouver les bons mots aux bons moments. Merci à David Touitou, pourvoyeur d'idées lumineuses et puits sans fond de connaissances : les débats que l'on a sur des sujets de pointe m'ont grandement aidé à m'élever intellectuellement et, de ce fait, à atteindre le grade de docteur. Enfin, merci à Nouredine, d'abord camarade, puis ami et pour finir collègue, pour son soutien, sa réflexivité et cette relation propice à la sérendipité depuis des années.

Merci à mes proches : merci à ma famille, mes parents et mes sœurs, pour m'avoir épaulé, soutenu sans faille et pour toutes les tâches ingrates que vous vous êtes récupérées à ma place pour que je sois pleinement concentré sur mon travail. Merci à mes amis également, pour avoir été ceux qui écoutaient toutes mes plaintes sur la difficulté de l'exercice qu'est la thèse de doctorat.

Je tiens à remercier également l'Université Sorbonne Paris-Nord et le laboratoire LIPN pour m'avoir accueilli et pour m'avoir permis de travailler dans un environnement propice à la recherche.

Enfin, cette thèse CIFRE est financée en partie par l'ANRT, que je tiens à remercier pour leur soutien financier.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations

ASP . . *Answer set programming*
CNF . . *Conjunctive normal form*
DNF . . *Disjunctive normal form*
IA . . . . *Intelligence artificielle*
lgg . . . *Least general generalization*
LPO . . *Logique du premier ordre*
mce . . . *Minimal common explanation*
MCS . . *Minimal correction subset*
MDP . . *Markov decision process*
MHS . . *Minimal hitting set*
MUS . . *Minimal unsatisfiable subset*
PAC . . *Probablement approximativement correct*
PI . . . . *Prime implicant*
PLI . . . *Programmation logique inductive*
UPG . . *Unificateur le plus général*

Liste des notations

$C \models D$	D est conséquence logique de C
$C \preceq_\theta D$	C θ-subsume D
$C\theta$	C après substitution de ses variable par la substitution θ
$C \preceq_i D$	D est une instance de C
$C \prec_i D$	D est une instance stricte de C
$[p(X), q(X), r(X)]$	le motif relationnel $p(X) \wedge q(X) \wedge r(X)$
$[lgg(O), m]$	le motif $lgg(O)$ concaténé au motif m
$\dot{x}$	la conjonction des littéraux formés des atomes de l'interprétation x

Résumé

L'intelligence artificielle prend une place de plus en plus importante dans notre quotidien, posant de nombreux défis techniques, notamment dans le développement de systèmes capables d'intégrer la connaissance humaine, de considérer divers points de vue et d'expliquer leurs décisions. Assurer que l'IA et l'humain s'améliorent mutuellement est un enjeu majeur pour le développement de systèmes d'IA plus efficaces et plus éthiques. Cette thèse se concentre sur deux contributions principales pour relever ces défis : développer un modèle d'apprentissage supervisé intégrant l'identité des annotateurs et générer des explications compréhensibles pour les décisions de l'IA.

La première contribution s'intéresse à la prise de décision multijoueur au bridge, où différents joueurs annotent un ensemble d'observations et peuvent avoir différentes stratégies d'annotation face à des situations similaires. Nous avons développé un algorithme qui prend efficacement en compte l'identité des joueurs lors de la construction du modèle, appliqué à un scénario de bridge où les désaccords des joueurs face à une situation sont fréquents. Notre approche démontre une meilleure performance et gère efficacement l'augmentation du nombre d'annotateurs. Nous avons également utilisé les règles de notre modèle pour regrouper les joueurs en fonction de leurs comportements, facilitant la recherche de partenaires de jeu compatibles.

La deuxième contribution se concentre sur la création d'explications pour les actions d'un agent artificiel dans un jeu de bridge simplifié. Nous proposons une méthode pour générer des explications en logique du premier ordre, à la fois diverses et concises, qui sont communes à des sous-ensembles d'observations. Cette approche permet d'obtenir des explications plus générales et compréhensibles pour les humains, facilitant ainsi la compréhension du processus de décision de l'IA.

Mots clés : Apprentissage supervisé, Intelligence artificielle, Explicabilité, Bridge, Multijoueur, Multiannotateur, Programmation Logique Inductive, Logique.

Abstract

Artificial intelligence is becoming an increasingly important part of our daily lives, posing many technical challenges, not least in the development of systems capable of integrating human knowledge, considering diverse points of view and explaining their decisions. Ensuring that AI and humans improve each other is a major challenge for the development of more efficient and ethical AI systems. This thesis focuses on two main contributions to meeting these challenges : developing a supervised learning model incorporating annotator identity and generating comprehensible explanations for AI decisions.

The first contribution focuses on multiplayer decision making in bridge, where different players annotate a set of observations and may have different annotation strategies in front of similar situations. We have developed an algorithm that effectively takes player identity into account during model construction, applied to a bridge scenario where player disagreements about a situation are frequent. Our approach demonstrates better performance and effectively handles the increase in the number of annotators. We also used our model's rules to group players according to their behaviors, facilitating the search for partners.

The second contribution focuses on the creation of explanations for the actions of an artificial agent in a simplified bridge game. We propose a method for generating diverse and concise first-order logic explanations that are common to subsets of observations. This approach results in more general explanations that are understandable to humans, thus facilitating understanding of the AI's decision-making process.

Keywords : Supervised Learning, Artificial Intelligence, Explainability, Bridge, Multiplayer, Multiannotator, Inductive Logic Programming, Logic.

Introduction

L'intelligence artificielle (IA) prend de plus en plus de place dans notre vie quotidienne et ouvre un vaste champ de défis. Au cœur de ces défis se trouve l'ambition de créer une IA qui soit à la fois performante, mais qui soit également profitable à ses utilisateurs en leur faisant par exemple acquérir de nouvelles connaissances, dans le respect de principes éthiques. Cette thèse s'intéresse à des questions techniques concernant le développement d'une telle IA, d'une part capable d'intégrer de la connaissance humaine en tenant compte de la pluralité des points de vue. D'autre part, cette thèse s'intéresse au développement d'une IA capable d'expliquer ses décisions, ce qui représente un pilier essentiel pour une IA qui apprend des choses à ses utilisateurs. En effet, il est crucial que l'humain puisse comprendre les bases sur lesquelles l'IA prend ses décisions, pour pouvoir lui faire confiance, mais également pour avoir accès à de nouvelles connaissances.

Nous sommes convaincus que l'interaction entre l'humain et l'intelligence artificielle peut engendrer des bénéfices considérables des deux côtés. Pour l'IA, une interaction plus développée avec l'humain peut se traduire par une amélioration de ses performances et de sa sécurité, qui comprend de nombreux aspects tels que la résilience aux attaques adverses, la robustesse vis-à-vis des données bruitées, mais également sujet d'actualité, l'adaptation culturelle des modèles d'IA. En effet, les récentes avancées en IA ont montré que les modèles d'IA peuvent être biaisés envers certaines cultures. Par exemple, les agents conversationnels qui sont en plein essor, sont pour la plupart entraînés sur des corpus où l'anglais est prédominant. Par conséquent, ces agents peuvent montrer des biais latents favorisant les cultures occidentales [Tao+24], et le besoin de développer des modèles qui prennent en compte la diversité culturelle de ses utilisateurs est de plus en plus pressant.

Du côté humain, le développement d'une interaction entre l'humain et l'IA permettra un accès enrichi à des connaissances nouvelles, facilité non seulement par l'observation, mais aussi par un dialogue avec la machine. En 2017, AlphaGo [Sil+16], un programme d'intelligence artificielle développé par DeepMind, a non seulement marqué l'histoire en étant le premier ordinateur à battre un champion du monde de Go, mais a également permis aux joueurs de tous niveaux de découvrir de nouvelles stratégies pour le Go. Dans un match contre Lee Sedol, un joueur professionnel de Go, le programme a effectué un coup nommé « coup 37 » original et créatif qu'aucun humain n'avait jamais fait [1]. Ce coup, inexistant dans la base d'entraînement du programme et contraire à la théorie enseignée vieille de 5500 ans, a encouragé les joueurs de Go à revoir leur compréhension du jeu.

Ces exemples montrent que premièrement, malgré la complexité et la performance des modèles actuels, il est important de maintenir une interaction entre l'humain et la

1. https://www.johnmenick.com/writing/move-37-alpha-go-deep-mind.html

machine afin de garantir non seulement le progrès des modèles, mais également de veiller à ce que ces progrès se fassent dans le respect des valeurs et de la diversité de notre société. Deuxièmement, ces exemples soulignent l'importance de l'apprentissage mutuel : l'IA apprend de l'humain à travers des corrections, tandis que l'humain acquiert de nouvelles connaissances en analysant le comportement d'une IA qui est plus performante que lui dans un domaine.

Cependant, dans la plupart des interactions entre la machine et l'humain, le dialogue reste pour le moment limité. Par exemple, pour comprendre le coup 37 d'AlphaGo, il a fallu que des joueurs observent le déroulement de la partie pour comprendre pourquoi le modèle a choisi ce coup. Il convient donc de développer des modèles capables de communiquer leur raisonnement de façon plus explicite, afin de faciliter l'interaction entre l'humain et la machine. Pour que ces interactions se rapprochent autant que possible de la communication humaine, il faut qu'elles s'appuient sur des représentations *symboliques*, c'est-à-dire des formats compréhensibles par l'humain, afin de fluidifier l'échange d'informations. En s'appuyant sur un langage symbolique, nous pourrions faciliter la communication entre l'humain et la machine en favorisant une compréhension mutuelle plus intuitive, permettant aux machines de communiquer leur raisonnement, mais également permettant à l'humain de comprendre plus facilement les décisions que la machine effectue.

NukkAI, l'entreprise au sein de laquelle cette thèse a été réalisée, développe des méthodes d'IA qui « gardent l'humain dans la boucle ». Les méthodes réalisées chez NukkAI doivent être capables de prendre en compte l'expertise humaine et de fournir des explications compréhensibles par l'humain. Pour développer de telles méthodes et les tester, NukkAI a choisi le jeu de bridge comme « bac à sable ». Ce choix a été effectué en raison de la complexité de ce jeu considéré comme le roi des jeux de l'esprit, offrant un panel de possibilités de recherche alignées sur le développement d'une IA qui « garde l'humain dans la boucle », comme en atteste l'article [VT17]. Dans cet article, les auteurs mettent en lumière les défis posés par le bridge, dans un premier temps en parlant des défis techniques, avec la nature d'information incomplète du jeu et le fait que le jeu se joue à quatre joueurs répartis en équipe de deux. Deuxièmement, coté humain, on retrouve également des difficultés avec la nécessité que les joueurs explicitent leur raisonnement, mais également des difficultés psychologiques où les joueurs doivent connaître leur adversaire aussi bien au niveau du style de jeu que du niveau de jeu. De plus, les équipes formant des paires de joueurs sont souvent amenées à changer au cours d'une carrière de joueur de bridge, alors les joueurs doivent s'adapter à leur nouveau partenaire ce qui nécessite une phase d'apprentissage essentielle. Dans [LRV18], les auteurs soulignent également l'importance de l'apport de la connaissance humaine dans un problème d'apprentissage supervisé pour une décision au bridge, en montrant que l'utilisation d'une théorie du domaine (définie par l'humain) par les méthodes de Programmation Logique Inductive permettaient une amélioration des performances en test par rapport aux méthodes d'apprentissage supervisé classiques.

Dans [VT17], il est également souligné que la conception d'une IA surpassant les champions mondiaux de bridge représenterait une avancée majeure dans le domaine de l'intelligence artificielle. Dans une récente avancée, l'entreprise NukkAI a démontré les progrès significatifs réalisés dans le domaine des robots de bridge. Au cours d'un challenge,

le robot de bridge développé chez NukkAI a battu huit champions du monde dans le jeu de la carte. Cette performance fait suite à d'importantes contributions à la recherche, présentées dans [BRV20 ; LTV20 ; CLV21]. Lors du challenge, un module d'explicabilité, appelé l'*explainer* et développé par NukkAI a été utilisé [2]. Il prend en entrée des traces de parties jouées par des joueurs ou par le robot de bridge et explicite les stratégies qui ont été suivies, en langage naturel.

Les contributions de cette thèse explorent les deux axes présentés ci-dessus : nous proposons deux cadres dans lesquels l'interaction entre l'humain et la machine leur permet de s'améliorer mutuellement. En prenant comme cas d'étude le bridge, nos travaux s'inscrivent dans la recherche effectuée chez NukkAI. La première contribution intègre de la connaissance humaine dans l'apprentissage d'un modèle afin de le rendre plus performant et permet d'établir des groupes de joueurs pertinents ayant des styles de jeu similaires, et la seconde contribution génère des explications de l'optimalité des actions prises par un agent artificiel au bridge, pouvant alors s'appliquer au robot de bridge développé chez NukkAI pour expliquer ses décisions, permettant ainsi aux personnes qui voudraient apprendre du robot de comprendre pourquoi il a pris une décision.

Financement. Cette thèse est financée en partie par l'ANRT à travers une convention CIFRE. Le financement est également assuré par l'entreprise NukkAI, qui est le partenaire industriel de cette thèse.

Résumés des contributions

Apprentissage multiannotateur supervisé

Dans la première contribution de cette thèse, nous nous intéressons à la construction de modèles prédictifs qui s'adaptent à la diversité des points de vue des annotateurs et utilisateurs. Pour ce faire, nous utilisons le jeu du bridge, dans lequel les joueurs peuvent avoir des styles de jeu très différents. Nous avons à notre disposition un ensemble de données qui correspondent à des situations de bridge face auxquelles différents annotateurs doivent prendre une décision. La différence d'âge, de sexe, de niveau de jeu, ou de style de jeu des joueurs peut entraîner des styles de jeu très différents et donc des décisions différentes face à des situations similaires. Le but est d'apprendre un modèle capable de prédire la décision d'un joueur face à une nouvelle situation, nous avons appelé ce cadre l'*apprentissage multiannotateur supervisé*. Nous montrons dans cette contribution la nécessité de prendre en compte l'identité de l'annotateur lorsqu'on se trouve dans ce cadre, ce qui nous a mené à développer un algorithme d'apprentissage qui prend en compte de façon efficace l'identité de l'annotateur.

Contexte applicatif. Nous avons à notre disposition un ensemble de données qui correspondent à des mains de bridge face auxquelles 10 annotateurs, qui sont des joueurs de bridge, doivent prendre une décision et les annotateurs n'étiquettent pas forcément les mêmes données. Cette situation au bridge se déroule juste après la phase des enchères,

2. `https://youtu.be/yrDZ_ssqxl0?si=HMo8JMX3sMyMMW16&t=420`

plus précisément à l'entame, où le joueur doit décider quelle carte jouer en premier. Il a le choix entre jouer une carte majeure (♠, ♡) ou une carte mineure (♣, ♢). Le joueur doit prendre en compte le fait qu'il a 3 cartes pour chaque couleur majeure et 4 cartes pour une des couleurs mineure. Lorsque cette situation est présentée face à des joueurs de bridge, il y a des désaccords sur la couleur à jouer, car les couleurs majeures sont souvent plus longues dans le camp des défenseurs, alors qu'à l'inverse les mineures sont souvent plus courtes, alors que le joueur qui entame a lui même plus de cartes mineures que de majeures.

Cadre d'apprentissage. Nous nous plaçons dans le cadre de l'apprentissage supervisé de concept [Mit82]. Le but est d'apprendre un modèle sous forme d'hypothèse composée de règles $c \leftarrow t$, où c est le concept à apprendre et t est une conjonction d'atomes issus d'un langage d'hypothèse. Nous disposons d'un ensemble E d'exemples avec la valeur de vérité de c, répartis en exemples positifs (E^+), où c est vrai, et en exemples négatifs (E^-), où c est faux. L'objectif est de trouver une hypothèse qui couvre tous les exemples positifs et rejette tous les exemples négatifs.

Méthode. Les exemples sont de la forme (j, x), où j est l'identifiant du joueur et x est la situation de jeu. Ils sont répartis en E^+ pour les cas où le joueur a joué une carte majeure, et E^- pour les cas où il a joué une carte mineure. L'apprentissage multiannotateur vise à trouver une hypothèse H qui couvre les exemples positifs et rejette les exemples négatifs, où chaque règle $r_i \in H$ est de la forme $c \leftarrow 1_C \wedge t$, où 1_C est l'ensemble des annotateurs pour lesquels la règle s'applique.

L'approche standard, appelée approche *directe*, inclut l'identité du joueur dans le langage des hypothèses. Notre approche, appelée approche *parcimonieuse*, cherche des règles parcimonieuses, où C est le sous-ensemble de joueurs pour lesquels la règle est *porteuse* et *valide*. Une règle est porteuse pour un joueur si elle couvre au moins un exemple positif de ce joueur et valide si elle rejette tous ses exemples négatifs. Nous présentons un algorithme pour construire des règles parcimonieuses, utilisant l'identité du joueur uniquement dans la fonction de guidage, sans l'inclure dans l'espace de recherche des règles.

Résultats. Dans nos expérimentations, nous comparons l'approche parcimonieuse avec différentes approches, notamment l'approche directe, mais également une approche *individuelle* dans laquelle une règle peut concerner soit un seul joueur, soit tous les joueurs, ainsi que l'approche *ignorante* qui ne prend pas du tout en compte l'identité du joueur. Les résultats montrent que l'approche parcimonieuse permet de construire un modèle avec une meilleure justesse que les différentes approches comparées, excepté la forêt aléatoire, mais une forêt aléatoire n'est pas interprétable contrairement à un modèle par approche parcimonieuse. Nous montrons également tout le sens de notre approche en augmentant artificiellement le nombre de joueurs, avec un effet sur le temps d'apprentissage négligeable, tandis que les approches comparatives sont fortement impactée par l'augmentation du nombre d'annotateurs et certaines deviennent rapidement inutilisables. Nous montrons également à l'aide d'un apprentissage non supervisé sur les joueurs à partir règles ap-

prises par notre algorithme la pertinence des associations de joueurs dans le contexte des règles en remplaçant leur identité dans les données par l'identifiant de leur cluster, et en apprenant un modèle sur ces données, permettant d'établir des groupes de joueurs qui ont des stratégies similaires.

Explications abductives communes minimales

Dans notre seconde contribution, nous nous intéressons à la génération d'explications des décisions d'un agent artificiel jouant à un jeu de bridge simplifié. Nous proposons une méthode de génération d'explications en logique du premier ordre qui explique l'optimalité d'une action effectuée par cet agent. Le domaine des explications en IA est un domaine en pleine expansion, et il est crucial de développer des méthodes qui génèrent des explications compréhensibles par l'humain. Notre méthode de génération d'explications est basée sur la génération d'*explications communes minimales*, c'est-à-dire des explications qui sont partagées par un groupe d'observations, ce qui les rend plus générales, donc plus proches des explications que pourraient donner un humain. Cette contribution aborde donc le problème de l'amélioration des connaissances de l'humain par l'observation de la machine, où l'objectif est de construire des systèmes d'IA capables de générer des explications en termes compréhensibles, facilitant ainsi le passage de l'information de la machine vers l'humain et permettant à l'humain d'accéder à de nouvelles connaissances.

Contexte applicatif. Nous nous intéressons à un jeu de cartes simple qui est une restriction du jeu de bridge à une couleur et 13 cartes. Toujours dans la phase du jeu de la carte, on s'intéresse au point de vue du déclarant, qui est le joueur qui a remporté la phase des enchères et qui doit réaliser un certain nombre de plis. Nous avons à notre disposition un agent artificiel pour jouer en tant que déclarant qui connaît la récompense associée à chaque action, et qui doit choisir l'action optimale, c'est-à-dire celle qui maximise la récompense attendue en nombre de plis. L'objectif est de générer des explications qui expliquent l'optimalité de l'action choisie par l'agent.

Cadre des explications. Nous nous plaçons ici dans le cadre des explications locales abductives. *Locales*, car nous cherchons à expliquer une décision d'un agent pour une observation donnée. *Abductives*, car on cherche les raisons qui ont mené l'agent à prendre sa décision sur l'observation. Ici, l'observation est l'état du jeu au moment où l'agent doit jouer une carte et la décision est la carte jouée par l'agent. Une explication locale est couramment obtenue en entraînant un modèle explicatif (appelé modèle *surrogé*) pour approximer le modèle de décision original dans le voisinage de l'observation à expliquer. Il existe un cadre formel pour les explications locales abductives où l'on considère que le modèle surrogé doit être exprimable par une formule logique. Ce formalisme permet de générer un nombre élevé d'explications d'une même décision, qui présentent des garanties de concision, élément essentiel pour la compréhension humaine de l'explication. Ce cadre formel a été défini dans [SCD18] et [DH20], et sera présenté dans l'état de l'art des explications en IA dans le chapitre 5 et c'est dans ce cadre que nous nous placerons.

Méthode. Dans le but d'expliquer l'optimalité des décisions de l'agent artificiel, nous proposons une notion originale des explications communes minimales dans le cadre présenté ci-dessus. Ces explications sont communes à un ensemble d'observations qui partagent la même étiquette que l'observation à expliquer. Pour notre problème, ces observations sont l'ensemble des trajectoires de jeu optimales partant de l'action de l'agent artificiel à expliquer.

Nous présentons une formalisation de telles explications en logique du premier ordre, en exprimant tout d'abord nos observations comme étant des interprétations d'un langage du premier ordre. Une explication commune à un ensemble d'observations de même étiquette exprimées en logique du premier ordre est alors un motif relationnel dont ces observations sont modèles, et dont les observations d'une étiquette différente ne sont pas modèles. Nous introduisons alors la notion d'explications communes subset-minimales. Une explication commune est subset-minimale si tout motif strictement inclus dans cette explication n'est pas une explication commune. Cette définition des explications commune subset-minimales provient de la nécessité qu'une explication soit concise, cette propriété étant l'une des quatre propriétés attendues d'une explication.

Toutefois, dans le cadre de la logique du premier ordre, et en raison de notre objectif qui est de créer des explications qui sont communes à un ensemble d'observations, des variables apparaissent dans les explications et peuvent rendre la compréhension du motif ardue dans certains cas. Par conséquent, nous avons introduit un concept supplémentaire, les *explications communes leq-minimales*, qui sont les instanciations maximales des explications communes subset-minimales. Ces explications sont plus facilement compréhensibles lorsqu'elles sont présentées à un humain. Nous avons défini les explications communes leq-minimales d'un ensemble d'observations comme les sous-ensembles minimaux de la *lgg* de cet ensemble d'observations, qui est le motif le plus spécifique qui le couvre et proposé plusieurs algorithmes pour les générer efficacement. Enfin, notre méthode génère un grand nombre d'explications, nous avons donc proposé une méthode de sélection d'un nombre restreint d'explications leq-minimales qui soient diverses.

Résultats. Nous avons testé notre méthode sur deux situations de bridge différentes, correspondant à deux distributions de cartes différentes. Les résultats montrent que notre méthode d'approximation de la *lgg* est efficace pour générer l'explication commune la plus grande, et que notre méthode de recherche des explications subset-minimales et d'instanciation maximale permet de générer des explications communes leq-minimales. Nous montrons également que notre méthode de sélection d'explications diverses permet de sélectionner, en fonction des préférences de l'utilisateur, des explications qui sont différentes les unes des autres, et qui sont plus compréhensibles pour un humain. Enfin, nous posons les bases des perspectives de cette contribution, en analysant les explications communes aux deux distributions de cartes, donnant des pistes pour la définition d'explications communes dans un monde partiellement observable.

Organisation du document

Ce manuscrit est organisé de la façon suivante : les chapitres 1 2 et 3 qui forment la partie I, détaillent les prérequis pour les contributions de cette thèse. Le chapitre 1 est

un chapitre contenant les notions élémentaires de bridge nécessaires pour comprendre les contributions de cette thèse, ainsi que les notions des processus de décision markoviens. Le chapitre 2 est un chapitre rappelant les notions de logique propositionnelle et du premier ordre utilisées dans cette thèse. Le chapitre 3 est un chapitre présentant les notions d'apprentissage supervisé et les notions de Programmation Logique Inductive qui sont utilisées dans cette thèse. La partie II contient les contributions de cette thèse, avec le chapitre 4 qui présente la première contribution, appelée *Apprentissage multiannotateur supervisé*, le chapitre 5 présente un état de l'art des explications en IA permettant d'introduire la deuxième contribution alors développée dans le chapitre 6, appelée *Explications abductives communes minimales* et enfin le chapitre 7 présente les perspectives de recherche futures liées à cette thèse.

Première partie

Prérequis

Chapitre 1

Bridge et processus de décision Markovien

Dans ce chapitre, nous introduisons des notions diverses nécessaires à la compréhension des contributions présentées dans cette thèse mais qui ne s'intègrent pas directement dans les chapitres suivants des prérequis, ce chapitre parlera de deux sujets très différents : tout d'abord nous présenterons les bases du jeu du bridge en section 1.1, afin de rendre familier le lecteur avec les règles du jeu pour comprendre les deux contributions présentées dans les chapitres 4 et 6, puis nous introduirons les processus de décision Markoviens en section 1.2, qui seront utilisés dans le chapitre 6.

1.1 Jeu du bridge

Le bridge est un jeu de cartes qui se joue à quatre joueurs Nord, Sud, Est et Ouest répartis en deux équipes de deux partenaires que l'on appelle le *déclarant* (Nord, Sud) et le *défenseur* (Est, Ouest), les partenaires étant assis face à face. Le jeu se joue avec 52 cartes classiques à quatre couleurs (trèfle ♣, carreau ♢, cœur ♡ et pique ♠) et de hauteur allant du 2 à l'As.

Le bridge se divise en deux phases principales : les enchères et le jeu de la carte. Aux enchères, les joueurs communiquent pour déterminer le contrat final, tandis que le jeu de la carte est une phase de jeu où l'équipe du déclarant tente de réaliser le contrat fixé pendant les enchères, alors que l'équipe du défenseur tente de l'en empêcher. Nous allons décrire de façon plus détaillée le déroulement de ces deux phases.

Les enchères

Les enchères sont une séquence de communication codée entre les partenaires dont l'objectif est de déterminer le contrat final, c'est-à-dire le nombre de plis que l'équipe du déclarant s'engage à réaliser et la couleur qui sera l'atout (ou s'il s'agira d'un contrat à Sans-Atout), l'effet de l'atout sera détaillé dans la section suivante.

La séquence d'enchères est circulaire, le donneur prend la parole en premier et les joueurs doivent parler à tour de rôle. Chaque joueur ayant la possibilité de passer ou d'enchérir un contrat. Les paroles possibles sont les suivantes : (i) De 1♣ à 7♣, (ii) De

South	West	North	East
1♣	P	1♡	1♠
2♡	2♠	P	P
P			

TABLE 1.1 – Exemple de séquence d'enchères au bridge

1♢ à 7♢, (iii) De 1♡ à 7♡, (iv) De 1♠ à 7♠, (v) De 1SA à 7SA qui veut dire sans atout, (vi) *Passe* (P).

Le chiffrage de l'enchère correspond au nombre de plis que le déclarant s'engage à réaliser, plus six. Par exemple, une enchère de 2♢ signifie que le déclarant s'engage à réaliser huit plis avec les carreaux comme atout. De plus, les couleurs sont ordonnées de la plus faible à la plus forte : ♣, ♢, ♡, ♠, SA. Une enchère doit être supérieure à la dernière enchère effectuée, par exemple 3♢ est supérieur à 2♢, et 2♡ est supérieur à 2♢. Un exemple de séquence d'enchères est donné dans le tableau 1.1. Le joueur ayant nommé la couleur du contrat final en premier sera le déclarant. Dans l'enchère de la table 1.1, c'est *East* qui sera déclarant.

Les enchères sont une phase de jeu très codifiée, suivant des conventions précises. Les décisions d'un joueur dans une séquence d'enchères sont un moyen de communiquer des informations à son partenaire. Ces informations sont codifiées dans la convention choisie par l'équipe et tous les joueurs de la partie y ont accès.

Les éléments du jeu de bridge sont visibles Figure 1.1 avec à gauche, le support pour la communication aux enchères, en bas à droite une main de bridge étalée et en haut à droite la séquence d'enchères qui a mené au contrat 1♡. Dans l'étude de cas du chapitre 6, nous utiliserons la représentation abstraite de la figure 1.2 pour décrire les états du jeu de bridge.

FIGURE 1.1 – Les différents éléments d'un jeu du bridge

Le jeu de la carte

La phase du jeu de la carte est la seconde phase principale du bridge et fait suite à la phase d'enchères. C'est durant cette phase que l'équipe du déclarant cherche à réaliser le

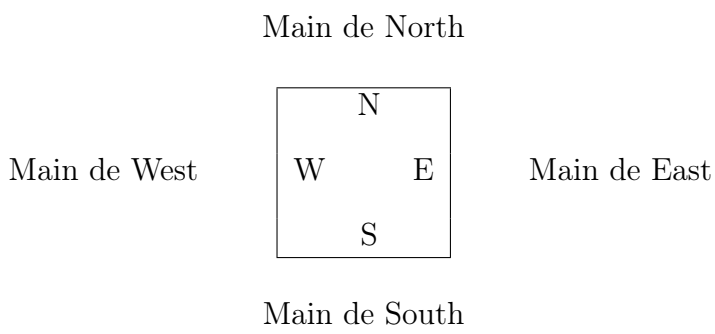


FIGURE 1.2 – Représentation schématique d'une table de bridge

contrat fixé lors des enchères et où l'équipe du défenseur cherche à l'en empêcher.

Après la conclusion des enchères, le joueur à gauche du déclarant joue la première carte, ouvrant le premier pli. Le partenaire du déclarant, appelé le « mort », étale alors toutes ses cartes face visible sur la table. Le mort ne participera pas activement à la phase du jeu de la carte, c'est le déclarant qui jouera à la fois ses propres cartes et celles du mort.

Chaque joueur, à son tour dans le sens des aiguilles d'une montre, joue une carte de sa main pour essayer de remporter le pli. Les règles du jeu exigent que les joueurs suivent la couleur demandée, c'est-à-dire qu'ils jouent une carte de la même couleur que la première carte jouée pour ce pli, s'ils en ont une. Si un joueur n'a pas de carte de la couleur demandée, il peut jouer une carte d'une autre couleur, y compris une carte de l'atout. La carte la plus haute remporte le pli sauf si un atout a été joué, auquel cas l'atout le plus haut remporte le pli. Le joueur qui remporte le pli commence le pli suivant.

Quelques mots de vocabulaire

Ici, nous listons quelques mots de vocabulaire utilisés dans le bridge et dans les différentes contributions de cette thèse :

- *Donne* : La distribution des cartes à chaque joueur.
- *Honneurs* : Les cartes du Valet au Roi.
- *Gros honneurs* : Les cartes Roi et As.
- *Petites cartes* : Les cartes de 2 à 10.
- *Pli* : Un ensemble de quatre cartes jouées, une par joueur. Le joueur qui a joué la carte la plus haute de la couleur demandée remporte le pli.
- *Atout* : La couleur qui a été déclarée comme étant plus forte que les autres couleurs. Une carte d'atout bat n'importe quelle carte d'une autre couleur.
- *Points d'honneur* : Les points d'honneur sont attribués aux cartes de l'As au Valet. L'As vaut 4 points, le Roi 3 points, la Dame 2 points et le Valet 1 point.
- *Distribution d'une main* : La distribution d'une main est le nombre de cartes dans chaque couleur.
- *Main équilibrée* : Une main est équilibrée si elle possède une distribution en (4,3,3,3), (4,4,3,2) ou (5,3,3,2).

— *Main semi-équilibrée* : Une main est semi-équilibrée si elle possède une distribution en (5,4,2,2) ou (6,3,2,2).

— *Couleur majeure* : Les couleurs majeures sont le ♡ et le ♠.

— *Couleur mineure* : Les couleurs mineures sont le ♣ et le ♢.

1.2 Processus de décision Markovien

Dans le chapitre 6, l'agent que nous utilisons pour générer des explications prend ses actions en résolvant un processus de décision markovien (MDP).

Un MDP [BCC57 ; How60] est un modèle utilisé pour décrire des environnements où un agent prend des décisions en fonction d'un ensemble d'états, d'actions et de récompenses. Son but est d'optimiser un certain critère, généralement la somme espérée des récompenses futures.

Définition 1.1 (MDP)**.** *Un MDP est défini par un n-uplet $\langle S, A, T, R\rangle$ tel que :*

1. *S est l'ensemble de tous les états possibles dans lesquels l'agent peut se trouver.*
2. *A est l'ensemble de toutes les actions que l'agent peut effectuer.*
3. *$T : S \times A \times S \to [0, 1]$ est la fonction de transition qui décrit la probabilité de passer de l'état s à l'état s' après avoir effectué l'action a.*
4. *$R : S \times A \to \mathbb{R}$ est la fonction de récompense qui donne la récompense immédiate reçue après avoir effectué une action a dans l'état s.*

On considère dans ce qui nous intéresse que la récompense future a autant d'importance que la récompense immédiate, ainsi nous n'introduisons pas de facteur de décompte de la récompense.

Le but d'un MDP est d'apprendre une *politique* qui associe à chaque état s une action a (ou une distribution de probabilité sur les actions). L'objectif est de trouver une politique optimale π^* qui maximise la valeur attendue des récompenses cumulées.

La *fonction de valeur d'état* associée à une politique π, notée $V^\pi(s)$, est définie comme étant l'espérance des récompenses cumulées obtenues en suivant la politique π depuis l'état s. La relation entre la valeur d'un état et les valeurs des états qui peuvent lui succéder est décrite récursivement n par l'équation de Bellman :

$$V^\pi(s) = R\big(s, \pi(s)\big) + \sum_{s' \in S} T\big(s, \pi(s), s'\big)\, V^\pi(s'),$$

La *fonction de qualité* (ou fonction de *q-values*) associée à une politique π, notée $Q^\pi(s, a)$, est définie comme étant l'espérance des récompenses cumulées obtenues en prenant l'action a depuis l'état s, puis en suivant la politique π et est définie comme suit :

$$Q^\pi(s, a) = R(s, a) + \sum_{s' \in S} T(s, a, s')\, V^\pi(s').$$

L'une des méthodes fondamentales pour résoudre un MDP est la technique d'*itération de valeur*. Cette méthode vise à approximer la fonction de valeur de manière itérative

jusqu'à la convergence, qui se base sur la fonction de valeur, qui est mise à jour pour chaque itération en utilisant l'équation de Bellman. Une autre approche pour la résolution de MDP est l'*itération de politique*, qui se concentre directement sur l'optimisation d'une politique initiale de façon itérative, jusqu'à converger vers la politique optimale π^*.

1.3 Résumé

Dans ce chapitre, nous avons introduit deux sujets très différents : tout d'abord nous avons présenté les notions élémentaires de bridge en section 1.1, afin de rendre familier le lecteur avec les règles du jeu pour comprendre les notions de bridge utilisées dans les chapitres 4 et 6. Les règles présentées dans ce chapitre ne sont qu'un sous-ensemble des règles du bridge, mais sont suffisantes pour comprendre les contributions de cette thèse qui présentent des études de cas qui se déroulent durant la phase du jeu de la carte. En revanche, l'étude de cas de la contribution du chapitre 4 se déroule à l'entame, juste après une suite d'enchères spécifique qui a une influence sur la décision du joueur à l'entame. Il est donc nécessaire de comprendre également les règles des enchères pour comprendre les motivations de cette contribution.

Ensuite, nous avons introduit les processus de décision Markoviens en section 1.2, qui seront utilisés dans l'étude de cas du chapitre 6, où le but est de générer des explications pour les décisions prises par un agent artificiel durant la phase du jeu de la carte, qui résout un MDP. Cet agent en revanche n'inclut pas les enchères dans son modèle, mais se concentre uniquement sur la phase du jeu de la carte.

Chapitre 2

Logique

Dans cette thèse, la logique est un outil central pour la représentation des connaissances et des données. En particulier, nous utilisons la logique pour représenter les données et les modèles appris des chapitres 4 et 6, mais également en tant que formalisme pour caractériser les explications dans le chapitre 6. Dans ce chapitre, nous introduisons les notions de logique propositionnelle et de logique du premier ordre en reprenant en partie la structure proposée par [Nie+97]. Pour un approfondissement sur ces sujets, nous recommandons aux lecteurs de consulter l'ouvrage de référence [Nie+97].

2.1 Logique propositionnelle

La logique propositionnelle, dont les bases ont été posées par Boole [Boo09] en 1847, est un formalisme capable d'exprimer une forme simple de raisonnement, manipulant des *propositions*. Une proposition est un fait qui peut prendre la valeur *vrai* ou *faux*. L'intérêt de la logique propositionnelle est de se débarrasser de l'ambiguïté de la langue naturelle pour l'expression d'énoncés. Par exemple, l'énoncé en langue naturelle « jeudi et vendredi sont des jours de réunion » est ambigu. Il peut soit vouloir dire « il y a des réunions les deux jours à la fois », soit « il y a des réunions soit le jeudi, soit le vendredi, soit les deux ». En logique propositionnelle, on peut exprimer ces deux énoncés par respectivement $j \wedge v$ et $j \vee v$, où j et v sont des propositions qui dénotent « il y a une réunion le jeudi » et « il y a une réunion le vendredi ».

2.1.1 Syntaxe

En logique propositionnelle, la syntaxe est caractérisée par un *alphabet*. Cet alphabet inclut un ensemble non vide d'atomes tels que p, q, ..., divers connecteurs : $\neg$ pour la négation, $\wedge$ pour le ET logique, $\vee$ pour le OU logique, $\rightarrow$ pour l'implication, et $\leftrightarrow$ pour la réciprocité. En complément, l'alphabet comprend des symboles de ponctuation tels que les parenthèses – (– et –) –. Les constantes $\top$ et $\bot$ désignent resp. une proposition qui est toujours vraie et une proposition qui est toujours fausse.

En utilisant des séquences de ces symboles, on peut construire des *formules*. Les formules dont la syntaxe est correcte sont appelées les *formules bien formées*. Un *littéral*

dans une formule est soit un atome, auquel cas, c'est un *littéral positif*, soit la négation d'un atome auquel cas, c'est un *littéral négatif*.

Pour plus de détails sur ces formules, nous proposons au lecteur de se référer à [Nie+97].

Définition 2.1 (Langage propositionnel)**.** *Le langage propositionnel donné par un alphabet est l'ensemble des formules bien formées qui peuvent être construites à partir des symboles de l'alphabet.*

Définition 2.2 (Vocabulaire d'un langage propositionnel)**.** *Le vocabulaire $\mathcal{V}$ d'un langage propositionnel est l'ensemble des atomes de ce langage.*

2.1.2 Sémantique

La sémantique de la logique propositionnelle désigne le processus qui permet d'évaluer une formule à *vrai* ou *faux*. Son évaluation est déterminée par l'attribution de valeurs booléenne à l'ensemble des sous-formules composant une formule. Ainsi, si l'on doit déterminer la valeur de $(p \vee q) \rightarrow r$, on doit déterminer la valeur de vérité de $(p \vee q)$ et la valeur de r et pour déterminer la valeur de $(p \vee q)$, on doit déterminer la valeur de p et la valeur de q. La valeur d'une formule dépend donc de la valeur des formules dont la granularité est la plus petite, c'est-à-dire les atomes, et l'affectation à *vrai* ou *faux* des atomes est appelée une *interprétation*.

Définition 2.3 (Interprétation)**.** *Soit L un langage propositionnel. Soit $\mathcal{V}$ l'ensemble des atomes de L. Une interprétation de L est une fonction qui associe à chaque atome de $\mathcal{V}$ une valeur de vérité, soit vrai (noté T), soit faux (noté F).*

On représente de manière simplifiée une interprétation I comme étant l'ensemble des atomes dont la valeur de vérité est *vrai* : $I = \{a \in \mathcal{V} \mid I(a) = T\}$.

Définition 2.4 (Modèle d'une formule)**.** *Soit ϕ une formule et I une interprétation. On dit que I est un modèle de ϕ si I rend vrai ϕ. On dit alors que I satisfait ϕ ou que ϕ a I comme modèle.*

Dans le cas où l'on a un ensemble de formules, le modèle d'un ensemble de formules est défini de la façon suivante :

Définition 2.5 (Modèle d'un ensemble de formules)**.** *Soit Σ un ensemble de formules et I une interprétation. I est un modèle de Σ si I est un modèle de toutes les formules $\phi \in \Sigma$.*

Exemple 2.1. *Soit un vocabulaire $\mathcal{V} = \{p, q, r\}$ et l'ensemble de formules $\Sigma = \{r; (p \vee q); (\neg r \vee q)\}$. L'interprétation $I = \{r, q\}$ (r et q sont vrais dans I) est un modèle de Σ, car elle est un modèle de toutes les formules de Σ. L'interprétation $I' = \{r, p\}$ en revanche n'est pas un modèle de Σ, car elle n'est pas un modèle de la troisième formule de Σ.*

L'introduction des modèles d'un ensemble de formules nous permet d'introduire un concept important qui est la notion de *conséquence logique* définie de la façon suivante :

Définition 2.6 (Conséquence logique)**.** *Soit Σ un ensemble de formules et ϕ une formule. ϕ est dite conséquence logique de Σ (écrit $\Sigma \models \phi$) si tous les modèles de Σ sont aussi des modèles de ϕ.*

Si ϕ n'est pas conséquence logique de Σ, alors on note $\Sigma \not\models \phi$.

On parlera également d'*implication logique* pour désigner la conséquence logique, qui diffère de l'implication syntaxique ($\rightarrow$). Le théorème de la déduction permet de faire le lien entre la conséquence logique et l'implication syntaxique.

Théorème 2.1. *Soit Σ un ensemble de formules, et ϕ et ψ deux formules. On a alors $\Sigma \cup \{\phi\} \models \psi$ (noté également $\Sigma, \{\phi\} \models \psi$) si et seulement si $\Sigma \models (\phi \rightarrow \psi)$.*

De plus, on peut également définir la relation entre la conséquence logique et l'insatisfiabilité :

Proposition 2.1. *Soit Σ un ensemble de formules et ϕ une formule. $\Sigma \models \phi$ si et seulement si $\Sigma \cup \{\neg\phi\}$ est insatisfiable, noté $\Sigma \cup \{\neg\phi\} \models \bot$.*

Définition 2.7 (Équivalence logique)**.** *Deux formules ϕ et ψ sont logiquement équivalentes si $\phi \models \psi$ et $\psi \models \phi$. De même, deux ensembles de formules Σ et Γ sont logiquement équivalents si $\Sigma \models \Gamma$ et $\Gamma \models \Sigma$. On note alors $\phi \equiv \psi$ et $\Sigma \equiv \Gamma$. Les deux formules ont alors les mêmes modèles.*

2.1.3 Clauses et motifs propositionnels

Dans cette thèse, nous manipulons principalement deux types de formules appelées *clauses* et *motifs*. Lorsqu'il y aura besoin de précisions, les termes *clause propositionnelle* et *motif propositionnel* désigneront resp. une clause exprimée en logique propositionnelle et un motif exprimé en logique propositionnelle. Définissons dans un premier temps les clauses :

Définition 2.8 (Clause propositionnelle)**.** *Une clause est une disjonction finie de littéraux et de la forme $l_1 \vee \ldots \vee l_n$ avec $n \in \mathbb{N}$.*

La clause $S = h_1 \vee \ldots \vee h_n \vee \neg b_1 \vee \ldots \vee \neg b_m$, avec $n, m \in \mathbb{N}$ peut être notée de façon équivalente comme : $S = (h_1 \vee \ldots \vee h_n) \leftarrow (b_1 \wedge \ldots \wedge b_m)$, avec $(h_1 \vee \ldots \vee h_n)$ la *tête* de la clause, notée $head(S)$ qui est une disjonction de littéraux et $(b_1 \wedge \ldots \wedge b_m)$ le *corps* de la clause, noté $body(S)$ qui est une conjonction de littéraux. La clause vide est notée $\Box$, et un ensemble fini de clauses est appelé une *théorie clausale*.

Exemple 2.2. *Soit un vocabulaire $\mathcal{V}$ composé des atomes $\{p, q, r\}$ et $S = q \wedge r \rightarrow p$. On a alors $head(S) = p$ et $body(S) = q \wedge r$.*

Nous introduisons ici la restriction des clauses souvent utilisée dans le domaine de la Programmation Logique Inductive, qui sont les *clauses de Horn*. Cette restriction n'autorise que des clauses avec au plus un littéral positif. Par exemple, on ne peut pas exprimer $p \vee q$.

Définition 2.9 (Clause définie)**.** *Une clause est dite définie si elle contient un seul littéral positif.*

Définition 2.10 (But défini)**.** *Un but est une clause avec uniquement des littéraux négatifs.*

Définition 2.11 (Clause de Horn)**.** *Une clause de Horn est soit une clause définie, soit un but défini.*

Définition 2.12 (Fait)**.** *Une clause de Horn avec un seul littéral positif est un fait.*

Dans cette thèse, nous manipulons également les motifs, que nous définissons ici dans leur version propositionnelle.

Définition 2.13 (Motif propositionnel)**.** *Un motif propositionnel est une conjonction de littéraux en logique propositionnelle.*

2.1.4 Formes normales

Les formes normales sont des formes canoniques qui permettent de représenter des formules logiques de façon unique. Dans cette thèse, nous nous intéressons à deux formes normales : la forme normale conjonctive et la forme normale disjonctive.

Définition 2.14 (Forme normale conjonctive et disjonctive)**.** *Une formule est dite en forme normale disjonctive (DNF) si elle est une disjonction de motifs. Une formule est dite en forme normale conjonctive (CNF) si elle est une conjonction de clauses. Une formule en CNF est équivalente à une théorie clausale.*

Exemple 2.3. *Soit un vocabulaire* $\mathcal{V} = \{p, q, r\}$ *:*

- *une formule en CNF à partir de* $\mathcal{V}$ *est :* $(p \vee q) \wedge (\neg p \vee r)$
- *une formule en DNF à partir de* $\mathcal{V}$ *est :* $(p \wedge q) \vee (\neg p \wedge r)$

2.1.5 Procédures de preuve

Le problème de savoir si une formule ϕ est conséquence logique d'un ensemble de formules Σ est un problème important en logique. Il existe des procédures appelées *procédures de preuve* qui permettent de répondre à cette question. Ces procédures consistent en un certain nombre de courtes étapes au cours desquelles on applique des *règles d'inférence*. Ces règles permettent de déduire de nouvelles formules, appelées *conclusions*, à partir de formules existantes, appelées *prémisses*.

Un exemple de règle d'inférence couramment utilisée est le *modus ponens*, et se base sur la règle d'inférence suivante :

$$\frac{\phi \rightarrow \psi \qquad \phi}{\psi}$$

Cette règle d'inférence est utilisée pour dériver la conclusion ψ, à partir des prémisses $\phi \rightarrow \psi$ et ϕ. Soit l'ensemble des prémisses Σ, lorsqu'une formule ψ est dérivable à partir de Σ en utilisant une procédure de preuve P, on note $\Sigma \vdash_P \psi$. Par exemple, à partir de $\Sigma = \{p, (p \rightarrow q)\}$, on peut dériver q en utilisant le modus ponens (noté mp) : $\Sigma \vdash_{mp} q$.

Résolution. La *résolution* [BG01] est une procédure de preuve basée sur la règle d'inférence suivante :

$$\frac{\phi \vee \psi \qquad \neg\phi \vee \chi}{\psi \vee \chi}$$

Cette règle d'inférence est une généralisation du modus ponens car elle s'applique à des clauses quelconques. Elle prend en prémisses deux clauses, chacune contenant une *paire complémentaire*, c'est-à-dire deux littéraux dont le signe est opposé, qui est représentée dans la règle ci-dessus par ϕ et $\neg\phi$. Si on peut dériver la clause C par résolution à partir d'un ensemble de clauses Σ, alors C est appelé le *résolvant* de Σ, noté $\Sigma \vdash_r C$.

2.1.6 Problème SAT

Le problème *SAT*, ou *problème de satisfaction booléenne* est un problème de décision sur les formules logiques propositionnelles, dont l'objectif est de répondre si une formule admet un modèle.

Définition 2.15 (SAT)**.** *Le problème SAT pour une formule quelconque ϕ est de répondre s'il existe ou non un modèle de ϕ.*

Exemple 2.4. *On considère la formule propositionnelle $(p \vee q) \wedge (\neg p \vee r)$. Un modèle de ϕ est $I = \{q, r\}$, la réponse au problème SAT est donc oui, car ϕ admet un modèle.*

Problème MAXSAT

On définit également le problème MAXSAT, qui est une généralisation du problème SAT, appliqué aux formules propositionnelles CNF. Les formules CNF étant des ensembles de clauses, l'objectif de ce problème est de répondre s'il existe un modèle qui satisfait un certain nombre fixé de clauses dans un ensemble de clauses.

Définition 2.16 (MAXSAT)**.** *Le problème MAXSAT pour une formule CNF ϕ et un entier k est de déterminer s'il existe un modèle de ϕ qui satisfait au moins k clauses de ϕ. Le problème d'optimisation associé est de trouver un modèle de ϕ qui satisfait le plus grand nombre de clauses de ϕ.*

Exemple 2.5. *On considère l'ensemble de clauses $\phi = \{\neg p; p; p \vee q; p \vee r\}$, et $k = 3$. L'interprétation $I = \{p, q, r\}$ satisfait toutes les clauses sauf la première et vérifie donc k clauses. I est une solution optimale au problème d'optimisation associé car il est impossible de satisfaire à la fois la première et la deuxième clause de ϕ.*

Weighted MAXSAT

Le problème Weighted MAXSAT étend le problème d'optimisation lié à MAXSAT en introduisant des poids pour chaque clause dans une formule CNF. Cela permet de donner une importance relative aux clauses dans la formule.

Définition 2.17 (Weighted MAXSAT)**.** *Soit une formule CNF ρ, composée d'un ensemble de clauses C où chaque clause $c_i \in C$ est associée à un poids $w_i \in \mathbb{N}$. L'ensemble des paires clause-poids est noté $\{(c_1, w_1), (c_2, w_2), ..., (c_n, w_n)\}$. Le problème Weighted MAX-SAT consiste à trouver un modèle des clauses de ρ qui minimise la somme des poids des clauses non satisfaites.*

Exemple 2.6. *Considérons une formule CNF ρ avec les clauses et poids suivants : $\{(p \vee q, 3), (p \vee \neg q, 2), (\neg p, 5)\}$. L'interprétation $I = \{q\}$, satisfait la première et la troisième clause mais pas la deuxième, qui a le poids le plus faible, et il n'y a pas d'interprétation satisfaisant les trois clauses. I est donc la solution optimale au problème.*

Partial MAXSAT

Le problème Partial MAXSAT pour une formule propositionnelle est un cas particulier de Weighted MAXSAT où un sous-ensemble des clauses pondérées de ρ est associé à un poids infini. De fait, ce sous-ensemble de contraintes doit obligatoirement être satisfait et est appelé les *contraintes dures* C_{hard}, tandis que le sous-ensemble de ρ dont les clauses sont associées à un entier naturel est dit ensemble des *contraintes souples* C_{soft}, ces contraintes souples doivent être satisfaites autant que possible.

Définition 2.18 (Partial MAXSAT)**.** *Soit ρ un ensemble de paires clause-poids. Dans le problème Partial MAXSAT, $\rho = C_{hard} \cup C_{soft}$. Chaque clause dans C_{hard} est associée à un poids infini, tandis que chaque clause dans C_{soft} est associée à un poids fini. L'objectif est de trouver un modèle qui satisfait toutes les clauses dans C_{hard} et qui minimise le poids des clauses insatisfaites dans C_{soft}.*

Remarque 2.1. *Il est d'usage de nommer le problème Partial MAXSAT lorsque les poids associés aux contraintes souples sont tous égaux à 1, tandis que le problème Weighted Partial MAXSAT est utilisé lorsque certains poids associés aux contraintes souples sont différents de 1.*

2.2 Logique du premier ordre

La logique du premier ordre (LPO) est une généralisation de la logique propositionnelle qui une expressivité plus élevée que cette dernière. En effet, la logique du premier ordre permet de représenter des connaissances plus complexes, car les atomes d'une formule ont des arguments, permettant de représenter des relations entre les objets, mais également de les quantifier à l'aide des quantificateurs universels et existentiels. Dans cette thèse, la logique du premier ordre est un outil central pour la représentation des connaissances et des données. En particulier, nous utilisons la logique du premier ordre pour représenter les données, les modèles et les explications générées dans le chapitre 6. Pour désigner le contexte de la logique du premier ordre, nous utilisons également l'adjectif *relationnel*, par exemple une clause en logique du premier ordre est appelée une *clause relationnelle*. Dans cette section, nous présentons les éléments et les notations que nous allons utiliser tout au long du chapitre 6.

2.2.1 Syntaxe

Dans la logique du premier ordre, tout comme en logique propositionnelle, les formules sont construites à partir d'un *alphabet* comprenant divers symboles. Cet alphabet se compose des éléments suivants : des variables (ex. : $X, Y, \ldots$) commençant par une majuscule ; des symboles de fonction (ex. : $f, g, \ldots$) avec une arité associée, les symboles de fonction d'arité nulle étant des constantes ; des symboles de prédicat (ex. : $p, q, \ldots$) avec une arité notée p/x ; en plus des connecteurs logiques identiques à la logique propositionnelle deux quantificateurs, à savoir $\exists$ (existentiel) et $\forall$ (universel).

Les *formules* en logique du premier ordre sont des séquences de ces symboles. Dans cette thèse, nous nous intéressons à une restriction de la LPO dans laquelle on n'a pas d'autres symboles de fonction que des constantes. De la même façon que pour la section précédente, nous proposons au lecteur non avisé de s'en référer à [Nie+97] pour de plus amples détails.

On définit également l'ensemble des *termes* qui sont les constantes et les variables. Un *terme instancié* est un terme ne contenant aucune variable. Une *formule instanciée* est une formule ne contenant aucune variable libre.

Définition 2.19 (Langage du premier ordre)**.** *Le langage du premier ordre donné par un alphabet est l'ensemble des formules bien formées qui peuvent être construites à partir des symboles de l'alphabet.*

Définition 2.20 (Vocabulaire d'un langage du premier ordre)**.** *Le vocabulaire $\mathcal{V}$ d'un langage du premier ordre est l'ensemble des symboles de prédicats et des symboles de fonction de ce langage.*

2.2.2 Sémantique

En LPO, les formules obéissent aux mêmes règles sémantiques que celles de la logique propositionnelle. Nous proposons au lecteur de se référer à la section 2.1 pour plus de détails, les définitions 2.4 à 2.14 sont également valides pour les formules en LPO, excepté la définition 2.8 et la définition 2.13 que nous allons adapter à la LPO. Le théorème 2.1 et la proposition 2.1 sont également valables pour la LPO. Les différences se font en revanche dans la définition d'une interprétation et de la calculabilité de la conséquence logique.

Interprétations et modèles de Herbrand. En logique du premier ordre, on passe également par une interprétation pour évaluer la valeur de vérité d'une formule. À la différence de la logique propositionnelle, une interprétation en LPO doit prendre en compte les termes présents dans une formule logique lors de l'affectation à vrai ou faux des atomes de la formule. En effet, un terme dans une formule se réfère à une « chose » et une formule est une affirmation à propos de ces choses. L'ensemble de ces « choses » est appelé *domaine de discours* et peut être n'importe quel ensemble d'objets usuels (les entiers naturels, les marques de voitures, un ensemble de fruits, etc.). Ainsi, chaque terme d'un langage du premier ordre doit être appairé à un élément dans le domaine de discours. Ensuite, il faut donner une affectation de $\{T, F\}$ aux appariements pour chaque prédicat afin de déterminer quels atomes sont vrais ou faux dans la formule, ce qui permet de construire

une interprétation. Enfin, vient l'évaluation de la formule étant donné les deux étapes précédentes grâce aux connecteurs et aux quantificateurs.

Une formule quantifiée existentiellement $\exists X\phi$, avec X étant une variable apparaissant dans ϕ, est vraie si au moins un élément du domaine de discours auquel X est affecté rend ϕ vraie et une formule quantifiée universellement $\forall X\phi$ est vraie si tous les éléments du domaine de discours auquel X est affecté rend ϕ vraie.

Dans le cadre de cette thèse pour la logique du premier ordre, nous nous concentrons sur les modèles de Herbrand, qui offrent une approche plus intuitive pour interpréter des formules du premier ordre. Dans les modèles de Herbrand, on considère que les constantes sont affectées à elles-mêmes dans le domaine de discours. Le domaine du discours est alors l'ensemble des termes instanciés apparaissant dans le langage considéré. Cet ensemble est appelé l'*univers de Herbrand*, et l'ensemble des atomes instanciés pouvant être formés à partir de l'univers de Herbrand est appelé la *base de Herbrand*.

Définition 2.21 (Univers et base de Herbrand)**.** *Soit un vocabulaire $\mathcal{V}$. L'univers de Herbrand $U_\mathcal{V}$ pour $\mathcal{V}$ est l'ensemble de tous les termes instanciés qui peuvent être formés à partir des constantes et symboles de fonction dans $\mathcal{V}$. Si $\mathcal{V}$ ne contient pas de constante, on ajoute une constante arbitraire à $U_\mathcal{V}$.*

La base de Herbrand $B_\mathcal{V}$ pour $\mathcal{V}$ est l'ensemble de tous les atomes instanciés que l'on peut construire à partir des symboles de prédicat dans $\mathcal{V}$ et des éléments de l'univers de Herbrand $U_\mathcal{V}$.

Exemple 2.7. *Soit le vocabulaire $\mathcal{V}$ composé des constantes $\{a, b\}$ et des prédicats $\{p/1, q/2\}$.*

L'univers de Herbrand est $U_\mathcal{V} = \{a, b\}$ et la base de Herbrand est $B_\mathcal{V} = \{q(a,b), q(a,a), q(b,b), q(b,a), p(a), p(b)\}$.

Une interprétation de Herbrand I est une assignation à $\{T, F\}$ des éléments de la base de Herbrand que l'on peut représenter par le sous-ensemble des éléments de la base de Herbrand assignés à T dans cette interprétation. Lorsque la base de Herbrand est finie, cette assignation est ensuite utilisée pour évaluer la formule en utilisant les tables de vérité de la même façon qu'en logique propositionnelle. Une interprétation I de Herbrand est un *modèle de Herbrand* pour une formule ϕ (resp. un ensemble de formules Σ) si elle rend vraie ϕ (resp. Σ).

Définition 2.22 (Modèle de Herbrand)**.** *Soit L un langage du premier ordre, Σ un ensemble de formules de L, et I une interprétation de Herbrand de L. Si I est un modèle de Σ, alors I est un modèle de Herbrand de cette formule.*

Exemple 2.8. *On considère le langage L utilisé dans l'exemple 2.7. Soit $\Sigma = \{\forall X(\neg q(X,X) \vee p(X)); q(a,b)\}$ et les interprétations de Herbrand suivantes de Σ :*

- $I_1 = \{p(a), p(b), q(a,b)\}$
- $I_2 = \{p(a), q(a,a), q(b,b), q(a,b)\}$
- $I_3 = \{q(a,a), q(b,b), q(a,b)\}$
- $I_4 = \{p(a), q(a,a), q(a,b)\}$

I_1 et I_4 sont des modèles de Herbrand de Σ tandis que I_2 et I_3 ne le sont pas.

Exemple 2.9. *On considère toujours le langage L de l'exemple 2.7. Soit $\Sigma = \{\exists X(\neg q(X,X) \vee p(X)); q(a,b)\}$ et on considère les interprétations I_1, I_2, I_3 et I_4 de l'exemple 2.8.*

I_1, I_2 et I_4 sont des modèles de Herbrand de Σ, tandis que I_3 ne l'est pas.

Calculabilité de la conséquence logique En logique propositionnelle, il est possible de vérifier si $\Sigma \models \phi$. En effet, il suffit d'énumérer toutes les 2^n interprétations possibles des n atomes apparaissant dans Σ et ϕ et de vérifier si les modèles de Σ sont également des modèles de ϕ. Cela veut dire qu'en un temps fini, on peut décider si $\Sigma \models \phi$, ce qui rend la conséquence logique *décidable* en logique propositionnelle.

Cependant, en logique du premier ordre, la conséquence logique n'est pas décidable [NW97]. En effet, il peut exister une infinité d'interprétations, donc une infinité de modèles possibles pour une formule ce qui rend leur énumération impossible. Il n'existe aucun algorithme qui peut décider si $\Sigma \models \phi$ dans le cas général.

2.2.3 Clauses et motifs relationnels

En logique du premier ordre, de la même façon que pour la logique propositionnelle, une clause est une disjonction finie de littéraux. Lorsqu'il y aura besoin de précisions, le terme *clause relationnelle* désignera une clause exprimée en logique du premier ordre. Les clauses sont un élément très important en LPO, car les ensembles de clauses permettent d'exprimer les théories du domaine dans la Programmation Logique Inductive qui sera présentée dans le chapitre suivant, dans lequel on manipule des clauses sous *forme standard* :

Définition 2.23 (Forme standard)**.** *Une formule est sous forme normale conjonctive prénexe si elle est de la forme $q_1X_1 \ldots q_nX_n(S_1 \wedge \ldots \wedge S_m)$ où chaque q_i est un quantificateur ($\forall$ ou $\exists$), les X_i sont toutes les variables apparaissant dans la formule, et chaque S_j est une clause (notons qu'il n'y a pas de variable non quantifiée dans une formule sous forme normale conjonctive prénexe).*

Une formule est sous forme standard si elle est sous forme normale conjonctive prénexe et que les quantificateurs sont uniquement universels ($\forall$).

On ne considère donc par la suite dans cette thèse que les clauses sous forme standard. Une *clause relationnelle* est alors de la forme $\forall X_1 \ldots \forall X_n(S)$, où les X_i sont toutes les variables apparaissant dans la clause, et S est une disjonction de littéraux. Par exemple, la clause $\forall X \forall Y(r(X) \vee \neg p(X) \vee \neg q(Y))$ est sous forme standard, que l'on peut également noter $\forall(r(X) \vee \neg p(X) \vee \neg q(Y))$.

Comme en logique propositionnelle, une clause relationnelle $S = \forall[h_1 \vee \ldots \vee h_m \vee \neg b_1 \vee \ldots \vee \neg b_n]$ peut être notée sous la forme $S = \forall[(h_1 \vee \ldots \vee h_m) \leftarrow (b_1 \wedge \ldots \wedge b_m)]$ où $h_1 \vee \ldots \vee h_m$ représente la tête de S notée $head(S)$ et $b_1 \wedge \ldots \wedge b_m$ est le corps de la clause S noté $body(S)$.

La restriction des clauses relationnelles aux clauses de Horn est similaire à celle des clauses propositionnelles : une clause relationnelle de Horn est une clause relationnelle qui a au plus un littéral positif. Cette perte d'expressivité renforce la tractabilité des

procédures de preuve qui l'utilisent en LPO, ce qui rend son utilisation très courante dans la Programmation Logique Inductive.

Un deuxième type de formules couramment utilisé en Programmation Logique Inductive, les conjonctions existentiellement quantifiées appelées *motifs relationnels*.

Définition 2.24 (Motif relationnel)**.** *Un motif relationnel est une conjonction de littéraux existentiellement quantifiée.*

Notations alternatives pour les clauses et les motifs relationnels

Dans cette thèse, nous utilisons deux notations simplifiées pour les clauses et les motifs relationnels qui omettent les quantificateurs. Nous illustrons les différentes notations pour les clauses par l'exemple $\forall X \forall Y (p(X, a) \vee \neg q(X) \vee \neg p(b, Y) \vee q(b))$. Les clauses relationnelles peuvent se noter de façon alternative :

1. Comme une disjonction de littéraux dont on omet le quantificateur $\forall$: $p(X, a) \vee \neg q(X) \vee \neg p(b, Y) \vee q(b)$;
2. Comme une implication syntaxique : $p(X, a) \vee q(b) \leftarrow \neg q(X) \wedge \neg p(b, Y)$ ou $p(X, a), q(b) \leftarrow q(X), \neg p(b, Y)$.

Concernant les motifs, et l'exemple : $\exists X \exists Y (p(a, a) \wedge q(X) \wedge \neg p(X, Y))$, les notations alternatives sont les suivantes :

1. Comme une conjonction de littéraux dont on omet le quantificateur $\exists$: $p(a, a) \wedge q(X) \wedge \neg p(X, Y)$;
2. Comme une liste de littéraux quand le contexte est clair : $[p(a, a), q(X), \neg p(X, Y)]$.

2.2.4 Ordres sur les clauses et les motifs

Les clauses et les motifs peuvent être comparés deux à deux grâce à des relations que l'on appelle *ordres partiels*. Dans cette sous-section, nous commençons par présenter quelques notions sur les ordres partiels, qui sont ceux qui nous intéressent, puis nous présentons les ordres que nous allons utiliser dans cette thèse.

Notions sur les ordres

Définition 2.25 (Ordre partiel)**.** *Soit R une relation binaire sur un ensemble E (noté $\langle E, R \rangle$). R est un ordre partiel ssi R est réflexive, transitive et antisymétrique.*

On note les ordres partiels par le symbole $\preceq$. Si l'on considère un ensemble E partiellement ordonné, un sous-ensemble $S \subseteq E$ peut admettre des bornes supérieures et inférieures.

Définition 2.26 (Bornes supérieures et inférieures)**.** *Soit $\langle E, \preceq \rangle$ un ensemble partiellement ordonné et $S \subseteq E$. Un élément $x \in E$ est un minorant de S ssi pour tout $y \in S$, $x \preceq y$. Un minorant $x' \in E$ est la borne inférieure de S ssi pour tout minorant x de S, $x \preceq x'$.*

Un élément $x \in E$ est un majorant de S ssi pour tout $y \in S$, $y \preceq x$. Un majorant $x' \in E$ est la borne supérieure de S ssi pour tout majorant x de S, $x' \preceq x$.

Définition 2.27 (Treillis)**.** *Soit $\langle E, \preceq \rangle$ un ensemble partiellement ordonné. Si pour toute paire d'éléments $x, y \in E$ une borne inférieure de $\{x, y\}$ et une borne supérieure de $\{x, y\}$ existe, alors $\langle E, \preceq \rangle$ est appelée un treillis.*

Subsomption

La *subsomption* est une relation d'ordre entre les clauses ou les motifs. Par abus de langage, nous parlons d'ordre dans cette thèse pour parler de préordre (pas d'antisymétrie). Nous nous intéressons ici plus spécifiquement à la θ-subsomption. La θ-subsomption [Rob65] est une relation d'inclusion reposant sur la notion de *substitution* qui est le fait de remplacer, dans une formule du premier ordre, des variables par des termes.

Définition 2.28 (Substitution)**.** *Une substitution θ est un ensemble fini de la forme*

$$\{X_1/t_1, \ldots, X_n/t_n\}, n \geq 0,$$

où $X_1, \ldots, X_n$ sont les variables substituées par les termes $t_1, \ldots, t_n$. Une substitution est dite close si tous les termes $t_1, \ldots, t_n$ sont des constantes. On note $S\theta$ l'application de la substitution θ à la formule S.

Exemple 2.10. *Soit $C = p(X) \vee q(Y)$ et $\theta = \{X/a, Y/X\}$. On a alors $C\theta = p(a) \vee q(X)$.*

Nous pouvons maintenant définir la θ-subsomption :

Définition 2.29 (θ-subsomption)**.** *Soit C et D deux clauses ou deux motifs relationnels. On dit que C θ-subsume D, noté $C \preceq_\theta D$ s'il existe une substitution θ telle que $C\theta \subseteq D$. Par ailleurs, C et D sont subsume-equivalents si $C \preceq_\theta D$ et $D \preceq_\theta C$, noté $C \sim_\theta D$.*

Exemple 2.11. *Soit $C = \{p(X) \vee q(X)\}$ et $D = \{p(a) \vee q(a) \vee r(a)\}$. $C \preceq_\theta D$, car $C\theta \subseteq D$ avec $\theta = \{X/a\}$.*

Exemple 2.12. *Soit $C = \{p(X, Y, Z) \vee p(X, X, X)\}$ et $D = \{p(X, X, X)\}$. $C \sim_\theta D$, car $C \preceq_\theta D$ avec $\theta = \{Y/X, Z/X\}$ et $D \preceq_\theta C$ avec $\theta = \emptyset$.*

Dans le cas général :

— pour les clauses : si $C \preceq_\theta D$, alors $C \models D$, mais la réciproque n'est pas vraie.

— pour les motifs : si $C \preceq_\theta D$, alors $D \models C$, mais la réciproque n'est pas vraie.

En revanche, pour les clauses *non récursives*, c'est-à-dire les clauses dans lesquelles il n'y a pas un littéral positif et un littéral négatif qui partagent le même symbole de prédicat, la θ-subsomption est équivalente à la conséquence logique [Ide95].

Remarque 2.2. *Le problème de déterminer s'il existe une substitution θ telle que $C \preceq_\theta D$ est un problème NP-complet [KN86 ; Coo71], ce qui veut dire qu'il n'existe pas d'algorithme pour résoudre ce problème en temps polynomial dans tous les cas.*

Il existe une relation d'ordre partiel qui est un cas particulier de la θ-subsomption. La *subsomption sous Object Identity* (*OI*) [Fer+02] repose sur l'*hypothèse OI* qui stipule que les termes (et les variables) d'une clause ou d'un motif ne peuvent pas pointer sur une même constante.

Exemple 2.13. *La clause* $C = p(X) \leftarrow q(X,X) \wedge q(Y,a)$ *sous hypothèse OI est une abréviation de la clause* $C_{OI} = p(X) \leftarrow q(X,X) \wedge q(Y,a) \mid [X \neq Y], [X \neq a], [Y \neq a]$. *(exemple tiré de [Fer+02])*

Définition 2.30 (OI-subsomption)**.** *Soit* C *et* D *deux clauses ou deux motifs relationnels.* D *subsume* C *sous OI (noté* $D \preceq_{OI} C$*) si et seulement s'il existe une substitution* θ *telle que* $D_{OI}\theta \subseteq C_{OI}$.

Instance

La deuxième relation d'ordre sur les clauses et les motifs que nous allons utiliser dans cette thèse est la relation d'instance.

Définition 2.31 (Instance)**.** *Soit* C, D *deux clauses ou motifs.* D *est une instance de* C *ssi il existe une substitution* θ *telle que* $C\theta = D$ *à un renommage de variable près, qu'on écrit* $C \preceq_i D$*. Si* $C \neq D$ *à un renommage de variable près, alors* D *est une instance stricte de* C*, que l'on note* $C \prec_i D$*. Si* D *est instancié, alors* D *est une instance close de* C.

La relation $\preceq_i$ est une relation d'ordre partiel sur les clauses et les motifs, telle que si $C \preceq_i D$ alors $C \preceq_\theta D$.

Exemple 2.14. $p(X,X)$ *est une instance de* $p(X,Y), p(Y,Z)$ *étant donné que* $(p(X,Y), p(Y,Z))\theta = p(X,X)$ *avec* $\theta = \{Y/X, Z/X\}$.

$p(X,a)$ *est une instance de* $p(X,Y)$ *avec* $\theta = \{Y/a\}$ *et* $p(b,a)$ *est une instance close de* $p(X,Y)$ *avec* $\theta = \{X/b, Y/a\}$.

2.2.5 Unification

L'*unification* est une opération qui permet de trouver une substitution, appelée l'*unificateur le plus général*, qui rend deux formules logiques identiques. L'unification est une opération centrale dans les procédures de preuve en logique du premier ordre. Bien que nous ne nous intéressions pas directement à l'unification dans cette thèse, nous présentons dans le chapitre 3 l'opération inverse appelée l'*anti-unification*, ce qui justifie la présentation de l'unification dans cette section.

Définition 2.32 (Unification)**.** *Soit* Σ *un ensemble fini de formules logiques. Une substitution* θ *est un unificateur de* Σ *si* $\Sigma\theta$ *est un singleton. S'il existe un unificateur* θ *de* Σ*, alors* Σ *est dit unifiable.*

Exemple 2.15. *Soit* $\Sigma = \{p(X,b); p(a,Y)\}$. $\theta = \{X/a, Y/b\}$ *est un unificateur de* Σ.

Définition 2.33 (Unificateur le plus général)**.** *Si* θ *est un unificateur de* Σ*, alors* θ *est l'unificateur le plus général (UPG) de* Σ *si pour tout autre unificateur* σ *de* Σ*, il existe une substitution* γ *telle que* $\sigma = \theta\gamma$*.* θ *est unique à un renommage près des variables.*

Exemple 2.16. *Soit* $\Sigma = \{p(X,X); p(Z,Y)\}$. $\theta = \{Z/X, Y/X\}$ *est un unificateur de* Σ. *La substitution* $\sigma = \{X/a, Z/a, Y/a\}$ *est un unificateur de* Σ*, mais n'est pas un UPG car il n'existe pas de substitution* γ *telle que* $\sigma\gamma = \theta$.

L'algorithme d'unification est décrit dans [NW97], et prend en entrée un ensemble de formules simples (des littéraux ou des termes), et retourne leur unificateur le plus général unique (à un renommage près des variables) s'il existe. Pour présenter l'algorithme de l'unification, nous introduisons la notion d'*ensemble de désaccord*. Nous décrivons l'algorithme de l'unification dans l'Algorithme 1.

Définition 2.34 (Ensemble de désaccord)**.** *Soit Σ un ensemble de formules simples ayant les mêmes symboles de prédicat. L'ensemble de désaccords de Σ est l'ensemble des termes les plus à gauche de chaque formule de Σ qui n'ont pas le même symbole.*

Exemple 2.17. *Soit $\Sigma = \{p(X, Y, Z); p(X, a, b); p(X, W, b)\}$. L'ensemble de désaccords de Σ est $\{Y, a, W\}$.*

Algorithme 1 Algorithme d'unification

Entrée: Un ensemble fini de formules simples Σ ayant les mêmes symboles de prédicat.
Sortie: L'unificateur le plus général θ de Σ s'il existe.

1: $\theta \leftarrow \emptyset$
2: **tant que** $\Sigma\theta$ n'est pas un singleton **faire**
3: Trouver l'ensemble de désaccord D de $\Sigma\theta$
4: **si** Il existe X et t dans D tels que X est une variable qui n'apparaît pas dans t **alors**
5: $\theta \leftarrow \theta \cup \{X/t\}$
6: **sinon**
7: **retourne** *échec*
8: **fin si**
9: **fin tant que**
10: **retourne** θ

2.2.6 Clauses réduites

Une clause *réduite* peut-être vue comme le représentant le « plus petit » de sa classe d'équivalence selon un ordre partiel, celui auquel on s'intéressera ici est la θ-subsomption. Les clauses réduites sont utiles, car elles sont plus petites que les autres membres de leur classe d'équivalence en raison de l'absence de redondance des littéraux au sens de la θ-subsomption. Dans cette section, nous utiliserons les clauses comme objet d'étude, mais toutes les définitions et l'algorithme présentés dans cette sous-section sont également valables pour les motifs relationnels.

Définition 2.35 (Clause θ-réduite)**.** *Une clause C est dite θ-réduite s'il n'existe aucun sous-ensemble propre D de C ($D \subset C$) tel que $D \sim_\theta C$. Une clause θ-réduite D telle que $C \sim_\theta D$ et $D \subseteq C$ est appelée une θ-réduction de C.*

Nous utiliserons le terme *clause réduite* et *réduction* pour désigner resp. une clause θ-réduite et l'opération de θ-réduction, car nous nous intéressons uniquement à la réduction sous la θ-subsomption dans cette thèse.

Exemple 2.18. *$C = p(X,Y) \vee p(Y,X)$ est réduite. $D = p(X,X) \vee p(X,Y) \vee p(Y,X)$ n'est pas réduite, car $D' = p(X,X)$ est un sous-ensemble propre de D tel que $D' \sim_\theta D$. D' est une réduction de D étant donné la substitution $\theta = \{Y/X\}$.*

Dans [Plo70], l'auteur propose un algorithme permettant de calculer la réduction d'une clause. L'algorithme est présenté dans l'Algorithme 2. L'algorithme prend en entrée une clause C et retourne une réduction de C. L'algorithme commence par trouver un littéral L et une substitution θ tels que $C\theta \subseteq C \setminus \{L\}$. Si un tel littéral est trouvé, alors on retire ce littéral de la clause. On répète ce processus jusqu'à ce qu'aucun littéral ne puisse être retiré. L'algorithme retourne alors la clause réduite.

Algorithme 2 ReduceClause(C)

Entrée: Une clause C
Sortie: Une réduction D de C

```
1: trouvé ← true ;
2: D ← C
3: tant que trouvé faire
4:     Trouver un littéral L ∈ D et une substitution θ tels que Dθ ⊆ D \ {L}.
5:     si L n'a pas été trouvé alors
6:         trouvé ← false
7:     sinon
8:         D ← D \ {L}
9:     fin si
10: fin tant que
11: retourne D
```

Remarque 2.3. *Le problème de décider si une clause est réduite a été prouvé co-NP-complet [GF93], qui est la classe de complexité complémentaire à NP-complet. Ce qui signifie que vérifier qu'il n'existe aucune réduction d'une clause est un problème difficile. Des travaux ont proposé des algorithmes pour calculer plus efficacement les réductions de clauses [GF93 ; MS04 ; Hir03], mais restent limités par la complexité du problème.*

2.3 Impliquants et impliqués premiers

2.3.1 Impliquants premiers

Le concept des *impliquants premiers* occupe une position centrale dans la littérature sur les explications, développée dans le chapitre 5. Introduit initialement par W. V. Quine [Qui59], un impliquant premier est défini comme une conjonction de littéraux donc un motif, notée t, qui implique une formule F, telle que la suppression de tout littéral de t conduit à une nouvelle conjonction qui n'implique pas F. Si F est une formule propositionnelle, alors t est un motif propositionnel, et si F est une formule du premier ordre, t est un motif relationnel. Plus formellement, on définit un impliquant premier en utilisant la définition présentée dans [Mar91] et [RK87].

Définition 2.36 (Impliquant premier)**.** *Soit L un langage (resp. propositionnel ou du premier ordre) composé d'un nombre infini d'atomes, soit t un motif (resp. propositionnel ou relationnel), et une formule $F \subseteq L$. t est un impliquant de F si $t \models F$.*

t est un impliquant premier de F si :

- *t est un impliquant de F ;*
- *si $t' \models F$ et $t \models t'$ alors $t' \models t$.*

La définition ci-dessus nous dit qu'un impliquant t de la formule F est premier si aucun autre impliquant $t' \neq t$ de F n'est impliqué par t.

Exemple 2.19. *Soit $\mathcal{V}$ un vocabulaire propositionnel contenant les atomes $\{p, q, r\}$ et une formule $F = (p \wedge q) \vee (\neg p \wedge r)$.*

Le motif propositionnel $t = p \wedge q \wedge r$ est un impliquant de F, car $t \models F$. Ce n'est pas un impliquant premier, car $t' = p \wedge q$ est également un impliquant de F et $t \models t'$, mais $t' \not\models t$.

t' est un impliquant premier de F, car il n'existe pas d'autre impliquant t'' de F tel que $t' \models t''$ et $t'' \not\models t'$.

$\neg p \wedge r$ et $q \wedge r$ sont également des impliquants premiers de F.

Exemple 2.20. *Soit $\mathcal{V}$ un vocabulaire du premier ordre composé du prédicat $\{p/2\}$ et des constantes $\{a, b, c\}$ et une formule $F = \exists(\neg p(X, Y) \vee \neg p(Y, Z) \vee p(X, Z))$.*

Le motif relationnel $t = \neg p(c, b) \wedge p(a, b)$ est un impliquant de F, car $t \models F$. Ce n'est pas un impliquant premier, car $t' = p(a, b)$ est également un impliquant de F et $t \models t'$, mais $t' \not\models t$.

t' est un impliquant premier de F, car il n'existe pas d'autre impliquant t'' de F tel que $t' \models t''$ et $t'' \not\models t'$.

2.3.2 Impliqués premiers

Les *impliqués premiers* sont une notion duale des impliquants premiers. En effet, on a $t \models F$ ssi $\neg F \models \neg t$. Ainsi, les impliquants de F sont la négation des impliqués de $\neg F$, et les impliquants premiers de F sont la négation des impliqués premiers de $\neg F$. Un impliqué premier t d'une formule F est défini comme une clause impliquée par F et qui implique toutes les autres clauses impliquées par F. Si F est une formule propositionnelle, alors t est une clause propositionnelle, si F est une formule du premier ordre, alors t est une clause relationnelle.

Définition 2.37 (Impliqué premier)**.** *Soit L un langage (resp. propositionnel ou du premier ordre) composé d'un nombre infini d'atomes, soit t une clause (resp. propositionnelle ou relationnelle), et une formule $F \subseteq L$. t est un impliqué de F si $F \models t$.*

t est un impliqué premier de F si :

- *t est un impliqué de F ;*
- *si $F \models t'$ et $t' \models t$ alors $t \models t'$.*

Exemple 2.21. *Soit $\mathcal{V}$ un vocabulaire propositionnel composé des atomes $\{p, q, r\}$ et une formule $F = (\neg p \vee \neg q) \wedge (p \vee \neg r)$.*

La clause propositionnelle $t = \neg p \vee \neg q \vee \neg r$ est un impliqué de F, car $F \models t$. Ce n'est pas un impliqué premier, car $t' = \neg q \vee \neg r$ est également un impliqué de F et $t' \models t$, mais $t \not\models t'$.

t' est un impliqué premier de F, car il n'existe pas d'autre impliqué t'' de F tel que $t'' \models t'$ et $t' \not\models t''$.

Exemple 2.22. *Soit $\mathcal{V}$ un vocabulaire du premier ordre composé du prédicat $\{p/2\}$ et des constantes $\{a, b, c\}$ et une formule $F = \exists(p(X, Y) \wedge p(Y, X) \wedge \neg p(X, Z))$.*

La clause relationnelle $t = \neg p(a, b) \vee p(c, b)$ est un impliqué de F, car $F \models t$. Ce n'est pas un impliqué premier, car $t' = \neg p(a, b)$ est également un impliqué de F et $t' \models t$, mais $t \not\models t'$.

t' est un impliqué premier de F, car il n'existe pas d'autre impliqué t'' de F tel que $t'' \models t'$ et $t' \not\models t''$.

2.4 Résumé

Dans ce chapitre, nous avons introduit les notions de bases de logique propositionnelle et de logique du premier ordre. La logique propositionnelle est un formalisme qui permet de représenter des connaissances en utilisant des propositions atomiques et des connecteurs logiques. La logique du premier ordre est une extension de la logique propositionnelle qui permet de représenter des connaissances plus complexes en utilisant des variables, des prédicats et des quantificateurs (on ignore ici les symboles de fonctions autres que les constantes).

Dans cette thèse, nous manipulons deux types de formules : les clauses, qui sont des disjonctions de littéraux et les motifs, qui sont des conjonctions de littéraux. Ces deux types de formules peuvent être propositionnels (logique propositionnelle) ou relationnels (logique du premier ordre). Dans cette thèse, on considère que la forme standard d'une clause relationnelle est quantifiée universellement ($\forall$) et la forme standard d'un motif relationnel est quantifiée existentiellement ($\exists$).

Les clauses et les motifs peuvent s'ordonner selon une relation d'ordre partiel. La relation d'ordre partiel qui sera principalement utilisée dans cette thèse est la θ-subsomption. Cette relation d'ordre est la plus utilisée dans la Programmation Logique Inductive, car contrairement à l'implication logique, la θ-subsomption est décidable bien que la complexité de sa décision soit NP-complète. Deux algorithmes centraux ont été présentés sur les formules du premier ordre : l'algorithme d'unification qui permet de trouver une substitution qui rend un ensemble de formules égales et l'algorithme de réduction qui permet de supprimer des littéraux d'une formule sans changer ses modèles.

Enfin, les concepts d'impliquant premier et d'impliqué premier sont centraux dans la littérature sur les explications présentée dans le chapitre 5. Un impliquant premier d'un ensemble de clauses est un impliquant de cet ensemble de clauses qui n'est pas impliqué par les autres impliquants. De même, un impliqué premier d'un ensemble de clauses est un impliqué de cet ensemble de clauses et qui n'est pas impliqué par les autres impliqués.

Nous réutiliserons ces notions dans les chapitres suivants, premièrement dans le chapitre 3 les cadres d'apprentissage supervisé que nous présentons construisent des modèles en logique, deuxièmement pour représenter les données et les modèles appris dans les chapitres 4 et 6, et enfin pour caractériser les explications abductives en ordre un dans le chapitre 6, précédé d'un état de l'art sur les explications dans le chapitre 5 qui utilise également ces notions.

Chapitre 3

Apprentissage supervisé

Dans ce chapitre, nous nous intéressons à l'apprentissage supervisé, qui est un cadre d'apprentissage automatique dans lequel un système apprend à partir d'exemples étiquetés. Nous commençons par définir le concept d'apprentissage supervisé de concept, qui est la version la plus simple de l'apprentissage supervisé. Ensuite, nous présentons la Programmation Logique Inductive, qui est une approche d'apprentissage supervisé de concept basée sur la logique du premier ordre, et enfin nous présenterons le programme de Programmation Logique Inductive NukLear utilisé dans cette thèse. Enfin, nous présenterons le concept de lgg (*least general generalization*) qui est un concept clé dans la programmation logique inductive et qui sera utilisé dans le chapitre 6.

3.1 Apprentissage supervisé en logique propositionnelle

Dans le chapitre 4, nous nous intéressons à la version la plus simple de l'apprentissage automatique, à savoir l'apprentissage supervisé de concept [Mit82]. Le cadre ici présenté est l'apprentissage supervisé de concept en logique propositionnelle. Dans cette section, nous allons introduire les définitions et notations usuelles du cadre de la contribution présentée dans le chapitre 4.

3.1.1 Formulation du problème

On formule le problème de la façon suivante, on considère que l'on a :

1. Un *langage d'hypothèses* représenté par L, exprimé à partir d'un ensemble d'atomes A ;
2. Un *concept cible* noté c, avec $c \notin A$;
3. Un *univers d'observation* $\mathcal{U}$, chaque observation $x \in \mathcal{U}$ étant une interprétation sur A en logique propositionnelle. Théoriquement, $\mathcal{U}$ est partitionné en deux sous-ensembles : $\mathcal{U}^+$ (où c est vrai) et $\mathcal{U}^-$ (où c est faux), mais nous ne connaissons pas cette partition pour l'ensemble complet $\mathcal{U}$.

4. Un sous-ensemble $E \subseteq \mathcal{U}$ tel que pour chaque observation $x \in E$, on sait si c est vrai pour x. On note E^+ l'ensemble des observations de E pour lesquelles c est vrai et E^- l'ensemble des observations de E pour lesquelles c est faux.

Les éléments de E sont appelés *exemples*, divisés en exemples *positifs* dans E^+ et *négatifs* dans E^-. On pose que pour tout élément $H \in L$ et pour toute observation $x \in O$ on peut savoir si H est vrai pour x, on dira alors que H *couvre* x. On note $\text{ext}_\text{X}(H)$ la *couverture* de H dans $X \subseteq \mathcal{U}$ formée des éléments de X pour lesquelles H est vraie.

Le problème posé est de trouver une hypothèse H dans L telle que $\text{ext}_\mathcal{U}(H) = \mathcal{U}^+$, c'est-à-dire une hypothèse de L vraie pour les mêmes observations de $\mathcal{U}$ que le concept-cible c. Comme on sait si c est vrai ou non seulement sur les observations de E, on cherchera une solution H telle que :

$$\text{ext}_\text{E}(H) = E^+ \tag{3.1}$$

et on supposera alors que $\text{ext}_\mathcal{U}(H) = \mathcal{U}^+$.

Exemple 3.1. *Soit $A = \{a, b, c\}$ un ensemble d'atomes et z le concept cible. L'univers $\mathcal{U}$ contient $2^{|A|} = 8$ observations partitionnées en $\mathcal{U}^z$ et $\mathcal{U}^{\neg z}$, mais dont on ne connaît pas la partition. Soit $E = \{x_1, x_2, x_3\}$ un sous-ensemble des observations de $\mathcal{U}$ pour lesquelles on connaît la valeur de z. On suppose $E^z = \{x_1, x_2\}$ l'ensemble des observations pour lesquelles on sait que z est vrai et $E^{\neg z} = \{x_3\}$ l'ensemble des observations pour lesquelles on sait que z est faux. On cherchera alors une hypothèse H telle que $\text{ext}_\text{E}(H) = E^z$ et on supposera que $\text{ext}_\mathcal{U}(H) = \mathcal{U}^z$ et que $\mathcal{U}^{\neg z} = \mathcal{U} \setminus \mathcal{U}^z$.*

3.1.2 Hypothèse

Une hypothèse H est ici un ensemble de règles concluant sur c. On aura $H = \{r_1, \ldots, r_n\}$ où une règle r_i est de la forme $c \leftarrow t$ où t est un motif. Nous introduisons la notation $\dot{x}$, qui est le motif construit des atomes de A dont x est modèle.

Définition 3.1 (Représentation d'une observation par une conjonction de littéraux)**.** *Soit O un univers d'observations, $x \in O$ une interprétation et A un ensemble d'atomes. x est représenté par une conjonction de littéraux $\dot{x}$ telle que $\dot{x} = \bigwedge_{a \in A} a$.*

Exemple 3.2. *Soit $A = \{a, b, c\}$ un ensemble d'atomes et x une interprétation telle que $x(a) = vrai$, $x(b) = faux$ et $x(c) = vrai$. La représentation de x est $\dot{x} = a \wedge \neg b \wedge c$.*

La conjonction t est alors vraie pour x si tous les atomes de t sont inclus dans $\dot{x}$. Il en résulte que H est vraie pour x si et seulement s'il existe une règle de H, $r_i = c \leftarrow t_i$, telle que t_i vrai pour x. De manière équivalente H est vrai pour x si on peut déduire c de $\dot{x}$:

$$\dot{x}, r_1, \ldots r_n \models c \tag{3.2}$$

Une hypothèse H est alors une solution de l'équation 3.1 si et seulement si :

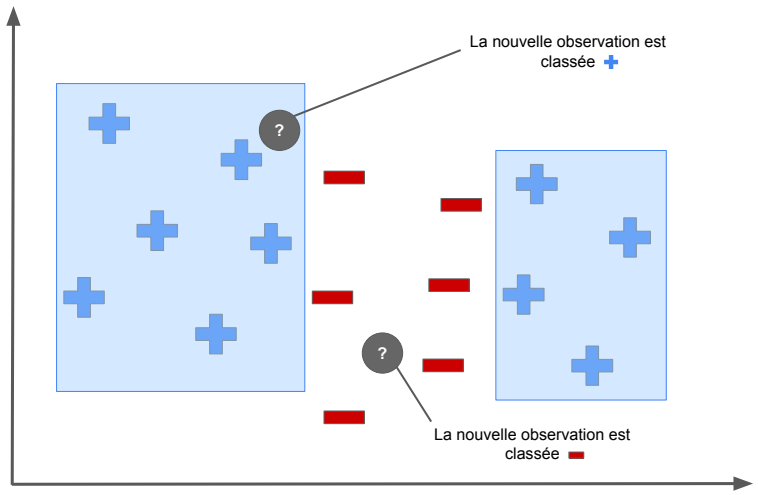


FIGURE 3.1 – Exemple d'apprentissage supervisé binaire

1. Toute règle r_i de H est *valide* sur E :

$$\forall x \in E^-, \forall r_i \in H, t_i \text{ faux pour } x \tag{3.3}$$

et ne conclut donc jamais c pour les objets de E^-, ce qu'on peut aussi écrire $\text{ext}_{\text{E}^-}(r) = \emptyset$ et, cela étant vrai pour toutes les règles de H, on a $\text{ext}_{\text{E}^-}(H) = \emptyset$

2. H est *complète* sur E : pour tout exemple x de E^+ il existe une règle r_i dans H qui conclut sur c

$$\forall x \in E^+, \exists r_i \in H \text{ telle que } t_i \text{ vrai pour } x \tag{3.4}$$

et on a donc $\text{ext}_{E^+}(H) = E^+$.

L'ensemble des règles valides sur E constitue l'ensemble des règles *candidates* : toute solution est constituée de règles candidates. Parmi toutes les solutions on cherchera une solution H de complexité la plus faible possible, mesurée en nombre de règles.

Considérons un objet x non étiqueté (on ne sait pas si c est vrai pour x), s'il existe r_i dans H telle que t_i est vrai pour x, on peut déduire c de H et de $\dot{x}$ et on décide alors que c est vrai pour x, sinon on décide que c est faux pour x. Cette règle ne commet par définition pas d'erreur sur l'ensemble E.

Dans la figure 3.1, on illustre un exemple d'apprentissage supervisé binaire de façon simple. On considère les deux classes $(+, -)$. L'apprentissage supervisé est effectué en apprenant les règles qui couvrent les exemples de la classe $+$ et qui ne couvrent pas les exemples de la classe $-$. Dans cette figure, deux règles disjointes sont apprises et leur couverture dans l'espace des objets est représentée par les zones en bleu. Les nouvelles observations représentées par des points gris peuvent être classés en utilisant les règles apprises. S'ils sont couverts par l'une des deux règles, alors ils sont classés $+$. Sinon, ils sont classés $-$.

Plus généralement, l'apprentissage peut ne pas être *réalisable*, c'est-à-dire qu'il n'existe pas de solution H qui satisfasse l'équation 3.1, ou bien, même s'il est réalisable, on peut

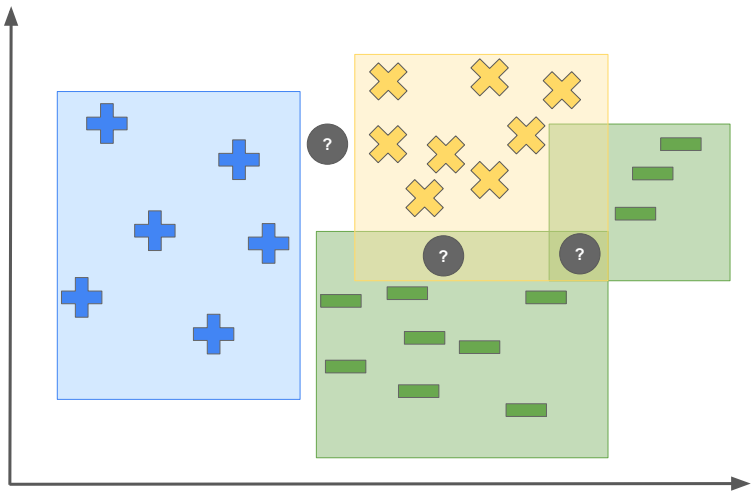


FIGURE 3.2 – Exemple d'apprentissage supervisé multiclasse

préférer un H ne satisfaisant pas les équations 3.3 et 3.4, mais représentant un compromis entre sa justesse (proportion d'objets de E pour lesquels la prédiction est correcte) et sa complexité, en utilisant par exemple une fonction de pénalisation de l'hypothèse croissante avec sa complexité. Dans tous les cas, on évaluera *a posteriori* la qualité de la solution retenue H sur des exemples qui n'ont pas été utilisés en apprentissage, formant l'ensemble de test. Pour cela on utilise une fonction d'évaluation, comme l'estimation de sa justesse en dehors de E.

Apprentissage supervisé multiclasse. Le problème de l'apprentissage de concept peut être étendu à l'apprentissage de labels de classes c_i avec $i \in 1 \dots n$. Dans ce contexte, une hypothèse peut conclure sur différentes classes et le problème de l'apprentissage de concept peut être vu comme un problème de prédiction pour une partition des objets en $\{c_i, \neg c_i\}$. Cette stratégie est connue sous le nom de « one versus all ».

Le but est de trouver une hypothèse $H = \{h_1, \dots, h_n\}$, où chaque h_i est un ensemble de règles, telle que pour toute sous-hypothèse $h_i \in H$, h_i est valide et complète pour c_i. Pour construire une sous-hypothèse h_i, on effectue un apprentissage supervisé classique, décrit dans la section précédente, en considérant les ensembles d'exemples positifs $E^+_{c_i}$ et négatifs $E^-_{\neg c_i}$. Les règles d'une sous-hypothèse h_i sont de la forme $c_i \leftarrow t$, où c_i est la classe.

Dans la figure 3.2, on illustre un exemple d'apprentissage supervisé multiclasse. On considère trois classes $(+, -, \times)$. Les carrés bleus, vert et jaune représentent les règles qui ont été apprises pour chaque classe respective. Cette figure illustre les différents problèmes liés à la décision de la classe d'un nouvel exemple. Une observation peut être couvert par plusieurs règles concluant sur différentes classes, ou bien n'être couvert par aucune règle. Dans ces cas, de nombreuses stratégies peuvent être utilisées pour décider de la classe de l'observation, dont nous allons citer les plus communes. Si une nouvelle observation est couverte par plusieurs règles qui concluent sur des classes différentes, il est possible

d'effectuer une classification par vote, où la classe attribuée à la nouvelle observation est la classe sur laquelle concluent le plus grand nombre de règles qui couvrent cette observation. S'il y a égalité, on peut attribuer la classe qui a le plus grand nombre d'exemples d'apprentissage couverts par les règles qui couvrent l'observation. Si une nouvelle observation n'est couverte par aucune règle, on peut par exemple attribuer la classe qui a le plus grand nombre d'exemples d'apprentissage.

3.2 Apprentissage supervisé en logique du premier ordre

Dans cette section, nous présentons une introduction à l'apprentissage supervisé en logique du premier ordre, en nous intéressant à la Programmation Logique Inductive (PLI). La PLI est un sous-domaine de l'apprentissage automatique supervisé introduit en 1994 par Stephen Muggleton [MR94], dont l'objectif est d'apprendre des programmes logiques de premier ordre et des règles logiques à partir d'exemples et d'une théorie du domaine. La PLI se distingue d'autres méthodes d'apprentissage automatique par le fait que les modèles appris sont des programmes logiques exprimés en logique du premier ordre. Ainsi, de la même façon que les programmes d'apprentissage en logique propositionnelle, on peut classer la PLI dans la catégorie des programmes d'apprentissage symbolique, lesquels utilisent une représentation humainement compréhensible des données en entrée et produisent des résultats de même nature. Pour plus de détails sur la PLI, on peut se référer à [MR94 ; NW97 ; Cro+21].

3.2.1 Cadre d'apprentissage en PLI

Un programme d'apprentissage en PLI essaie d'apprendre un programme logique à partir d'un ensemble d'exemples positifs et négatifs, qui est un ensemble de clauses permettant d'expliquer ou de classifier ces exemples. Pour cela, comme dans l'apprentissage supervisé de concept, la PLI utilise deux ensembles d'informations : l'ensemble des exemples positifs et l'ensemble des exemples négatifs. Les exemples positifs sont des objets de l'univers pour lesquels le concept cible que l'on cherche à apprendre est vrai, tandis que les exemples négatifs sont des objets pour lesquelles ce concept cible est faux.

La PLI utilise une *théorie du domaine* B (en anglais *background knowledge*). Cette théorie B fournit un ensemble d'informations sur les différents objets du domaine de discours. La théorie du domaine peut inclure des faits et des règles donnant des informations pour le domaine d'application. L'objectif principal de la PLI est donc d'apprendre un programme logique qui *couvre* l'ensemble des positifs, et *ne couvre pas* l'ensemble des négatifs. Les notions de complétude, de validité et de couverture seront détaillées dans les sections suivantes, car elles dépendent du cadre d'apprentissage considéré.

Il existe différents cadres d'apprentissage pour la PLI, qui se distinguent sur trois paramètres [QI92] :

— Le langage décrivant les hypothèses ;

— Le langage décrivant les exemples d'apprentissage ;

— La relation de couverture d'une hypothèse sur les exemples.

Parmi les différents types de cadres, les deux les plus utilisés que nous décrivons ci-dessous sont le cadre d'*apprentissage par implication logique* [MR94], et le cadre d'*apprentissage par interprétation* [Blo+00 ; RD94]. D'autres cadres d'apprentissage comme l'*apprentissage par satisfiabilité* [DD98], et plus récemment l'*apprentissage par transitions* [IRS14 ; Eva+19 ; Rib+21], et l'*apprentissage par answer set* [LRB14] sont développés dans la littérature, mais ne font pas l'objet d'une étude dans cette thèse. Pour plus de détails sur ces cadres, nous redirigeons le lecteur vers les articles cités.

Apprentissage par implication logique

Nous présentons dans cette section le cadre classique de la PLI, à savoir l'apprentissage par implication logique [MR94]. Dans ce cadre, on considère un prédicat cible c représentant le concept à apprendre. Les exemples sont des faits dont le symbole de prédicat est c et forment l'ensemble des exemples E. On distingue les exemples qui sont vrais, qui forment l'ensemble des exemples positifs E^+ et les exemples qui sont faux, qui forment l'ensemble des exemples négatifs E^-. La théorie du domaine B est un ensemble fini de clauses et l'hypothèse H est un ensemble de clauses définies dont la tête est un littéral ayant c comme symbole de prédicat. Dans ce cadre, la relation de couverture d'une hypothèse H sur un exemple e est déterminée par l'implication logique : H couvre e si e est conséquence logique de B, H, noté $B, H \models e$, et plus précisément s'il existe une clause $S \in H$ telle que $B, S \models e$. La couverture d'une clause S sur un ensemble d'observations E est définie comme $cov(S, E) = \{e \in E \mid S \text{ couvre } e\}$.

Étant donné E^+ et E^- les ensembles d'exemples positifs et négatifs, et B la théorie du domaine, le but est d'apprendre une hypothèse H qui est un ensemble de clauses définies et telle que :

1. pour tout $e \in E^+ : B, H \models e$
2. pour tout $e \in E^- : B, H, \neg e \not\models \bot$

La condition 1 assure que l'hypothèse apprise est complète, c'est-à-dire qu'elle couvre tous les exemples positifs. La condition 2 assure que l'hypothèse apprise est valide, c'est-à-dire qu'elle ne couvre pas les exemples négatifs.

Exemple 3.3. *Considérons un ensemble de faits sur des individus, leurs relations familiales et leur genre. Notre objectif est d'apprendre une hypothèse qui nous permet de déduire si un individu est le grand-père d'un autre.*

Théorie du domaine B :

$parent(jean, pierre)$
$parent(jean, eve)$
$parent(pierre, luc)$
$parent(eve, david)$
$parent(sarah, pierre)$
$parent(jacques, emma)$
$parent(marie, emma)$
$parent(emma, sophie)$
$homme(jean)$
$homme(pierre)$
$homme(jacques)$
$homme(david)$
$femme(marie)$
$femme(emma)$
$femme(sarah)$

Exemples positifs E^+ :

$grandpere(jean, luc)$
$grandpere(jean, david)$
$grandpere(jacques, sophie)$

Exemples négatifs E^- :

$grandpere(pierre, sophie)$
$grandpere(marie, emma)$
$grandpere(sarah, luc)$

En utilisant l'apprentissage par implication logique, on peut apprendre une hypothèse H composée de la règle suivante :

$$\forall X \forall Z(grandpere(X, Z)) \leftarrow \exists Y(parent(X, Y) \wedge parent(Y, Z) \wedge homme(X))$$

Cette règle stipule que pour tout individu X et tout individu Z, X est le grand-père de Z s'il existe un individu Y tel que X est le parent de Y qui est lui-même le parent de Z, et si X est un homme.

Apprentissage par interprétations

L'*apprentissage par interprétation* [RD94 ; Blo+00] est un cadre de la PLI différent de l'apprentissage par implication logique présenté ci-dessus. Les exemples sont des *interprétations de Herbrand*, au lieu d'être des faits. Toutes les informations concernant l'exemple doivent donc être contenues dans l'interprétation et le système d'apprentissage ne se base que sur ces interprétations pour construire une théorie. Dans ce cadre, les exemples ne doivent plus être impliqués par l'hypothèse apprise et la théorie du domaine, mais doivent être des modèles de l'hypothèse apprise. Parmi les systèmes de PLI les plus connus dans le cadre par interprétations, on peut citer ICL [RL95], CLAUDIEN [DD97] et TILDE [BR98]. Différentes versions de l'apprentissage par interprétation sont disponibles dans la littérature, nous choisissons de présenter celle basée sur les travaux de [RL95 ; Rae97].

On dispose alors des éléments suivants :

— Un concept cible c étant un prédicat d'arité nulle ;

— Un ensemble d'exemples $\mathcal{I}^+$ où chaque élément $e \in \mathcal{I}^+$ est une interprétation de Herbrand pour laquelle le concept c est vrai ;

— Un ensemble d'exemples $\mathcal{I}^-$ où chaque élément $e \in \mathcal{I}^-$ est une interprétation de Herbrand pour laquelle le concept c est faux.

Dans le cadre de l'apprentissage par interprétation, une hypothèse H pour un concept c est sous forme DNF, $H = m_1 \vee m_2 \vee \ldots \vee m_n$ avec m_k un motif relationnel. Un exemple e est couvert par une hypothèse H ssi e est un modèle de H, c'est-à-dire qu'il existe un motif m_k dans H tel que e est un modèle de m_k. Par ailleurs, la couverture d'un motif relationnel m sur un ensemble d'observations $\mathcal{I}$ est définie comme $cov(m, \mathcal{I}) = \{e \in \mathcal{I} \mid m \text{ couvre } e\}$.

Ainsi, le but est de trouver H telle que :

— pour tout $e \in \mathcal{I}^+$: e est un modèle de H ;

— pour tout $e \in \mathcal{I}^-$: e n'est pas un modèle de H.

Exemple 3.4. *Soit les interprétations de Herbrand* $e_1 = \{p(1), q(1)\}$ *et* $e_2 = \{q(2), r(2)\}$ *qui forment l'ensemble des exemples positifs, et* $e_3 = \{p(2), q(3), r(4)\}$ *un exemple négatif. On peut apprendre une hypothèse* H *sous forme de DNF telle que :*

$$H = \exists(p(X) \wedge q(X)) \vee \exists(q(Y) \wedge r(Y))$$

e_1 *est un modèle de* H*, car il est un modèle de* $\exists(p(X) \wedge q(X))$*,* e_2 *est un modèle de* H*, car il est un modèle de* $\exists(q(Y) \wedge r(Y))$*, et* e_3 *n'est pas un modèle de* H*.*

Dans certaines approches de l'apprentissage par interprétation, on peut utiliser une théorie du domaine B pour compléter les connaissances contenues dans les exemples. Les exemples sont alors considérés comme des interprétations partielles. On utilise donc un mécanisme d'inférence pour *saturer* la représentation d'un exemple. La saturation [Rou94] d'un exemple est une opération qui consiste à ajouter dans l'interprétation de l'exemple tous les faits qui peuvent être déduits de l'exemple et de la théorie du domaine B. L'exemple ci-dessous montre un exemple d'apprentissage par interprétation dans le cas où les exemples sont des interprétations partielles. On note $sat(u)$ l'interprétation saturée de l'exemple e.

Exemple 3.5. *On considère un problème d'apprentissage par interprétation où l'on cherche à apprendre un programme logique qui modélise la décision d'un joueur dans un jeu de cartes. Nous sommes dans un scénario où le joueur peut décider de jouer le premier coup du jeu ou bien de passer. Les exemples correspondent à la description de la main du joueur et l'étiquette considérée est le prédicat* $c = open$ *qui indique si le joueur ouvre. On a alors :*

Théorie du domaine B *:*

$$big\_card(as) \qquad small\_card(7)$$
$$big\_card(roi) \qquad small\_card(8)$$
$$big\_card(dame) \qquad small\_card(9)$$
$$big\_card(valet) \qquad small\_card(10)$$

Ensemble des exemples positifs E^+*, composé de trois exemples pour lesquels* c *est vrai :*

$$e_1 = \{has\_card(as), has\_card(roi), has\_card(7)\}$$
$$e_2 = \{has\_card(roi), has\_card(dame), has\_card(10)\}$$
$$e_3 = \{has\_card(valet), has\_card(8), has\_card(9)\}$$

Ensemble des exemples négatifs E^-*, composé de trois exemples pour lesquels* c *est faux :*

$$e_4 = \{has\_card(7), has\_card(8)\}$$
$$e_5 = \{has\_card(10), has\_card(7)\}$$
$$e_6 = \{has\_card(8), has\_card(9), has\_card(10)\}$$

Ici par exemple, la saturation de l'exemple e_1 *est* $sat(e_1) = \{has\_card(as), has\_card(roi), has\_card(7), big\_card(as), big\_card(roi), small\_card(7)\}$.

En utilisant l'apprentissage par interprétation, on peut apprendre une hypothèse H *composée du motif suivant :*

$$\exists(has\_card(X) \wedge big\_card(X)).$$

Ce motif rejette l'ensemble des exemples négatifs et couvre l'ensemble des exemples positifs.

Ce cadre présente certaines limitations. Premièrement, il n'est pas possible d'apprendre une hypothèse composée de clauses récursives. On retrouve par exemple ce type de clauses dans la manipulation de listes. Deuxièmement, le cadre de l'apprentissage par interprétation n'autorise pas les modèles de Herbrand infinis, restreignant ainsi les langages utilisables à ceux contenant un nombre fini de constantes, et n'ayant pas de symboles de fonction d'arité supérieure à 1.

Cette perte d'expressivité est compensée par un gain en facilité d'apprentissage, en effet, il a été montre dans [RD94] que le cadre de l'apprentissage par interprétation montrait de plus fortes garanties en termes de PAC-apprenabilité que le cadre général, avec no-

tamment des résultants montrant que les *théories jk-clausales* privées de fonctions d'arité ≥ 1, c'est-à-dire des théories contenant des clauses avec au plus k littéraux d'arité au plus j, sont PAC-apprenables en temps polynomial [Val84] dans l'apprentissage par interprétation.

Passage d'un cadre à un autre

Nous nous intéressons dans cette sous-section aux liens que présentent les cadres d'apprentissage par interprétation et par implication logique. Il est possible de réécrire un problème d'apprentissage par interprétation dans un cadre d'apprentissage par implication logique. Le but de cette transformation est de pouvoir utiliser des algorithmes d'apprentissage opérant dans le cadre par implication logique pour résoudre des problèmes du cadre par interprétations.

Dans une première étape, on réécrit le formalisme de l'apprentissage par interprétation en un formalisme équivalent, mais dans lequel l'hypothèse est un ensemble de clauses (donc sous forme de CNF), au lieu d'être une disjonction de motifs relationnels. L'hypothèse est alors de la forme $H = (c \leftarrow m_1) \wedge (c \leftarrow m_2) \wedge \ldots \wedge (c \leftarrow m_n)$ avec m_k, $k \in 1 \ldots n$ un motif relationnel et c le littéral représentant le concept cible. Les exemples positifs et négatifs sont toujours des interprétations de Herbrand, et on note $\dot{e}$ la conjonction des littéraux formés par les atomes de l'interprétation e : si un atome est assigné à vrai dans e alors c'est un littéral positif dans $\dot{e}$, sinon c'est un littéral négatif dans $\dot{e}$. Un exemple e est couvert par une hypothèse H ssi $\dot{e}, H \models c$.

Soit $\mathcal{I}^+$ l'ensemble des exemples positifs et $\mathcal{I}^-$ l'ensemble des exemples négatifs et c le concept cible. L'objectif de l'apprentissage par interprétation est de trouver une hypothèse H telle que :

— pour tout $e \in \mathcal{I}^+ : \dot{e}, H \models c$

— pour tout $e \in \mathcal{I}^- : \dot{e}, H \not\models c$

L'exemple suivant est l'adaptation de l'exemple 3.4 à la nouvelle représentation de l'hypothèse.

Exemple 3.6. *Soit les interprétations de Herbrand* $e_1 = \{p(1), q(1)\}$ *et* $e_2 = \{q(2), r(2)\}$ *qui forment l'ensemble des exemples positifs, et* $e_3 = \{p(2), q(3), r(4)\}$ *un exemple négatif. On peut apprendre une hypothèse* H *sous forme de CNF telle que :*

$$H = (c \leftarrow p(X) \wedge q(X)) \wedge (c \leftarrow q(Y) \wedge r(Y))$$

e_1 *est couvert par* H*, car* $\dot{e}_1, H \models c$*,* e_2 *est couvert par* H*, car* $\dot{e}_2, H \models c$*, et* e_3 *n'est pas couvert par* H*, car* $\dot{e}_3, H \not\models c$*. Les variables* X *et* Y *sont quantifiées existentiellement, mais sont omises pour plus de clarté.*

Dans une deuxième étape, on adapte les exemples pour les rendre compatibles avec le cadre par implication logique. Pour cela, chaque littéral d'un exemple est transformé en fait et est ajouté à la théorie du domaine B. Afin de distinguer les propriétés de chaque exemple, on ajoute un argument qui est la clef de l'exemple à tous les faits concernant

cet exemple. Par exemple, le premier exemple positif de l'exemple 3.6 est réécrit dans le cadre par implication logique comme suit :

$$p(e_1, 1).$$
$$q(e_1, 1).$$

Ces deux faits sont ajoutés à la théorie du domaine B, de même pour les autres exemples.

Il reste maintenant à construire les ensembles d'exemples positifs et négatifs E^+ et E^-, qui pour rappel, sont des ensembles de faits dans le cadre de l'apprentissage par implication logique. Pour former ces ensembles, on ajoute un argument qui est la clef de l'exemple à chaque prédicat cible. Dans notre exemple, *open* devient $open(e_1)$ et $open(e_2)$ sont des faits de E^+, et $open(e_3)$ est un fait de E^-. Ci-dessous, on présente la transformation complète de l'exemple 3.6 dans le cadre par implication logique.

Exemple 3.7. *Soit les interprétations de Herbrand* $e_1 = \{p(1), q(1)\}$ *et* $e_2 = \{q(2), r(2)\}$ *qui forment l'ensemble des exemples positifs, et* $e_3 = \{p(2), q(3), r(4)\}$ *un exemple négatif. On peut réécrire ces exemples dans le cadre par implication logique comme suit :*

La théorie du domaine *B est alors :*

$$p(e_1, 1).$$
$$q(e_1, 1).$$
$$q(e_2, 2).$$
$$r(e_2, 2).$$
$$p(e_3, 2).$$
$$q(e_3, 3).$$
$$r(e_3, 4).$$

Les exemples positifs E^+ *sont :*

$$open(e_1).$$
$$open(e_2).$$

Les exemples négatifs E^- *sont :*

$$open(e_3).$$

L'hypothèse H apprise par un programme dans le cadre d'apprentissage par implication logique peut être la suivante :

$$H = \forall A(open(A) \leftarrow \exists X(p(A, X) \wedge q(A, X))) \wedge (open(A) \leftarrow \exists Y(q(A, Y) \wedge r(A, Y))).$$

Cette représentation est maintenant compatible avec le cadre par implication logique. On peut alors utiliser des algorithmes opérant dans ce cadre pour résoudre des problèmes de l'apprentissage par interprétation. La question des liens et des différentes transformations possibles entre les cadres d'apprentissage de la PLI est étudiée de façon plus approfondie dans [Rae97].

3.2.2 Systèmes d'apprentissage en PLI

Nous avons présenté les bases théoriques de la PLI et les différents cadres d'apprentissage. Nous allons maintenant introduire les critères distinctifs des systèmes d'apprentissage en PLI, à savoir les *biais* et les *stratégies de recherche*.

Biais

Le *biais* d'un algorithme d'apprentissage constitue un ensemble de contraintes de l'espace de recherche d'une hypothèse. Les biais sont importants en PLI, car ils permettent de réduire l'espace de recherche, souvent très grand, et de guider la recherche vers des hypothèses que l'on préfère. Il est à noter qu'il y a un compromis à trouver entre la restriction de l'espace de recherche et la perte de qualité des hypothèses apprises. En effet, un biais trop restrictif peut empêcher de trouver une hypothèse correcte, tandis qu'un biais trop permissif peut rendre la recherche infaisable en pratique.

On peut distinguer trois types de biais tels que décrits dans [Néd+96], les *biais de langage*, les *biais de recherche* et les *biais de validation*.

Biais de langage. Le biais de langage concerne les restrictions sur les clauses (ou les motifs dans le cadre par interprétations) dans l'espace de recherche. Par exemple, on peut restreindre le nombre de littéraux ou le nombre de variables dans une clause ou un motif, ou même se restreindre aux clauses de Horn dans le cadre général. La forme de biais de langage la plus utilisée parmi les systèmes de PLI (TILDE [BR98], ALEPH [Sri99]) est la *déclaration de modes* [Mug95a]. La déclaration de modes permet de restreindre l'espace des hypothèses en restreignant la forme des règles de l'hypothèse, en déclarant les prédicats qui peuvent apparaître dans la règle, le type des arguments dans les prédicats (variable ou constante), le nombre d'apparitions d'un même prédicat dans une règle, les appariements des variables autorisés, etc. *modeh* indique les prédicats autorisés et leur mode dans la tête de la règle et *modeb* pour le corps. L'ensemble des hypothèses possibles étant donné un biais de langage est appelé l'*espace des hypothèses* [Mit77].

Ci-dessous, on présente un exemple de biais de langage dans le cadre général au format d'ALEPH. ALEPH adopte une stratégie de « diviser pour régner », qui consiste à ajouter des règles une par une à l'hypothèse jusqu'à ce que tous les exemples soient couverts, le biais de langage d'ALEPH restreint donc les règles individuelles de l'hypothèse. Dans l'exemple, on restreint les prédicats autorisés dans la tête de la règle à *grandpere*, et dans le corps de la règle à *parent*, *homme* et $femme$. L'argument le plus à gauche de chaque mode indique le nombre de fois que le prédicat en deuxième argument peut apparaître dans la règle et le plus à droite indique les types d'arguments autorisés dans le prédicat. Les variables dites de *sortie* ayant en préfixe − sont les variables qui peuvent être introduites dans la règle en construction sans qu'elles apparaissent déjà dans la règle et les variables dites d'*entrée* ayant en préfixe + sont les variables qui doivent être déjà présentes dans la règle en construction. Les constantes seront quant à elles notées par #. On peut également typer les variables grâce aux mots-clef, ici $personne$, permettant d'indiquer que deux arguments dans différents prédicats peuvent partager une même variable.

$$modeh(1, grandpere(+personne, -personne)).$$
$$modeb(*, parent(+personne, -personne)).$$
$$modeb(2, homme(+personne)).$$
$$modeb(2, femme(+personne)).$$

Biais de recherche. Le biais de recherche concerne les stratégies de recherche utilisées pour explorer l'espace des hypothèses. En effet, malgré le fait que l'espace des hypothèses soit restreint par le biais de langage, il reste souvent très grand, et il est nécessaire d'utiliser des stratégies de recherche pour explorer plus efficacement cet espace. L'un des principaux biais de recherche est l'*opérateur de raffinement* [NW97] choisi pour *spécialiser* (opérateur descendant) ou *généraliser* (opérateur ascendant) une clause durant la recherche. Parmi les opérateurs de raffinement, on peut prendre comme exemple l'ajout ou la suppression d'un littéral de la règle en construction, l'instanciation ou la généralisation d'un terme, etc. Savoir si une clause C est une spécialisation ou une généralisation d'une clause D est déterminé par un *ordre de spécificité*. L'ordre de spécificité est décrit plus en détails dans la section 3.3.

L'algorithme de recherche choisi est également un des principaux biais de recherche et définit la manière dont la solution sera cherchée dans l'ordre de spécificité choisi. Par exemple, l'ensemble des hypothèses forme un *treillis* (une borne inférieure et une borne supérieure, voir sous-section 2.2.4) si l'ordre choisi est la θ-subsomption et le choix de la stratégie de recherche définira la façon par laquelle le treillis est exploré. Ce choix de la stratégie de recherche est souvent utilisé pour classer les différents programmes de PLI dans la littérature, on distingue les approches *descendantes* qui partent d'une hypothèse générale et la spécialisent, les approches *ascendantes* qui partent d'une hypothèse spécifique et la généralisent, les approches *mixtes* qui combinent les deux approches, et les approches *meta-level* qui délèguent la recherche de l'hypothèse à un solveur externe. Les principaux programmes de PLI et leur stratégie de recherche associée sont présentés de façon plus détaillée ci-dessous.

Par ailleurs, on peut citer d'autres biais de recherche comme la manière dont les exemples sont sélectionnés pour être utilisés dans la recherche, la fonction de guidage permettant d'évaluer une hypothèse dans l'espace de recherche, etc.

Biais de validation. Le biais de validation concerne les critères utilisés pour évaluer une hypothèse apprise. Ce biais est un critère d'arrêt de la recherche qui permet de déterminer si une hypothèse est suffisamment bonne pour être acceptée et pour arrêter la recherche. Parmi les biais de validation, on peut citer par exemple, l'arrêt de la recherche lorsque tous les exemples positifs sont couverts.

Algorithmes de recherche

En PLI, les algorithmes de recherche d'une hypothèse peuvent se classer dans deux grandes familles, les approches *descendantes* (*top-down*) et les approches *ascendantes*

(*bottom-up*). Ces deux types d'approches diffèrent par le sens dans lequel les hypothèses sont construites dans l'ordre de spécificité sur les clauses considéré. Le sens de la recherche influe directement sur la fréquence de l'utilisation d'un type d'opérateur de raffinement (par exemple dans une approche descendante, on utilisera le plus souvent un opérateur de raffinement descendant). Des approches dites « mixtes » utilisent les deux types d'approches en ayant une étape ascendante pour générer l'espace de recherche, puis descendante pour effectuer la recherche. Une quatrième famille d'approches plus récente a fait son apparition et se nomme *meta-level* [CRL11 ; Ino16] et concerne les approches raisonnant sur les programmes directement, en déléguant le plus souvent la recherche de l'hypothèse à un solveur externe. Nous décrivons plus en détails ci-dessous les approches ascendantes, descendantes, mixtes et meta-level, en renseignant les différents programmes de PLI utilisant chacun de ces types d'approches.

Approches descendantes. Dans les approches descendantes (FOIL [Qui90], Claudien [DD97], TILDE [BR98], QuickFOIL [ZPP14]), on parcourt l'espace de recherche du plus général au plus spécifique. On part d'une clause S de l'hypothèse H tel que S est la plus générale, couvrant donc à la fois E^+ et E^-. On raffine ensuite S en la spécialisant pour qu'elle couvre moins d'exemples négatifs, tout en couvrant toujours les exemples positifs.

Approches ascendantes. Dans les approches ascendantes (Golem [MF90], XHAIL [Ray09]), on parcourt l'espace de recherche du moins général au plus général. On part d'une clause S telle que S est la plus spécifique qui couvre un exemple de E^+ et rejette tous les exemples de E^-. On raffine ensuite en généralisant S pour qu'elle couvre plus d'exemples positifs, tout en ne couvrant toujours aucun exemple négatif.

Approches mixtes. Les approches mixtes sont des approches qui sont descendantes, mais qui utilisent l'approche ascendante afin de borner la recherche. Le système Progol [Mug95b] introduit cette stratégie en construisant la *bottom clause* d'un exemple graine de l'ensemble d'apprentissage, c'est-à-dire la clause la plus spécifique qui couvre l'exemple relativement aux modes et à la théorie du domaine, construite par l'algorithme de saturation [Rou94]. Cette clause définit l'espace de recherche d'une hypothèse dont les éléments sont les littéraux de la bottom clause. La recherche de la meilleure hypothèse est alors effectuée en utilisant une approche descendante sur cet espace de recherche, de façon à ce qu'elle couvre le plus d'exemples positifs et aucun négatif.

Approches meta-level. Le terme désigne de nombreuses approches qui n'ont pas beaucoup de choses en commun. Il est malgré tout possible de présenter les approches *meta-level* (Metagol [MLT15], δILP [EG17], Popper [MC21]) comme étant des approches qui n'utilisent pas directement un algorithme descendant ou ascendant pour construire une hypothèse. Les approches classiques (ascendantes et descendantes) utilisant une stratégie gloutonne, elles sont souvent limitées en efficacité par un espace de recherche qui est souvent très grand. Au contraire, dans les approches meta-level, la recherche d'un

programme logique couvrant tous les exemples positifs et rejetant tous les exemples négatifs est formulé comme un problème de satisfaction de contraintes syntaxiques sur les programmes. La recherche de programme satisfaisant les contraintes est alors déléguée à un solveur externe, permettant d'utiliser les programmes de l'état de l'art, dont les plus couramment utilisés par les approches meta-level sont les solveurs *answer set programming* (ASP). Le raisonnement sur les programmes tel qu'effectué dans les approches meta-level permettent de réduire l'espace de recherche de façon plus efficace que les approches classiques, permettant, dans le cas par exemple de Popper [MC21], de réduire significativement le temps d'apprentissage.

Pour une étude plus approfondie des différentes méthodes et programmes dans le cadre du meta-level, nous invitons le lecteur à se référer à [Cro+21].

3.2.3 Un système de PLI : NukLear

NukLear est un système d'apprentissage en PLI propriétaire développé par l'entreprise NukkAI. NukLear a les mêmes objectifs que le système Aleph qui est l'un des systèmes d'apprentissage PLI les plus utilisés, car très bien documenté. C'est un système d'apprentissage en PLI qui permet d'apprendre des programmes logiques de premier ordre à partir d'exemples positifs et négatifs, et d'une théorie du domaine, et opère dans le cadre de l'apprentissage par implication logique.

Paramètres

NukLear prend en entrée un ensemble d'exemples positifs E^+ et négatifs E^- qui sont des faits, ainsi qu'une théorie du domaine B. Chaque exemple est une instance du prédicat cible. La théorie du domaine est un ensemble de faits et de clauses du premier ordre décrivant les connaissances globales du domaine considéré. À cela, s'ajoute un ensemble de contraintes M sur la syntaxe des clauses à apprendre (biais de langage), sous forme de déclarations de modes. L'objectif est d'apprendre une hypothèse $H = \{r_1, \ldots, r_k\}$, un ensemble de clauses définies concluant sur le prédicat cible, telle que :

— H respecte les modes M ;
— pour tout $e^+ \in E^+$, $H, B \models e^+$;
— pour tout $e^- \in E^-$, $H, B \not\models e^-$.

Une particularité du système NukLear est qu'il est capable d'apprendre des règles sous hypothèse OI.

Algorithme

L'algorithme de NukLear est similaire à celui d'Aleph. Il utilise la stratégie « diviser pour régner », c'est-à-dire qu'il construit des règles une à une, qui sont valides, qu'il ajoute à l'hypothèse jusqu'à ce qu'elle couvre la totalité des exemples positifs. Plus précisément, l'étape de construction d'une règle se déroule comme suit :

1. Un exemple graine non couvert par l'hypothèse courante est choisi. S'il n'existe pas, l'algorithme est arrêté et on retourne l'hypothèse courante ;

2. La *bottom clause* est construite à partir de l'exemple graine, c'est la clause la plus spécifique couvrant l'exemple, vérifiant les contraintes de modes dans M ;
3. Recherche d'une règle plus générale que la bottom clause en cherchant parmi les sous-ensembles de littéraux de la bottom clause ceux qui ont le meilleur *score* ;
4. La règle est ajoutée à l'hypothèse.

Bottom clause. Durant l'étape 2 de l'algorithme, on construit la bottom clause, qui est la clause la plus spécifique couvrant l'exemple graine et dont la taille est bornée par un ensemble de modes. Elle permet de former alors un treillis représentant l'espace de recherche d'une règle, où la borne inférieure est la bottom clause et la borne supérieure est la clause vide. Les nœuds intermédiaires du treillis sont les sous-ensembles de littéraux de la bottom clause. NukLear cherche une règle dans le treillis formé par la bottom clause. La construction de cette clause est décrite dans [Mug95b].

Exemple 3.8. *On considère un sous-ensemble de la théorie du domaine de l'exemple 3.5 décrite dans le cadre de l'apprentissage par implication logique. On a alors :*

Théorie du domaine B *:*

$$
\begin{aligned}
&big\_card(as)\\
&big\_card(roi)\\
&small\_card(7)\\
&has\_card(ex1, as)\\
&has\_card(ex1, roi)\\
&has\_card(ex1, 7).
\end{aligned}
$$

On considère également l'exemple positif $ex1$ *qui est un élément de l'ensemble des exemples positifs* E^+ *:*

$$open(ex1)$$

On considère également l'ensemble des modes suivants :

$$
\begin{aligned}
&modeh(1, open(+ex))\\
&modeb(3, has\_card(+ex, -card))\\
&modeb(3, has\_card(+ex, \#card))\\
&modeb(1, big\_card(+card))\\
&modeb(1, small\_card(+card)).
\end{aligned}
$$

La bottom clause construite à partir de l'exemple $ex1$ *est alors :*

$$open(A) \leftarrow has\_card(A, as) \wedge has\_card(A, roi) \wedge has\_card(A, 7) \wedge \\ has\_card(A, B) \wedge big\_card(B) \wedge has\_card(A, C) \wedge small\_card(C)$$

Recherche d'une règle. L'algorithme de recherche de NukLear est un algorithme descendant, il effectue sa recherche d'une règle en partant de la plus générale et la spécifie jusqu'à ce qu'elle rejette tous les exemples négatifs. Cet algorithme est basé sur la stratégie de *recherche en faisceau* (*beam search*), qui est un cas particulier de la stratégie de *séparation et évaluation* (*branch and bound*) dont l'algorithme est tiré de [PS82]. Le but de l'algorithme est de trouver une solution parmi toutes les solutions qui ait un *coût* minimal. La plupart du temps, l'espace des versions est très grand, ce qui rend leur énumération complète impossible. Le biais de recherche permet alors de restreindre cet espace en imposant des contraintes sur les solutions possibles.

Dans le cas général, l'algorithme effectue un parcours en largeur borné par le biais de recherche en générant et testant des règles de plus en plus spécifiques, formant un arbre de recherche, ce qui lui permet de trouver les solutions les plus générales possibles étant donné un biais de recherche. À chaque nœud de l'arbre de recherche, l'algorithme génère un ensemble de fils qui sont des spécialisations de la règle associée au nœud parent en ajoutant un littéral de la bottom clause. Les fils sont ensuite classés selon une *fonction de guidage* qui permet de choisir ceux qui vont être développés en priorité. La particularité de la recherche en faisceau est qu'elle introduit un entier k qui limite aux k-meilleurs nœuds, selon la fonction de guidage, ceux qui vont être développés à chaque étape. Dans NukLear, le score d'une règle calculé par la fonction de guidage est $P - N$, avec P le nombre d'exemples positifs couverts par la règle et N le nombre de négatifs couverts par la règle.

Une règle est solution si elle vérifie les conditions de validation du biais choisi, elle est évaluée selon une *fonction de coût* qui maximise le nombre d'exemples couverts par la règle qui ne sont pas encore couverts par l'hypothèse.

3.3 Least General Generalization

Dans le cadre de l'apprentissage par implication logique, la notion de *moindre généralisation* (*least general generalization* en anglais, *lgg*) est une notion centrale. Dans la contribution présentée en chapitre 6, nous utilisons la *lgg* pour construire le motif le plus spécifique qui décrit un ensemble d'observations. Afin de définir la notion de *lgg* plus précisément, nous allons discuter des ordres de spécificité sur les clauses et les motifs, et présenter l'algorithme de construction de la *lgg*. Cette section est basée sur les travaux de Plotkin [Plo70] et Reynolds [C70].

3.3.1 Ordre de spécificité sur les clauses et les motifs

L'ordre de spécificité est un ordre sur les clauses et les motifs permettant de déterminer si une clause ou un motif est plus général(e) ou plus spécifique qu'un(e) autre. Cet ordre

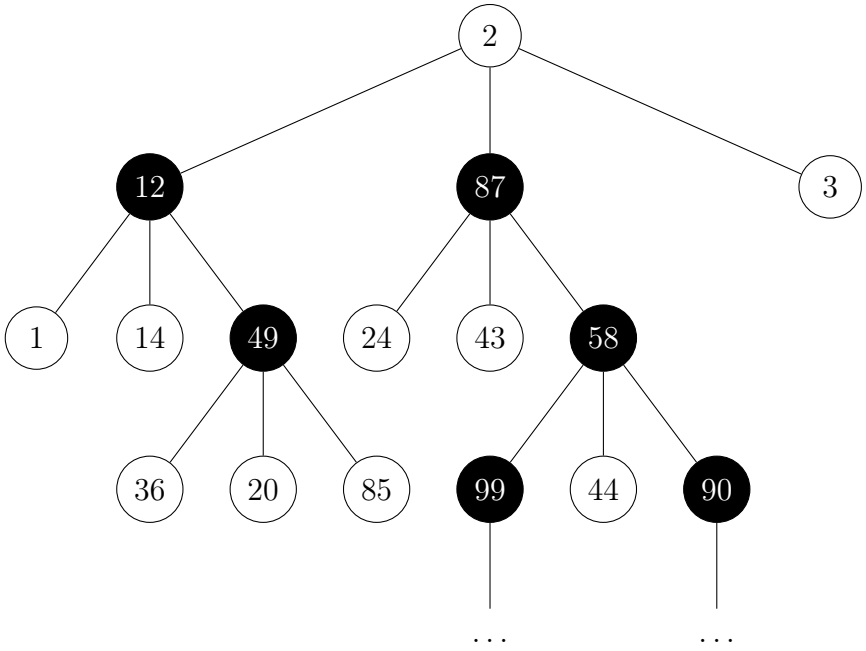


FIGURE 3.3 – Exemple de recherche en faisceau avec $k = 2$, la valeur dans chaque noeud est donnée par la fonction de guidage et l'algorithme cherche à maximiser cette valeur.

est noté par le symbole $\preceq_S$. Soit C et D deux clauses ou motifs, si $C \preceq_S D$, alors D est plus spécifique que C, C est plus général que D. Dans la PLI, l'ordre de spécificité que l'on utilise est le plus souvent basé sur la θ-subsomption. Si $C \preceq_\theta D$, alors $C \preceq_S D$.

Soit Σ un ensemble de clauses ou de motifs, $E \subseteq \Sigma$, on utilise la nomenclature suivante [NW97] :

— Soit $C, D \in \Sigma$. Si $C \preceq_\theta D$, alors C est plus générale que D, et D est plus spécifique que C. C est une *généralisation* de D et D est une *spécialisation* de C.

— Une borne inférieure (voir définition 2.26) $C \in \Sigma$ de $\langle E, \preceq_\theta \rangle$, si elle existe, est appelée un *moindre généralisé* (en anglais *least general generalization*, *lgg*).

— Une borne supérieure (voir définition 2.26) $C \in \Sigma$ de $\langle E, \preceq_\theta \rangle$, si elle existe, est appelée une *plus grande spécialisation*.

Dans la section suivante, nous présentons plus en détails la *lgg* sous θ-subsomption et nous présentons l'algorithme permettant de la construire.

3.3.2 Lgg sous θ-subsomption

Dans le cas de la θ-subsomption, qui est l'ordre que nous allons utiliser pour construire la *lgg* dans le chapitre 6, il existe une borne inférieure [Plo70], donc la *lgg* est unique. Ce n'est pas le cas, par exemple, de la OI-subsomption [LFF02] où l'unicité de cette *lgg* n'est pas garantie.

Dans cette section, nous présentons l'algorithme qui permet de calculer la *lgg* de deux clauses ou motifs sous θ-subsomption, après avoir redéfini la notion de *lgg* pour un

ensemble de clauses ou de motifs selon la θ-subsomption pour une meilleure compréhension de cette notion.

Définition 3.2 (*lgg* de deux clauses ou motifs)**.** *Une clause (ou un motif) S est la lgg de deux clauses (ou motifs) C et D (dénoté $S = lgg(C, D)$) ssi $S \preceq_\theta C$ et $S \preceq_\theta D$ et pour tout $C' \preceq_\theta C$ et $D' \preceq_\theta D$, $C' \preceq_\theta S$ et $D' \preceq_\theta S$.*

La définition suivante adapte la définition 3.2 pour la *lgg* d'un ensemble de clauses ou de motifs.

Définition 3.3 (*lgg* d'un ensemble de clauses ou motifs)**.** *Une clause (ou motif) S est la lgg d'un ensemble de clauses (ou motifs) Σ (dénoté $S = lgg(\Sigma)$) ssi pour tout $C_i \in \Sigma$, $S \preceq_\theta C_i$ et pour tout G_j tel que $G_j \preceq_\theta C_i$ pour tout C_i, $G_j \preceq_\theta S$.*

L'intuition de cette définition est que la *lgg* d'un ensemble de clauses (ou de motifs), est la clause (ou le motif) la plus spécifique qui θ-subsume cet ensemble de clauses (ou de motifs), et toute clause (ou motif) différent de la *lgg* qui θ-subsume la *lgg* est plus général que la *lgg*.

Exemple 3.9. *Considérons l'ensemble Σ composé des trois motifs suivants :*

$$m_1 : q(a) \wedge q(b) \wedge r(a) \wedge w(b) \wedge s(a)$$
$$m_2 : q(c) \wedge q(d) \wedge r(c) \wedge w(d) \wedge t(c)$$
$$m_3 : q(c) \wedge q(e) \wedge r(c) \wedge w(e) \wedge u(c)$$

Le motif $S = lgg(\Sigma)$ est alors :

$$S : \exists(q(X) \wedge r(X) \wedge q(Y) \wedge w(Y))$$

3.3.3 Construction de la *lgg* sous θ-subsomption

L'algorithme de la *lgg* sous θ-subsomption est un algorithme de construction itératif qui construit la *lgg* de deux clauses ou motifs en comparant les atomes de chaque clause ou motif. L'algorithme compare les atomes des deux formules ayant le même symbole de prédicat deux à deux, et s'il existe une généralisation possible entre les deux atomes, alors la généralisation est ajoutée à la *lgg*. L'algorithme qui généralise deux atomes est l'algorithme de l'anti-unification, qui effectue l'opération inverse de l'unification décrite dans la section 2.2.5. La construction de la *lgg* utilise l'algorithme de l'anti-unification pour généraliser les atomes deux à deux.

Anti-unification. L'anti-unification de deux atomes est l'équivalent de la *lgg* entre ces deux atomes. L'algorithme d'anti-unification prend en entrée deux atomes *conventionnels* (c'est-à-dire de la forme $p(t_1, \ldots, t_n)$ avec t_i un terme) A et B ayant le même symbole de prédicat et qui renvoie un atome L qui est un moindre généralisé de A et B, on a alors $L \preceq_\theta A$ et $L \preceq_\theta B$.

L'algorithme d'anti-unification [NW97] est décrit dans l'algorithme 3. Il utilise la notion d'ensemble de désaccord définie dans la définition 2.34.

Algorithme 3 Algorithme d'anti-unification de deux atomes

Entrée: Deux atomes conventionnels A et B avec le même symbole de prédicat.
Sortie: Un moindre généralisé de A et B.

$\theta \leftarrow \emptyset$, $\sigma \leftarrow \emptyset$, $A' \leftarrow A$, $B' \leftarrow B$, $i \leftarrow 0$.
Soit $\{Z_1, Z_2, \ldots\}$ un ensemble de variables n'apparaîssant ni dans A ni dans B.
tant que $A' \neq B'$ **faire**
    Soit k la position l'ensemble de désaccord de A' et B'.
    Soit t_A et t_B les termes à la position i de A' et B' respectivement.
    si pour un j avec $1 \leq j \leq i$, $Z_j\theta = t_A$ et $Z_j\sigma = t_B$ **alors**
        Remplacer t_A et t_B par Z_j dans A' et B' à la position k.
    sinon
        $i \leftarrow i + 1$
        Remplacer t_A et t_B par Z_i dans A' et B' à la position k.
        $\theta \leftarrow \theta \cup \{Z_i/t_A\}$.
        $\sigma \leftarrow \sigma \cup \{Z_i/t_B\}$.
    fin si
fin tant que
retourne A'

Exemple 3.10. *Considérons les deux atomes $A = p(a, b, b, c)$ et $B = p(a, d, d, e)$. On considère également l'ensemble des variables $\{X_1, X_2, \ldots\}$. L'exécution de l'algorithme de l'anti-unification sur ces deux atomes est effectuée de la façon suivante :*

1. *$A' = p(a, b, b, c)$ et $B' = p(a, d, d, e)$, $i = 0$, $\theta = \emptyset$, $\sigma = \emptyset$.*
   L'ensemble de désaccord est $\{b, d\}$ à la position 2. On incrémente i avec $i = 1$. On remplace donc b et d par X_1 dans A' et B' respectivement à la position 2. On ajoute $\{X_1/b\}$ à θ et $\{X_1/d\}$ à σ.
2. *$A' = p(a, X_1, b, c)$ et $B' = p(a, X_1, d, e)$, $\theta = \{X_1/b\}$, $\sigma = \{X_1/d\}$ et $i = 1$.*
   L'ensemble de désaccord est $\{b, d\}$ à la position 3. $j = 1$ est un j tel que $1 \leq j \leq 1$ et $X_j\theta = b$ et $X_j\sigma = d$, donc on remplace b et d par $X_1 = X_j$ dans A' et B' respectivement à la position 3.
3. *$A' = p(a, X_1, X_1, c)$ et $B' = p(a, X_1, X_1, e)$, $\theta = \{X_1/b\}$, $\sigma = \{X_1/d\}$ et $i = 1$.*
   L'ensemble de désaccord est $\{c, e\}$ à la position 4. Il n'existe pas de j tel que $1 \leq j \leq 1$ et $X_j\theta = c$ et $X_j\sigma = e$. On a donc $i = 2$, et on remplace c et e par X_2 dans A' et B' respectivement à la position 4. On ajoute $\{X_2/c\}$ à θ et $\{X_2/e\}$ à σ.
4. *$A' = p(a, X_1, X_1, X_2)$ et $B' = p(a, X_1, X_1, X_2)$, $\theta = \{X_1/b, X_2/c\}$, $\sigma = \{X_1/d, X_2/e\}$ et $i = 2$.*
   On a $A' = B'$, l'algorithme retourne le moindre généralisé de A et B qui est $p(a, X_1, X_1, X_2)$.

Nous avons vu comment l'algorithme d'anti-unification permettait de construire le moindre généralisé de deux atomes. Dans la section suivante, nous présentons l'algorithme de construction de la *lgg* de deux clauses ou motifs, qui est une généralisation de l'algorithme de l'anti-unification à deux ensembles de littéraux qui forment les clauses ou motifs.

Algorithme de Plotkin. L'algorithme de construction de la *lgg* est décrit dans [Plo70 ; NW97]. Il prend en entrée deux formules qui sont des clauses ou des motifs et renvoie la *lgg* de ces deux formules. On utilise dans cette section le terme *formule* pour désigner une clause ou un motif. Pour construire la *lgg* de ces deux formules, l'algorithme utilise une *représentation atomique* des formules et applique l'algorithme de l'anti-unification sur cette représentation. La représentation atomique des formules est obtenue en effectuant un ensemble de *sélections* sur les littéraux des formules. Deux littéraux forment une *sélection* s'ils ont le même symbole de prédicat et le même signe.

Définition 3.4. *Soit C et D deux clauses ou motifs. Une sélection de C et D est une paire de littéraux de la forme (l, m) avec $l \in C$ et $m \in D$ et l et m ont le même symbole de prédicat et le même signe.*

Les sélections de deux formules C et D sont une séquence de paires de littéraux de la forme :

$$l_1 m_1, \ldots, l_n m_n \tag{3.5}$$

Exemple 3.11. *Considérons les deux formules C et D sous forme d'ensembles de littéraux $C = \{p(a), p(b), \neg p(c)\}$ et $D = \{p(d), q(e), \neg p(f)\}$. La séquence $(p(a), p(d)), (p(b), p(d)), (\neg p(c), \neg p(f))$ constitue les sélections de C et D.*

De 3.5, on obtient la représentation atomique de C et D par les séquences des littéraux respectifs de C et D dans les paires de littéraux de la sélection : $C' = l_1, \ldots, l_n$ et $D' = m_1, \ldots, m_n$.

L'algorithme de construction de la *lgg* calcule alors l'anti-unification de C' et D' en appliquant l'algorithme 3 avec $A = C'$ et $B = D'$. L'algorithme renvoie un ensemble de littéraux $S = \{n_1, \ldots, n_n\}$ de C et D où n_i est le résultat de l'anti-unification de m_i et l_i. La disjonction (ou conjonction) des littéraux de S forme la *lgg* de C et D. On introduit la notation $\vee(L)$ avec L un ensemble de littéraux, pour désigner la disjonction des littéraux de L, et $\wedge(L)$ pour désigner la conjonction des littéraux de L. L'algorithme 4 décrit la construction de la *lgg* de deux clauses ou motifs.

Algorithme 4 Algorithme de construction de la *lgg* de deux clauses ou motifs

Entrée: Deux clauses (ou motifs) C et D.
Sortie: Une *lgg* S de C et D.

1: Construire l'ensemble des sélections de C et D $(l_1, m_1), \ldots, (l_n, m_n)$.
2: Construire la représentation atomique de C et D : $C' = (l_1, \ldots, l_n)$ et $D' = (m_1, \ldots, m_n)$.
3: Calculer l'anti-unification de C' et D' : $S' = \text{AntiUnification}(C', D')$.
4: $S \leftarrow \vee(S')$ si C et D sont des clauses, $S \leftarrow \wedge(S')$ si C et D sont des motifs.
5: $S \leftarrow$ ReduceClause(S)
6: **retourne** S

Exemple 3.12. *On veut calculer la lgg des deux motifs suivants :* $m_1 = q(a) \wedge r(a) \wedge q(b) \wedge w(b) \wedge s(a)$ *et* $m_2 = q(c) \wedge r(c) \wedge q(d) \wedge w(d) \wedge t(c)$. *Les sélections de* m'_1 *et* m'_2 *sont* $(q(a), q(c)), (r(a), r(c)), (q(a), q(d)), (q(b), q(c)), (q(b), q(d)), (w(b), w(d))$. *La représentation atomique de ces deux motifs est donc* $m'_1 = (q(a), r(a), q(a), q(b), q(b), w(b))$ *et* $m'_2 = (q(c), r(c), q(d), q(c), q(d), w(d))$. *On considère l'ensemble des variables* $\{X_1, X_2, ...\}$. *L'anti-unification de* m'_1 *et* m'_2 *est calculée de la manière suivante :*

1. $A' = q(a), r(a), q(a), q(b), q(b), w(b)$ *et* $B' = q(c), r(c), q(d), q(c), q(d), w(d)$, $i = 0$, $\theta = \emptyset$, $\sigma = \emptyset$.
   L'ensemble de désaccord est $\{a, c\}$ *à la position 1. On incrémente* i *avec* $i = 1$. *On remplace donc* a *et* c *par* X_1 *dans* A' *et* B' *respectivement à la position 1. On ajoute* $\{X_1/a\}$ *à* θ *et* $\{X_1/c\}$ *à* σ.
2. $A' = q(X_1), r(a), q(a), q(b), q(b), w(b)$ *et* $B' = q(X_1), r(c), q(d), q(c), q(d), w(d)$, $\theta = \{X_1/a\}$, $\sigma = \{X_1/c\}$ *et* $i = 1$.
   L'ensemble de désaccord est $\{a, c\}$ *à la position 2.* $j = 1$ *est un* j *tel que* $1 \leq j \leq i$ *et* $X_j\theta = a$ *et* $X_j\sigma = c$, *donc on remplace* a *et* c *par* X_1 *dans* A' *et* B' *respectivement à la position 2.*
3. $A' = q(X_1), r(X_1), q(a), q(b), q(b), w(b)$ *et* $B' = q(X_1), r(X_1), q(d), q(c), q(d), w(d)$, $\theta = \{X_1/a\}$, $\sigma = \{X_1/c\}$ *et* $i = 1$.
   L'ensemble de désaccord est $\{a, d\}$ *à la position 3. Il n'existe pas de* j *tel que* $1 \leq j \leq 1$ *et* $X_j\theta = a$ *et* $X_j\sigma = d$. *On a donc* $i = 2$, *et on remplace* a *et* d *par* X_2 *dans* A' *et* B' *respectivement à la position 3. On ajoute* $\{X_2/a\}$ *à* θ *et* $\{X_2/d\}$ *à* σ.
4. *En déroulant l'algorithme jusqu'à ce que* $A' = B'$, *l'algorithme retourne le moindre généralisé de* A *et* B *qui est* $S = q(X_1), r(X_1), q(X_2), q(X_3), q(X_4), w(X_4)$.

Soit après un renommage des variables, la lgg des deux motifs est $q(X) \wedge r(X) \wedge q(Y) \wedge q(Z) \wedge q(W) \wedge w(W)$. *Ici la lgg n'est pas réduite. Il faut maintenant calculer la réduction en appliquant l'algorithme 2 de réduction :*

1. $D = q(X) \wedge r(X) \wedge q(Y) \wedge q(Z) \wedge q(W) \wedge w(W)$, *trouvé* $= true$.
   En prenant $L = q(Y)$ *et* $\theta = \{Y/Z\}$, *on a* $D\theta = D \setminus \{q(Y)\} = q(X) \wedge r(X) \wedge q(Z) \wedge q(W) \wedge w(W)$.
2. $D = q(X) \wedge r(X) \wedge q(Z) \wedge q(W) \wedge w(W)$, *trouvé* $= true$.
   En prenant $L = q(Z)$ *et* $\theta = \{Z/W\}$, *on a* $D\theta = D \setminus \{q(Z)\} = q(X) \wedge r(X) \wedge q(W) \wedge w(W)$.
3. $D = q(X) \wedge r(X) \wedge q(W) \wedge w(W)$, *trouvé* $= true$.
   Il n'y a plus de littéraux à réduire, trouvé $= false$.

L'algorithme retourne alors la lgg réduite $S' = q(X) \wedge q(W) \wedge r(X) \wedge w(W)$.

Il est important de noter que le nombre de sélections entre deux formules C et D dont tous les littéraux sont de même signe et même prédicat est de $|C| \times |D|$ [NW97 ; Plo70], ce qui fait que la *lgg* aura également une taille de $|C| \times |D|$, autrement dit il y a autant de littéraux dans la *lgg* que de sélections. Dans le cas où l'on veut calculer la *lgg* d'un ensemble de n formules de taille m dont tous les littéraux sont de même signe et de même symbole de prédicat, le nombre de sélections sera de m^n, de même que pour la taille de la *lgg*. La taille de la *lgg* peut donc augmenter très rapidement en fonction du nombre de littéraux partagés entre les formules de l'ensemble.

3.4 Résumé

Dans ce chapitre, nous avons vu comment les méthodes d'apprentissage automatique supervisé peuvent être utilisées pour apprendre des concepts à partir de données. Nous avons introduit les notions de couverture et d'extension et avons vu comment les hypothèses peuvent être représentées par des règles logiques. Nous avons également introduit le concept de Programmation Logique Inductive, qui permet d'apprendre des programmes logiques à partir de données. Nous avons introduit également les notions de généralisation et de spécialisation, qui sont définies à partir d'un ordre de spécificité donné. Ces ordres de spécificité admettent parfois une borne inférieure, qui dans le cas des clauses est appelée la moindre généralisation (ou *lgg*). Nous avons étudié plus en détails la *lgg* sous θ-subsomption, qui garantit l'unicité de cette *lgg*. Cette *lgg* occupe une position centrale dans la contribution présentée dans le chapitre 6, nous avons donc vu comment la *lgg* pouvait être calculée en utilisant l'algorithme de l'anti-unification et avons présenté les limites dans la construction de la *lgg*, qui peut s'avérer très grande en nombre de littéraux, d'où l'utilité de calculer une approximation de cette *lgg*, comme présenté dans le chapitre 6.

Deuxième partie

Contributions

Chapitre 4

Apprentissage multiannotateur supervisé

Introduction

Dans le cadre de l'apprentissage supervisé, il est courant que les données soient étiquetées par de multiples sources. Ces sources peuvent être différentes personnes, différents capteurs, différents algorithmes, etc. Apprendre un modèle sur toutes les données annotées devient difficile lorsque les différentes sources se comportent de façon différente face à des situations similaires. Dans ce chapitre, nous traitons d'un cas particulier de l'apprentissage supervisé dans lequel l'univers des objets, qui contient l'ensemble d'apprentissage, est constitué d'objets provenant de différentes sources et où la valeur prédite par le modèle dépend, dans une certaine mesure, de la source de l'objet.

Notre cas d'étude dans ce chapitre est une situation bien spécifique de l'entame dans le bridge, c'est-à-dire la première carte déposée par un joueur durant la phase du jeu de la carte. La particularité de cette situation tient au fait qu'elle fait suite à une enchère fixée bien connue du monde du bridge et les joueurs, même experts, peuvent avoir des décisions différentes dans cette situation. Le but est de prédire, étant donné une situation et une main, si un joueur va jouer une carte d'une couleur majeure (♠, ♡) ou une couleur mineure (♢, ♣). En effet, certains joueurs peuvent avoir des préférences pour les couleurs majeures, d'autres pour les couleurs mineures. Nous disposons de données annotées par 10 joueurs de bridge différents dont le niveau est élevé, ces joueurs n'ont pas annoté les mêmes observations et le but est d'apprendre un modèle sur l'ensemble de toutes les données annotées, qui prédit la décision d'un joueur face à une nouvelle situation.

Ce problème peut se ramener à un problème d'apprentissage supervisé classique, dans lequel l'identité du joueur est utilisée en tant que caractéristique présente dans l'espace de recherche. Mais nous montrerons qu'une augmentation du nombre de joueurs rend rapidement l'apprentissage très long, ce qui limite les cas d'usages dans lesquels on voudrait apprendre des stratégies sur des centaines de joueurs, ou plus généralement de sources. Nous proposons ainsi dans cette contribution une approche dite *parcimonieuse*, qui n'inclut pas explicitement l'identité du joueur dans l'espace de recherche, mais qui l'utilise tout de même pour l'apprentissage et la prédiction.

Le cas d'étude considéré se situe dans le cadre que nous avons appelé *Apprentissage*

multijoueur supervisé dans l'article [Kaz+24] qui constitue le corps de cette contribution. Ce cadre d'apprentissage peut s'appliquer à n'importe quelle situation multiannotateur telle que celle mentionnée ci-dessus, ce qui explique l'utilisation du terme « multiannotateur » dans le titre de cette contribution. Nous allons toujours utiliser les mots « joueur » pour parler des sources d'annotation, mais il est important de noter que ce terme peut être remplacé par n'importe quelle autre source d'annotation. De même, nous allons utiliser le terme « apprentissage multijoueur » pour parler de l'apprentissage multiannotateur lorsque les sources d'annotation sont des joueurs de bridge.

4.1 Préliminaires

4.1.1 Présentation du problème

Le cas d'étude considéré dans ce chapitre est celui de l'entame au bridge. Les enchères viennent de se terminer, on se place du point de vue de *West*, dans l'équipe du défenseur qui est celui qui doit jouer la première carte de la phase du jeu de la carte. Dans l'expérience, on propose à différents joueurs experts de bridge de jouer l'entame étant donné les informations suivantes :

— **Sa main composée de 13 cartes :** Dans cette situation, les distributions des mains sont fixées : on n'autorise que les nombres de cartes suivants pour chaque couleur respective dans $(\spadesuit, \heartsuit, \diamondsuit, \clubsuit)$: $(3, 3, 4, 3)$ ou $(3, 3, 3, 4)$. Plus précisément, on autorise 3 cartes dans les couleurs majeures $(\spadesuit, \heartsuit)$ et 4 cartes dans une des couleurs mineures $(\diamondsuit, \clubsuit)$. La décision binaire que l'on veut modéliser est *« est-ce que West va jouer une carte d'une couleur majeure ou jouer la carte de la couleur mineure contenant 4 cartes ? »*. On dénomme la couleur dans laquelle *West* a 4 cartes *« couleur mineure quatrième »*. Dans cette expérience, le joueur est tiraillé entre entamer avec sa couleur la plus longue (une mineure) ou entamer dans une majeure, mais dans laquelle il possède moins de cartes.

— **La séquence d'enchères :** La séquence d'enchères utilisée dans notre cas d'études est décrite en annexe A, et renseigne les joueurs sur de nombreuses informations à propos des mains des joueurs adverses. Ces informations sont également décrites dans cette même annexe. Dans notre cas, le dilemme réside dans le fait que grâce aux contraintes d'enchères décrites, West sait que les adversaires devraient avoir moins de majeures que de mineures, or West a bien 4 cartes dans une couleur mineure ce qui devrait le tenter de jouer dans cette couleur. Au bridge, l'entame est d'une importance capitale, car elle définit comment le reste de la partie pourrait se dérouler et est souvent décisive quant au résultat final de la donne. De manière générale, les joueurs essayent de jouer leurs cartes des couleurs les plus longues, car plus on a de cartes dans une couleur, moins les adversaires en ont.

4.1.2 Données

On dispose d'un jeu de données contenant 7263 exemples, étiquetés par 10 joueurs différents. Chaque exemple correspond à un état de jeu et les caractéristiques associées à

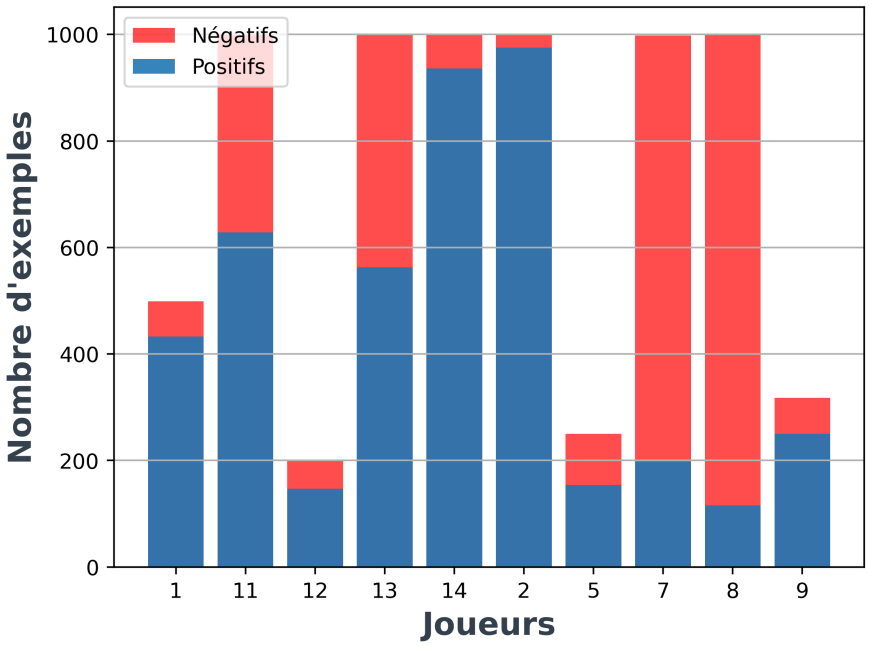


FIGURE 4.1 – Répartition des exemples en fonction des joueurs

chaque exemple sont des descriptions de la main face à laquelle le joueur doit prendre sa décision d'entamer avec une majeure ou une mineure. Les enchères sont implicites et ne sont pas décrites dans les exemples. On définit les exemples positifs comme étant « *West* entame avec une couleur majeure », ou de façon plus simple « majeure » et les exemples négatifs comme étant « West entame avec une carte mineure quatrième » ou « mineure ». Sur la figure 4.1, on peut voir la répartition des exemples en fonction des différents joueurs, avec également la répartition des exemples positifs et négatifs pour chaque joueur. On peut voir que les répartitions entre exemples positifs et négatifs sont très différentes en fonction des joueurs. Les exemples positifs sont très peu représentés pour les joueurs 7 et 8, alors que les exemples négatifs sont quant à eux très peu représentés pour les joueurs 1, 12, 5, 9, 14 et 2. Les joueurs 11 et 13 ont une répartition équilibrée relativement aux autres. On peut donc s'attendre à ce que la décision du joueur soit très différente en fonction de son identité.

Le format des données et leur pré-traitement sont décrits dans l'annexe B.

4.2 Apprentissage multijoueur

Le problème considéré est le suivant. On dispose d'un ensemble d'états S étiquetées par un ensemble de joueurs J. Soit $j \in J$ et $x \in S$. L'objectif est d'apprendre un modèle qui décide pour une paire (j, x) si c est vrai. On se place dans le cadre de l'apprentissage supervisé décrit en section 3.1, et l'apprentissage multijoueur supervisé en est un cas particulier.

Un exemple positif est une paire (j, x) telle que c est vrai, un exemple négatif est une paire (j, x) telle que c est faux. E^+ est l'ensemble des exemples positifs et E^- l'ensemble des exemples négatifs. Chaque état est une interprétation sur un ensemble d'atomes A, et on représente un état x par la conjonction des atomes vrais dans l'interprétation notée $\dot{x}$.

Le but est de trouver une solution H à l'équation 3.1 que l'on rappelle ici :

$$\mathrm{ext}_{\mathrm{E}}(H) = E^+ \tag{4.1}$$

Une solution sera dans ce contexte $H = \{r_1, \ldots, r_n\}$ où chaque règle r_i est de la forme $c \leftarrow 1_C \wedge t$ où C est un sous-ensemble de joueurs et 1_C est vrai pour (j, x) si $j \in C$ et t est une conjonction d'atomes inclus dans A. Pour ce chapitre, on omettra le connecteur $\wedge$ pour les conjonctions d'atomes pour décrire l'état.

Exemple 4.1. *Soit $J = \{j_1, j_2, j_3\}$, $A = \{a, b, d, e\}$. $1_{\{j_1,j_2\}} \wedge bd$ est vrai pour la paire (j_1, bde) car $j_1 \in \{j_1, j_2\}$ et $bd \subseteq bde$.*

4.2.1 Approche directe

L'approche standard pour la résolution de ce type de problème, que l'on nomme *approche directe* (*Dir*), consiste à prendre en compte l'identité du joueur dans l'espace de recherche. La composante 1_C d'une règle sera représentée par une conjonction d'atomes. On représente les $2^{|J|}$ sous-ensembles possibles de joueurs par $|J|$ atomes, dont le codage est le suivant :

Soit $1_{\overline{k}}$ l'atome prenant sa valeur vrai pour (j, x) si $j \neq k$. 1_C est équivalent à la conjonction des atomes $1_{\overline{k}}$ tels que $k \notin C$.

Exemple 4.2. *Soit $J = \{j_1, j_2, j_3\}$. $1_{\{j_1,j_2\}}$ s'écrit $1_{\overline{j_3}}$ et $1_{\{j_2\}}$ s'écrit $1_{\overline{j_1}} \wedge 1_{\overline{j_3}}$.*

4.2.2 Approche parcimonieuse

Dans l'approche que nous proposons, on ne cherchera que les solutions dont les règles r sont *parcimonieuses*, c'est-à-dire telles que C est le sous-ensemble C_t de tous les joueurs j tels que $c \leftarrow t$ est *valide* et *porteuse* pour le joueur j au sens suivant :

Définition 4.1. *Soit E_j le sous-ensemble de E des paires (j, x) du joueur j,*

1. *$c \leftarrow t$ est valide sur E_j si et seulement si*

$$\textit{Pour toute paire } (j, x) \in E_j^-, t \textit{ est faux pour } x \tag{4.2}$$

2. *$c \leftarrow t$ est porteuse sur E_j, si et seulement si*

$$\textit{Il existe une paire } (j, x) \in E_j^+, \textit{ telle que } t \textit{ est vrai pour } x \tag{4.3}$$

3. *$c \leftarrow 1_C \wedge t$ est parcimonieuse si et seulement si*

$$C = C_t \textit{ où } C_t = \{j \in J \mid c \leftarrow t \textit{ valide et porteuse pour } j\} \tag{4.4}$$

Exemple 4.3. *On a 9 paires (joueur, état) concernant 3 joueurs j_1, j_2, j_3 et 4 états $x_1 \dots x_4$. Un état est ici une situation de jeu, et le concept-cible correspond au choix d'entamer d'une couleur majeure (majeure) ou mineure (¬majeure) lors de la phase du jeu de la carte. Chaque état est décrit par les valeurs de vérités prises par les atomes de l'ensemble $A = \{a, b, d, e\}$. La table 4.1 présente E. Chaque paire $p_k = (joueur\ j, état\ x)$ est associée à l'étiquette + (majeure) ou − (¬majeure).*

Joueurs/Etats	$\dot{x}_1 = bde$	$\dot{x}_2 = ade$	$\dot{x}_3 = abe$	$\dot{x}_4 = abd$
j_1	p_1^-	p_2^+	p_3^+	
j_2	p_4^+	p_5^-	p_6^+	
j_3	p_7^+		p_8^+	p_9^-

TABLE 4.1 – Paires (joueur, état) dans E pour l'exemple 4.3. La première ligne montre la représentation des états et la première colonne montre les joueurs. Dans la ligne l et la colonne c on trouve la paire $p_k = (j_l, x_c)$ avec son étiquette + ou −.

On constate ainsi dans la table que les joueurs j_2 et j_3 jouent de la même manière dans l'état x_1 (majeure) alors que j_1 joue différemment d'eux (¬majeure). Considérons les conjonctions t et t' suivantes :

- *$t = ae$ est vrai pour les états x_2 et x_3 et donc pour les paires $\{p_2^+, p_3^+\} \subseteq E_1$, $\{p_5^-, p_6^+\} \subseteq E_2$ et pour la paire $\{p_8^+\} \subseteq E_3$. En conséquence, la règle $r = c \leftarrow ae$ est valide et porteuse pour les joueurs j_1 et j_3, et donc la règle $r = c \leftarrow 1_{C_{ae}}, ae$ est parcimonieuse avec $C_{ae} = \{j_1, j_3\}$ et permet de décider majeure pour les paires $\{p_2^+, p_3^+, p_8^+\}$.*
- *$t' = be$ est vrai pour les états x_1 et x_3 et donc pour les paires $\{p_1^-, p_3^+\} \subseteq E_1$, $\{p_4^+, p_6^+\} \subseteq E_2$ et $\{p_7^+, p_8^+\} \subseteq E_3$ et est donc valide et porteuse pour j_2 et j_3. La règle $r = 1_{\{j_2, j_3\}} \wedge be$ est parcimonieuse et permet de décider majeure pour $\{p_4^+, p_6^+, p_7^+, p_8^+\}$.*

On en conclut que l'hypothèse $H = \{r, r'\}$ décide majeure pour les paires $\{p_2^+, p_3^+, p_4^+, p_6^+, p_7^+, p_8^+\}$, c'est-à-dire E^+ et ne décide majeure pour aucune paire de E^-, elle est donc solution du problème d'apprentissage multijoueur posé.

■

La restriction de la recherche des solutions aux solutions parcimonieuses est justifiée par le résultat suivant :

Proposition 4.1. *Soit H une solution de l'approche directe, et H' l'hypothèse parcimonieuse obtenue en remplaçant chaque règle $r = c \leftarrow 1_C \wedge t$ de H par la règle parcimonieuse $r' = c \leftarrow 1_{C_t} \wedge t$, alors H' est aussi une solution avec $|H'| = |H|$.*

Démonstration. La règle parcimonieuse associée r' est telle que $C_t = V_t \cap P_t$ où V_t est le sous-ensemble de joueurs pour lesquels $c \leftarrow t$ est valide et P_t le sous-ensemble de joueurs pour lesquels $c \leftarrow t$ est porteuse. Comme r appartient à H, elle est valide, donc en particulier $c \leftarrow t$ est valide pour tous les joueurs de C. En conséquence, on a (i) $C \subseteq V_t$. De même si $c \leftarrow t$ n'est pas porteuse pour un joueur j de C alors ce joueur ne participe pas à la couverture de r, c'est-à-dire $\text{ext}_{E_j^+}(r) = \emptyset$. Soit alors $C'' = C \cap P_t$ et $r'' = c \leftarrow 1_{C''} \wedge t$,

on a (ii) $\text{ext}_{E^+}(r) = \text{ext}_{E^+}(r'')$. De (i) et (ii) on déduit que $\text{ext}_E(r) = \text{ext}_E(r'')$ et comme $C'' \subseteq C_t$ on a aussi $\text{ext}_E(r') \supseteq \text{ext}_E(r'')$ et en conséquence $\text{ext}_{E^+}(r') \supseteq \text{ext}_{E^+}(r)$. Comme cela est vrai pour toutes les règles r de H on a donc $\text{ext}_E(h') \supseteq \text{ext}_E(h) = E^+$ et les règles de H' étant valides on a $\text{ext}_E(h') = E^+$. On en conclut que H' est aussi solution. □

Corollaire 4.1. *Soit H une solution de taille minimale alors la solution parcimonieuse associée H' est aussi de taille minimale.*

Pour trouver une solution de taille minimale il suffit donc de parcourir les hypothèses parcimonieuses. L'espace R' des règles parcimonieuses est de taille $|R'| = 2^{|A|}$, exponentiellement plus petit (en nombre de joueurs) que l'espace R de toutes les règles dans l'approche directe et dont la taille est $|R| = 2^{|J|} * 2^{|A|}$.

S'il existe r_i telle que $j \in C_{t_i}$ et t_i vrai pour x, on peut déduire c. La règle de décision est définie dans la définition 4.2 et ne commet pas d'erreur sur E

Définition 4.2 (Règle de décision multijoueur)**.** *Soit une paire (j, x). S'il existe r dans H telle que $j \in C_{t_i}$ et $t \subseteq \dot{x}$, on décide c pour (j, x) sinon on décide $\neg c$ pour (j, x)*

4.3 Un algorithme d'apprentissage multijoueur

L'apprentissage multijoueur étant un cas particulier d'apprentissage supervisé à base de règles, on peut utiliser pour l'approche *Dir* un algorithme de parcours des règles descendant standard en parcourant l'espace des motifs de la forme $1_C \wedge t$ qui décrit également la règle associée $c \leftarrow 1_C \wedge t$. Le même schéma algorithmique peut-être utilisé dans l'approche parcimonieuse en parcourant l'espace des motifs t portant sur les états et donc l'ensemble des règles parcimonieuses $c \leftarrow 1_{C_t} \wedge t$. Nous présentons ci-dessous l'algorithme général avec les notations multijoueur : un objet est une paire (j, x) et une règle est notée (t, C).

4.3.1 Algorithme d'apprentissage glouton

Dans l'algorithme glouton par couverture de l'apprentissage de concept à base de règles, on ramène la recherche d'une meilleure solution $H = \{r_1 \dots r_m\}$ à la recherche itérative d'une meilleure règle qui soit *candidate*, on rappelle qu'une règle est candidate si elle ne couvre aucun exemple négatif. L'algorithme s'arrête lorsqu'on a trouvé une solution $H = \{r_1 \dots r_m\}$ qui ne peut-être améliorée. La couverture $ext_{E^+}(H)$ de la solution partielle courante H augmente à chaque itération, et on peut alors réduire à l'étape suivante l'ensemble des exemples positifs à ceux qui ne sont pas couverts par cette solution partielle. Dans la variante présentée ci-dessous on utilise à chaque itération un exemple positif particulier, non encore couvert, appelé « graine ». On n'explore alors à chaque itération que les motifs couvrant cette graine. Dans cette variante, très utilisée en particulier en Programmation Logique Inductive [Mug95b ; Sri99], l'espace des motifs à explorer est réduit. Pour la recherche d'une meilleure règle candidate nous utilisons une recherche par faisceau de largeur k. L'algorithme général illustre la boucle classique mise en œuvre par les algorithmes de type « diviser pour régner ».

Algorithme 5 *multiPlayerLearning*(E, k, d)

Entrée: Ensemble d'apprentissage $E = \{E^+, E^-\}$, largeur du faisceau k, profondeur maximale de la recherche d
Sortie: H : solution

```
1: X ← E+
2: tant que X ≠ ∅ faire
3:     seed ← head(X)
4:     (t, C) ← bestRule(seed, E−, X, k, d)
5:     si t = faux alors
6:         X ← X \ {seed}
7:     sinon
8:         X ← X \ coverage((t, C), X)
9:         H ← H ∪ (t, C)
10:    fin si
11: fin tant que
```

— X est l'ensemble des exemples positifs non couverts par la solution partielle H et pouvant encore l'être : sont exclus de X les exemples pour lesquels aucune règle valide et porteuse n'a été trouvée lorsqu'ils jouaient le rôle de graine lors d'une itération précédente.

— *bestRule*$(seed, E^-, X, k, d)$ renvoie la meilleure règle (t, C) candidate couvrant la graine *seed* pour l'ensemble X d'exemples positifs courant et l'ensemble d'exemples négatifs E^-. La meilleure règle est la règle candidate (donc valide et porteuse) maximisant le nombre d'exemples positifs de X. (t, C) est ajoutée à la solution partielle courante H. Si la recherche échoue ($t = faux$) la solution partielle n'est pas modifiée et la graine est retirée de X. Cette fonction est détaillée dans l'algorithme 6.

— *coverage*$((t, C), X)$ renvoie l'ensemble des exemples de X couverts par la règle sélectionnée (t, C) et incluent la graine *seed* de l'itération courante. Ils sont retirés de X.

Remarquons que si l'apprentissage est réalisable (s'il existe une solution) la recherche d'une meilleure règle candidate n'échoue jamais et H couvre E^+. Cependant, d'une part, on peut ajouter des contraintes, par exemple un budget b en nombre d'itérations : on aura alors au plus b règles dans H, ou encore un nombre d'atomes maximum $\#a$ dans une règle, ce qui conduit parfois à l'échec de la recherche d'une meilleure règle et à renvoyer une solution partielle ne couvrant pas tout E^+. D'autre part, même sans contraintes supplémentaires, l'apprentissage peut ne pas être réalisable, par exemple si pour un même joueur j et une même situation x on trouve deux exemples avec les étiquettes c et $\neg c$. L'algorithme décrit ci-dessus renvoie éventuellement une solution partielle, valide, mais ne couvrant qu'une partie de E^+.

Dans le cas général *bestRule* est un algorithme descendant qui parcourt un espace de motifs à partir de $\top = vrai$, le spécialise en ajoutant des littéraux issus de la graine, puis renvoie la meilleure règle candidate trouvée, au sens d'une fonction s. Pour des raisons

de complexité, la recherche n'explore pas tout l'espace des motifs, c'est une recherche en *faisceau* (voir figure 3.3) de *largeur* k qui utilise une fonction de *guidage* g.

Algorithme 6 *bestRule*$(seed, E^-, X, k, d)$

Entrée: Graine *seed*, exemples négatifs E^-, Ensemble des exemples positifs non couverts encore X, largeur du faisceau k, fonction de score s, fonction de guidage g, profondeur maximale de la recherche d

Sortie: r : meilleure règle

1: $currentList \leftarrow \{vrai\}$
2: $r \leftarrow r_\perp$
3: $size \leftarrow 0$
4: **tant que** $currentList \neq \emptyset$ et $size < d$ **faire**
5: $\quad currentList \leftarrow \cup_{l \in currentList} minimalSpecializations(l, seed, d)$
6: $\quad Cands \leftarrow candidates(currentList, E^-, X)$
7: $\quad r \leftarrow bestCandidate(s, r, Cands)$
8: $\quad currentList \leftarrow currentList \setminus Cands$
9: $\quad currentList \leftarrow kMostPromisingTerms(k, g, currentList)$
10: $\quad size \leftarrow size + 1$
11: **fin tant que**

À une itération les éléments de la liste courante *currentList* sont spécialisés pour former une nouvelle liste *currentList*, on retire de L l'ensemble *Cands* des motifs associés à des règles candidates, et si la meilleure règle issue de *Cands*, au sens de la fonction s, est meilleure que la meilleure règle candidate courante r elle la remplace.

— *minimalSpecializations*$(l, seed, d)$ renvoie l'ensemble des motifs spécialisant l de taille $|l| + 1$ et couvrant la graine *seed*.

— *candidates*(L, E^-, X) renvoie la liste *Cands* des motifs de *currentList* dont la règle associée (C, t) est candidate.

— *bestCandidate*$(s, r, Cands)$ renvoie la meilleure règle candidate de l'ensemble $\{Cands \cup r\}$ au sens de le fonction de score s.

— *kMostPromisingTerms*(k, g, L) renvoie les k meilleures règles de L au sens de la fonction de guidage g.

g renvoie alors un score mesurant l'intérêt de spécialiser un motif et seules les k meilleurs motifs de *currentList* sont gardées pour l'itération suivante. Les fonctions g et s sont détaillées dans la section suivante.

4.3.2 Approche parcimonieuse versus approche directe

Les deux approches se différencient d'abord par l'espace de recherche exploré, plus petit dans l'approche parcimonieuse et par la fonction de guidage g et la fonction d'évaluation s à maximiser.

Dans le cas direct, un motif est constitué d'atomes portant sur le joueur, formant la composante 1_C et d'atomes portant sur l'état formant la composante t.

Dans le cas parcimonieux, un élément de L est constitué uniquement d'atomes portant sur l'état x. $candidates(L)$ renvoie l'ensemble des motifs t associés à des règles parcimonieuses (t, C_t), où C_t est l'ensemble des joueurs pour lesquels $c \leftarrow t$ est valide et porteuse, et $bestCandidate(s, r, Cands)$ met à jour la meilleure règle parcimonieuse courante.

Dans les deux cas, la fonction d'évaluation s des solutions candidates est la couverture : la meilleure règle candidate est celle couvrant le plus d'exemples de X (comme elle est candidate on a aussi $ext_{E^-}(r) = \emptyset$) :

$$s(r) = |ext_X(r)| \tag{4.5}$$

Dans l'approche directe, nous n'utilisons pas explicitement l'information sur le joueur dans la fonction de guidage g. Celle-ci est une adaptation de la précision $P/(P+N)$ telle qu'à précision égale, la règle ayant la plus petite couverture sur les négatifs sera préférée :

$$g(r) = \frac{P - N}{P + N} \tag{4.6}$$

où $P = |ext_X(r)|$ et $N = |ext_{E^-}(r)|$

Dans l'approche parcimonieuse nous avons utilisé pour fonction de guidage la séparation de E en sous-ensembles $\{E_1, \dots E_{|J|}\}$ de même joueur :.

$$g_P(r) = Max_{j \in J}\ g_j(t) \tag{4.7}$$

où $P_j = |ext_{X_j}(c \leftarrow t)|$, $N_j = |ext_{E_j^-}(c \leftarrow t)|$ et $g_j(t) = \dfrac{P_j - N_j}{P_j + N_j}$,

Remarquons que g_P n'utilise que t, sous la forme d'une règle $c \leftarrow t$, et considère tous les joueurs de J. En effet, l'intérêt de la règle $r = (t, C_t)$ se mesure aussi en fonction de joueurs pour lesquels la règle n'est pas encore valide, mais peut le devenir au cours d'une spécialisation. En revanche, concernant l'évaluation des règles candidate (t, C_t), on utilise bien la règle $c \leftarrow 1_{C_t} \wedge t$ comme dans l'approche directe.

4.4 Apprentissage multijoueur multiclasse

Dans l'apprentissage multiclasse, on a un ensemble de classes possibles, exclusives, pour chaque observation x et un exemple étiqueté s'écrit donc (x, c). Dans le cas multijoueur, un exemple étiqueté s'écrira donc $((j, x), c)$ ou plus simplement (j, x, c), où c est la classe de l'état x pour le joueur c. Parmi les manières d'aborder ce cas avec des modèles d'apprentissage à base de règles, la plus simple consiste à construire un modèle pour chaque classe c_k en l'opposant aux autres, c'est-à-dire en prenant $E^+ = E^{c_k}$ et $E^- = \bigcup_{m \neq k} E^{c_m}$. Il faut alors adapter la règle de décision. Le problème abordé dans les expériences concerne une seule classe c, donc opposée à $\neg c$ (voir Exemple 4.3) et il est apparu intéressant de considérer ce problème comme un problème à deux classes $c_1 = c, c_2 = \neg c$ et de construire donc deux hypothèses h_1 et h_2 où h_1 est solution du problème d'apprentissage multijoueur initial et h_2 est solution du problème multijoueur opposé où $E^+ = E \setminus E^c$ et $E^- = E^c$. Par souci de simplicité, nous notons par abus de

langage $h, (j, x) \models c$ le fait qu'il existe une règle dans h qui couvre la paire (j, x). La règle de décision envisage alors 4 situations.

Définition 4.3 (Règle de décision multijoueur multiclasse)**.** *Soit une paire* (j, x)

1. *Si* $h_1, (j, x) \models c_1$ *et* $h_2, (j, x) \not\models c_2$ *décider* c_1 *pour* (j, x)
2. *Si* $h_1, (j, x) \not\models c_1$ *et* $h_2, (j, x) \models c_2$ *décider* c_2 *pour* (j, x)
3. *Si* $h_1, (j, x) \models c_1$ *et* $h_2, (j, x) \models c_2$ *décider* $maxCov(h_1, h_2, (j, x))$
4. *Si* $h_1, (j, x) \not\models c_1$ *et* $h_2, (j, x) \not\models c_2$ *décider* $majoritaire(h_1, h_2, (j, x))$

La fonction *majoritaire* décide la classe majoritaire entre c_1 et c_2 parmi les exemples d'entraînement du joueur j. Pour la fonction *maxCov*, on a un ensemble $R_{(j,x)}$ de règles de la forme $c_1 \leftarrow t_1$ et $c_2 \leftarrow t_2$ qui couvrent la paire (j, x). Dans ce cas, l'algorithme décide de classer (j, x) dans la classe de la règle de $R_{(j,x)}$ qui couvre le plus d'exemples d'apprentissage.

4.5 Expérimentations

La figure 4.2, indique la mesure d'accord entre les différents joueurs sur la tâche de décision. L'histogramme mesure le pourcentage d'exemples par rapport à la taille de la plus grande coalition, où une coalition est un nombre de joueurs qui prennent la même décision face à un exemple.

Pour effectuer cette mesure, en raison du fait que les joueurs qui annotent les données n'ont pas annoté les mêmes données, nous avons entraîné un modèle *parcimonieux* (voir section 4.2.2), qui apprend une solution composée de règles de décision spécifiques à des sous-ensembles de joueurs. Ainsi, pour chaque exemple (j, x, c_x^j) dans l'ensemble de test, on construit pour l'état x les paires (j', x) pour chaque joueur j' différent de j, et à chaque paire (j', x) on associe $c_x^{j'}$ l'étiquette prédite par le modèle. On obtient donc les exemples étiquetés $(1, x, c_x^1), (2, x, c_x^2), \ldots, (j-1, x, c_x^{j-1}), (j, x, c_x^j), \ldots, (10, x, c_x^{10})$ qui sont ensuite présentés au modèle dont on récolte les décisions pour chaque joueur. L'histogramme mesure donc un degré d'accord relatif à une solution h, et non un degré d'accord absolu entre les joueurs. Le modèle est entraîné sur 1016 exemples, et on mesure le désaccord sur 2904 exemples de test.

On remarque un très faible nombre d'exemples (0.4%) sur lesquels tous les joueurs en accord. On a une majorité (92.5%) des exemples sur lesquels on a entre 5 et 8 joueurs qui sont en accord. Cette mesure indique que les joueurs ont des comportements très différents en fonction des exemples et justifie l'apprentissage d'un modèle multijoueur.

4.5.1 Comparaison des approches en fonction du nombre d'exemples d'apprentissage

Nous considérons pour des ensembles d'apprentissage de taille $|E| = n$ croissante, avec n allant de 1 à 45% des exemples étiquetés, la justesse des prédictions de plusieurs modèles multiclasse $c_1 =$*majeure*, $c_2 =$ *mineure*. Chaque justesse est calculée sur 20 essais, chacun associé à une permutation différente de l'ensemble de tous les exemples. On prend

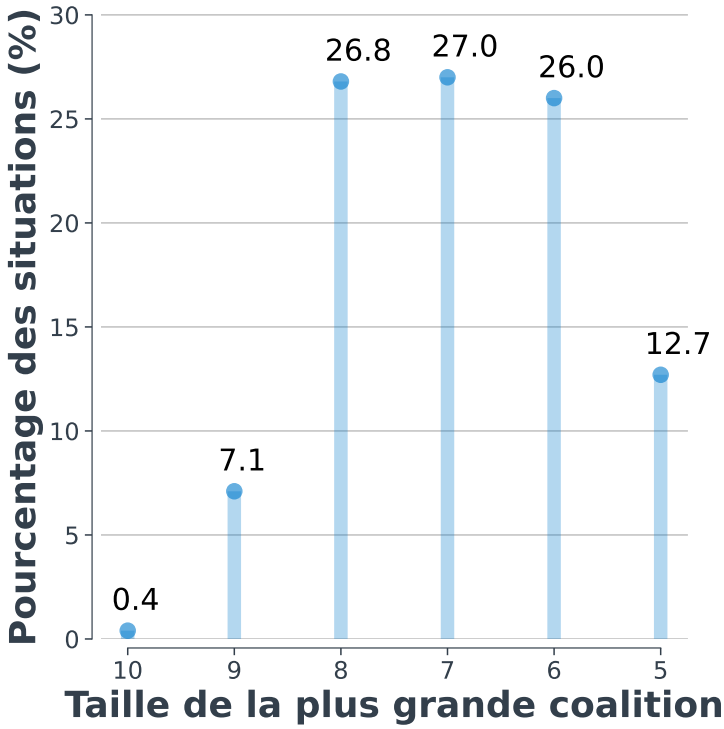


FIGURE 4.2 – Accord entre les joueurs. En abscisse on a le nombre de joueurs dont les exemples ont la même étiquette pour le même état.

pour chaque essai les n premiers éléments de la permutation pour former E et les k derniers pour former l'ensemble test sur lequel la justesse est évaluée. On utilise le même algorithme présenté ci-dessus selon plusieurs approches :

- **Approche directe** *Dir*
- **Approche parcimonieuse** *Par*
- **Approche directe individuelle** *Ind* où le contexte ne peut être que de la forme $C = \{j\}$ ou $C = J$, c'est-à-dire qu'on apprend des règles propres à chaque joueur individuellement et des règles indépendantes du joueur.
- **Approche ignorante** *Ign* où on décrit la paire (j, x) uniquement selon l'état x

Enfin, nous ajoutons le résultat de deux méthodes de référence *Random forest* et *Decision tree*, dans une version utilisant l'identité du joueur. La figure 4.3 représente les justesses obtenues avec les différentes approches.

On observe que l'approche *Ign* a une justesse n'excédant pas 0.63 lorsque le pourcentage des exemples de E sur lesquels le modèle est construit augmente, montrant le désaccord entre joueurs représenté Figure 4.2 et donc la pertinence de prendre en compte l'identité du joueur. On constate aussi que l'approche *Dir* a une mauvaise performance par rapport aux approches *Ind* et *Par*, alors qu'elle utilise pleinement l'identité du joueur. On analyse les approches pour 45% des exemples (le dernier point sur les abscisses de la figure 4.3), nous n'allons pas plus loin en raison de l'apparition d'un plateau de performance. L'approche *Ind*, dont les règles sont bien moins expressives concernant l'identité

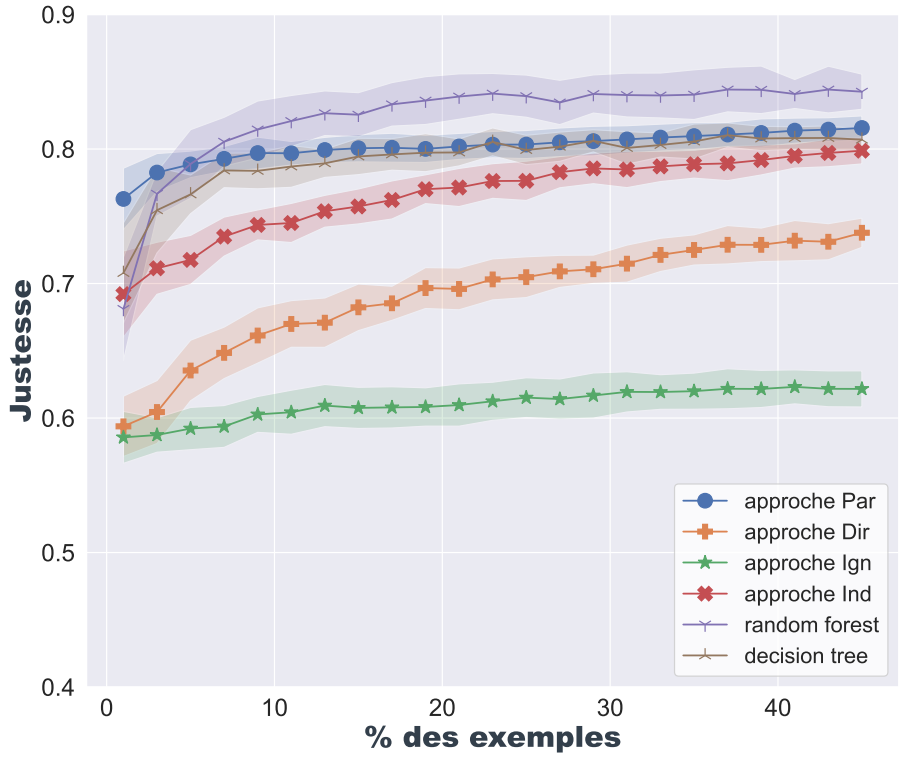


FIGURE 4.3 – Justesses des différentes approches en fonction du nombre d'exemples d'apprentissage

du joueur que celles l'approche *Dir*, a une bien meilleure justesse (≈ 0.80). Sa justesse est cependant moins bonne que celle de l'approche *Par* (≈ 0.815). La forêt aléatoire obtient un résultat supérieur (≈ 0.84). Enfin, l'arbre de décision obtient une performance légèrement en dessous (≈ 0.805) de celle de l'approche *Par*. On remarque :

1. Le bon résultat de la forêt aléatoire dans sa version informée doit être relié à sa nature ensembliste : la forêt est constituée d'une centaine d'arbres, rendant l'interprétation du modèle difficile.
2. Le résultat de l'arbre de décision est très proche de celui de l'approche *Par*, mais ne le dépasse pas. On remarque en revanche qu'avec un faible nombre d'exemples d'apprentissage, l'approche *Par* est plus performante que l'arbre de décision.
3. Le mauvais résultat de l'approche *Dir*, aussi expressive que l'approche *Par*, est lié à la très grande taille de l'espace de règles, la recherche d'une meilleure règle allant alors vers de mauvais optimums locaux. Dans l'approche *Ind* la contrainte sur le joueur ne peut être que nulle ou l'identité exacte du joueur, et l'algorithme de recherche d'une meilleure règle explore un espace plus petit et se dirige mieux.

Complexité syntaxique des solutions La Figure 4.4 montre la complexité mesurée des solutions produites par les différentes approches à base de règles en fonction du nombre d'exemples pour 10 joueurs, de 5% des exemples ($n = 363$) à 45% des exemples ($n = 3268$). La complexité des solutions des différentes approches est mesurée en nombre de règles du modèle appris, on précise que dans ce cas, on considère qu'un exemple non-couvert par la solution est une règle.

Dans le cas des forêts aléatoires, sa complexité est mesurée en nombre de feuilles moyen par arbre de la forêt (équivalent à un nombre de règles). La forêt étant constituée de 100 arbres, le modèle est cependant bien plus complexe, avec en moyenne 1848 feuilles par arbre. Pour l'arbre de décision, le nombre de feuilles est égal au nombre de règles des solutions *Par* (si l'on ne considère pas les exemples non couverts comme des règles), mais la taille moyenne des règles des solutions *Par* est bien inférieure à la taille moyenne des branches de l'arbre de décision. Pour 11% des exemples en apprentissage, la taille moyenne des règles *Par* est de 4.2 atomes tandis que la taille moyenne des chemins dans l'arbre est de 13.4 tests.

On remarque que les approches *Dir* et *Par* fournissent des solutions plus simples que l'approche *Ind*, avec une pente de la courbe de la complexité supérieure pour l'approche *Ind* que les deux autres approches. Cela vient du fait que les règles de la solution *Ind* ne peuvent concerner qu'un seul joueur, faisant ainsi augmenter le nombre de règles redondantes dont la seule différence est le joueur associé. Cette analyse est confirmée par la figure 4.5, qui montre en bleu le nombre de règles de la solution *Ind* qui sont incluses dans une autre règle ou incluent une autre règle de la même solution et dont le joueur associé est différent, et en rouge le nombre de groupes de ces mêmes règles, appelés *factorisations*. Pour construire ces factorisations, on considère deux règles r_1 et r_2 respectivement de la forme $c \leftarrow 1_j \wedge t$ et $c \leftarrow 1_{j'} \wedge t'$ incluses dans une solution obtenue à partir d'une approche *Ind*, et telles que $t \subseteq t'$. Alors r_1 et r_2 appartiennent à la même factorisation si $c \leftarrow 1_j \wedge t'$ est valide, c'est-à-dire qu'elle ne couvre aucun négatif. On peut considérer que les règles de la même factorisation compteraient comme une seule règle dans l'approche *Par*.

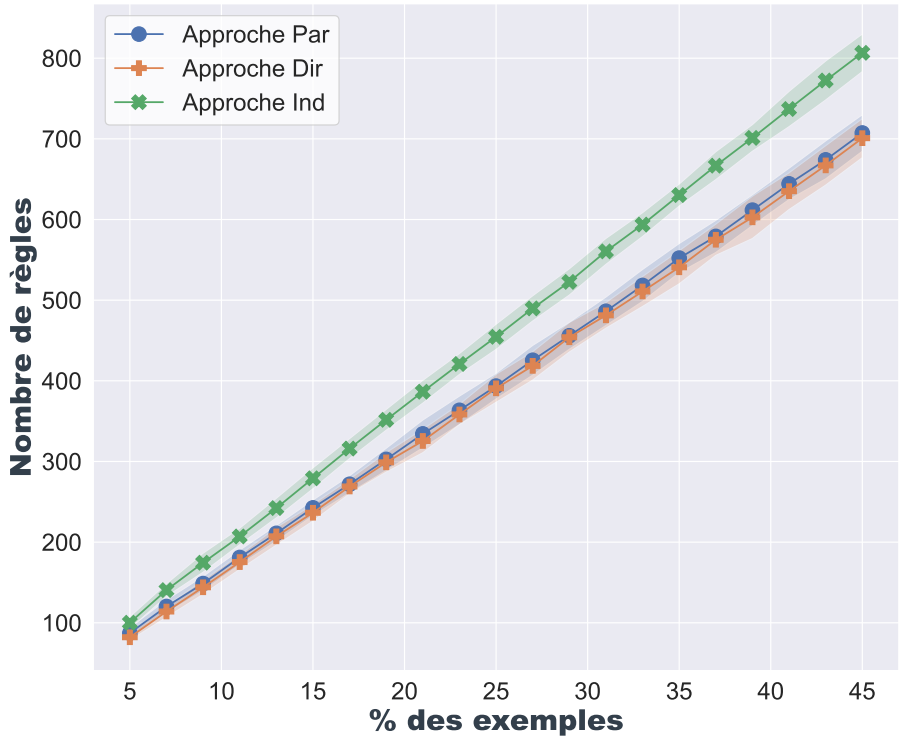


FIGURE 4.4 – Nombre de règles dans les solutions fournies par les différentes approches en fonction du nombre d'exemples d'apprentissage.

Dans la Figure 4.5, on a en vert le nombre de règles dans la solution *Ind*. On remarque ainsi qu'en moyenne, la moitié des règles des solutions issues de l'approche *Ind* appartiennent à au moins une factorisation. En moyenne, on constate que chaque factorisation contient 2.17 règles de la solution *Ind*. Il est à noter également qu'une même règle peut appartenir à plusieurs factorisations. On remarque que la part des règles appartenant à au moins une factorisation augmente avec le nombre d'exemples, ce qui confirme l'analyse de la complexité syntaxique des solutions.

4.5.2 Comparaison des approches en fonction du nombre de joueurs

Une des motivations de l'apprentissage multijoueur parcimonieux est le traitement de problèmes dans lesquels on a un grand nombre de joueurs. Pour étudier le comportement des approches *Dir* et *Par* en fonction du nombre de joueurs, nous avons artificiellement altéré le jeu de données d'origine de la manière suivante : nous définissons pour chaque joueur j du jeu de données original un nouvel ensemble de k joueurs et chaque exemple de E_j, incluant son étiquette, est attribué à tous ces nouveaux joueurs. Ainsi, on associe à chaque joueur original k joueurs artificiels se comportant de la même manière que le joueur original. Par exemple, supposons que l'ensemble d'apprentissage soit un singleton $E = \{e_1\}$ avec $e_1 = (j_1, x)$. Pour $k = 2$ on a ainsi un nouveau jeu de données $E' = \{e_{1,1}, e_{1,2}\}$ avec $e_{1,1} = (j_{1,1}, x)$ et $e_{1,2} = (j_{1,2}, x)$. On a donc un jeu de données de taille k

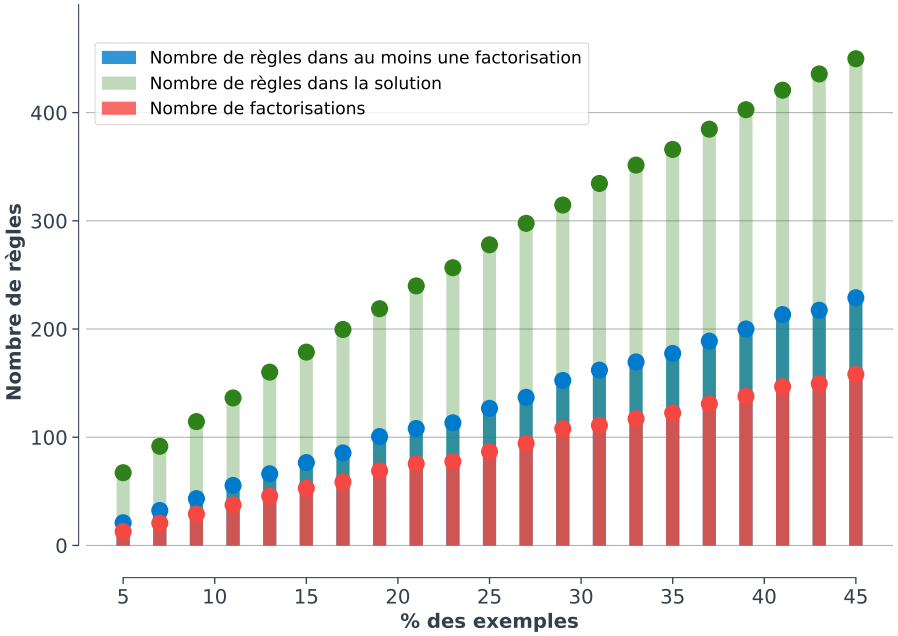


FIGURE 4.5 – Part des règles redondantes des solutions *Ind*, en fonction du nombre d'exemples.

fois plus grand que le jeu de données original.

Justesse et temps CPU. Pour un ensemble d'apprentissage original de taille $n = 1380$ (19%) nous avons comparé les approches *Dir* et *Par* et montré les résultats dans la table 4.2. Ce nombre d'exemples a été choisi pour que le temps CPU de l'approche *Dir* reste raisonnable.

# de joueurs	Justesse Par	CPU Par (min)	Justesse Dir	CPU Dir (min)	n
10	0.80	15	0.68	125	1380
20	0.795	46	0.646	659	2760
30	0.79	116	0.652	2707	4140
40	0.799	236	0.667	8836	5520

TABLE 4.2 – Justesses moyennes et temps CPU des approches *Dir* et *Par* pour un nombre croissant de joueurs et pour 19% ($n = 1380$) exemples originaux

L'approche *Par* montre une décroissance négligeable de la justesse en fonction du nombre de joueurs et pour un nombre croissant d'exemples d'apprentissage. C'est ce à quoi on peut s'attendre en raison du fait que les exemples sont doublons, et qu'une règle valide et porteuse pour un joueur artificiel j' issu d'un joueur original j le sera également pour tous les joueurs artificiels issus de j. On remarque en revanche une baisse plus substantielle de la justesse de l'approche *Dir*, en particulier lorsqu'on introduit uniquement un joueur artificiel (ligne 2 de la table 4.2), avec une baisse de 3.4 points en pourcentage.

Concernant le temps CPU, comme attendu, on remarque une augmentation du temps d'apprentissage pour l'approche *Dir* supérieure à celle de l'approche *Par* en raison de

l'augmentation de l'espace de recherche, contrairement à l'approche *Par* où le nombre de joueurs dans le jeu de données n'influe pas sur la taille de l'espace de recherche.

Complexité syntaxique. La figure 4.6 représente les complexités syntaxiques en nombre de règles des solutions obtenues avec les approches *Par* et *Ind* en fonction du nombre de joueurs, pour 19% des exemples ($n = 1380$). Pour l'approche *Ind*, on remarque une augmentation linéaire du nombre de règles dans la solution en fonction du nombre de joueurs. Cela est dû au fait que chaque règle d'une solution de l'approche *Ind* ne peut concerner qu'un joueur unique ou bien tous les joueurs. Ainsi, chaque règle est dupliquée du nombre de joueurs artificiels créés rendant l'approche linéaire en fonction du nombre de joueurs.

L'approche *Par* montre son efficacité en termes de complexité des solutions ici, avec une baisse de la pente en fonction du nombre de joueurs. Chaque règle d'une solution de l'approche *Par* pouvant être valide et porteuse pour un sous-ensemble de joueurs, on s'attend effectivement à une complexité moindre de cette approche par rapport à l'approche *Ind* en augmentant le nombre de joueurs.

Cette analyse est confirmée par la figure 4.7, qui mesure le nombre de règles appartenant à au moins une factorisation sur le nombre total de règles dans la solution en fonction du nombre de joueurs. Cette analyse a été effectuée, cette fois, sans dupliquer les exemples d'apprentissage pour une comparaison équitable. On remarque alors qu'en augmentant le nombre de joueurs, le nombre de règles appartenant à au moins une factorisation s'approche des 100% des règles de la solution pour l'approche *Ind*. Cette analyse montre alors l'utilité de l'approche *Par* pour des jeux de données avec un grand nombre de joueurs.

4.5.3 Utilisation des résultats de l'apprentissage multijoueur pour l'analyse des groupes de joueurs

Nous avons utilisé les résultats de l'approche *Par* afin de construire des groupes de joueurs se comportant de la même manière. Pour cela, nous avons utilisé l'algorithme de clustering K-Means sur les matrices joueur-règle obtenues par l'approche *Par*. Pour construire cette matrice, nous avons récupéré l'ensemble des règles de la solution obtenue par l'approche *Par* et pour chaque joueur, nous avons construit un vecteur binaire indiquant si la règle est valide et porteuse pour le joueur ou non. Nous avons ensuite appliqué un K-Means sur la matrice obtenue.

Dans la figure 4.8, nous avons représenté par ce dendrogramme le regroupement hiérarchique des joueurs en fonction du nombre de clusters choisi à l'exécution de l'algorithme K-Means. À chaque étape, les deux clusters « les plus proches » sont fusionnés. Nous avons fait varier les nombres de clusters de 2 à 9. Nous pouvons alors analyser les groupes de joueurs obtenus pour affirmer la ressemblance des joueurs entre eux. Afin d'étudier l'effet du clustering, pour chaque nombre de clusters analysé, nous avons effectué un nouvel apprentissage avec les données où l'identité des joueurs a été remplacée par leur cluster, pour chaque nombre de clusters. Cette approche est nommée *ParClust*.

Afin de comparer l'approche *ParClust* avec une approche « témoin », nous avons réalisé

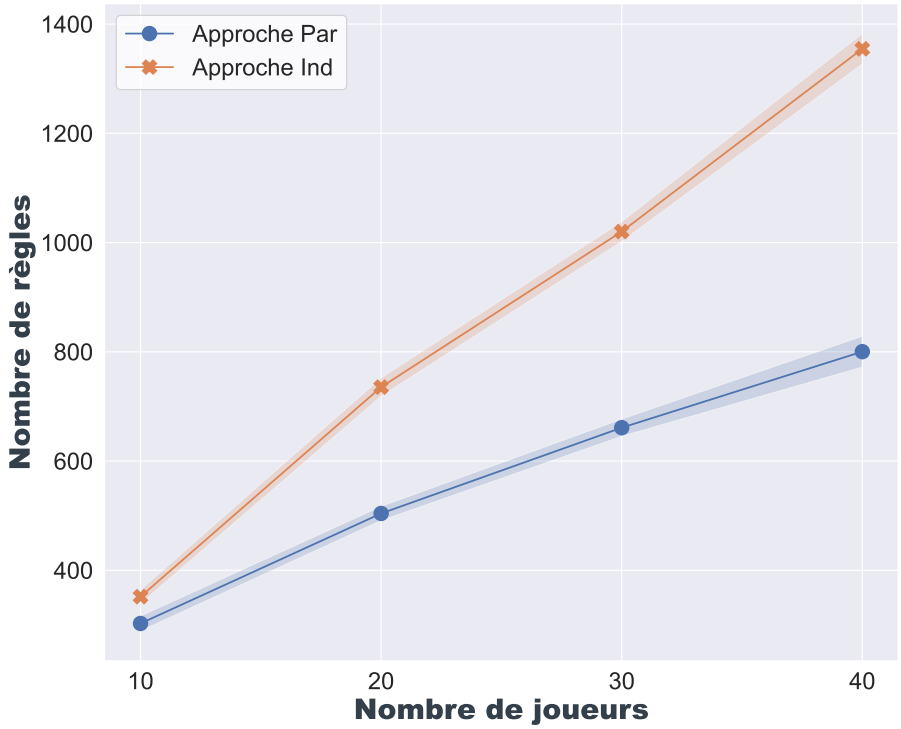


FIGURE 4.6 – Nombre de règles des solutions fournies par les approches *Par* et *Ind* en fonction du nombre de joueurs, pour 19% des exemples ($n = 1380$).

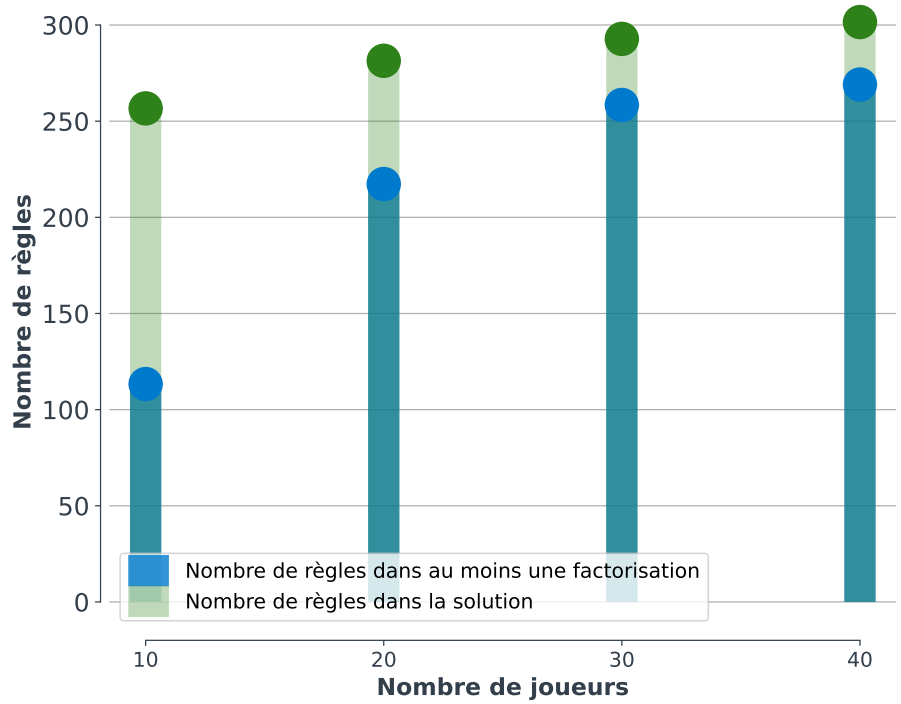


FIGURE 4.7 – Part des règles redondantes des solutions *Ind* en fonction du nombre de joueurs, pour 23% des exemples ($n = 1670$).

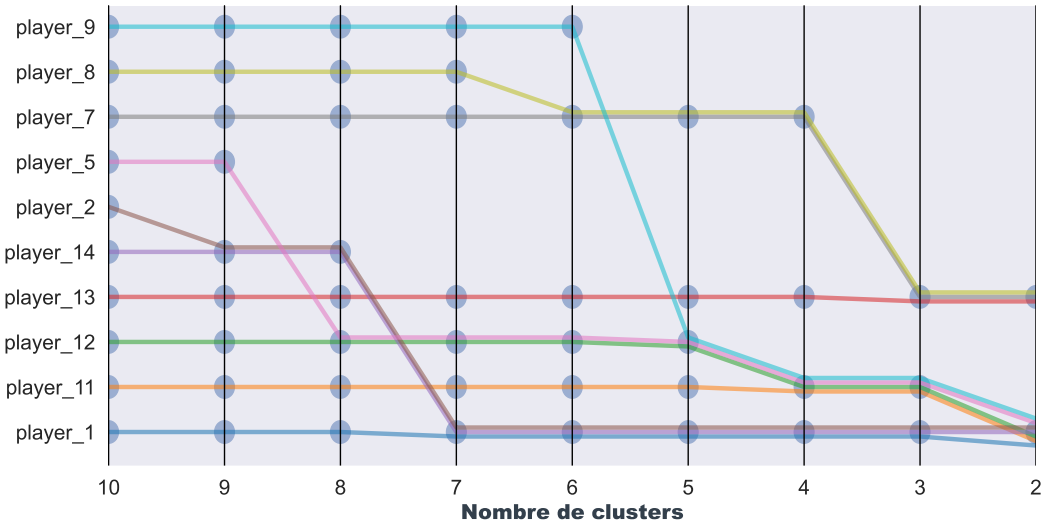


FIGURE 4.8 – Dendrogramme du regroupement hiérarchique des joueurs.

un K-Means sur les données initiales, sans apprentissage multijoueur. Pour cela, nous avons représenté chaque joueur par un vecteur de caractéristiques correspondant à la concaténation du vecteur de la distribution sur les caractéristiques des exemples qu'il a étiquetés positivement et du vecteur de la distribution sur les caractéristiques des exemples qu'il a étiquetés négativement. Nous avons ensuite appliqué un K-Means sur la matrice obtenue. La figure 4.9 montre le dendrogramme du regroupement hiérarchique des joueurs obtenu par l'approche témoin.

La figure 4.10 montre la justesse des modèles appris avec remplacement de l'identité des joueurs par leur cluster dans les données d'apprentissage avec 45% des exemples. La ligne rouge en pointillés indique la justesse de l'approche *Par* avec 10 clusters correspondant aux joueurs originaux. On remarque que la justesse de l'approche *ParClust* est la même que l'approche *Par* pour un nombre de clusters supérieur ou égal à 4. On peut alors considérer que si l'on souhaite réduire le nombre de joueurs ou établir des clusters de joueurs se comportant de la même manière sans perte de justesse, alors le nombre de clusters donnant un regroupement optimal des joueurs est 4.

On remarque également que la justesse de l'approche *ParClust* est supérieure à celle de l'approche témoin pour un nombre de clusters inférieur à 8. Cela indique que le clustering à partir des résultats de l'apprentissage multijoueur est plus pertinent que le clustering à partir des données initiales.

4.6 Travaux associés

Certains travaux ont abordé le problème de l'annotation par différents experts lorsqu'il y a un problème de désaccord entre les étiquettes attribuées par chacun des experts. Dans [Wan+23], les auteurs présentent un cadre d'apprentissage où le désaccord entre les experts est modélisé et utilisé de manière à régulariser le classifieur, dans [Yan+14] les

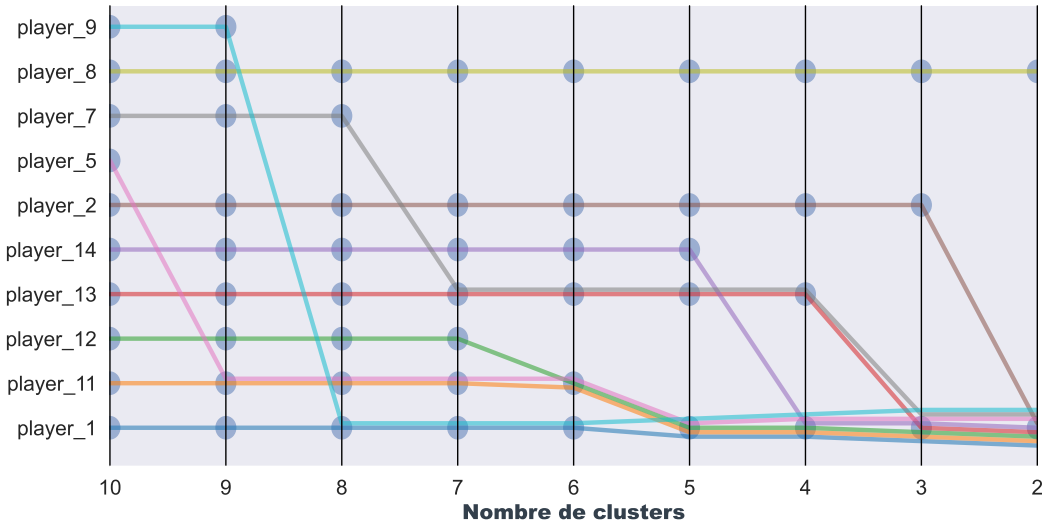


FIGURE 4.9 – Dendrogramme du regroupement hiérarchique témoin des joueurs.

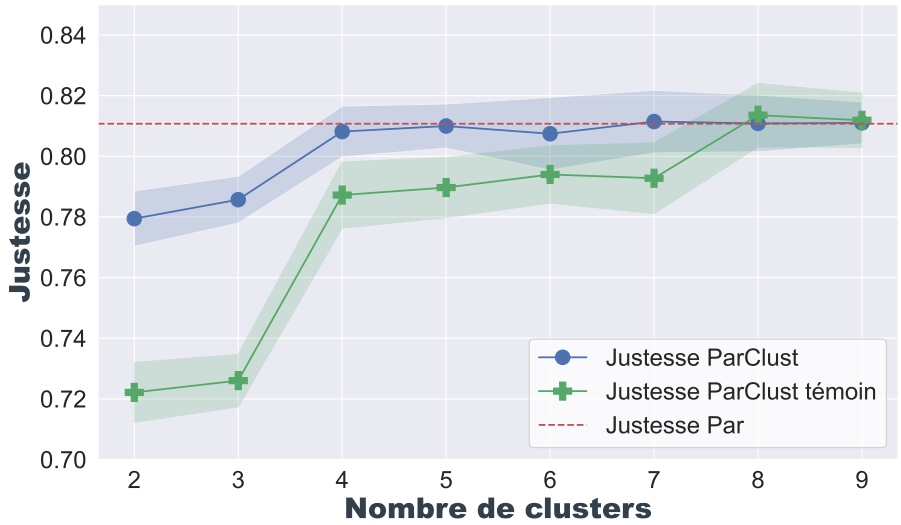


FIGURE 4.10 – Justesse des modèles appris avec remplacement de l'identité des joueurs par leur cluster dans les données pour 45% des exemples ($n = 3268$).

auteurs présente une mesure de l'expertise des différents annotateurs de manière à utiliser cette information pour déterminer une étiquette à partir d'un consensus entre les annotateurs pour chaque observation. Les auteurs de [SL13] présentent une étude comparative des différentes méthodes de calcul d'un consensus sur l'annotation des données par différents experts. Mais ces méthodes s'appuient sur le fait que les annotateurs étiquettent les mêmes données et construisent un modèle général où l'identité de l'annotateur n'est pas prise en compte dans la décision face à une nouvelle observation. À notre connaissance, aucune étude sur l'utilisation implicite de l'identité de l'annotateur dans le cadre simple de l'apprentissage supervisé de concept n'a été effectuée. Enfin, l'idée d'utiliser séparément l'identité de l'annotateur et l'objet annoté était déjà présente dans les travaux sur l'apprentissage abstrait [Sol11].

4.7 Conclusion

Dans ce chapitre, nous nous sommes intéressés au cadre de l'apprentissage supervisé de concept, dans lequel les données peuvent être étiquetées de différentes manières par différents annotateurs. Nous considérons le cas où les annotateurs n'annotent pas forcément les mêmes données, ce qui rend impossible de calculer un consensus sur l'étiquette d'une donnée. Notre but a été alors d'entraîner un modèle capable de prédire la décision d'un annotateur face à un nouvel exemple non vu pendant l'entraînement. Entraîner un modèle performant sur ces données nécessite de prendre en compte la spécificité de chaque annotateur dans la construction du modèle et d'utiliser cette information pour prédire sur de nouvelles données.

Pour ce faire, nous avons adapté le cadre de l'apprentissage supervisé de concept pour prendre en compte la spécificité des annotateurs, en considérant qu'une règle de décision est valable pour un sous-ensemble des annotateurs. Ainsi, face à un nouvel exemple associé à un annotateur, le modèle prédit la décision de l'annotateur en utilisant uniquement les règles qui lui sont associées. Nous avons nommé cette adaptation du cadre classique l'*apprentissage multiannotateur supervisé*.

L'approche classique pour résoudre ce problème est d'inclure l'identité de l'annotateur dans l'espace de recherche des hypothèses, mais cette approche augmente la complexité du modèle et le temps d'apprentissage lorsque le nombre de joueurs est élevé. Pour résoudre ce problème, nous avons proposé un algorithme d'apprentissage multiannotateur par approche parcimonieuse, qui prend en compte l'identité de l'annotateur dans la fonction de guidage lors de la recherche d'hypothèses sans qu'elle soit incluse dans l'espace de recherche. L'approche parcimonieuse construit des règles qui sont valides et porteuses pour un sous-ensemble des annotateurs et cette information est utilisée par la suite dans la prédiction sur de nouvelles données.

Pour illustrer ce cadre, nous avons pris comme cas d'étude un problème d'ouverture au bridge, où nous devions produire un modèle qui prédit l'entame, qui est la première carte qu'un joueur va jouer dans la phase du jeu de la carte. Dans le bridge, les joueurs ne prennent pas toujours les mêmes décisions face à une même situation et certains joueurs peuvent avoir des stratégies globales complètement différentes. La situation à laquelle nous nous sommes intéressés fait suite à une suite d'enchère bien spécifique, face à laquelle il y

a une grande disparité dans le choix de la carte à jouer, rendant ainsi la construction d'un modèle commun à tous les joueurs délicate, car il est nécessaire de prendre en compte l'identité du joueur dans la décision.

Nous avons montré que l'approche parcimonieuse, produit des solutions moins complexes en nombre de règles et plus performantes dans la prédiction que les approches comparatives, excepté la forêt aléatoire, qui est une méthode par ensemble de modèles. En revanche, bien que la performance de la forêt aléatoire soit supérieure à celle d'un arbre de décision seul, elle sacrifie son interprétabilité présente dans les arbres de décision. Notre méthode est par ailleurs plus performante qu'un arbre de décision unique.

Nous avons montré également l'intérêt principal de l'approche parcimonieuse, qui est d'être peu sensible à l'augmentation du nombre d'annotateurs, que ce soit en termes de justesse, de temps d'apprentissage, mais également de complexité syntaxique.

Nous avons enfin évalué la pertinence de l'association des sous-ensembles de joueurs aux règles construites par approche parcimonieuse, en construisant un clustering sur l'ensemble des joueurs représentés par les règles qui les concernent, puis en remplaçant l'identité des joueurs par le cluster auquel ils appartiennent. Cette méthode nous a permis d'expliciter le nombre minimal de clusters sans perte de justesse du modèle, et nous avons montré que les règles de la solution parcimonieuse permettait de réduire sensiblement le nombre de joueurs sans perte d'information, indiquant que les groupes de joueurs formés se comportent réellement de la même façon.

Chapitre 5

Explications : état de l'art

Introduction

En avril 2016, l'adoption du règlement général sur la protection des données (RGPD) par l'Union Européenne crée le « droit à l'explication » des décisions prises par les algorithmes. Ce droit implique de donner la possibilité à un citoyen Européen, faisant face à une décision prise par un algorithme d'apprentissage automatique, d'exiger la logique sous-jacente à la décision prise par l'algorithme [1]. Cette loi montre une volonté réelle des régulateurs de protéger les utilisateurs de décisions automatiques qui pourraient être biaisées. Ce cas est arrivé aux États-Unis en 2016, où un outil d'évaluation des risques nommé COMPAS, prédisant le risque de récidive a été critiqué pour ses préjugés contre les individus noirs [WBL16].

L'explicabilité est donc devenu un sujet très étudié dans l'IA, en particulier au niveau du droit commun. Mais nous pensons que l'explicabilité est un élément essentiel pour l'accès à de nouvelles connaissances pour l'humain, car elle facilite la transmission d'informations de la machine vers l'humain. Comme on l'a vu dans le chapitre d'introduction, les IA performantes telles qu'AlphaGo avec le coup 37, ont pour la plupart apporté de nouvelles connaissances dans leurs domaines spécifiques. Comprendre les raisons des décisions prises par les algorithmes pourrait donc s'avérer crucial dans l'accès à de nouvelles connaissances pour l'humain, et comme le note [Mil19], la fonction primaire d'une explication est de faciliter l'apprentissage de quelque chose par l'interlocuteur. Lorsque l'interlocuteur pose une question, c'est pour mieux comprendre le monde qui l'entoure et l'explication est un moyen de lui faire apprendre des connaissances sur ce monde. L'importance de l'explicabilité résonne avec une caractéristique fondamentale de l'humain, qui est sa bonne capacité à établir un modèle mental de son environnement pour pouvoir le comprendre et le prédire.

En IA, l'explicabilité est un domaine de recherche où de nombreuses méthodes ont été développées [RSG16 ; LL17 ; MI22a ; RSG18 ; DH20 ; Aud+22b]. Dans ce chapitre, nous présentons une vue générale de ces méthodes d'explicabilité récentes. Nous étudierons principalement le cas des explications *locales*, dont le but est d'expliquer la décision prise par un modèle sur une observation, et qui peuvent être classées en deux catégories : les

1. `https://www.cnil.fr/fr/reglement-europeen-protection-donnees/chapitre3#Article13`

explications *non formelles* et les explications *formelles*. Pour les deux types de méthodes, on entraîne un modèle qui approxime le modèle original, appelé modèle *surrogé*, dans le voisinage de l'observation à expliquer, et on utilise ce modèle pour générer des explications. Les méthodes non formelles sont plébiscitées, mais présentent certaines limites qui peuvent entraver le passage de l'information de la machine à l'humain, notamment pour certaines caractéristiques principales que l'on attend d'une explication dont la stabilité et la non-contradiction. Le domaine des explications formelles s'intéresse beaucoup à la question de la concision, en exprimant le modèle surrogé comme une formule logique, ces méthodes proposent des calculs pour réduire la taille des explications pour garantir des explications minimales donc concises. Cependant, ces méthodes s'intéressent uniquement aux explications d'une seule observation, résultant en des explications très spécifiques qui ne permettent pas de généraliser le comportement du modèle.

5.1 Qu'est-ce qu'une explication ?

Pour introduire le type d'explications auquel on s'intéresse, il est nécessaire de définir ce qu'est une explication. Une explication est une interaction entre deux entités, où l'*expliquant* doit faire comprendre à un *interlocuteur* les raisons d'un évènement. Une explication répond à la question du « pourquoi ? » [Mil19] posée par l'interlocuteur à l'expliquant. Par exemple, l'interlocuteur peut demander à l'expliquant « pourquoi est-ce que les feuilles des arbres changent de couleur en automne ? ». Pour répondre à cette question, l'expliquant passe par une première étape de raisonnement dans le but de déterminer toutes les causes possibles : « parce que la chlorophylle se décompose en automne », « parce qu'il y a une réduction de la luminosité en automne », etc. Dans une deuxième étape, se déroule l'échange entre l'expliquant et l'interlocuteur pour répondre à la question de ce dernier. Cette étape est le processus social qui a pour objectif de faire accepter à l'interlocuteur la réponse de l'expliquant. L'expliquant a alors le choix de donner une seule explication, d'en donner plusieurs en même temps, de les donner toutes, ou bien d'en donner uniquement une partie, en fonction de l'objectif de l'interaction. Par exemple, si l'interaction est dans un but pédagogique, on préfère souvent donner plusieurs explications, pour permettre à l'interlocuteur de comprendre les causes de l'évènement sous différents angles.

En IA, les explications sont souvent utilisées pour comprendre une décision prise par un modèle sur une observation particulière. Par exemple, prenons l'exemple d'un modèle de décision de l'attribution d'un crédit à un client par une banque. L'observation est la description complète du client (son âge, son revenu, sa situation maritale, son emploi, etc.), et une explication est de donner les facteurs que le modèle a pris en compte dans sa décision de lui accorder un crédit ou non (le fait que le client ait un emploi stable, un revenu élevé, etc.). Une explication peut également avoir pour but de comprendre le comportement général d'un modèle permettant notamment d'expliciter les biais d'un modèle. Par exemple, si un modèle de prédiction de risque de récidive d'un détenu est biaisé envers les individus noirs, une explication globale permettra de mettre en évidence les caractéristiques des individus noirs qui sont prises en compte par le modèle pour attribuer un risque de récidive plus élevé. Une explication peut alors être vue comme

étant les informations nécessaires pour comprendre le comportement d'un modèle sur une observation particulière ou sur un ensemble d'observations.

5.1.1 Explicabilité vs. interprétabilité

Il est essentiel de distinguer clairement entre *explicabilité* et *interprétabilité*, deux concepts souvent utilisés de façon interchangeable, mais différents. La norme ISO [2] a tenté de clarifier cette distinction en définissant l'explicabilité comme le niveau de compréhension des raisons qui ont mené un modèle à produire un résultat spécifique et l'interprétabilité comme l'accessibilité à la compréhension du fonctionnement de ce modèle lui-même.

L'*interprétabilité* mesure la facilité pour un humain à comprendre le fonctionnement du modèle, par exemple les arbres de décision et les réseaux neuronaux n'ont pas la même interprétabilité. Pour les arbres de décision, il est aisé pour un humain de comprendre leur processus de décision, puisqu'il suffit de suivre les branches de l'arbre que l'observation « allume ». En revanche, pour les réseaux neuronaux, il est plus difficile de comprendre comment le modèle a pris sa décision, car la décision d'un réseau neuronal est une multitude de calculs matriciels et de transformations non linéaires qui sont difficiles à suivre pour un humain. Ainsi, l'arbre de décision est plus interprétable que le réseau de neurones, mais les deux modèles peuvent être expliqués de la même manière. En revanche, même si l'on dispose d'un modèle interprétable, la décision du modèle étant donné une observation peut souvent être difficile à comprendre. Prenons les arbres de décision, qui peuvent être très profonds, le chemin parcouru par une observation dans un arbre peut contenir beaucoup de tests inutiles, dont l'issue n'influe pas la classe attribuée à l'observation et les raisons véritables de l'attribution de sa classe sont noyées dans ces tests inutiles.

L'*explicabilité* est l'ensemble de méthodes permettant d'expliciter les facteurs que prend en compte un modèle quelconque lors de l'attribution d'une classe à une observation ou de comprendre le comportement général de ce modèle face à l'ensemble des observations. Une explication ne contient que les attributs qui ont réellement un impact sur les décisions, permettant une meilleure compréhension du comportement du modèle par l'interlocuteur, et souvent, cela implique l'utilisation d'une représentation des données compréhensible par les humains, par exemple en utilisant des classements d'importance des attributs ou des règles logiques. Aucun consensus n'a été trouvé sur la forme que doit prendre une explication, mais certains critères ont été définis pour évaluer la qualité d'une explication, qui seront développées en section 5.1.2. L'explicabilité est donc un concept plus large que l'interprétabilité, car elle peut être appliquée à des modèles interprétables et non interprétables.

5.1.2 Caractéristiques attendues d'une explication

Pour qu'une explication soit utile pour un interlocuteur, il est nécessaire qu'elle soit de bonne qualité. Dans [Che+22b], les auteurs effectuent une revue des différentes caractéristiques citées dans la littérature permettant de définir une bonne explication et ont identifié 4 métriques principales pour évaluer une explication :

2. `https://www.iso.org/obp/ui/en/#iso:std:iso-iec:tr:29119:-11:ed-1:v1:en`

— *La robustesse :* La robustesse d'une explication est la capacité de l'explication à rester valable malgré des changements mineurs d'une observation donnée. Les utilisateurs auxquels on fournit une explication non robuste peuvent perdre leur confiance dans un modèle. On associe également à la robustesse le critère de *stabilité*, qui requiert que deux exécutions de la méthode d'explication sur une même observation donne une seule explication.

— *La fidélité :* La fidélité d'une explication est la capacité de l'explication à refléter le vrai raisonnement du modèle dans sa prise de décision. Les auteurs notent qu'une explication non fidèle peut mener à des erreurs de jugement de la part de l'utilisateur, ce qui peut être problématique dans des domaines critiques.

— *La simplicité/concision :* La simplicité d'une explication est la capacité de l'explication à être facilement compréhensible par un utilisateur, mesurée par la quantité d'efforts cognitifs nécessaires par l'utilisateur pour comprendre l'explication. Dans [RB18], les auteurs présentent différentes métriques pouvant être utilisées pour mesurer la simplicité d'une explication, comme la longueur de l'explication (le nombre de règles, le nombre de mots, le nombre de caractéristiques), la lisibilité de l'explication (dépendante de l'audience recevant l'explication), etc.

— *L'homogénéité :* L'homogénéité désigne la capacité d'un modèle à générer des explications de bonne qualité pour toutes les différentes observations : peu importe l'observation analysée, l'explication doit être robuste, fidèle et simple. L'homogénéité est pertinente pour l'équité, car elle concerne la capacité d'une explication à rester précise même lorsqu'une observation se trouve dans une région de faible densité ou bien proche de la frontière de décision. Si un modèle est biaisé, une explication qui respecte l'homogénéité doit être capable de refléter ce biais.

5.1.3 Explications locales

Les *explications locales* sont un type d'explications qui visent à expliquer la décision d'un modèle pour une observation particulière, sans avoir besoin de la connaissance de l'architecture du modèle (on dit que les explications locales sont agnostiques à l'architecture du modèle). Ces explications sont dites également des explications *a postériori*, car elles visent à expliquer une décision déjà prise par le modèle, notons que les méthodes d'explication locales ont besoin d'accéder au modèle afin de produire une explication. La plupart des méthodes d'explications locales fonctionnent en utilisant le voisinage de l'observation à expliquer afin d'entraîner un modèle explicatif qui approxime le modèle original à expliquer dans le voisinage de l'observation, appelé modèle *surrogé*. Le modèle surrogé est quant à lui interprétable, il peut être un arbre de décision, une régression linéaire ou un ensemble de règles logiques. On distingue alors deux types d'approches parmi les explications locales : les explications *non formelles* qui vont directement interpréter le modèle surrogé en considérant que l'interprétation de ce modèle surrogé se suffit à elle-même pour générer des explications. Par exemple, si le modèle surrogé est une régression linéaire, les coefficients de la régression linéaire peuvent être interprétés comme les importances des caractéristiques dans la décision du modèle original. Et les explications *formelles* qui vont exprimer le modèle surrogé par des formules logiques, sur

lesquelles on peut effectuer des opérations de simplification pour obtenir des explications minimales.

Les explications locales ont été définies par contraste avec les *explications globales*, qui à l'inverse, visent à expliquer le comportement global du modèle et non pas sur une observation particulière. Ces explications sont souvent utilisées pour comprendre comment le modèle prend ses décisions en général, permettant alors de détecter les biais éventuels du modèle afin de le rendre plus juste.

5.2 Méthodes d'explications non formelles

Le terme *explications non formelles* désigne les méthodes qui visent à offrir des explications de la décision d'un modèle en interprétant directement le modèle surrogé entraîné dans le voisinage de la décision à expliquer. Ce terme a été introduit dans les travaux de Marques-Silva [MI22a] en contraste avec ce que ces auteurs ont appelé les explications *formelles*. Les méthodes non formelles cherchent à apporter une simplification du modèle original en se reposant sur un échantillonnage des observations pour entraîner un modèle surrogé interprétable qui imite le modèle original. Parmi les méthodes non formelles, des méthodes phares comme LIME [RSG16], SHAP [LL17] et Anchors [RSG18] trouvent leur utilisation dans un nombre élevé de domaines industriels, comme la finance [GG21 ; HL23], la médecine [Che+22a ; Oh+22], la psychologie [Aki+22 ; Li+22], la manufacture [Bru+23 ; MN22], etc. Dans cette section, nous allons donc présenter les principales méthodes d'explications non formelles, puis nous présenterons les travaux qui parlent des limitations de ces méthodes. En effet, malgré leur popularité croissante et leur capacité à rendre les modèles d'apprentissage automatique plus accessibles à un public plus large, ces méthodes présentent certaines limites intrinsèques qui peuvent poser des problèmes dans des contextes critiques.

5.2.1 Méthodes d'attribution additive des caractéristiques

Les méthodes d'explications non formelles les plus connues et utilisées ont été classées par [LL17] dans la catégorie des *méthodes d'attribution additive des caractéristiques* et sont des méthodes dans lesquelles le modèle surrogé est un modèle linéaire, dont les coefficients sont les importances des caractéristiques dans la décision du modèle original sur l'observation à expliquer. Le modèle surrogé g est une fonction linéaire de variables binaires :

$$g(o) = \phi_0 + \sum_{i=1}^{M} \phi_i o_i \tag{5.1}$$

où l'observation à expliquer est $o \in \{0,1\}^M$, M est le nombre de caractéristiques de o, et $\phi_i \in \mathbb{R}$ sont les coefficients du modèle linéaire, dont la valeur est interprétée comme l'importance de la caractéristique i dans la décision du modèle original. Les méthodes d'attribution additive des caractéristiques visent donc à calculer la contribution de chaque caractéristique à la prédiction du modèle. Ces méthodes sont très populaires, car elles

permettent de donner une explication simple de la décision du modèle, en attribuant une valeur d'importance attribuable à chaque caractéristique de l'observation.

LIME

LIME [RSG16] est l'un des principaux systèmes d'explications locales, son principe est d'apprendre à un modèle linéaire à approximer le modèle original dans le voisinage de l'observation dont la prédiction est à expliquer. Afin de rendre interprétables les explications, LIME applique une fonction de mapping qui transforme l'observation o en une observation o' représentée par un vecteur de 0 et de 1, dénotant la représentation de l'observation originale en observation interprétable. Par exemple, cette nouvelle représentation peut être une absence ou une présence d'un mot, d'un pixel, etc. Pour expliquer la décision sur o, LIME génère des observations synthétiques voisines de o', dont les caractéristiques sont des sous-ensembles des caractéristiques de o, et calcule un modèle linéaire appris sur ces observations synthétiques. Les coefficients ϕ de l'équation 5.1 sont alors les coefficients du modèle linéaire appris.

Afin de calculer le modèle linéaire, LIME minimise une fonction de perte $\mathcal{L}(f, g, \pi_o)$, où f est le modèle original, et g est le modèle surrogé, exprimant l'infidélité de l'approximation de f par g, dans la localité définie par π_o. Les auteurs définissent également une mesure de complexité $\Omega(g)$ qui permet de pénaliser une explication qui serait trop complexe, et qui peut être par exemple la profondeur d'un arbre, le nombre de caractéristiques non-nulles, etc.

La formule de calcul du modèle linéaire est la suivante :

$$\phi = \arg\min_{g} \mathcal{L}(f, g, \pi_o) + \Omega(g) \tag{5.2}$$

SHAP

SHAP [LL17] est un autre programme d'explication locale et repose sur les valeurs de Shapley. Les *Valeurs de Shapley* sont une méthode issue de la théorie des jeux, dont l'objectif est l'attribution d'une récompense à un joueur au sein d'une équipe proportionnelle à sa participation dans le résultat d'un jeu coopératif. Cette méthode a été introduite en 1953 par Shapley [Sha53] et a été depuis adaptée en IA pour mesurer l'importance des caractéristiques dans la décision d'un modèle. L'adaptation est faite en considérant qu'une caractéristique de l'observation est un joueur dans la méthode de Shapley, et l'importance d'une caractéristique est vue comme la récompense attribuée à un joueur. De façon sommaire, les valeurs de Shapley sont les contributions marginales moyennes de chaque caractéristique calculées à partir de toutes les *coalitions* possibles de caractéristiques : une coalition est un sous-ensemble de caractéristiques dont la valeur est fixée à leur valeur dans l'observation à expliquer, les autres caractéristiques sont remplacées par des valeurs aléatoires à partir des données. La valeur de Shapley d'une caractéristique est calculée en évaluant sa contribution sur la prédiction du modèle, en ajoutant cette caractéristique dans chaque coalition possible, puis en faisant la moyenne de ces contributions. Cela offre une représentation de l'importance relative de chaque caractéristique en fonction de ses interactions avec les autres caractéristiques.

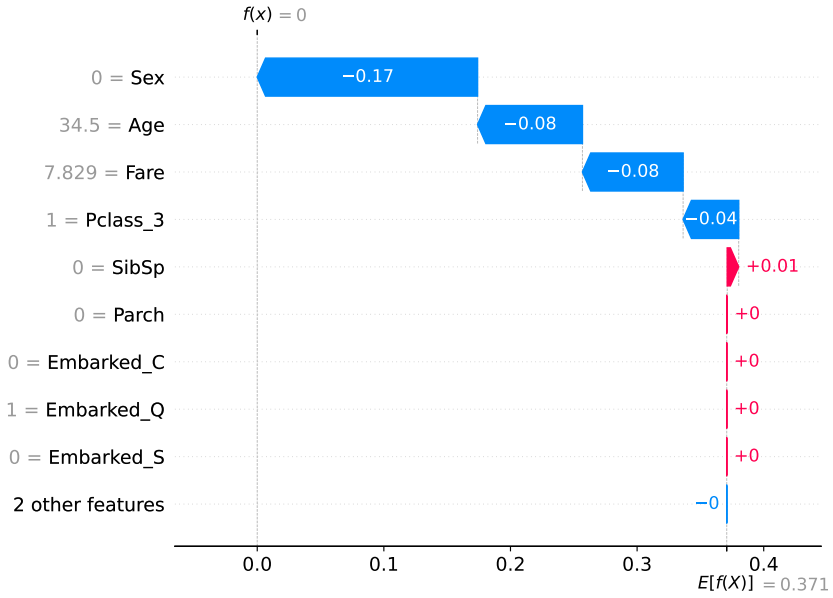


FIGURE 5.1 – Exemple de valeurs de Shapley calculées par SHAP pour un arbre de décision entraîné pour la prédiction de survie d'un passager dans le jeu de données Titanic. (source de l'image : exécution du notebook au lien `https://www.kaggle.com/code/gundawrsharvari/titanic-survival-analysis-using-shap-values`)

Le calcul des valeurs de Shapley peut être très coûteux, surtout lorsque le nombre de caractéristiques est élevé étant donné le grand nombre de sous-ensembles de caractéristiques à évaluer. Des travaux ont développé des approches d'approximation des valeurs de Shapley [ŠK10 ; ŠK13 ; AJL21] afin de contrevenir à ce défaut. De plus, l'utilisation des valeurs de Shapley résulte toujours en des explications très longues, car elles sont calculées et présentées pour chaque caractéristique de l'observation à expliquer.

Le programme SHAP est décliné en différentes versions qui proposent toutes des approximations des valeurs de Shapley. La principale approche proposée est celle de KERNEL SHAP, qui est une méthode proposant une définition non heuristique des paramètres Ω, $\mathcal{L}$ et π_o de la formule 5.2, de façon à ce que les coefficients du modèle linéaire en résultat soient les valeurs de Shapley, permettant de garantir trois propriétés essentielles du résultat : (i) la *justesse locale* qui impose que le modèle surrogé ait la même prédiction que le modèle original pour une même observation, (ii) la *distribution linéaire* qui impose que les caractéristiques absentes ou nulles dans l'observation originale n'aient aucun impact dans sa décision, et (iii) la *consistance* qui impose que si nous changeons un modèle de telle sorte qu'une caractéristique devienne plus importante (quelle que soit la définition de l'importance), la valeur de Shapley de cette caractéristique ne devrait pas diminuer. Dans cette version de SHAP, la complexité du modèle surrogé n'est pas pénalisée.

Dans la figure 5.1, nous présentons un exemple d'explication de SHAP pour un arbre de décision entraîné pour la prédiction de survie d'un passager dans le jeu de données Titanic [3]. On cherche à expliquer la classification d'un passager particulier x dont les valeurs

3. `https://www.kaggle.com/c/titanic`

des caractéristiques sont données sur l'axe des ordonnées. Les valeurs de Shapley sont représentées par des barres horizontales, où les barres rouges indiquent une contribution positive à la prédiction de survie et les barres bleues une contribution négative. Les caractéristiques sont ordonnées par leur importance. La contribution de chaque caractéristique à la prédiction (la prédiction du modèle original sur le passager x est 0 donc prédit non survécu) est calculée à partir la probabilité a priori de la classe à prédire représentée par $E[f(X)]$ sur la figure (dans l'exemple, la survie d'un passager a une probabilité a priori égale à 0.371).

La variable la plus influente est le sexe du passager (0 signifie masculin), avec une contribution négative de -0.17, ce qui signifie que le fait que le passager soit un homme a une forte influence négative sur la prédiction de survie. Viennent ensuite l'âge et le prix du billet, chacun avec une contribution négative, ce qui suggère que l'âge du passager x et le prix de son billet aient également contribué à réduire la probabilité de survie du passager attribuée par le classifieur. À l'inverse, la caractéristique *SibSp*, signifiant le nombre de frères et sœurs de x dans le bateau a contribué positivement à la prédiction de survie : ici le résultat est incohérent, car le passager n'a aucun frère ou sœur à bord, alors que la contribution est positive, bien que restant très faible.

5.2.2 Une méthode non formelle à base de règles : Anchors

Anchors [RSG18] est une méthode de génération d'explications locales, où le modèle surrogé est cette fois un ensemble de règles logiques dénommées *ancres*. L'idée d'Anchors est d'expliquer l'attribution d'une classe c à une observation o par un modèle. Contrairement à LIME où l'on apprend un modèle linéaire dont les coefficients sont les importances des caractéristiques expliquant la décision dans le voisinage de o, on va apprendre un modèle composé de règles en logique propositionnelle afin d'approximer le modèle original dans le voisinage de o. Le corps de la règle couvrant le plus d'exemples est alors utilisé en tant qu'explication du classement de l'observation par le modèle original.

Le modèle surrogé appris par Anchors est équivalent aux modèles appris par les méthodes d'apprentissage supervisé classiques présentées dans le chapitre 3 section 3.1, le modèle est une hypothèse composée d'un ensemble de règles concluant sur c, et l'ensemble des règles est appris sur un ensemble d'exemples positifs et négatifs. Pour construire les règles qui composent l'hypothèse, l'observation o et les observations voisines de o sont mappées vers une représentation atomique et les atomes de cette représentation forment l'espace de recherche des règles.

Le programme apprend alors un modèle surrogé sur l'ensemble des données voisines de o' en utilisant un algorithme spécifique de recherche de règles proposé par les auteurs. L'ensemble des exemples positifs est l'ensemble des observations voisines de o pour lesquelles le modèle original prédit c, et l'ensemble des exemples négatifs est l'ensemble des observations voisines de o pour lesquelles le modèle original ne prédit pas c. Une fois l'hypothèse construite, le corps la règle couvrant le plus d'exemples positifs et le moins d'exemples négatifs est sélectionnée comme ancre, donc comme explication.

On prend ici l'exemple développé dans [RSG18] où l'on veut expliquer la décision d'un modèle de classification binaire sur des critiques de films. L'observation o à expliquer est une critique de film sous format textuel et les atomes représentent la présence d'un mot

dans le texte.

Exemple 5.1. *On considère la classe + où sont classées les observations si leur sentiment est prédit comme positif, et − sinon. Soit o = « this movie is not bad », classée en + par le modèle. Soit E l'ensemble des observations voisines étant donné une mesure de distance. E^+ est l'ensemble des observations classées + par le modèle, et E^- est l'ensemble des observations classées −, par exemple :*

$$E^+ = \{\text{« this movie is not bad »} ; \text{« this movie is cool »} ; \ldots\}$$
$$E^- = \{\text{« this movie is bad »} ; \text{« this movie is not good »} ; \ldots\}$$

On suppose que la règle $+ \leftarrow not \wedge bad$ couvre une majorité relative des exemples de E^+ et une minorité relative des exemples de E^-. L'ancre est alors {not,bad}, indiquant que la présence des mots not et bad dans la critique est la raison pour laquelle le modèle a classé la critique o comme positive.

5.2.3 Limitations des méthodes non formelles

Malgré leur grande popularité, les méthodes d'explications non formelles présentent des limites qui peuvent poser problème dans des contextes critiques. Tout d'abord, ces méthodes présentent des difficultés techniques comme évoquées dans [Mol18]. Le premier problème pour LIME et Anchors, c'est qu'il est nécessaire de définir tout d'abord le « voisinage » d'une observation, ce qui est une tâche délicate lorsqu'on a à notre disposition des données tabulaires, car la notion de distance entre observations n'est pas toujours évidente à définir. SHAP quant à lui, dans sa version KernelSHAP, est très coûteux en temps de calcul.

En ce qui concerne les inconvénients qualitatifs, et ce sont ceux qui nous intéressent le plus, les explications non formelles peuvent ne pas vérifier les propriétés d'une bonne explication énoncées dans la section 5.1.2 :

— *Fidélité :* Ces méthodes peuvent être sensibles aux attaques adversariales [Dim+20 ; Sla+20], ce qui veut dire qu'il est possible de manipuler un modèle possédant des biais discriminatoires sur certaines caractéristiques (par exemple, un modèle d'attribution de crédit qui se base principalement sur le sexe de la personne), pour obtenir une explication qui suggère que le modèle ne se base pas sur ces caractéristiques. Par ailleurs, il a été montré dans [Ign20] que même pour des jeux de données simples, les méthodes non formelles pouvaient fournir une même explication pour deux observations de classes différentes.

— *Stabilité :* Ces méthodes fonctionnent autour d'un échantillonnage de données synthétiques pour l'entraînement d'un modèle surrogé. Cet échantillonnage étant effectué de façon stochastique, elles peuvent ne pas prendre en compte les corrélations entre les caractéristiques des observations, et donc ne pas synthétiser des observations réalistes. Cette méthode d'échantillonnage est considérée dans [Kel+23] comme étant à l'origine de l'instabilité de SHAP, qui peut produire dans plusieurs exécutions de SHAP des explications différentes pour une même observation.

— *Robustesse :* Dans [AJ18] les auteurs montrent que les méthodes non formelles ne sont pas robustes, c'est-à-dire que des changements mineurs dans l'observation à expliquer, entraînant très peu ou aucun changement dans la prédiction du modèle, peuvent entraîner des changements importants dans l'explication fournie par le modèle explicatif.

Face à ces limitations, des travaux se sont penchés sur l'amélioration de ces méthodes d'explications, par exemple, il a été proposé dans [Sai+20] et dans [Kel+23] de meilleures méthodes d'échantillonnage pour LIME et SHAP, respectivement, permettant de répondre aux problèmes de stabilité de ces méthodes. En parallèle, un domaine de recherche s'est développé autour des méthodes d'explications formelles, qui visent à expliquer le comportement d'un modèle par des justifications logiques. Ces méthodes offrent des garanties plus fortes que les méthodes non formelles et sont présentées dans la section suivante.

5.3 Méthodes d'explications formelles

On étudie ici les méthodes d'explications formelles, définies dans [MI22b] comme étant des méthodes qui visent à expliquer le comportement d'un modèle par des justifications logiques. Dans les explications formelles, le modèle surrogé doit être exprimé par une formule logique, permettant de caractériser les explications d'un classifieur comme des *impliquants premiers* de la formule logique qui lui est associée. Les impliquants premiers sont un concept largement étudié dans la littérature et de nombreux algorithmes existent pour les calculer.

On distingue deux types d'explications formelles, les *explications abductives* et les *explications contrastives*. Les explications abductives cherchent à donner les raisons du classement d'une observation par un classifieur, tandis que les explications contrastives cherchent à donner les raisons pour lesquelles une observation n'a pas été classée dans une autre classe.

Dans cette section, nous allons présenter une vue générale des travaux sur les explications formelles, tout d'abord en logique propositionnelle, puis en logique du premier ordre. Pour chaque formalisme, nous présentons les explications abductives dans les détails et les explications contrastives brièvement, car ces dernières ne sont pas liées aux contributions dans cette thèse.

5.3.1 Généralités

Nous posons ici les bases des explications abductives et contrastives. Nous présentons brièvement les concepts qui sous-tendent ces types d'explications, et nous définissons les termes qui seront utilisés dans la suite de ce chapitre.

Les explications abductives. Nous nous intéressons dans nos travaux aux explications abductives d'un phénomène. Ce type d'explications répond à la question du « *pourquoi ?* » et cherche à donner les raisons de ce phénomène. Dans les explications abductives, les raisons du phénomène sont un ensemble de faits qui ont été observés, et il arrive souvent qu'un sous-ensemble de ces faits suffise à expliquer le phénomène.

Exemple 5.2. *Eugénie demande un prêt immobilier à la banque « Le Crédit Français ». Eugénie possède un revenu de 2000 euros par mois, a un emploi stable, mais n'a pas d'apport personnel. Le Crédit Français possède des règles très strictes pour l'octroi de prêts immobiliers. Notamment, cette banque n'accorde un crédit immobilier qu'aux personnes gagnant au moins 2500 euros par mois, qui ont un emploi stable et qui ont un apport personnel. Eugénie se voit alors refuser le prêt immobilier. Eugénie demande alors à son conseiller bancaire « pourquoi me suis-je vue refuser le prêt immobilier ? ». Une explication abductive de ce refus pourrait être qu'elle ne gagne pas assez d'argent. C'est une raison suffisante pour expliquer le refus du prêt immobilier, mais il existe une autre explication abductive, qui est qu'elle n'a pas d'apport personnel et qui est une autre raison suffisante.*

Dans cet exemple, on voit que les explications abductives utilisées pour expliquer le refus du prêt immobilier sont un sous-ensemble des faits qui ont participé à la décision de refus du prêt immobilier et ces faits sont bien observés. Cette notion d'explication abductive est centrale dans nos travaux, car nous cherchons à donner les raisons du classement d'une observation par un classifieur.

Le raisonnement abductif est différent de la production d'explications abductives. Il convient ici de distinguer le *raisonnement abductif*, qui est l'un des trois types de raisonnements proposés par Charles Sanders Peirce (induction, abduction, déduction) [4], des explications abductives qui nous intéressent dans nos travaux.

Dans [Mil19], les auteurs précisent que le raisonnement abductif et les explications sont deux notions différentes malgré une certaine proximité. On définit le raisonnement abductif comme suit :

« Étant donné un fait B, et la connaissance que A implique B, alors on considère que A est possiblement vrai »

À la différence des explications abductives au sens que l'on entend dans cette thèse, on ne cherche pas dans le raisonnement abductif, à expliquer le classement d'une observation, mais à produire une hypothèse probable étant donné un phénomène observé et une connaissance, appelée théorie. Intuitivement, le raisonnement abductif part du phénomène, cherche les règles qui impliquent ce phénomène, à partir desquelles il abduit des faits. On remarque que le raisonnement abductif n'affirme pas de vérité quant aux hypothèses produites, on dit que ces hypothèses sont *possiblement* vraies. Le raisonnement abductif produit donc des hypothèses, tandis que les explications abductives sont des ensembles de faits observés qui suffisent à expliquer un phénomène.

Remarque 5.1. *On prend ici le parti de parler d'abduction lorsqu'on abduit un fait à partir d'une théorie, et de parler d'explication abductive lorsqu'on cherche à expliquer le classement d'une observation par un classifieur.*

Les explications contrastives. Le deuxième type d'explications considéré dans la littérature concerne *les explications contrastives*. Ces explications répondent à la question du « *pourquoi pas ?* » et intègrent un raisonnement contrastif dans la question à laquelle

4. `https://plato.stanford.edu/entries/abduction/peirce.html`

l'explication doit répondre. La question de « *pourquoi P ?* » devient « *pourquoi P et pas Q ?* ». On appelle P le *fait*, et Q le *contraste*. De nombreux auteurs en sciences cognitives [Lip90 ; Mil19 ; Bar94] considèrent que les explications devraient toujours répondre à une question contrastive, car les explications pour un fait diffèrent pour deux contrastes différents. Par exemple, les explications à la question « *pourquoi Alice a ouvert la fenêtre ?* » ne sont pas les mêmes si la question est « *pourquoi Alice a ouvert la fenêtre au lieu de la laisser fermée ?* », à laquelle on peut répondre « *parce que Alice a chaud* », ou si la question est « *pourquoi Alice a ouvert la fenêtre au lieu d'allumer la climatisation ?* », où l'explication « *parce que Alice a chaud* » ne fonctionne plus. Ce type d'explications est particulièrement étudié dans la plupart des travaux sur les explications formelles, ce qui justifie que nous le traitions dans cet état de l'art (même si on ne travaille pas directement sur ce type d'explications).

5.3.2 Explications formelles en logique propositionnelle

Abduction en logique propositionnelle. Une abduction en logique propositionnelle est une explication d'un phénomène observé c au regard de ce qui est connu, modélisé par une théorie Σ, où c est un littéral et Σ est un ensemble de clauses.

Afin de caractériser les abductions, nous utilisons la définition présentée dans [Mar91], où une abduction en logique propositionnelle de c par rapport à Σ est un motif propositionnel t, tel que $t, \Sigma \models c$.

Définition 5.1. *Un motif propositionnel t est une abduction de c étant donné une théorie Σ si $t, \Sigma \models c$.*

Exemple 5.3. *Soit un vocabulaire propositionnel $\mathcal{V} = \{a, b, c\}$, et $\Sigma = \{c \leftarrow a \wedge b\}$ une théorie composée d'une seule clause. Le motif $t = a \wedge b$ est une abduction de c par rapport à Σ, car $t, \Sigma \models c$.*

Il peut exister un nombre très élevé d'abductions lorsque la théorie à partir de laquelle on cherche à abduire est composée de plusieurs clauses, ainsi il est nécessaire de se restreindre aux explications vérifiant une contrainte de *minimalité*. La minimalité est ici définie au sens de l'implication.

Définition 5.2. *Une abduction t de c par rapport à Σ est minimale si pour toute abduction t' de c par rapport à Σ, si $t \models t'$ alors $t' \equiv t$.*

On cherche donc à engendrer toutes les abductions minimales de c par rapport à Σ. La notion de minimalité est reliée dans ces travaux aux impliquants premiers décrits dans la section 2.3.1. En partant de la définition 5.1, on a $t, \Sigma \models c$. Or $t, \Sigma \models c$ si et seulement si $t \models (\Sigma \rightarrow c)$. Les abductions t de c par rapport à Σ sont alors les impliquants de $(\Sigma \rightarrow c)$ (voir section 2.3), et les abductions *minimales* t_{min} de c par rapport à Σ sont les impliquants premiers de $(\Sigma \rightarrow c)$. Une propriété intéressante des impliquants premier, est qu'une formule propositionnelle est équivalente à la disjonction de ses impliquants premiers, il n'y a donc aucune perte d'information quand on représente une telle formule par ses impliquants premiers.

L'équivalence $t \models (\Sigma \rightarrow c)$ ssi $(\Sigma \wedge \neg c) \models \neg t$ montre une dualité qui dit que les impliquants premiers de $(\Sigma \rightarrow c)$ sont la négation des impliqués premiers de $(\Sigma \wedge \neg c)$. La génération des abductions d'une formule en logique propositionnelle peut donc se réduire à un problème de génération d'impliqués premiers.

Pour générer les impliqués premiers de $F = (\Sigma \wedge \neg c)$, une méthode souvent utilisée est la *résolution* (voir section 2.1.5) proposée dans ce cadre initialement dans les travaux de Tison [Tis67]. Le procédé est le suivant : (i) on transforme F en CNF (ii) on génère les résolvants de toutes les clauses de la CNF (iii) on retire les résolvants qui sont impliqués par d'autres résolvants. Les résolvants restants sont alors les impliqués premiers de la formule. Par la négation des impliqués premiers en sortie, on obtient alors l'ensemble des impliquants premiers de $(\Sigma \rightarrow c)$, qui sont les impliquants premiers et donc les abductions minimales de c par rapport à Σ. Un exemple d'utilisation de la résolution pour générer les impliquants premiers d'une formule est présenté dans la rubrique suivante.

Cette approche présente des limites computationnelles importantes, en effet, dans [Pro90] l'auteur montre que le nombre d'impliqués premiers d'une formule propositionnelle est exponentiel en la taille de la formule dans le pire cas, et que le pire cas est atteint dans presque toutes les formules propositionnelles. Des travaux ultérieurs ont cherché à développer des algorithmes plus efficaces pour générer les impliqués premiers d'une formule, notamment en sélectionnant plus efficacement les clauses sur lesquelles on applique l'opération de résolution pour limiter le nombre d'opérations de ce type [KT90 ; Jac92], d'autres utilisent des structures de données plus efficaces pour diverses optimisations, notamment le temps de calcul de l'implication entre les impliqués générés [Kle92] ou le nombre d'opérations de résolution [SV01]. D'autres méthodes de génération d'impliquants premiers dans le cadre propositionnel sont étudiées plus en détails dans [Tou16].

Les impliquants premiers sont donc une notion très liée aux abductions en logique propositionnelle, et constituent un élément central dans la génération des explications abductives qui seront présentées dans le paragraphe suivant.

Explications abductives minimales. Les *explications abductives minimales*, également appelées *P(rime)I(mplicants)-explications* [SCD18], ou *raisons suffisantes* [DH20] dans la littérature, expliquent le classement d'une observation particulière par un classifieur. Une explication abductive minimale du classement d'une observation par un classifieur, est un sous-ensemble minimal des caractéristiques de cette observation qui est un impliquant premier du modèle. Ce sous-ensemble est tel que, si ses valeurs restent constantes, la décision du modèle concernant cette observation restera inchangée.

Définition 5.3. *Soit un langage propositionnel L et le vocabulaire $\mathcal{V}$ qui lui est associé, une observation $o = \{x \text{ ou } \neg x \mid x \in \mathcal{V}\}$ et D un classifieur représenté par une formule propositionnelle dans le langage L et c un atome représentant la décision de D sur o telle que $o, D \models c$. Une explication abductive t de c pour o via D est un motif propositionnel inclus dans o tel que $t, D \models c$. t est une explication abductive minimale si c'est un impliquant premier de $(D \rightarrow c)$.*

Exemple 5.4. *Soit un vocabulaire propositionnel $\mathcal{V} = \{x, y, z, k\}$ et un ensemble d'étiquettes $C = \{+, -\}$. On considère l'arbre de décision décrit dans la Figure 5.2, dans lequel une instance suit l'arc droit d'un test portant sur la variable v si $v = Vrai$ et l'arc*

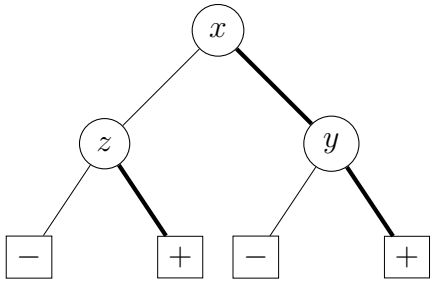


FIGURE 5.2 – Arbre de décision de l'exemple 5.4

gauche si $v = Faux$. La formule propositionnelle correspondant à l'étiquette + de l'arbre est $D = \{+ \leftarrow x \wedge y; + \leftarrow \neg x \wedge z\}$. L'observation $o = \{x, y, z, k\}$ est classée + par D et correspond à la feuille la plus à droite de l'arbre, une explication abductive du classement de o en + via D est donc $[x, y]$.

Le calcul des explications abductives minimales est effectué en appliquant la négation des impliqués premiers de $(D \wedge \neg +)$, qui est équivalent à $(\neg x \vee \neg y) \wedge (x \vee \neg z)$. On montre ensuite le calcul d'un impliqué par la dérivation suivante en utilisant la résolution :

$$\frac{\neg x \vee \neg y \qquad x \vee \neg z}{\neg y \vee \neg z}$$

$\neg y \vee \neg z$, mais également $\neg x \vee \neg y$ et $x \vee \neg z$ sont des impliqués de $(D \rightarrow +)$. Par application de la négation de $\neg y \vee \neg z$, on obtient l'impliquant $y \wedge z$ qui est premier, car aucun de ses sous-ensembles n'est un impliquant de $(D \rightarrow +)$. De même pour $\neg x \vee \neg y$ et $x \vee \neg z$ qui donnent les impliquants premiers $x \wedge y$ et $\neg x \wedge z$.

Ceux inclus dans o sont $x \wedge y$ et $y \wedge z$, $[x, y]$ et $[y, z]$ sont donc les explications abductives minimales de o.

Il est à noter que ces travaux se distinguent des travaux sur l'abduction par le fait que l'on cherche ici à expliquer la décision d'un classifieur sur une observation, et non pas à abduire un fait à partir d'une théorie. Contrairement à une abduction qu'on qualifie de « possiblement vraie », une explication abductive est un sous-ensemble des caractéristiques de l'observation dont on peut affirmer la véracité, car elles ont bien été observées.

Calcul des explications abductives minimales. Durant ces cinq dernières années, de nombreuses recherches ont été menées afin de développer des algorithmes capables de fournir les explications abductives minimales pour la décision d'un classifieur sur une observation. Certains de ces algorithmes se sont avérés très efficaces sur certains types de classifieurs notamment les classifieurs Naive Bayes [SCD18], les graphes [Hua+21], les arbres de décision et forêts aléatoires [Aud+22b ; Aud+22a ; IIM22 ; IIM20].

Nous présentons ici la méthode proposée dans les études [Aud+22b ; Aud+22a], où les auteurs proposent qu'au lieu de calculer toutes les explications abductives minimales en raison de leur nombre très élevé, de se focaliser sur les explications abductives minimales *de plus petite taille*, c'est-à-dire celles qui ont le plus petit nombre d'atomes. Cette approche est motivée par le fait que cette catégorie d'explications contient des explications plus

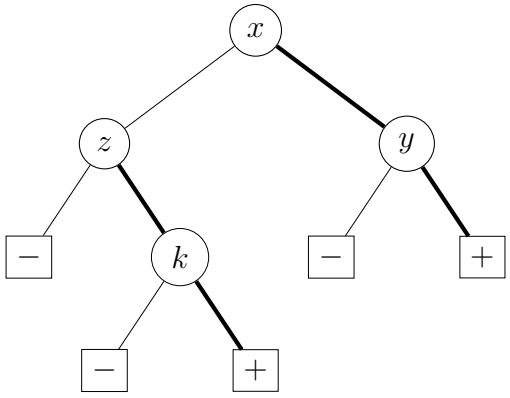


FIGURE 5.3 – Arbre de décision de l'exemple 5.5

simples à comprendre pour un humain grâce à leur concision, mais également par le fait qu'il y en a moins que des explications abductives minimales.

Cette méthode transforme la formulation de la tâche de trouver les explications abductives de plus petite taille en un problème d'optimisation sous contraintes. Le problème est traduit en problème Partial MAXSAT, consistant en une paire de contraintes (C_{soft}, C_{hard}), où C_{soft} et C_{hard} sont des ensembles finis de clauses. L'objectif est de trouver un modèle qui viole un minimum de contraintes de C_{soft} tout en vérifiant toutes les contraintes de C_{hard}.

Proposition 5.1. *Soit un vocabulaire $\mathcal{V}$, une observation $o = \{x \text{ ou } \neg x \mid x \in \mathcal{V}\}$, c l'étiquette attribuée à o par un arbre de décision binaire, et D la formule propositionnelle dont les modèles sont les observations classées dans c par D. Soit (C_{soft}, C_{hard}) une instance du problème Partial MAXSAT telle que :*

$$C_{soft} = \{\neg x \mid x \in o\} \cup \{x \mid \neg x \in o\} \text{ et } C_{hard} = \{s \cap o \mid s \in CNF(D)\}.$$

Une explication abductive minimale de plus petite taille de l'observation o est l'intersection entre o et l'interprétation I qui est une solution au problème Partial MAXSAT posé par (C_{soft}, C_{hard}).

Exemple 5.5. *Dans cet exemple, on considère l'arbre de décision décrit dans la figure 5.3. La formule propositionnelle dont les modèles sont les observations classées dans l'étiquette + est $D = (x \wedge y) \vee (\neg x \wedge z \wedge k)$. On reprend l'observation $o = \{x, y, z, k\}$ classée + par l'arbre de décision, on a alors $C_{soft} = \{\neg x, \neg y, \neg z, \neg k\}$.*

On a :

$$CNF(D) = (x \vee z) \wedge (x \vee k) \wedge (y \vee \neg x) \wedge (y \vee z) \wedge (y \vee k).$$

On a alors $C_{hard} = \{x \vee z, x \vee k, y \vee z, y \vee k, y\}$, équivalent à $\{x \vee z, x \vee k, y\}$.

L'interprétation $I = \{x, y\}$ est celle violant le moins de contraintes de C_{soft} tout en vérifiant toutes les contraintes de C_{hard}, $[x, y]$ correspond donc à la plus petite explication abductive minimale de la décision du classifieur sur o.

La caractérisation du problème de génération d'explications abductives dans le problème Partial MAXSAT décrit ci-dessus possède deux avantages :

— Premièrement, il est possible de générer toutes les explications abductives minimales en ajoutant la négation de la solution en tant que clause dans C_{hard} et en relançant la recherche d'une nouvelle solution. La nouvelle solution sera alors garantie de ne pas être un sur-ensemble d'une solution déjà trouvée. L'opération est répétée jusqu'à ce que l'optimiseur ne trouve aucune solution.

— Deuxièmement, il est possible de ne considérer que les explications abductives minimales de plus petite taille en incorporant dans C_{hard} une contrainte de cardinalité qui encode la taille de la dernière solution trouvée. De ce fait les solutions suivantes sont garanties d'avoir une taille inférieure ou égale à celle de la dernière solution.

Explications contrastives en logique propositionnelle Les explications contrastives sont des explications qui visent à expliquer pourquoi un modèle a effectué une certaine décision et pas une autre. Ces explications cherchent donc à donner des raisons d'une décision dont le changement garantirait de faire basculer la décision vers une autre.

Dans [Ign+20], les auteurs définissent les explications contrastives lorsqu'une observation est décrite par un ensemble de caractéristiques en logique propositionnelle, rejoignant la définition des explications abductives minimales décrites précédemment. Une explication contrastive de la décision d'un modèle pour une observation est un sous-ensemble minimal de caractéristiques dont le changement de valeur ferait basculer la décision du modèle pour cette observation.

Définition 5.4. *Soit un langage propositionnel L et le vocabulaire $\mathcal{V}$ qui lui est associé, une observation $o = \{x \text{ ou } \neg x \mid x \in \mathcal{V}\}$, D un classifieur représenté par une formule propositionnelle dans L et c la décision du modèle telle que $o, D \models c$. Une explication contrastive de c est un sous-ensemble de $t \subseteq o$ tel que $o', D \not\models c$ avec $o' = \{o \setminus t\} \cup \{\neg x \mid x \in t\}$.*

Une explication contrastive est minimale si tout sous-ensemble propre de t n'est pas une explication contrastive.

Exemple 5.6. *En reprenant l'exemple 5.4, on a l'observation $o = \{x, y, z, k\}$ classée en + par D. Une explication contrastive minimale de cette décision est $t = y$, car $o' = \{x, \neg y, z, k\}$ est classée − par D. Une autre explication contrastive minimale est $t = \{x, z\}$, car $o'' = \{\neg x, y, \neg z, k\}$ est classée − par D, et tout sous-ensemble propre de $\{x, z\}$ n'est pas une explication contrastive.*

Dualité entre les explications contrastives et abductives Comme vu plus haut, une explication abductive minimale pour le classement d'une observation o en c par un modèle D est un impliquant premier t de $(D \rightarrow c)$ inclus dans o. En prenant $D_c = (D \rightarrow c)$ on a notamment $t \models D_c$, équivalent à $t \wedge \neg D_c \models \bot$.

Ces propriétés ont permis de définir la génération des explications abductives minimales comme étant un problème de maintien de l'insatisfiabilité d'une formule. Le but est de trouver le sous-ensemble minimal de littéraux $t \subseteq o$ tel que $t \wedge \neg D_c \models \bot$. Ce sous-ensemble minimal est appelé un *Minimal Unsatisfiable Subset* (MUS) [Ign+20].

En parallèle, trouver une explication contrastive minimale du classement de o en c revient à trouver un sous-ensemble minimal de littéraux $t \subseteq o$ tel que $o' \wedge \neg D_c \not\models \bot$ avec $o' = \{o \setminus t\} \cup \{\neg x \mid x \in t\}$. Ce sous-ensemble minimal est appelé un *Minimal Correction Subset* (MCS) [Ign+20].

Les auteurs de [Ign+20] mettent en lumière la dualité qui existe entre les explications abductives et les explications contrastives, définie par la relation d'*ensemble intersectant*, également appelés *Minimal Hitting Set*.

5.3.3 Explications formelles en logique du premier ordre

Motivations. La logique propositionnelle est un formalisme qui permet de développer des techniques efficaces de génération d'explications comme on l'a vu dans la sous-section précédente, mais l'expressivité de ce formalisme reste limitée.

Prenons en exemple la phrase suivante : (a) « le joueur joue l'As au pli 1 et le Roi au pli 2 ». Pour exprimer cette phrase en logique propositionnelle, il faudrait introduire deux atomes distincts, c'est-à-dire a pour « le joueur joue l'As au pli 1 » et k pour « le joueur joue le Roi au pli 2 », et exprimer la phrase par $a \wedge k$. Or, dans ces phrases, il y a plusieurs objets qui peuvent avoir des relations complexes entre eux. Dans la vie de tous les jours, il arrive souvent qu'il y ait besoin de généraliser lorsqu'on cherche à décrire une situation ou expliquer une action. Par exemple, l'As et le Roi sont toutes les deux des cartes qui sont appelées *gros honneur* au bridge, et dans de nombreux cas, si un joueur possède ces deux cartes dans sa main, le choix de jouer une des deux cartes avant l'autre influe peu. Dans ce cas, on abstrait très souvent en disant que (b) « le joueur joue une carte qui est un gros honneur au pli 1 ». Le formalisme de la logique propositionnelle ne permet pas d'atteindre une granularité suffisante pour décrire cette phrase en réutilisant les atomes a et k déjà introduits. La logique du premier ordre, en revanche, permet une granularité assez forte pour faire apparaître ces relations dans une formule : la phrase (a) peut s'exprimer par la formule $\mathit{played}(\mathit{as}, \mathit{pli1}) \wedge \mathit{played}(\mathit{king}, \mathit{pli2})$ en logique du premier ordre. Si l'on veut exprimer la phrase (b), on peut introduire une variable existentielle *Card* qui représente une carte : $\mathit{played}(\mathit{Card}, \mathit{pli1}) \wedge \mathit{bigHonor}(\mathit{Card})$. Le prédicat *bigHonor* permet d'indiquer que la carte *Card* est un gros honneur, sans pour autant indiquer de quelle carte il s'agit exactement.

Cette expressivité plus élevée de la logique du premier ordre permet d'exprimer des relations plus complexes entre les éléments d'un domaine, et de capturer des connaissances plus riches, rendant ce formalisme adapté à la production d'explications plus riches et plus expressives. Il convient donc de s'intéresser à la question de la génération des explications en logique du premier ordre.

Abductions en logique du premier ordre. Dans la continuité des abductions en logique propositionnelle, la question des abductions en logique du premier ordre a été abordée pour la première fois par Marquis dans [Mar91]. Une abduction est dans le cas de la logique de premier ordre, un motif relationnel, c'est-à-dire une conjonction de littéraux existentiellement quantifiée.

Définition 5.5 (Abduction en LPO)**.** *Un motif relationnel t est une abduction de c étant donné une théorie Σ si $t, \Sigma \models c$.*

Exemple 5.7. *Soit un vocabulaire du premier ordre $\mathcal{V}$ composé des prédicats $\{p/1, q/1, r/2\}$ et des constantes $\{a, b\}$, et une théorie $\Sigma = \{r(X, Y) \leftarrow p(X) \wedge q(Y)\}$. On considère l'atome $c = r(a, b)$. Une abduction de c par rapport à Σ est $t = p(a) \wedge q(b)$, car $(p(a) \wedge q(b)), (r(X, Y) \leftarrow p(X) \wedge q(Y)) \models r(a, b)$.*

De même que pour les abductions en logique propositionnelle, dans l'objectif de générer des abductions concises, on ne va considérer que les abductions minimales. Ici également, la minimalité est définie au sens de l'implication.

Définition 5.6 (Abduction minimale en LPO)**.** *Une abduction t de c par rapport à Σ est minimale si pour toute abduction t' de c par rapport à Σ, si $t \models t'$ alors $t' \equiv t$.*

On cherche alors à générer l'ensemble des abductions minimales de c par rapport à Σ, qui on le rappelle, sont définies comme les impliquants premiers de $(\Sigma \rightarrow c)$, et la recherche des impliquants premiers de $(\Sigma \rightarrow c)$ peut se réduire à la recherche des impliqués premiers de $(\Sigma \wedge \neg c)$ (voir section 5.3.2).

Dans [Mar91], l'auteur note que dans le cas général en LPO, contrairement au cas propositionnel, cette méthode présente deux principaux problèmes pour la logique du premier ordre :

1. Toutes les conséquences logiques d'une formule du premier ordre ne sont pas nécessairement impliquées par l'ensemble des impliqués premiers de cette formule, ce qui fait que la résolution seule n'est pas suffisante pour générer toutes les abductions d'un phénomène ;
2. Il y a possiblement une infinité d'impliquants/impliqués premiers d'une formule du premier ordre, et ce, même si l'on considère uniquement les symboles de fonction d'arité nulle (les constantes).

L'auteur propose une restriction aux formules du premier ordre qui sont des 2-clauses (clauses à deux littéraux) et qui sont monodiques (prédicats unaires), pour lesquelles le point 1 est tout de même vérifié. Cette restriction présente en revanche l'inconvénient d'une expressivité restreinte des formules logiques considérées.

Générer les impliqués premiers est donc un défi hors de portée dans le cadre de la logique du premier ordre dans le cas général. Néanmoins, il est possible de définir les abductions minimales comme étant des motifs qui ne sont θ-subsumés par aucun autre motif. On passe donc d'une minimalité définie sur l'ordre partiel de l'implication, qui est indécidable en LPO, à l'ordre partiel de la θ-subsomption qui est décidable en LPO. Le procédé sera alors semblable à la génération des impliqués premiers en logique propositionnelle. Pour générer les abductions minimales au sens de la θ-subsomption (θ-minimales) de c par rapport à Σ : (i) on génère l'ensemble T des conséquences logiques de $(\Sigma \wedge \neg c)$ (ii) on retire de T les clauses qui sont θ-subsumées par d'autres clauses de T. La négation des clauses de T restantes seront alors les abductions θ-minimales de c par rapport à Σ. Il reste en revanche un problème pour le point (i), c'est que l'ensemble des conséquences logiques d'une formule du premier ordre peut être infini.

Pour pallier cela, Inoue a introduit la *résolution linéaire ordonnée avec règle de saut* (*résolution-SOL*) [Ino91], un algorithme qui permet de générer l'ensemble des conséquences logiques d'une formule F étant donné un critère de sélection (la taille, le nombre de variables, etc.), qui ne sont θ-subsumées par aucune autre conséquence logique de F. De telles conséquences logiques sont appelées les *clauses caractéristiques* et ne considérer que ce type de clauses permet alors d'éviter d'en générer une infinité. En prenant $F = (\Sigma \wedge \neg c)$, l'application de cet algorithme permet alors de générer l'ensemble des conséquences logiques θ-minimales de F qui satisfont un critère de sélection, et dont la négation est l'ensemble des abductions θ-minimales de c par rapport à Σ. La résolution-SOL sera par la suite implémentée avec diverses optimisations dans le programme SOLAR [Nab+10].

Explications contrastives en logique du premier ordre. Des travaux récents [RSS22] proposent une approche basée sur les *near-miss* [Win70] dans le cadre de l'apprentissage par implication logique en PLI afin de générer des explications contrastives. La méthode présentée dans ces travaux génère des explications *near-miss*, c'est-à-dire que l'explication donnée au classement d'une observation dans une classe est une observation la plus proche possible de l'observation à expliquer, mais classée dans une étiquette différente. Dans le cas de [RSS22], un near-miss est produit en effectuant des variations de la règle déclenchée sur l'observation à expliquer. Prenons l'exemple 3.3, avec H l'hypothèse apprise et B la théorie du domaine, et on cherche à expliquer $grandpere(jean, luc)$. La clause $grandpere(jean, luc) \leftarrow homme(jean), parent(jean, pierre), parent(pierre, luc)$ ed.st la règle concluant sur l'exemple à expliquer. Cette règle est la clause apprise $S \in H$ avec la substitution $\theta = \{X/jean, Y/luc, Z/pierre\}$. Les auteurs définissent une explication near-miss $S'\theta'$ comme étant une variation minimale de $S\theta$ telle que $B, H \models body(S'\theta')$ et $B, H \not\models head(S'\theta')$. Par exemple, une explication near-miss de $S\theta$ est $S'\theta' = grandpere(sarah, luc) \leftarrow femme(sarah), parent(sarah, pierre), parent(pierre, luc)$, avec $\theta' = \{X/sarah, Y/luc, Z/pierre\}$ et $S' = grandpere(X, Z) \leftarrow femme(X), parent(X, Y), parent(Y, Z)$.

5.3.4 Avantages et limitations des explications formelles

Les explications formelles offrent des garanties que les explications non formelles n'offrent pas : premièrement, la concision est l'une des principales préoccupations des explications formelles et la caractérisation de ces explications comme étant des impliquants premiers permet de garantir une explication sans redondance tout en restant correcte vis-à-vis du classifieur. Deuxièmement, les explications formelles garantissent également la fidélité requise pour une bonne explication en se reposant sur la structure du classifieur pour expliciter son raisonnement, en enlevant les tests aux nœuds qui n'influent pas sur la décision du classifieur pour une observation. Enfin, les méthodes d'explications formelles génèrent une multitude d'explications pour une seule observation, ce qui permet de donner une vision plus large des raisons de la décision du classifieur et facilite sa compréhension par l'interlocuteur.

Malgré les garanties proposées, ces approches présentent certaines limitations :

— Les explications que ces méthodes peuvent générer sont valides uniquement pour une seule observation, or nous pensons que lorsqu'un humain demande une explication,

il la veut dans un certain niveau de généralité, c'est-à-dire qu'elle soit assez générale pour expliquer plus d'une observation. Mais ces méthodes ne s'intéressent qu'aux explications pour une seule observation ;

— On ne peut pas générer d'explications si l'on ne connaît pas le modèle surrogé, en effet les explications formelles se basent sur des calculs à partir de la formule logique qui exprime le modèle surrogé. Si l'on ne connaît pas cette formule, il est impossible de générer des explications ;

— Ces méthodes génèrent des explications concises, mais en même temps elles peuvent en générer un grand nombre pour une seule décision, ce qui peut rendre la tâche de sélection des explications à présenter à l'interlocuteur difficile.

5.4 Conclusion

Dans ce chapitre, nous avons présenté un panorama des différentes méthodes d'explicabilité récentes existant dans la littérature. Nous nous sommes focalisé sur les explications locales, qui sont des explications qui cherchent à expliquer la décision d'un modèle sur une observation. Nous avons vu qu'il y avait deux grandes familles d'explications locales : les explications non formelles et les explications formelles.

Les méthodes d'explications non formelles bénéficient d'une grande popularité en raison de leur accessibilité. Cependant, ce type de méthodes n'offre pas de garanties fortes sur les explications en sortie, pouvant entraver la compréhension de la décision du modèle par l'interlocuteur, ou même aller jusqu'à le tromper dans le pire des cas. De leur côté, les méthodes d'explications formelles, génèrent des explications les plus concises possibles, tout en garantissant la fidélité de l'explication vis-à-vis du modèle. Pour cela, les explications formelles expriment le modèle surrogé en une formule logique, dont les impliquants premiers inclus dans l'observation sont les différentes explications minimales de la décision du modèle. Cette formulation permet alors de générer de nombreuses explications d'une seule décision, donnant une vision plus large des raisons de la décision du modèle à l'interlocuteur.

Cependant, le domaine des explications formelles a été développé principalement dans le cadre de la logique propositionnelle. Des travaux ont été menés pour générer des abductions minimales au sens de la θ-subsomption d'une formule en logique du premier ordre, mais les explications de la décision d'un classifieur en logique du premier ordre restent un défi ouvert. Nous avons montré que la logique du premier ordre bénéficiait d'une expressivité supérieure à la logique propositionnelle, et qu'elle était plus adaptée à la génération d'explications riches et expressives. Notamment, les explications en logique du premier ordre permettent de raisonner de manière générale grâce à l'introduction de variables existentielles, se rapprochant ainsi de la manière dont les humains raisonnent.

Dans le chapitre suivant, nous présentons une approche de génération des explications abductives du premier ordre, en utilisant l'ensemble des observations possibles pour générer des explications abductives communes à plusieurs observations. Ces explications sont alors plus générales et l'utilisation de la logique du premier ordre est plus adaptée pour générer ce type d'explications.

Chapitre 6

Explications abductives communes minimales

6.1 Introduction

Dans ce chapitre, nous allons développer la deuxième contribution principale de cette thèse. Cette contribution reprend en majeure partie les récents travaux que nous avons publiés [Rou+23] et les étend par de nouvelles définitions et de nouveaux algorithmes. Nous nous intéressons ici aux explications en logique du premier ordre. Ces explications ont pour but d'expliquer pourquoi un agent a effectué une action a dans un état s. Le travail est illustré par un scénario dans lequel une machine joue une partie de bridge simplifié dans la position du déclarant contre un programme qui joue le rôle du défenseur. Elle peut être interrogée par un humain à tout moment du jeu pour répondre à la question : « pourquoi l'action choisie conduit-elle à une trajectoire de jeu dont la récompense est optimale ? ». L'optimalité est définie dans le sens des *q-values* d'un MDP (voir chapitre 1), dans lequel les q-values les plus élevées sont les trajectoires dans lesquelles le déclarant remporte le plus de plis. Nous nous intéressons plus particulièrement aux explications abductives communes d'un ensemble de trajectoires de jeu, qui seront utilisées pour expliquer l'observation demandée. De ce fait, on considère un agent qui sait planifier et générer l'ensemble des trajectoires de jeu à partir de (s, a), et qui sait décider si une trajectoire de jeu est optimale ou non. L'agent est guidé par le but c à atteindre à partir de l'état courant, c étant l'optimalité de la trajectoire de jeu. À partir de (s, a), il y a autant de manières de jouer de façon optimale, que de trajectoires de jeu optimales à partir de (s, a), mais présenter l'ensemble des trajectoires optimales à l'interlocuteur n'est pas une explication satisfaisante pour lui. À cela, nous proposons de regrouper ces trajectoires entre elles, en les expliquant par des *explications communes*. Par ailleurs, nous nous inscrivons dans la mouvance des explications formelles telles que décrites dans le chapitre 5, où il est primordial que les explications soient *minimales*, ce qui permet de présenter des explications concises à l'interlocuteur, vérifiant ainsi une des caractéristiques principales d'une bonne explication (voir section 5.1.2).

Dans [Rou+23], nous avons proposé une méthodologie pour construire de telles explications. La première étape consiste à regrouper les trajectoires optimales partant de (s, a) en plusieurs groupes, chaque groupe correspondant à une manière de jouer optima-

lement depuis (s, a). Expliquer l'action a revient alors à expliquer chacun de ces groupes de trajectoires, c'est-à-dire expliquer les différentes façons de jouer optimalement. Nos observations à expliquer sont maintenant les trajectoires de jeu optimales depuis (s, a) et sont définies comme des interprétations de Herbrand. Pour réaliser cette première étape, nous entraînons un classifieur D à base de règles logiques à décider si une trajectoire est optimale ou non. Ensuite, nous utilisons la couverture de chaque règle de D pour regrouper les trajectoires optimales entre elles. Une explication commune pour un groupe de trajectoires de même étiquette est défini comme un *motif relationnel* qui couvre les trajectoires de ce groupe.

Contrairement aux travaux sur les explications formelles, la minimalité de nos explications n'est pas définie au sens des impliquants premiers de D, mais au sens de la couverture sur un groupe de trajectoires à expliquer. Nous avons donc introduit les *explications communes subset-minimales* qui sont des explications pour lesquelles on a la garantie qu'elles ne contiennent pas de littéraux redondants. La deuxième étape de notre méthodologie consiste à construire la *lgg* du sous-ensemble de trajectoires que l'on cherche à expliquer, qui est le motif relationnel le plus spécifique couvrant toutes les trajectoires de ce sous-ensemble. Nous avons étendu également les travaux sur les explications formelles en introduisant la formule d'univers U, dont les modèles sont l'ensemble de toutes les observations possibles. Cette formule d'univers permet alors de retirer les littéraux redondants de la *lgg*, la dernière étape consistait alors à extraire les explications communes subset-minimales de la *lgg* en utilisant un algorithme de recherche de motifs minimaux. Dans [Rou+23], nous avons proposé deux contributions algorithmiques : (i) un algorithme de calcul de l'approximation de la *lgg* qui permet de réduire les coûts de calcul de la *lgg* en imposant des contraintes, et (ii) un algorithme de recherche des motifs subset-minimaux qui est une adaptation de l'algorithme de [SR17] à la LPO.

L'approximation de la *lgg* implique automatiquement une perte de spécificité, ce qui implique que les motifs minimaux extraits de la *lgg* approximée puissent être plus généraux que certains des motifs minimaux de la vraie *lgg*. Or, la notion de spécificité est essentielle dans les explications. En effet, il est apparu que, bien que très utiles à la généralité, les variables contenues dans les explications pouvaient parfois nuire à la compréhension des explications. Ainsi, dans ce chapitre, nous actualisons la notion d'*explications préférées* définie dans [Mar91], en préférant les explications maximalement instanciées parmi les explications minimales.

Dans ce chapitre, nous reprenons les travaux présentés dans [Rou+23] et les étendons de plusieurs manières, donnant lieu aux contributions suivantes :

— Nous définissons les *explications communes leq-minimales* (ou *leq-mces*) comme étant les explications maximalement instanciées parmi les explications minimales au sens ensembliste (*subset-minimales*) définies dans [Rou+23]. Dans [Rou+23], nous évoquions une préférence pour l'explication les plus spécifiques parmi les explications subset-minimales, sans inclure cette préférence dans la définition de la minimalité. Dans ce chapitre, nous présenterons la définition actualisée de la minimalité ;

— Nous présentons l'algorithme de calcul de l'approximation de la *lgg*, nommée *Bottom*, tel que décrit dans [Rou+23] ;

— Nous présentons l'algorithme de recherche des motifs minimaux qui est une adap-

tation de l'algorithme de [SR17] à la LPO ;

— Nous présentons l'algorithme d'instanciation maximale des explications communes subset-minimales, permettant de construire les explications communes leq-minimales ;

— Nous proposons une méthode de sélection des explications, de manière à construire le sous-ensemble maximal telle que la diversité est garantie. Cette méthode s'appuie un algorithme de sélection [SSB20] accompagné d'une mesure de la similarité syntaxique s entre deux motifs relationnels inspirée de [Fer+09]. Cette méthode renvoie les explications telles qu'étant donné deux d'entre elles m et m', on garantie $s(m, m') < \beta$;

— Enfin, nous présentons diverses expérimentations de cette méthode sur la version simplifiée du jeu de la carte au bridge, en étudiant les explications générées par notre méthode pour deux distributions de cartes différentes. Nous comparons enfin les explications entre ces deux distributions de cartes, pour mettre en avant les futurs travaux à réaliser sur les *explications multimonde*, proposées dans le chapitre des perspectives (chapitre 7).

6.2 Restrictions et notations

Nous utilisons dans notre expérience le langage DATALOG (c'est-à-dire la LPO sans symbole de fonction autre que des constantes). Les seuls termes admis sont donc les *constantes* et les *variables*. Nous nous plaçons dans le cadre de l'apprentissage par interprétations en PLI (voir section 3.2.1) : nos observations sont considérées comme des interprétations de Herbrand du vocabulaire $\mathcal{V}$.

De plus, on pose la restriction suivante : on se restreint aux motifs positifs, c'est-à-dire les motifs qui ne contiennent que des littéraux positifs. Ainsi, pour savoir si une interprétation o est un modèle d'un motif m, il suffit de ne considérer que les atomes vrais dans o. La notation $\dot{o}$ désigne alors maintenant la conjonction des atomes vrais dans o. Un motif m couvre une observation o si o est un modèle de m, c'est-à-dire si $\dot{o} \models m$.

On rappelle la notation des motifs, un motif relationnel $\exists a_1 \wedge \ldots \wedge a_n$ est noté par la liste suivante : $[a_1, \ldots, a_n]$. On écrit $[m, q]$ la concaténation de deux motifs m et q. On écrit $[m, l]$ par abus de notation la concaténation d'un motif m et d'un littéral l.

6.3 Explications

6.3.1 Explications pour des observations en LPO

Nous considérons les *explications abductives* présentées dans la section 5.3.1. Dans ces travaux, une observation est décrite par les valeurs que prennent un ensemble d'attributs propositionnels. L'explication du classement d'une observation o par un classifieur D dans une classe c est alors un sous-ensemble minimal des attributs de o qui est suffisant pour expliquer le classement de o en c. Nous allons adapter cette notion d'explication à notre contexte.

Dans un contexte relationnel, on décrit une observation o comme une interprétation de Herbrand à partir d'un vocabulaire $\mathcal{V}$, que l'on représente comme une conjonction de littéraux, c'est-à-dire un motif relationnel instancié construit sur $\mathcal{V}$. Le classifieur D est une formule logique du premier ordre construite sur $\mathcal{V} \cup C$ où C est un ensemble de prédicats d'arité nulle qui représentent les classes. L'explication du classement d'une seule observation dans une classe c est alors un motif relationnel instancié construit sur $\mathcal{V}$. En revanche, lorsqu'on veut des explications d'un groupe d'observations, il faut considérer des motifs relationnels plus généraux, c'est-à-dire possiblement non instancié.

L'ensemble des observations possibles $\mathcal{U}$ de notre problème, que nous appelons l'*univers*, n'est qu'une partie de l'ensemble entier des interprétations de Herbrand possibles à partir de $\mathcal{V}$. Ci-dessous, $\mathcal{U}$ est représenté par l'ensemble des modèles d'une formule U construite sur $\mathcal{V}$, que l'on appelle la *formule d'univers*. Un classifieur D assigne à toute interprétation de Herbrand u sur $\mathcal{V}$ exactement une classe c de l'ensemble C, et à u est donc associée exactement une interprétation de Herbrand u_D dans $\mathcal{V} \cup C$. On a donc $\dot{u}, D \not\models \bot$, vu que u_D est un modèle de $\dot{u}, D$ et $\dot{u}, D \models c$. Autrement dit, D classe toutes les observations et aucune observation ne falsifie D.

La connaissance dans notre problème est donc divisée en une partie U qui restreint les observations o qui sont autorisées et une partie D permettant d'inférer la classe de toute observation.

Définition 6.1 (Explication subset-minimale d'une observation)**.** *Une explication de l'assignation de la classe c à une observation o par un classifieur D et une formule d'univers U dont o est un modèle, est un sous-ensemble t de o tel que $t, U, D \models c$.*

Si pour tout $t' \subset t$ on a $t', U, D \not\models c$ alors t est une explication abductive subset-minimale de l'assignation de c à o par rapport à D et U.

En ajoutant en tant que formule U l'univers des observations possibles, on obtient des explications plus simples que celles obtenues en omettant U. Le résultat suivant généralise le théorème 1 de [GR22] et considère le cas plus général dans lequel seulement une partie des contraintes restreignant l'univers des observations possibles est connu et représenté comme une *formule d'univers partiel* U_g, c'est-à-dire une formule construite sur $\mathcal{V}$ telle que $U \models U_g$.

Proposition 6.1. *Soit U_g une formule d'univers partiel, u une observation qui satisfait U, et de label c étant donné un classifieur D. Alors :*

1. *Toute explication pour u étant donné U_g est aussi une explication pour u étant donné U ;*
2. *Pour toute explication subset-minimale e pour u par rapport à U_g il existe une explication subset-minimale e' pour u par rapport à U telle que $e' \subseteq e$.*

Démonstration. Le point 1 découle de la définition 6.1 : si on a $U \models U_g$ alors pour tout $e \subseteq \dot{u}$, de $e, U, D \models e, U_g, D$ et de $e, U_g, D \models c$ on infère $e, U, D \models c$. Notons que l'on a considéré que u est un modèle de U et est donc un modèle de U_g. Concernant le point 2, du point 1 on sait que e est une explication pour u étant donné U. Ceci veut dire que soit e est subset-minimal, ou qu'il existe une explication subset-minimale e' par rapport à U telle que $e' \subset e$. □

Nous illustrons cette proposition à l'aide d'un exemple en logique propositionnelle.

Exemple 6.1. *D est un ensemble de clauses définies concluant sur c, et on considère que si $o, D, U \not\models c$ alors o est classifiée en $\neg c$. On considère un langage propositionnel construit sur les atomes $\{a, p, q\}$ avec le classifieur $D = \{c \leftarrow p\}$, la formule d'univers $U = \{p \leftarrow q \wedge a; p \vee q \leftarrow\}$ et l'observation $o = \{p, q, a\}$ qui satisfait U. Considérons les formules d'univers partiel suivantes :*

- *$U_g = \{p \leftarrow q \wedge a\}$. Maintenant, les explications subset-minimales de o par rapport à D et U_g sont $[p]$ et $[q, a]$: grâce à la partie définie de U, on a $[q, a]$ comme explication subset-minimale en plus de l'évidente explication $[p]$.*
- *Considérons les explications subset-minimales de o par rapport à D et U entier. Premièrement, on a p qui est une explication subset-minimale. Deuxièmement, $[q, a]$ est toujours une explication, mais elle n'est plus subset-minimale, étant donné que $a, U \models p$. En effet, quand a est vrai on a soit (i) q est vrai et donc $a, q, U \models p$, ou (ii) q est faux et donc de $a, \neg q, U$ on infère p. Par conséquent, $[p]$ et $[a]$ sont les deux explications subset-minimales.*

On peut également considérer U_g comme étant le contexte général pour lequel D est construit est valide tandis que U représente les observations possibles dans le contexte courant. On peut également considérer U_g comme une formule plus générale que U, car certaines contraintes de U ne sont pas disponibles.

6.3.2 Explications communes leq-minimales

Nous définissons maintenant une *explication commune* de l'assignation d'une même classe c à un ensemble d'observation O par un classifieur comme étant un motif relationnel, contenant possiblement des variables.

Définition 6.2 (Explication commune à un ensemble d'observations)**.** *Une explication commune de l'assignation d'une même classe c à un ensemble d'observations O par rapport à un classifieur D et une formule d'univers U est u n motif relationnel e, tel que $\forall o \in O$, o est un modèle de e et $e, U, D \models c$.*

Si pour tout $e' \subset e$ on a $e', U, D \not\models c$ alors e est une explication commune subset-minimale de O (raccourcie en subset-mce).

Selon cette définition, tout motif relationnel est une explication commune pour O tant qu'il permet d'inférer c pour tout $o \in O$. En considérant les subset-mces, on se retrouve avec des explications plus concises, ce qui correspond à ce qu'on attend d'une explication. En effet, plus une explication est courte, plus il est facile d'inférer c pour un humain, mais également pour une machine. En revanche, les explications abductives sont définies comme étant des motifs instanciés, ce qui suggère que parmi les subset-mces pour $O = \{o\}$, on doive donc préférer celles qui sont instanciées. Une manière de satisfaire cette préférence est de préférer l'explication commune e_i à e quand e_i est une instance stricte de e (c'est-à-dire $e \prec_i e_i$) : e_i spécialise e sans ajout de nouvel atome, mais plutôt en remplaçant des variables par des constantes ou par des autres variables dans e. On définit alors les explications communes préférées comme maximales selon la concision et l'instanciation.

Ce qui nous mène à la définition des *explications communes leq-minimales*.

Définition 6.3 (Explication commune leq-minimale)**.** *Soit e une explication commune pour O.*

- *e est une explication commune instance-maximale pour O (notée instance-Mce) ssi pour tout m tel que $e \prec_i m$, m n'est pas une explication commune pour O.*
- *e est une explication commune leq-minimale pour O (notée leq-mce) ssi e est à la fois subset-mce pour O et une instance-Mce pour O.*

Exemple 6.2. *Soit l'ensemble suivant d'explications communes pour O : $\{[p(X,Y), p(Y,Z)], [p(X,X)], [r(X,X)], [r(1,1)], [p(X,X), r(X,X)]\}$. $[p(X,Y), p(Y,Z)]$ n'est pas une leq-mce car son instance $[p(X,X)]$ est une explication commune, $[r(X,X)]$ n'est pas une leq-mce car son instance $[r(1,1)]$ est une explication commune et $[p(X,X), r(X,X)]$ n'est pas une subset-mce. On obtient alors les leq-mces $\{[p(X,X)], [r(1,1)]\}$.*

On considère maintenant un exemple complet :

Exemple 6.3. *Soit p et r deux prédicats unaires et $\{p(1), p(2), r(1), r(2), z\}$ la base de Herbrand. On considère le classifieur D, la formule d'univers U et l'ensemble d'observations $O = \{o_1, o_2\}$ définis de la façon suivante :*

- $o_1 = \{p(1), p(2), r(1)\}$
- $o_2 = \{p(1), p(2), r(2)\}$
- $U = \{p(X) \leftarrow r(X); p(1) \vee p(2); \neg z\}$
- $D = \{c \leftarrow p(X), r(X); c \leftarrow p(2); c \leftarrow z\}$

On cherche dans un premier temps les motifs subset-minimaux permettant d'inférer c et qui sont vrais pour les deux observations dans O. Premièrement, de $r(X)$ et U on infère $p(X)$, et de $p(X), D$ on infère c. Deuxièmement, de $p(2)$ et D on infère c. $[r(X)]$ et $[p(2)]$ sont deux subset-mces et comme $[r(X)]$ n'a aucune instance qui soit une explication commune pour O, $[r(X)]$ et $[p(2)]$ forment l'ensemble des leq-mces pour $O = \{o_1, o_2\}$.

6.3.3 Identification des explications par étiquetage de l'univers

Dans ce qui suit, la formule d'univers U est connue à travers l'ensemble de ses modèles $\mathcal{U} = mod(U)$ partitionné selon les classes assignées par D en $\{\mathcal{U}^c \subseteq \mathcal{U} \mid c \in C\}$. Une explication commune e de l'assignation de c à $O \subseteq \mathcal{U}^c$ est telle qu'elle n'est satisfaite par aucune observation de $\mathcal{U}^{notc} = \mathcal{U} \setminus \mathcal{U}^c$.

On peut alors construire l'ensemble des subset-mces pour des groupes d'observations ayant la même classe c à partir de la partition $\{\mathcal{U}^c, \mathcal{U}^{notc}\}$ sans avoir à utiliser le classifieur :

Proposition 6.2. *Un motif relationnel e est une explication commune pour $O \subseteq \mathcal{U}^c$ ssi :*

- $\forall o \in O \neq \emptyset, \dot{o} \models e$
- $\forall u \in \mathcal{U}^{notc}, \dot{u} \not\models e$.

Démonstration. Lorsque e est une explication commune pour $O \neq \emptyset$, on a $\forall o \in O, \dot{o} \models e$. On doit donc montrer que dans ce cas, on a $e, U, D \models c$ ssi $\forall u \in \mathcal{U}^{notc}, \dot{u} \not\models e$.

— $\Rightarrow$: On considère $e, U, D \models c$ et un $u \in \mathcal{U}^{notc}$. Par définition, cela veut dire que $\dot{u}, D \not\models c$. Supposons maintenant que $\dot{u} \models e$. Comme u est un modèle de U, on a $\dot{u} \equiv \dot{u} \wedge U$, et donc $\dot{u}, D \equiv \dot{u}, U, D \models e, U, D \models c$. Mais $\dot{u}, D \models c$ contredit $u \in \mathcal{U}^{notc}$ et donc notre supposition est contredite. Par conséquent, on a que $\forall u \in \mathcal{U}^{notc}, \dot{u} \not\models e$.

— $\Leftarrow$: On considère maintenant que $\forall u \in \mathcal{U}^{notc}, \dot{u} \not\models e$. Comme $\forall o \in O, \dot{o} \models e$, il s'ensuit que e, U a un modèle et donc que e, U, D a également un modèle. Supposons que $e, U, D \not\models c$. Cela veut dire qu'il existe un modèle u' de e, U tel que $\dot{u}', D \not\models c$, qui à son tour veut dire que u' appartient à $\mathcal{U}^{notc}$. Cependant, u' étant un modèle de e, on a $\dot{u}' \models e$ et cela contredit notre hypothèse, et donc notre supposition $e, U, D \not\models c$ est fausse.

□

Exemple 6.4. *On reprend l'exemple 6.3, U a 8 modèles parmi lesquels seule l'observation $o^- = \{p(1)\}$ a la classe $-$. On retrouve alors les subset-mces de $\{o_1, o_2\}$, qui sont $[r(1)]$ et $[p(2)]$, qui (i) n'ont aucune instance qui couvre O et (ii) ne couvrent pas o^-.*

6.3.4 Enumération des explications communes leq-minimales (positives)

Nous présentons dans cette section une méthode d'énumération des leq-mces d'un ensemble d'observations, en utilisant la *lgg* de cet ensemble. On considère maintenant les éléments suivants :

— Le classifieur D est un ensemble de clauses définies ;

— La formule d'univers U est un ensemble de clauses quelconques ;

— Les explications sont des motifs relationnels composés de littéraux positifs.

Dans la suite du manuscrit, tant qu'on utilise la Proposition 6.2 afin d'identifier les explications communes leq-minimales, on représente une observation o par son motif instancié positif construit à partir des littéraux de $\dot{o}$. On peut alors dans la suite noter $e \preceq_\theta o$ au lieu de $e \preceq_\theta \dot{o}$ et on dit que e couvre o. On a alors le résultat suivant :

Proposition 6.3. *Les explications communes leq-minimales pour O sont des sous-ensembles (à un renommage de variables près) de $lgg(O)$.*

Démonstration. Soit m une leq-mce, comme m couvre toutes les observations de O, on a que (i) $m \preceq_\theta lgg(O)$ c'est-à-dire qu'il existe un θ tel que $m\theta \subseteq lgg(O)$ ce qui signifie que $m\theta$ couvre toutes les observations de O. Par ailleurs, on a que (ii) $m\theta$ est plus spécifique que m ou équivalente (au sens de la θ-subsomption) et m ne couvre aucune observation de $\mathcal{U}^{notc}$, par conséquent $m\theta$ ne couvre pas non plus d'observation de $\mathcal{U}^{notc}$. De (i) et (ii) on déduit que $m\theta$ est une explication commune pour O. Comme m est une leq-mce, aucune instance stricte de m n'est une explication commune pour O, ce qui veut dire que $m\theta$ n'est pas une instance stricte de m et donc θ est un renommage de variable. □

On a ensuite une proposition simple qui dit qu'il suffit de vérifier qu'un motif dans $lgg(O)$ ne couvre pas d'observations de $\mathcal{U}^{notc}$ pour assurer que m soit une explication commune de O.

Proposition 6.4. *Soit une subset-mce $m \subseteq lgg(O)$. m est une leq-mce ssi il n'existe pas de $m' \subseteq lgg(O)$ tel que m' soit une explication commune et que $m \prec_i m'$.*

Démonstration. — $\Rightarrow$: si m est une leq-mce alors par définition il n'y a aucune explication commune m' pour O telle que $m \prec_i m'$. Considérons $m' \subseteq lgg(O)$ telle que $m \prec_i m'$, m' couvre alors toutes les observations de O en tant que sous-ensemble de $lgg(O)$. Par ailleurs, en tant qu'instance de m, m' ne couvre aucune observation de $\mathcal{U}^{notc}$ et donc selon la Proposition 6.2, m' est une explication commune pour O, ce qui contredit le fait que m est une leq-mce. Donc m' n'existe pas.

— $\Leftarrow$: Soit $m \subseteq lgg(O)$ une subset-mce qui ne soit pas une leq-mce. Cela veut dire qu'il existe une explication commune $m_i = m\theta_1$ telle que $m \prec_i m_i$. On renomme ses variables par des variables n'apparaissant pas dans $lgg(O)$. Comme m_i et $lgg(O)$ couvrent tous les deux O, on a que $[lgg(O), m_i]$ couvre O. Par définition $lgg(O)$ est le motif le plus spécifique qui couvre O et donc on a $[lgg(O), m_i] \preceq_\theta lgg(O)$, c'est-à-dire qu'il existe θ_2 telle que $[lgg(O), m_i\theta_2] \subseteq lgg(O)$. Par conséquent, on a également $m_i\theta_2 \subseteq lgg(O)$. Soit $m' = m\theta_1\theta_2 = m\theta$. On a alors $m \neq m\theta$, c'est-à-dire $m \prec_i m'$ et m' appartient à $lgg(O)$.

□

Exemple 6.5. *Soit $O = \{o_1, o_2\}$ avec $o_1 = \{p(1), q(1)\}$ et $o_2 = \{p(1), p(2), q(2)\}$. Soit $\mathcal{U}^{notc} = o_3$ avec $o_3 = \{q(2)\}$.*

On a $lgg(O) = \{p(1), p(X), q(X)\}$. Les leq-mces de $lgg(O)$ sont $\{p(X), q(X)\}$ et $p(1)$. $p(X)$ est une explication commune pour O car le motif ne couvre pas o_3, mais n'est pas une leq-mce car $p(X) \prec_i p(1)$.

Par conséquent, pour énumérer les explications communes leq-minimales pour O, on doit dans un premier temps calculer $lgg(O)$, puis énumérer les explications communes subset-minimales qui sont des sous-ensembles de $lgg(O)$ et qui ne couvrent aucune observation de $\mathcal{U}^{notc}$ (en tant que sous-ensembles de $lgg(O)$, elles couvrent aussi toutes les observations de O), puis élaguer les explications subset-minimales dont les instances strictes sont des explications communes de O.

6.3.5 Explications communes leq-minimales en pratique

La *lgg* d'un ensemble d'observations peut être grande et chère en ressources de calcul. Dans cette section, nous proposons de construire une approximation *Bottom* plus courte de $lgg(O)$. Cette approximation doit couvrir toutes les observations de O et ne couvrir aucune observation de $\mathcal{U}^{notc}$, et donc doit θ-subsumer $lgg(O)$.

Cependant, en énumérant les leq-mces de *Bottom* comme suggéré ci-dessus, nous avons les inconvénients suivants :

— On risque de rater certaines subset-mces ;

— Il n'est pas suffisant de regarder les instances strictes des subset-mces dans *Bottom* pour générer les leq-mces.

Concernant le dernier point, on énumère en premier lieu les subset-mces incluses dans *Bottom*, puis pour chacune des ces subset-mces, on cherche ses instances maximales qui

sont des explications communes. Ces explications communes sont alors des candidates pour être des explications leq-minimales, cependant l'opération d'instanciation peut résulter en un motif qui n'est plus subset-minimale.

Exemple 6.6. *On considère l'ensemble des observations* $O = \{o_1, o_2\}$ *avec* $o_1 = \{p(1,b,a), q(1,b,a), p(3,d,a)\}$ *et* $o_2 = \{p(2,c,a), q(2,c,a), p(4,d,a)\}$ *et* $U_{notc} = \{o^-\}$ *avec* $o^- = p(1,2,4), q(3,2,4)$.

On a alors une subset-mce extraite de Bottom qui est $p(X,Y,Z), q(X,Y,Z)$ *:*

- $[p(X,Y,Z), q(X,Y,Z)]$ *est instanciée maximalement en* $[p(X,Y,a), q(X,Y,a)]$ *;*
- $[p(X,Y,a), q(X,Y,a)]$ *n'est pas subset-minimale, elle est donc réduite à la subset-mce* $[p(X,Y,a)]$ *;*
- $[p(X,Y,a)]$ *est maximalement instanciée en* $[p(X,d,a)]$.

$[p(X,d,a)]$ *est une leq-mce comme c'est une subset-mce et qu'aucune de ses instances n'est une explication commune.*

Il est donc nécessaire que l'algorithme d'énumération des explications communes leq-minimales à partir d'une approximation de la *lgg* doit alterner des étapes de recherche de subset-mces et d'instanciation maximale pour accéder aux leq-mces à partir de ces subset-mces.

Définition 6.4 (Accessibilité d'une leq-mce)**.** *Une leq-mce* l *est accessible à partir d'une subset-mce* e *ssi* $l = e$ *ou* l *est accessible à partir d'une subset-mce* $e' \subseteq e\theta$ *où* $e\theta$ *est une instance-Mce de* e*. Une leq-mce* l *est accessible à partir de Bottom ssi* l *est accessible depuis l'une de ses subset-mces.*

Une leq-mce est donc dite *accessible depuis Bottom* ssi elle est accessible depuis l'une de ses subset-mces. On a donc le résultat suivant :

Proposition 6.5. *Pour toute subset-mce* e *de Bottom il existe au moins une leq-mce accessible depuis* e*.*

Notons que l'on n'a toujours pas accès aux leq-mces de certains motifs dans *Bottom* qui ne sont pas des subset-mces.

Exemple 6.7. *Soit* $O = \{o_1, o_2\}$ *avec* $o_1 = \{p(1), p(2), q(2)\}$*,* $o_2 = \{p(1), p(3), q(3)\}$*.* $\mathcal{U}^{notc} = \{o^-\}$ *avec* $o^- = \{p(2), q(3)\}$*. Considérons Bottom* $= [p(X), q(Y), p(1)]$*. Ce Bottom correspond à la restriction où l'on autorise les variables à n'apparaître qu'une seule fois dans Bottom, une forme de contrainte que l'on utilise par ailleurs dans les expérimentations. La leq-mce* $[p(X), q(X)]$ *qui est une instance de* $[p(X), q(Y)]$ *ne peut pas être accédée à partir de* $[p(X), q(Y)]$ *car ce n'est pas une subset-mce (couvrant* o^-*).*

Cependant, quand l'approximation *Bottom* est suffisamment proche de la *lgg* exacte, une grande partie des leq-mces sont accessibles depuis *Bottom*.

Travaux associés

Comme présenté dans le chapitre 5 en section 5.3.3, les travaux de [Nab+10] proposent le programme SOLAR pour énumérer un ensemble de conséquences logiques d'une formule qui ne sont θ-subsumées par aucune autre conséquence logique. En prenant $F = (U \wedge D \wedge \neg c)$, on peut alors transformer le problème de la recherche des explications en énumération des abductions telle que décrite dans la section 5.3.3 en énumérant l'ensemble des conséquences logiques de F qui ne sont pas θ-subsumées par d'autres conséquences logiques, dont la négation sont les abductions de l'assignation de l'ensemble d'observations O par D et U à c. Les abductions obtenues sont alors θ-minimales, ce qui ne correspond pas à notre définition des explications communes leq-minimales préférées. En effet, si l'on a deux explications communes m_1 et m_2 avec $m_1 = [p(X)]$ et $m_2 = [p(a)]$ de O, SOLAR donnera m_1 comme explication préférée, car $m_1 \preceq_\theta m_2$, alors que nous avons défini les explications préférées comme étant les plus instanciées en plus de la concision, auquel cas l'explication préférée est m_2.

Dans le cas où l'on ne cherche pas les explications les plus instanciées, SOLAR pourrait être utilisé pour énumérer les explications communes leq-minimales à partir de D, U et c, mais il faudrait disposer de la formule U et non de son ensemble de modèles que nous utilisons dans notre cadre.

6.4 Construire les explications leq-minimales

Dans cette section, nous proposons l'algorithme 7 qui construit l'ensemble M des leq-mces pour O par rapport à $\mathcal{U}^{notc}$ et accessible depuis $Bottom \preceq_\theta lgg(O)$. Cet algorithme prend en paramètre *Bottom*, et calcule dans un premier temps l'ensemble G des subset-mces incluses dans *Bottom*, c'est-à-dire les motifs les plus petits par rapport à l'inclusion inclus dans *Bottom* et qui ont une couverture nulle dans $\mathcal{U}^{notc}$. Ce calcul requiert la fonction *minGenRel* décrite dans l'algorithme 9, ainsi que la fonction *locales* qui renvoie l'ensemble des locales d'un motif, c'est-à-dire des composantes connexes par variables des littéraux dans un motif. L'utilité de cette fonction sera détaillée plus tard dans la section 6.4.2. Enfin, l'algorithme 7 calcule l'ensemble M des leq-mces accessibles à partir de G (voir définition 6.4) en utilisant la fonction *fixedPoint* détaillée dans l'algorithme 11.

Nous détaillons la construction de *Bottom* ainsi que les différentes étapes de l'algorithme 7 dans les sous-sections suivantes. Notons que dans la boucle de la ligne 2 l'appel récursif peut trouver des leq-mces déjà présentes dans M. Par exemple, $[m(1,1)]$ peut avoir été accédé à partir $[m(X,1), p(X)]$ et donc ajouté à M tandis qu'il est également accessible à partir de $[m(1,Y), q(Y)]$ dans un appel récursif ultérieur. Cela veut dire que l'appel de l'union ensembliste dans la ligne 3 doit éviter les doublons.

6.4.1 Construction de l'approximation de la lgg

Le calcul de la *lgg* pour un ensemble d'observations O, dont l'algorithme a été présenté dans le chapitre 3 en section 3.3, a un coût trop élevé, en $\mathcal{O}(C^n)$ où C est la taille en nombre de littéraux de la plus grande observation et $n = |O|$. La *lgg* peut être utilisée en tant qu'opérateur de raffinement pour la recherche d'hypothèses en appliquant de

Algorithme 7 *allExplanations*($Bottom, O, \mathcal{U}^{notc}$)

Entrée: O : un ensemble d'observations d'étiquette c, $\mathcal{U}^{notc}$: un ensemble d'observations d'étiquette différente de c, $Bottom$: l'approximation de $lgg(O)$.
Sortie: Un ensemble de leq-mces pour O par rapport à $\mathcal{U}^{notc}$ accessibles depuis $Bottom$.
1: $G \leftarrow minGenRel([\,], locales(Bottom), \mathcal{U}^{notc}, \mathcal{U}^{notc})$
2: **pour** tout $m \in G$ **faire**
3: $\quad M \leftarrow M \cup fixedPoint(m, O, \mathcal{U}^{notc})$
4: **fin pour**
5: **retourne** M

façon itérative la *lgg* sur des observations. Cependant, la taille de la *lgg* d'un ensemble d'observations croît exponentiellement avec la taille de cet ensemble. Dans l'optique de garder la *lgg* d'un ensemble d'observations à une taille raisonnable, la θ-réduction (voir section 2.2.6) est souvent appliquée sur une *lgg* nouvellement construite. L'application de la θ-réduction ne garantit pas que la taille de la *lgg* augmente de façon polynomiale dans le pire des cas, mais elle permet de réduire de façon significative la taille de la *lgg* dans la plupart des cas [KSZ12]. En considérant le fait que nos observations dans l'étude de cas (voir section 6.6) ont 257 littéraux en moyenne, et que certains symboles de prédicats ont une occurrence par observation de 5 en moyenne, il a été primordial de développer un algorithme d'approximation de la *lgg*.

Travaux associés

Des recherches ont déjà été effectuées dans l'objectif de calculer de façon efficace une approximation de la *lgg* exacte sous θ-subsomption. Dans [KSZ12], les auteurs présentent une généralisation de la notion de *lgg*, qui permet de calculer des *lgg* qui sont θ-réductibles en temps polynomial. On peut également citer par exemple le cas d'Aleph [Sri99], qui calcule une bottom clause utilisée pour borner l'espace de recherche. Cette bottom clause est construite sur un exemple unique, appelée la graine, ce qui permet d'éviter l'explosion combinatoire du calcul des sélections (voir section 3.3). De plus, la bottom clause est contrainte par les modes, permettant de contraindre le nombre de littéraux partageant le même symbole de prédicat, de contraindre les positions des variables existentielles dans les prédicats ainsi que leur nombre, de choisir les prédicats à inclure dans la *lgg* à partir du vocabulaire de la théorie du domaine, etc.

Algorithme d'approximation de la *lgg*

Nous avons implémenté l'algorithme 8, un algorithme descendant flexible de type « générer et tester » qui calcule une approximation $Bottom$ de la *lgg* exacte d'un ensemble d'observations et dont la taille est bornée. Étant donné un ensemble d'observations O qui ont la même étiquette c, et un ensemble d'observations $\mathcal{U}^{notc}$ d'étiquette différente de c, $Bottom$ θ-subsume $lgg(O)$ et aucune observation $o \in \mathcal{U}^{notc}$ n'est modèle de $Bottom$. L'algorithme 8 utilise un biais de langage et borne la taille de la solution. Le biais de langage $\mathcal{B}$ est composé de (i) une liste $\mathcal{B}_t$ de symboles de prédicats associés à leurs types

d'arguments, (ii) un ensemble de contraintes $\mathcal{B}_c$ à satisfaire dans *Bottom*. L'algorithme commence par sélectionner une observation $s \in O$ en utilisant la fonction *head* qui renvoie le premier élément d'un ensemble. L'algorithme considère alors tous les éléments p de $\mathcal{B}_t$ tour à tour. Chaque prédicat p peut avoir plusieurs instances dans s (l'ensemble des littéraux de s qui ont comme symbole de prédicat p). L'opérateur de raffinement ρ prend en entrée le littéral courant l_i, le candidat courant pour *Bottom*, la substitution courante θ telle que $Bottom \preceq_\theta s$, ainsi que les contraintes $\mathcal{B}_c$. L'opérateur ρ génère des littéraux g qui subsument l_i et satisfont les contraintes de $\mathcal{B}_c$. L'algorithme sélectionne alors ceux qui couvrent O et les ordonne selon une fonction d'ordonnancement. Ces littéraux sont alors ajoutés de façon itérative dans le *Bottom* courant tant que $[Bottom, g]$ couvre toutes les observations dans O jusqu'à ce que k littéraux aient été ajoutés. Si le *Bottom* final couvre toujours des observations de $\mathcal{U}^{notc}$, l'algorithme échoue à la construction de l'approximation de la *lgg*, ce qui suggère des modifications : il faut alors soit adapter $\mathcal{B}$, soit changer les hyperparamètres de l'algorithme (la borne k, la fonction d'ordre ou les paramètres de ρ). Autrement, la réduction de *Bottom* est retournée. Par construction, la taille de *Bottom* est inférieure à $|s| * k$, et sa réduction possède la même borne.

Dans ce qui suit, une *nouvelle variable* d'un littéral g par rapport à *Bottom* est une variable qui n'apparaît pas dans *Bottom*, et un *lien* est une variable dans g qui apparaît dans *Bottom*. Les contraintes dans $\mathcal{B}_c$ appliquées aux littéraux g sont (i) limiter le nombre de variables dans g ou dans $[Bottom, g]$ (ii) limiter le nombre de liens entre g et *Bottom* (iii) déterminer les types d'arguments pour lesquels ces liens peuvent apparaître. La fonction d'ordonnancement que l'on utilise ordonne premièrement les candidats g dans l'ordre croissant du nombre de nouvelles variables dans g, puis les ordonne– par ordre croissant de couverture sur $\mathcal{U}^{notc}$. Cela assure que les littéraux les plus spécifiques sont ajoutés en premier dans *Bottom*.

Exemple 6.8. *Soit $\mathcal{V}$ un vocabulaire avec quatre constantes $\{1, 2, 3, 4\}$ et $\mathcal{B}_t$ contient trois symboles prédicats $\{p/2, r/1, q/1\}$ dont les arguments sont tous du même type. Considérons que l'on a deux observations $O = \{o_1, o_2\}$, avec $o_1 = \{p(1,2), p(2,3), r(2), q(3)\}$ et $o_2 = \{p(1,3), p(2,4), r(4), q(3)\}$. Considérons également $\mathcal{U}^{notc} = \{o^-\}$, avec $o^- = \{p(2,1), p(1,2), q(4)\}$. On considère que o_1 est la graine, k est fixé à 2, et $\mathcal{B}_c$ dit que ρ ne génère que des littéraux qui contiennent au plus une variable.*

Commençons par le symbole de prédicat p. La première instance de p dans o_1 est $p(1,2)$ qui donne $\rho(p(1,2), Bottom, \theta, \mathcal{B}_c) = \{p(1,2), p(X_0,2), p(1,X_1)\}$. $p(1,2)$ ne couvre pas o_2, ni $p(X_0,2)$, et en résultat, $p(1,X_1)$, qui couvre à la fois o_1 et o_2, est le seul élément dans Cands et est ajouté à Bottom, on a alors $Bottom = \{p(1,X_1)\}$ avec $\theta = \{X_1/2\}$. Considérons maintenant la seconde instance de p dans la graine o_1, $\rho(p(2,3), Bottom, \theta, \mathcal{B}_c)$ retourne $\{p(2,3), p(X_1,3), p(X_2,3), p(2,X_3)\}$. Comme tous ces littéraux, sauf $p(2,3)$ couvrent O, la nouvelle liste de candidats, une fois ordonnée est $Cands = \{p(X_1,3), p(X_2,3), p(2,X_3)\}$. La première tentative de spécialisation de Bottom (lignes 6–11) échoue comme $p(1,X_1), p(X_1,3)$ ne couvre pas O, mais les autres tentatives réussissent, et on a $Bottom = \{p(1,X_1), p(X_2,3), p(2,X_3)\}$ avec $\theta = \{X_1/2, X_2/2, X_3/3\}$. Dans les itérations suivantes de l'algorithme, $r(X_4)$ et $q(3)$ sont ajoutés à Bottom et $\{X_4/2\}$ à θ. On obtient finalement $Bottom = \{p(1,X_1), p(X_2,3), p(2,X_3), r(X_4), q(3)\}$ avec la substitution $\theta = \{X_1/2, X_2/2, X_3/3, X_4/2\}$ pour la graine o_1. Enfin, Bottom couvre

Algorithme 8 *ComputeBottom*$(O, \mathcal{U}^{notc}, \mathcal{B}, k)$

Entrée: O : un ensemble d'observations, $\mathcal{U}^{notc}$: un ensemble d'observations, $\mathcal{B}$: un biais de langage, *sort* : une fonction d'ordonnancement pour les littéraux, k : un entier.

Sortie: Un motif relationnel *Bottom* de taille $\leq k * |s|$, avec s le premier élément de O, qui θ-subsume tout $o \in O$ et qui rejette tout $o^- \in \mathcal{U}^{notc}$.

1: $s \leftarrow head(O)$; $Bottom \leftarrow [\,]$; $\theta \leftarrow \emptyset$
2: **pour** tout $p \in \mathcal{B}_t$ **faire**
3: **pour** tout l_i instance de p apparaissant dans s ($l_i \in s$) **faire**
4: $Cands \leftarrow sort(\{g \in \rho(l_i, Bottom, \theta, \mathcal{B}_c) \mid coverage(g, O) = O\})$
5: $added \leftarrow 0$
6: **pour** tout $g \in Cands$ tant que $added \leq k$ **faire**
7: **si** $[Bottom, g]$ couvre tout $o_j \in O$ **alors**
8: $Bottom \leftarrow [Bottom, g]$
9: $added \leftarrow added + 1$
10: **fin si**
11: **fin pour**
12: **fin pour**
13: **fin pour**
14: **si** *Bottom* couvre des observations dans $\mathcal{U}^{notc}$ **alors** échec
15: **fin si**
16: **retourne** $reduce(Bottom)$

par construction O, et a une couverture nulle dans $\mathcal{U}^{notc}$. L'algorithme réussit donc à retourner une explication commune.

Dans cet exemple, un algorithme exact de lgg aurait retourné $\{p(1, X_1), p(X_2, X_3), p(X_4, 3), p(2, X_5), r(X_3), q(3)\}$ avec $\theta = \{X_1/2, X_2/1, X_3/2, X_4/2, X_5/3\}$. Notons que la lgg exacte contient le littéral $p(X_2, X_3)$ qui n'est pas dans Bottom à un renommage de variables près.

6.4.2 Calcul des subset-minimaux

La tâche suivante est d'énumérer l'ensemble G des subset-mces pour O par rapport à $\mathcal{U}^{notc}$ qui sont inclus dans *Bottom*. Dans ce qui suit, on considère qu'un motif m est subset-minimal vis-à-vis d'un ensemble d'observations E quand aucun motif $m' \subset m$ a la même couverture dans E que m (voir définition 6.2). Lorsque le contexte est clair, nous ne mentionnons pas E.

La fonction récursive standard d'un parcours en profondeur pour chercher l'ensemble des motifs subset-minimaux contenant un motif m est comme suit : l'algorithme construit tout d'abord l'ensemble des motifs subset-minimaux contenant $[m, l]$ pour un $l \notin m$, puis cherche les motifs subset-minimaux contenant m, mais pas l. Soit $n = |m|$, on vérifie la subset-minimalité de m en vérifiant que tous les sous-motifs de taille $n - 1$ ont une couverture qui est strictement incluse dans la couverture du motif m. L'algorithme de recherche de l'ensemble des motifs subset-minimaux de *Bottom* est descendant et commence à partir de $m = [\,]$ (motif vide). Grâce à la monotonicité de la couverture, la

recherche est élaguée sous les motifs qui ont une couverture nulle dans E. Notons que l'ordonnancement des littéraux l par ordre croissant de couverture des motifs $[m, l]$ est important pour maximiser l'élagage.

Dans le cas où l'on n'a aucune variable, on a l'avantage de la propriété suivante : Soit m et q deux motifs sans variable, on a

$$cov([m, q], E) = cov(m, E) \cap cov(q, E) \tag{6.1}$$

Cette propriété a de fortes conséquences : tout sous-ensemble s d'un motif subset-minimal q est, lui aussi, subset-minimal. Cela résulte en un élagage efficace dans le processus algorithmique présenté ci-dessus : si m n'est pas subset-minimal, tout motif dans le sous-arbre dont la racine est m ne l'est pas non plus, ce qui permet d'élaguer ces sous-arbres durant la recherche.

Exemple 6.9. *Soit l'ensemble de littéraux $L = \{a, b, c, d\}$, l'ensemble des observations est $E = \{o_1\}$ avec $o_1 = \{a, b, c, d\}$. D'après l'équation 6.1, tous les motifs dans L couvrent o_1 et nous savons donc qu'il n'y a aucun motif subset-minimal avec une couverture nulle sur E.*

L'algorithme commence par le motif vide [ ] qui est subset-minimal, mais dont la couverture n'est pas nulle. L'algorithme fait alors un premier appel récursif avec le motif $[a]$ qui a la même couverture que le motif vide, et donc n'est pas subset-minimal. Le sous-arbre entier des motifs contenant $[a]$ est donc élagué et le premier appel retourne une liste vide de motifs. L'algorithme fait alors un second appel récursif avec le motif vide [ ] et la liste réduite de littéraux $\{b, c, d\}$. Pour les mêmes raisons qu'au-dessus, les appels récursifs pour $[b]$, $[c]$ et $[d]$ retourne également la liste vide. L'algorithme retourne à la fin une liste vide de solutions, qui est correcte. Notons que grâce à cet élagage efficace seulement 4 appels récursifs ont été effectués.

Dans le cas sans variables, plusieurs implémentations de cet algorithme de recherche en profondeur se basent sur de larges structures à mémoriser et accéder afin d'effectuer une vérification efficace de la subset-minimalité [CG05 ; Sza+09]. Cela vient du fait que la vérification de la subset-minimalité d'un motif de taille n requiert de calculer la couverture de tous les sous-motifs de taille $(n-1)$, impliquant un grand nombre de calculs redondants de couverture. Un algorithme efficace sans de telles structures a été implémenté dans [SR17] et s'est montré compétitif par rapport aux méthodes préexistantes.

Pour les motifs relationnels en général, qui eux peuvent contenir des variables, l'algorithme de recherche de motifs subset-minimaux doit être adapté, car on ne bénéficie pas de l'équation 6.1 : pour diminuer la couverture d'un motif m, on peut avoir besoin d'ajouter plusieurs littéraux à la fois qui partagent une variable [QC93], signifiant que l'on ne peut pas élaguer un sous arbre dont la racine est un motif qui n'est pas subset-minimal.

Exemple 6.10. *On considère maintenant le cas relationnel avec $L = \{a(X), b(X), c(X), c(Y), d(Y)\}$, $\mathcal{U}^{notc} = \{o_1\}$ et $o_1 = \{a(1), b(2), c(1), d(1)\}$. À nouveau, o_1 est couvert par tous les littéraux, qui ne sont donc pas subset-minimaux. Seulement, les motifs $[a(X), b(X)]$ et $[b(X), c(X)]$ sont subset-minimaux et ne couvrent pas o_1. Lorsque l'algorithme effectue le premier appel récursif avec $m = [a(X)]$, le sous-arbre de racine m n'est pas élagué et l'appel retournera $\{[a(X), b(X)], [b(X), c(X)]\}$. Le second*

appel récursif ne bénéficiera pas de l'élagage et devra donc considérer la plupart des sous-ensembles de $\{b(X), c(X), c(Y), d(Y)\}$ *avant de retourner l'ensemble vide.*

Plusieurs stratégies de *lookahead* ont été proposées en PLI pour résoudre ce problème, toutes ces stratégies reposant sur une recherche *ad-hoc* non exhaustive ou un biais de langage. Nous proposons ici une stratégie originale qui utilise une structuration de *Bottom* en *locales* [CJ95]. Ces locales sont les composantes connexes d'un graphe dont les arêtes lient deux littéraux de *Bottom* lorsque ceux-ci partagent au moins une variable. On peut retrouver *Bottom* en concaténant chaque locale, que l'on dénote par $Bottom = flat(locales(Bottom))$. La fonction *flat* retourne la représentation à plat d'une liste de locales. On définit comme *disjoints par variables* deux motifs qui ne partagent aucune variable entre eux, on a alors que deux motifs inclus dans deux locales disjointes sont disjoints par variables.

Exemple 6.11. *Soit* $Bottom = [a(X), b(X), c(X), c(Y), d(Y)]$. *L'ensemble des locales de Bottom est* $\{\{a(X), b(X), c(X)\}, \{c(Y), d(Y)\}\}$. *Les motifs* $[a(X), b(X)]$ *et* $[c(Y), d(Y)]$ *sont disjoints par variables.*

On bénéficie alors de l'équation 6.1 qui est vérifiée tant que m et q sont disjoints par variables. Pour exemple, on considère le cas sans variables : les locales sont alors des singletons, et donc l'équation 6.1 est vérifiée pour toute paire de motifs. L'algorithme 9 que l'on propose prend parti de la structure de *Bottom* en locales pour calculer l'ensemble des motifs subset-minimaux qui ont une couverture nulle dans $\mathcal{U}^{notc}$, et qui contiennent le motif subset-minimal m. L'algorithme étend m avec des motifs de locales qui n'ont pas encore été explorées, permettant de satisfaire l'équation 6.1. Pour ce faire, l'algorithme considère une nouvelle locale HLK et construit ligne 9 l'ensemble L_j de tous les motifs subset-minimaux par rapport à la couverture de m et qui ont une couverture strictement plus petite. Dans la preuve ci-dessous nous montrons que les motifs $m_j \in L_j$ subset-minimaux fournissent les seuls candidats $[m, m_j]$ pour des motifs subset-minimaux par rapport à $\mathcal{U}^{notc}$ qui étendent m avec des motifs de HLK. Nous ne détaillons pas dans cette section la fonction *subsetMinimal* qui vérifie la subset-minimalité du motif $[m, m_j]$ de taille n en comparant sa couverture avec celle de ses sous-motifs directs de taille $(n-1)$. Cette fonction est détaillée dans l'annexe C. L'exemple suivant illustre l'algorithme 9.

Exemple 6.12. *Considérons le cas décrit dans les exemples 6.10 et 6.11. L'algorithme considère tout d'abord le motif vide* [ ] *avec l'ensemble complet des locales* $\{\{a(X), b(X), c(X)\}, \{c(Y), d(Y)\}\}$. *Il sélectionne la première locale* $HLK = \{a(X), b(X), c(X)\}$ *qui possède une couverture nulle. Il construit ensuite l'ensemble des motifs subset-minimaux de HLK avec une couverture strictement inférieure à* $\{o_1\}$, *c'est-à-dire* $\{[a(X), b(X)], [b(X), c(X)]\}$. M_2 *explore alors* $m = \emptyset$ *le reste des locales, c'est-à-dire ici le singleton* $\{\{c(Y), d(Y)\}\}$. *L'appel récursif ne produit aucun résultat, comme* $[c(Y), d(Y)]$ *a la même couverture que le motif* [ ]. *L'algorithme retourne alors l'ensemble* $G = \{[a(X), b(X)], [b(X), c(X)]\}$.

Plus formellement, on considère maintenant une définition et certaines propriétés regroupées dans la proposition 6.6. On dit que deux motifs a et b sont *disjoints* lorsque leur intersection est nulle. Soit $LL(m)$ le plus grand sous-ensemble de locales dont les locales

Algorithme 9 $minGenRel(m, ResLK, NCE, \mathcal{U}^{notc})$

Entrée: m un motif subset-minimal par rapport à $\mathcal{U}^{notc}$, $ResLK$ un ensemble de locales telles que $m \cap \mathit{flat}(ResLK) = \emptyset$, NCE l'ensemble des observations de $\mathcal{U}^{notc}$ couvertes par m.

Sortie: Retourne l'ensemble des motifs subset-minimaux de $m \cup \mathit{flat}(ResLK)$ avec une couverture nulle dans $\mathcal{U}^{notc}$ et qui contiennent tous m.

1: **si** $cov(\mathit{flat}(ResLK), NCE) \neq \emptyset$ **alors retourne** $\emptyset$
2: **fin si**
3: **si** $NCE = \emptyset$ **alors retourne** $\{m\}$
4: **fin si**
5: **si** $ResLK = \emptyset$ **alors retourne** $\emptyset$
6: **fin si**
7: $HLK \leftarrow head(ResLK)$ ▷ HLK est la locale LK avec la plus petite couverture $cov([m, LK], \mathcal{U}^{notc})$
8: $TLLK \leftarrow tail(ResLK)$
9: $L_j \leftarrow minGenL(HLK, NCE)$ ▷ L'ensemble des subset-minimaux de HLK par rapport à NCE et avec une couverture $\neq NCE$
10: $M_1 \leftarrow \emptyset$
11: **pour** tout $m_j \in L_j$ **faire**
12: $NNCE \leftarrow cov(m_j, NCE)$ ▷ $cov([m, m_j], \mathcal{U}^{notc})$
13: **si** $subsetMinimal([m, m_j], NNCE, \mathcal{U}^{notc})$ **alors** ▷ $[m, m_j]$ est subset-minimal par rapport à $\mathcal{U}^{notc}$
14: $M_1 \leftarrow M_1 \cup minGenRel([m, m_j], TLLK, NNCE, \mathcal{U}^{notc})$
15: **fin si**
16: **fin pour**
17: $M_2 \leftarrow minGenRel(m, TLLK, NCE, \mathcal{U}^{notc})$
18: **retourne** $concatenate(M_1, M_2)$

intersectent le motif m, alors lorsque $LL(a) \cap LL(b) = \emptyset$, a et b sont dits locale-disjoints et sont par conséquent à la fois disjoints et disjoints par variable.

Proposition 6.6. *Soit a, b deux motifs et E un ensemble d'observations.*

1. *Si a et b sont locale-disjoints, alors $cov([a, b], E) = cov(b, cov(a, E))$.*
2. *Si a et b sont locale-disjoints, alors le fait que a ne soit pas subset-minimal par rapport à E implique que $[a, b]$ ne l'est pas non plus.*
3. *Si a et b sont locale-disjoints, alors le fait que a ne soit pas subset-minimal par rapport à $cov(a, E)$ implique que $[a, b]$ ne soit pas subset-minimal par rapport à E.*

On peut maintenant établir que l'appel $minGenRel([\,], locales(Bottom), \mathcal{U}^{notc}, \mathcal{U}^{notc})$ retourne l'ensemble des motifs subset-minimaux avec une couverture nulle sur $\mathcal{U}^{notc}$, en tant que conséquence du résultat suivant :

Proposition 6.7. *Soit L un ensemble de littéraux, Loc la liste de ses locales, $ResLK$ un suffixe de Loc et m un motif subset-minimal par rapport à $\mathcal{U}^{notc}$ tel que $[m \cap flat(ResLK)] = \emptyset$.*

$minGenRel(m, resLK, cov(m, \mathcal{U}^{notc}), \mathcal{U}^{notc})$ retourne alors sans doublons à un renommage de variables près l'ensemble des motifs dans $[m, flat(ResLK)]$ qui contiennent chacun m et qui sont chacun subset-minimaux par rapport à $\mathcal{U}^{notc}$ et ont une couverture nulle dans $\mathcal{U}^{notc}$.

Démonstration. Lorsqu'on parle de la subset-minimalité ou de la couverture d'un motif sans référence à un ensemble d'observations, nous faisons référence à la couverture de ce motif par rapport à l'ensemble d'observations $\mathcal{U}^{notc}$.

Les cas terminaux. À la ligne 1 l'algorithme retourne l'ensemble vide quand $[m, flat(ResLK)]$ n'a pas une couverture nulle, étant donné que tout sous-motif de $[m, flat(ResLK)]$ contenant m a une couverture nulle. Ceci est effectué afin d'économiser en calculs, selon la proposition 6.6-1 en ne considérant que les observations de $NCE = \mathcal{U}^{notc}$. À la ligne 3, l'algorithme retourne $\{m\}$ quand m possède une couverture nulle étant donné qu'aucun $m' \supset m$ ne peut être subset-minimal. La ligne 5 concerne le cas où m n'est ni subset-minimal, ni ne peut être spécialisé : l'ensemble des solutions est donc vide.

Les cas non terminaux. Nous supposons que l'algorithme est correct pour des appels avec $ResLK' = tail(ResLK)$ et un motif subset-minimal $m' = [m, m'']$ où $m'' \subseteq LK = head(ResLK)$, incluant lorsque $m' = m$. Notons que m'' et m sont par définition disjoint et disjoint par variable.

Les lignes 7-16 construisent l'ensemble M_1 des motifs subset-minimaux qui contiennent m et au moins un littéral de la première locale HLK avant de calculer à la ligne 17 l'ensemble M_2 des motifs subset-minimaux contenant m, mais aucun littéral de HLK.

M_2 est obtenu par un appel récursif à $minGenRel$ avec le motif subset-minimal $m' = m$ et $ResLK' = tail(ResLK)$ qui est considéré correct. À propos de M_1, l'algorithme considère à la ligne 13 les motifs $[m, m_j]$ avec $m_j \in HLK$ qui sont subset-minimaux, et fait pour chacun d'eux un appel récursif qui retourne l'ensemble M_j des motifs subset-minimaux contenant $[m, m_j]$ et joint ces M_j pour construire M_1. Pour considérer tous

les $[m, m_j]$ subset-minimaux, l'algorithme calcule premièrement ligne 9 l'ensemble L_j des motifs de *HLK* qui sont subset-minimaux par rapport à la couverture de m et avec une couverture strictement plus petite que m. D'après la proposition 6.6-3, seuls les m_j dans L_j sont candidats pour être les motifs subset-minimaux étendant m avec les motifs $m'' \subseteq HLK$.

Finalement, on rappelle que chaque appel récursif retourne une liste sans doublons, c'est-à-dire un ensemble. Par conséquent, M_2 n'a pas de doublons, de même que tout appel contribuant à construire M_1. Comme M_1 ne fait que de grandir ligne 14 en utilisant un opérateur d'union par ensemble, il n'y a pas de doublons dans M_1. Le résultat final est obtenu par la concaténation de M_1 et M_2. Comme par définition l'intersection de M_1 et M_2 est vide, l'ensemble retourné est sans doublons, à un renommage de variables près.

Terminaison. Chaque appel de *minGenRel* effectue un nombre fini d'appels récursifs, chacun avec un ensemble plus petit de locales que la fonction appelante. Ces appels finissent quand *ResLK* est vide. □

L'algorithme 10 concerne le cas général où l'ensemble des littéraux n'est pas divisé en locales. Cet algorithme est utilisé dans notre contexte quand l'ensemble de littéraux est une unique locale, plus précisément lorsqu'il existe une unique composante connexe par variables des littéraux dans *Bottom*. L'algorithme appelle une fonction *minSubsets* implémentée dans l'algorithme 13 décrit dans l'annexe C qui ensuite vérifie la subset-minimalité en utilisant la fonction *subsetMinimal* implémentée dans l'algorithme 14 également en annexe.

Algorithme 10 *minGenL*(L, E)

Entrée: L : un ensemble de littéraux dans *Bottom*, E : un ensemble d'observations
Sortie: retourne l'ensemble des motifs subset-minimaux de L par rapport à E excepté le motif vide [ ]
1: **retourne** *minSubsets*([ ], E, L, E, *True*, *True*) \ [ ]

6.4.3 Calcul des explications communes leq-minimales

Selon l'algorithme 7, pour obtenir l'ensemble M des leq-mces accessibles à partir de *Bottom*, nous avons besoin de commencer à partir des subset-mces dans G afin d'atteindre un point fixe sur les explications qui sont à la fois maximales selon $\preceq_i$ et qui sont subset-minimales. Pour ce faire, nous avons besoin (i) une fonction *selectMaxInst* telle que *selectMaxInst*(G) sélectionne dans une liste G de motifs ceux qui sont maximaux selon $\preceq_i$, (ii) une fonction permettant d'extraire les subset-mces d'une explication commune : on utilise pour ça la fonction *minGenRel* (voir algorithme 9), (iii) une fonction *maxInst*, qui retourne les instanciations maximales d'une subset-mce qui sont des instance-Mces pour O et (iv) une fonction de point fixe *fixedPoint* implémentée dans l'algorithme 11 qui, étant donné une subset-mce e, retourne les leq-mces accessibles depuis e. La fonction *maxInst* est est implémentée dans l'algorithme 15 en annexe C et est illustrée dans l'exemple ci-dessous.

Exemple 6.13. *Soit* $m = [p(X,Y)]$, $O = \{o_1, o_2\}$ *avec* $o_1 = p(1,2), p(2,1), p(1,1)$ *et* $o_2 = \{p(1,2), p(2,3)\}$. *maxInst*$([p(X,Y)], \{X,Y\}, O)$ *sélectionne* X *avec* $\{\theta = X/1, \theta = X/2, \dots\}$, *tel que* $m\theta \preceq_\theta o = o_1$.

- $[p(1,Y)]$ *couvre* O, *alors* $\{p(1,2)\}$ *est retourné par maxInst*$([p(1,Y)], \{Y\}, O)$
- $[p(2,Y)]$ *couvre* O, *alors* $\{p(2,Y)\}$ *est retourné par maxInst*$([p(2,Y)], \{Y\}, O)$
- $[p(Y,Y)]$ *ne couvre pas* O

Le dernier appel maxInst$([p(X,Y)], \{Y\}, O)$ *retourne* $\{[p(X,2)]\}$ *qui résulte en* $Max = \{[p(1,2)], [p(2,Y)], [p(X,2)]\}$ *desquels* $\{[p(1,2)], [p(2,Y)]\}$ *sont sélectionnés.*

L'algorithme 11 considère une subset-mce e et cherche les instance-Mces e' telles que $e \preceq_i e'$. La fonction *vars* renvoie l'ensemble des variables d'un motif. Quand e est sa propre instance-Mce, c'est une leq-mce et est retournée en tant que singleton. Autrement, pour chacune de ses instances-mce l'algorithme cherche les subset-mces qu'elle contient. Si m_i est une subset-mce, alors elle est ajoutée à la liste courante L des leq-mces accessibles à partir de m, et pour chaque subset-mce m_s contenue dans m_i *fixedPoint* retourne les leq-mces accessibles à partir de m_s et les ajoute à L.

Algorithme 11 *fixedPoint*$(m, O, \mathcal{U}^{notc})$

Entrée: m une subset-mce pour O par rapport à $\mathcal{U}^{notc}$
Sortie: retourne l'ensemble L des leq-mces accessibles depuis m ▷
M : explications communes maximales selon $\preceq_i$, S : subset-mces, L : leq-mces $M \leftarrow maxInst(m, vars(m), O)$
1: **si** $M = \{m\}$ **alors retourne** $\{m\}$ ▷ m à la fois $\preceq_i$-maximal et subset-minimal
2: **fin si**
3: $L \leftarrow \emptyset$
4: **pour** tout $m_i \in M$ **faire**
5: $S \leftarrow S \cup minGenRel([\,], locales(m_i), \mathcal{U}_{notc}, \mathcal{U}_{notc})$
6: **si** $S = \{m_i\}$ **alors** $L \leftarrow L \cup \{m_i\}$ ▷ m_i à la fois $\preceq_i$-maximal et subset-minimal
7: **sinon**
8: **pour** tout $m_s \in S$ **faire** $L \leftarrow L \cup fixedPoint(m_s, O, \mathcal{U}^{notc})$ ▷ les explications communes ajoutées sont des leq-mces
9: **fin pour**
10: **fin si**
11: **fin pour**
12: **retourne** L

Exemple 6.14. *On revient à l'exemple 6.6 de la section 6.3.5 et on simule* $fixedPoint(m, O, \mathcal{U}^{notc})$ *où* $m = [p(X,Y,Z), q(X,Y,Z)]$ *est une subset-mce.* m *contient une seule instance-Mce différente de* m *et donc on a une unique itération de la boucle externe,* $m_i = [p(X,Y,a), q(X,Y,a)]$ *qui contient l'ensemble des subset-mces* $S = \{[p(X,Y,a)], [q(X,Y,a)]\}$ *qui est retourné par l'algorithme.*

6.5 Sélection d'explications diverses

Le nombre de leq-mces peut être très élevé, et il est courant de retrouver les mêmes prédicats dans plusieurs leq-mces. Pour éviter de présenter des explications redondantes entre elles à un humain, nous proposons d'en sélectionner un sous-ensemble qui soit diversifié. Dans cette section, nous proposons une méthode de sélection d'explications diverses selon une mesure d'intérêt. Nous définissons cette mesure d'intérêt et proposons une mesure de similarité entre motifs, inspirée de travaux antérieurs et nécessaire pour mesurer la diversité des explications.

6.5.1 Similarité entre motifs relationnels

Nous définissons ci-dessous un score de similarité entre deux motifs relationnels. Nous requérons que ce score soit calculé rapidement, car il sera utilisé pour comparer un grand nombre de motifs. Pour cela, nous utilisons la fonction de similarité $s(x, y)$ proposée dans [Fer+09] qui repose sur la comparaison du nombre de caractéristiques partagées et non partagées entre deux motifs. Elle est définie comme suit :

$$s(x, y) = 0.5\frac{l+1}{l+m+2} + 0.5\frac{l+1}{l+n+2} \tag{6.2}$$

où

— l est le nombre de caractéristiques partagées entre x et y,

— m est le nombre de caractéristiques de y qui ne sont pas dans x,

— n est le nombre de caractéristiques de x qui ne sont pas dans y.

Il convient maintenant de déterminer ce qu'on considère comme une caractéristique. Dans notre cas, nous avons trois types de caractéristiques. Les *caractéristiques-constantes*, capturant la position et la valeur des constantes dans chaque littéral d'un motif. Deux caractéristiques-constantes sont partagées par deux motifs si elles ont la même position et la même valeur dans les deux motifs. Les *caractéristiques-variables*, capturant le chemin de chaque variable dans un motif, c'est-à-dire la séquence des prédicats et leur position dans lesquels elles apparaissent. Deux caractéristiques-variables sont partagées par deux motifs si elles ont le même chemin. Enfin, les *caractéristiques-prédicats*, capturant les prédicats présents dans le motif. Deux caractéristiques-prédicats sont partagées par deux motifs si elles ont le même prédicat.

Définition 6.5 (Caractéristiques d'un motif)**.** *Soit m un motif,*

— *Une caractéristique-constante $p.v.i$ veut dire que le prédicat p a la constante v à la position i ;*

— *Une caractéristique-variable $(p_1.i_1, \ldots, p_n.i_n)$ veut dire que la variable apparaît dans m dans le prédicat p_k à la position i_k pour tout $k \in 1 \ldots n$;*

— *Une caractéristique-prédicat p veut dire que le prédicat p apparaît dans m.*

Exemple 6.15. *Soit $m = [a(1), b(2, 3), c(X), d(X, Y)]$ un motif. Les caractéristiques de m sont :*

— *l'ensemble des caractéristiques-constantes* $\{a.1.1, b.2.1, b.3.2\}$;

— *l'ensemble des caractéristiques-variables* $\{(c.1, d.1)$,$d.2\}$;

— *l'ensemble des caractéristiques-prédicats* $\{a, b, c, d\}$.

Afin de calculer $s(x, y)$, on doit calculer les éléments l, m et n de l'équation 6.2. Pour cela, on considère séparément la contribution de chaque type de caractéristique (Constante, Variable, Prédicat) à l, m et n et on associe un poids γ aux prédicats. Ce poids sert à augmenter l'importance du fait que deux prédicats sont partagés par deux motifs dans le calcul de la similarité. De ce fait, on a $l = l_C + l_V + \gamma l_P$, $m = m_C + m_V + \gamma m_P$ et $n = n_C + n_V + \gamma n_P$, où par exemple l_C est le nombre de caractéristiques-constantes partagées par x et y.

Nous présentons un certain nombre d'exemples de la fonction de similarité s dans le tableau 6.1 avec $\gamma = 1$.

x	y	$s(x, y)$
$[p(X), q(X)]$	$[p(X), q(X)]$	0.8
$[p(X), q(X)]$	$[p(Y), q(Y)]$	0.8
$[p(X), q(X)]$	$[p(X), q(X), r(X)]$	0.675
$[p(X), q(X), r(X)]$	$[p(X), q(X), s(X)]$	0.57
$[p(X, a), q(X, a)]$	$[p(X, a), s(Y, b)]$	0.57
$[p(a), q(a)]$	$[p(b), q(b)]$	0.375
$[p(X), q(X)]$	$[r(Y), s(Y)]$	0.2

TABLE 6.1 – Exemples de similarité entre motifs

6.5.2 Algorithme de sélection

Soit M l'ensemble des explications communes leq-minimales. Nous proposons de sélectionner un ensemble d'explications diverses en utilisant la similarité entre motifs. Pour cela, nous utilisons l'algorithme $g\beta$ décrit dans [SSB20] afin de sélectionner un sous-ensemble $M' \subseteq M$ tel que pour toute paire de motifs $(x, y) \in M'$, $s(x, y) < \beta$. L'algorithme maximise (approximativement) la somme de l'intérêt individuel des motifs sélectionnés dans M' selon une mesure d'intérêt g. L'algorithme est glouton, il ne requiert pas la valeur de l'intérêt d'un motif $g(x)$, mais requiert un ordre des éléments de M selon leur intérêt décroissant. L'algorithme 12 détaille le fonctionnement de $g\beta$.

6.6 Expérimentations

6.6.1 Cadre de nos explications

Nous nous intéressons ici à la phase du jeu de la carte, dans la situation où le déclarant (*Noo*) a remporté la phase des enchères. La partie se déroule en plis, dans lesquels chaque joueur joue une de ses cartes. Le joueur qui a posé la plus grande carte remporte le pli et c'est à lui de commencer le pli suivant. Le contrat étant remporté par *South*, c'est donc

Algorithme 12 $g\beta(M, s, \beta, g)$

Entrée: Ensemble de leq-mces M, fonction de similarité s, seuil de similarité β, fonction d'intérêt g
Sortie: M' : Ensemble d'explications diverses

```
// M est ordonné selon g de manière décroissante
M' ← ∅ ;
tant que M ≠ ∅ faire
    Trouver le premier élément x de M tel que s(x, y) < β pour tout y ∈ M'
    si x a été trouvé alors
        M' ← M' ∪ {x}
    sinon
        M ← ∅
    fin si
fin tant que
retourne M'
```

à *West* d'entamer le premier pli. Nous nous plaçons dans un cadre restreint du bridge dans lequel il n'y a que des cartes d'une seule couleur (13 cartes numérotées de 2 à 14) réparties chez tous les joueurs dans une *donne*, et où le nombre de cartes par joueur n'est pas restreint. Nous nous plaçons du point de vue du déclarant, dont le but est de remporter le plus grand nombre de plis possibles.

Notre joueur artificiel *Noo* se place dans la peau du déclarant (NS) et cherche à maximiser le nombre de plis remportés. Il connait les mains de l'équipe de la défense (EW), les enchères sont terminées et on est dans la phase du jeu de la carte. *Noo* doit choisir dans un état du jeu une carte à jouer à Nord et à Sud à chaque pli

Le travail qui suit est motivé par le scénario suivant : Nous appelons notre joueur artificiel *Noo*, et son interlocuteur *M. X*. *M. X* demande des explications sur les actions qu'effectue *Noo* dans le jeu. On considère un scénario simple où *Noo* connaît les cartes de *West* et *East*. Il connaît également le modèle de jeu de *West* et *East*, tous les deux ont le même modèle de jeu qui est déterministe, c'est-à-dire qu'il n'y a qu'une réponse possible pour le défenseur à chaque état du jeu. Le modèle du défenseur est décrit dans l'Annexe D.

Quand *Noo* doit effectuer une action, il résout un MDP : *Noo* calcule pour chaque distribution *East-West* la récompense maximale $q(s, a)$ associée à chaque état-action (état s, action a) partant de l'état du jeu auquel on s'intéresse. Le déclarant maximise le nombre de plis qu'il gagne en jouant des suites d'actions, formant des trajectoires de jeu, qui sont optimales, c'est-à-dire qui maximisent $q(s, a)$. Le scénario de l'interaction d'un interlocuteur avec *Noo* se déroule comme suit :

1. *Noo* choisit une action optimale a dans l'état courant s maximisant $q(s, a)$;
2. *M. X* demande comment l'action a mène à une récompense maximale cumulée ;
3. *Noo* construit un modèle à base de règles logiques D pour l'optimalité des trajectoires et utilise D pour construire des groupes de trajectoires optimales partageant des explications communes.

4. *Noo* propose un ensemble d'explications leq-minimales afin de répondre à la requête ;
5. *Noo* joue l'action a ce qui l'amène à un nouvel état et le scénario reprend à l'étape 1.

Nous utiliserons dorénavant les termes *North* et *South* pour désigner le déclarant, mais il convient de rappeler que c'est un seul joueur artificiel qui joue les deux rôles. Les deux défenseurs sont *West* et *East*.

6.6.2 Protocole expérimental

Les données dont nous disposons sont issues d'un simulateur de parties, développé dans le cadre d'un stage de M2 à NukkAI, qui permet de générer toutes les parties possibles étant donné une distribution des cartes. Le programme permet d'obtenir en sortie un ensemble d'états-actions comprenant différentes informations : les mains des joueurs, l'historique des cartes jouées, la liste des cartes jouées dans le pli courant et l'action correspondante à l'état-action.

Pour chaque état, on connaît la récompense maximale attendue (en nombre de plis gagnés) pour le déclarant en fonction de l'action choisie. La récompense maximale finale est une q-value pour chaque état-action, qui permet de déterminer si l'action est optimale ou non. Nous cherchons à expliquer l'optimalité d'une action a_0 dans un état donné s_0. On considère qu'une action a est optimale dans un état s si la q-value pour l'état-action est égale au maximum du nombre de plis que le déclarant peut remporter à partir de cet état s.

Construction des observations

Les observations que nous utilisons afin d'expliquer l'optimalité de (a_0, s_0) sont l'ensemble des trajectoires partant de (s_0, a_0).

Une *trajectoire de jeu* est une séquence $p = s_0 a_0 \dots s_t a_t \dots s_n a_n$ où s_t est l'état du jeu observé au temps t et a_t est l'action jouée au temps t pour progresser vers l'état s_{t+1}. Le vocabulaire que l'on utilise contient des prédicats qui pour la plupart contiennent un argument temporel : l'atome est vrai ou faux à un instant t de la trajectoire. Pour une représentation plus compacte, l'argument temporel de certains prédicats est un intervalle de temps $[t_1, t_2]$ signifiant que l'atome est vrai à tous les instants de l'intervalle et faux en dehors. Le littéral positif $a([b, e])$ est donc vrai quand a est vrai pour tout $t \in [b, e]$ et faux au temps $b - 1$ et $e + 1$.

Exemple 6.16. *Considérons une trajectoire où a est vrai aux temps 1,2,3,5,6 et b est vrai aux temps 2,3,4,6,8,9. La valeur de vérité des deux atomes le long de la trajectoire est notée* $a([1, 3]), a([5, 6])$ *et* $b([2, 4]), b([6, 6]), b([8, 9])$.

Le temps t représente le moment où *North* ou *South* doit jouer et s_t est l'état courant du jeu au temps t. Ce qui veut dire que chaque pli est associé à deux pas de temps.

Exemple 6.17. *Considérons une trajectoire qui commence de la façon suivante : W8 (t1) N12 E6 (t2) S2. North joue le 12 (la reine) en* t_1 *avec une récompense nulle* $r(t_1, 12) = 0$.

Ensuite, South joue le 2 en t_2 avec une récompense $r(s_2, 2) = 1$, car North remporte le pli.

Remarquons qu'à chaque temps, c'est au déclarant de jouer.

Parmi les trajectoires partant de (s_0, a_0), on identifie les trajectoires optimales qui sont les séquences d'états actions $s_0a_0 \dots s_ta_t \dots s_na_n$ tels que pour tout $t \in 1 \dots n$, $q(s_t, a_t) = q(a_0, s_0)$. Les trajectoires non optimales sont celles qui ne vérifient pas cette condition. L'ensemble des trajectoires forme alors un arbre de trajectoires où chaque nœud est un état où le déclarant doit jouer une carte, et chaque branche est une action jouée par le déclarant et la réponse de la défense à cette action. Un exemple d'arbre de trajectoires est illustré dans la figure 6.2, et sera détaillé dans la section suivante. Le code de la construction des trajectoires a été réalisé en PYTHON.

Les observations sont exprimées en langage PROLOG par des faits, dont les symboles de prédicats sont tirés du vocabulaire décrit dans le vadémécum en annexe E. Les observations sont décrites dans des fichiers PROLOG au format adapté au programme NUKLEAR qui s'exécute dans l'environnement SWI-PROLOG.

Travaux associés sur les descriptions temporelles. En logique, la description temporelle des événements est un sujet très étudié. Les bases ont été développées par Allen [All83] qui a défini un ensemble de relations temporelles entre les intervalles de temps permettant de les comparer deux à deux, appelée *algèbre d'intervalles d'Allen*. Ces relations peuvent être représentées par des prédicats binaires dont les arguments sont soit des intervalles de temps, soit des événements se déroulant à un instant précis. Les prédicats des relations introduites das [All83] sont les suivantes :

- *before*(x, y) : si x se termine avant que y ne commence : *before*$([1, 3], [5, 6])$.
- *meets*(x, y) : si x se termine au moment où y commence : *meets*$([1, 3], [3, 5])$.
- *overlaps*(x, y) : si x et y se chevauchent : *overlaps*$([1, 3], [2, 4])$.
- *during*(x, y) : si x est entièrement inclus dans y : *during*$([2, 3], [1, 4])$.
- *starts*(x, y) : si x commence au moment où y commence et se termine avant que y ne se termine : *starts*$([1, 3], [1, 4])$.
- *finishes*(x, y) : si x se termine au moment où y se termine : *finishes*$([1, 3], [2, 3])$.
- *equals*(x, y) : si x et y sont identiques : *equals*$([1, 3], [1, 3])$.

Nous avons utilisé une restriction de l'algèbre d'Allen dans le vocabulaire utilisé pour décrire les observations, car les relations temporelles entre les actions du déclarant et de la défense sont souvent utilisées par les joueurs humains pour expliquer le déroulement d'une trajectoire : *« mon action est optimale, car je jouerai la Reine après que l'adversaire ait joué le Roi dans un pli précédent, puis je jouerai l'As et cela va me permettre de remporter deux plis »*. Le vocabulaire utilisé dans nos expérimentations permet d'exprimer certaines de ces relations temporelles. La relation *meets* est exprimée dans un motif lorsque la borne supérieure d'un intervalle du motif est une variable qui est partagée avec la borne inférieure d'un autre intervalle : $a([1, X]), b([X, 4])$. La relation *starts* peut être également exprimée dans un motif lorsque la borne inférieure d'un intervalle du motif est une variable qui est partagée avec la borne inférieure d'un autre intervalle : $a([X, 3]), b([X, 4])$. La relation

finishes peut être exprimée dans un motif lorsque la borne supérieure d'un intervalle du motif est une variable qui est partagée avec la borne supérieure d'un autre intervalle : $a([1, X]), b([2, X])$. Enfin, la relation *equals* peut être exprimée dans un motif lorsque deux intervalles du motif ont les mêmes variables pour les bornes inférieures et supérieures : $a([X, Y]), b([X, Y])$. Nous avons des prédicats « ponctuels » (voir annexe E) qui permettent de décrire des événements se déroulant à un instant précis. Les temps de ces prédicats peuvent se lier par une variable à des bornes d'intervalle, qui permet d'exprimer des choses qui se passent au début ou à la fin d'un intervalle.

Pour exprimer les autres relations temporelles, il faut pouvoir comparer les bornes d'intervalles deux à deux. Dans notre vocabulaire, nous disposons de deux prédicats : *geq/2* et *leq/2* permettant de comparer deux temps et définis respectivement comme $geq(X, Y)$ ssi $X \geq Y$ et $leq(X, Y)$ ssi $X \leq Y$ (on suppose alors que les relations ne sont pas strictes). Ainsi, on peut exprimer la relation *before* de la façon suivante : $a([1, X]), b([Y, 4]), leq(X, Y)$. On peut également exprimer la relation *overlaps* de la façon suivante : $a([1, X]), b([Y, 5]), geq(X, Y)$. Enfin, on peut exprimer la relation *during*, de la façon suivante : $a([X, Y]), b([Z, W]), geq(X, Z), leq(Y, W)$. Cependant, la comparaison des bornes d'intervalles deux à deux est une opération coûteuse dans le cadre de notre approche, car elle nécessite de comparer chaque paire de bornes deux à deux, ce qui retirerait l'intérêt de la décomposition de *Bottom* en locales, car on aurait une seule grande locale et rendrait le calcul des explications leq-minimales impossible. L'utilisation de l'algèbre d'Allen dans notre approche nécessiterait donc une adaptation de notre approche, ou bien d'introduire des contraintes supplémentaires pour limiter le nombre de comparaisons de bornes d'intervalles.

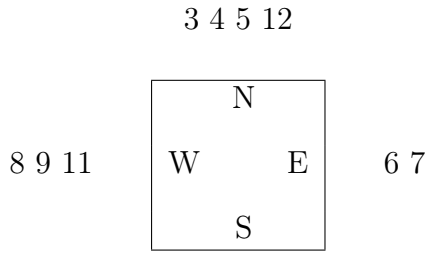
Situations de jeu

Dans nos expérimentations, nous nous sommes intéressé à deux situations de jeu issues de deux distributions initiales de cartes différentes. Les distributions initiales des cartes pour les deux situations qui nous intéressent sont illustrées dans les figures 6.1a et 6.1b. Les mains de *North* et *South* sont identiques dans les deux distributions. Ce qui diffère, c'est que dans la première distribution, le 11 est dans la main de *West* tandis que dans la seconde, le 11 est dans la main de *East*. Cette différence a été choisie pour illustrer les différences d'explications que l'on peut obtenir dans les deux cas, étant donné que l'optimalité des actions de *Noo* dépend essentiellement de l'emplacement du 11.

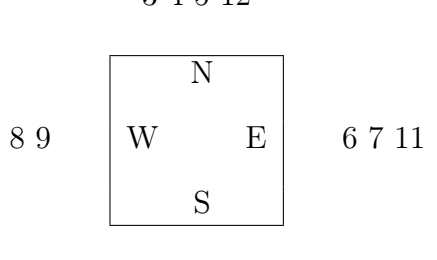
On rappelle que dans les deux cas d'étude qui nous intéressent, *West* est le premier à jouer une carte. Dans les deux cas, on va s'intéresser à expliquer l'optimalité d'une action de *North* qui suit l'action de *West*, mais il est important de noter que notre approche s'applique à n'importe quel état du jeu. Notons que pour les deux situations qui nous intéressent, le temps s'arrête en t_7, qui est le moment où la défense n'a plus de cartes en main et donc le déclarant remporte le reste des plis.

Pour chacune des deux situations, on effectue les étapes suivantes :

1. On génère les trajectoires partant de l'état-action que l'on veut expliquer ;
2. Nous utilisons le programme NukLear pour construire un classifieur D à partir des trajectoires de jeu : les règles de D couvrent chacune un sous-ensemble des



(a) Donne n°1 (W_{11}), où le 11 est dans la main de *West*



(b) Donne n°2 (E_{11}), où le 11 est dans la main de *East*

FIGURE 6.1 – Donnes pour l'expérience

trajectoires optimales et rejettent toutes les trajectoires non optimales. D est le même classifieur que le classifieur D utilisé dans la section 6.3 ;

3. Pour chaque règle de D, on cherche les leq-mces associées à l'ensemble des observations couvertes par la règle ;
4. On sélectionne un sous-ensemble des leq-mces qui soit diversifié et intéressant en utilisant l'algorithme de sélection $g\beta$.

6.6.3 Expérimentations avec la donne n°1 : le 11 est à *West*

Paramètres de l'expérience :

— **Nom** : W_{11}.

— **Distribution des cartes** : le 11 est dans la main de *West*.

— **Action optimale à expliquer** : jouer la carte 3 par *North* au temps t_1.

— **Nombre de trajectoires** : 144.

— **Nombre de trajectoires optimales** : 40.

— **Nombre de trajectoires non-optimales** : 104.

3 4 5 12

N

8 9 11 W E 6 7

S

2 10 13 14

On s'intéresse dans cette expérience, à la situation où le 11 est dans la main de *West* et où *West* joue la carte 8 au premier coup. C'est donc à *North* de jouer la carte suivante, et on choisit d'expliquer pourquoi l'action « jouer la carte 3 par *North* » est optimale. Pour cela, on génère les trajectoires partant de l'état-action (s, a) où s est l'état du jeu où *West* a joué la carte 8 et a est l'action jouer la carte 3 par *North*, noté N3. Les trajectoires forment un *arbre de trajectoires*. L'arbre des trajectoires partant de l'état-action N3 pour la donne n°1 est représenté dans la Figure 6.2. Dans cette figure, chaque nœud représente un état du jeu et chaque branche représente une action jouée par *North* ou *South*. Les nœuds sont étiquetés par le temps t_s auquel les actions du déclarant se déroulent dans la trajectoire dans l'état s, ce qui veut dire que chaque pli est associé à deux pas de temps. On remarque que le temps s'arrête en t_7, qui est le moment où la défense n'a plus de cartes en main et donc le déclarant remporte le reste des plis. Les actions optimales de la part du déclarant sont indiquées en noir, et les actions non optimales sont indiquées en rouge et leurs fils ne sont pas affichés. Cet arbre de trajectoires est compact, c'est-à-dire que chaque branche peut représenter plusieurs actions différentes considérées comme équivalentes. Deux actions sont considérées comme équivalentes si elles correspondent à

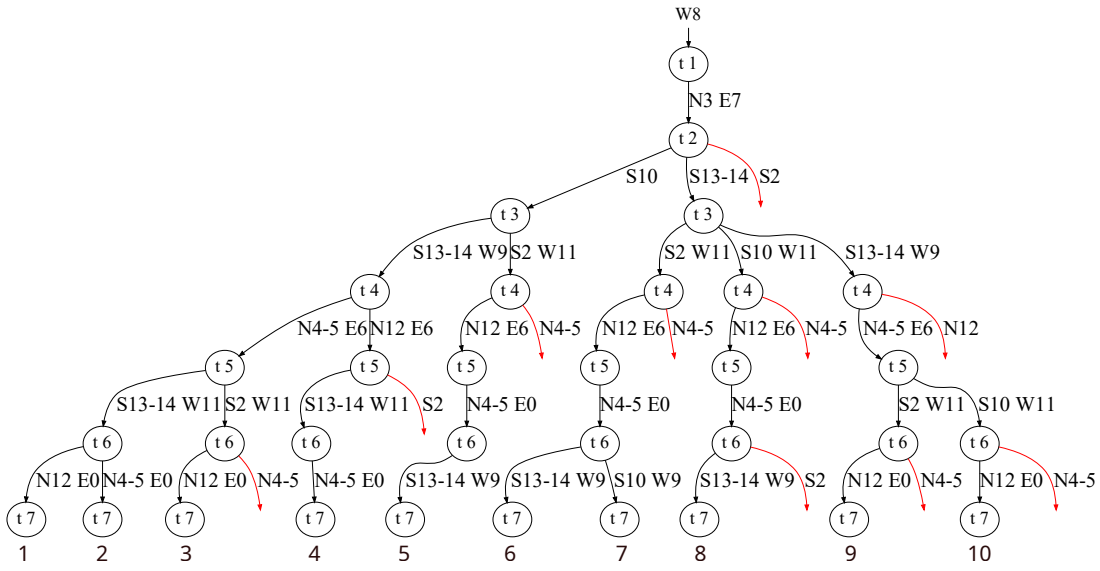


FIGURE 6.2 – Arbre de trajectoires partant de N3 pour la donne n°1

deux cartes consécutives dans la main du joueur. On dit que deux cartes $a < a'$ sont consécutives dans la main d'un joueur si dans s il n'y aucune carte a'' dans la main d'un autre joueur ou jouée dans le pli courant telle que $a < a'' < a'$. Les cartes consécutives sont alors interchangeables, car elles ont le même impact sur le nombre de plis remportés à la fin de la trajectoire. Il y a donc plusieurs trajectoires qui se terminent dans chaque feuille de l'arbre numérotée de 1 à 10.

Construction du classifieur

On exécute le programme NUKLEAR sur les 144 trajectoires partant de l'état-action N3, avec en exemples positifs les 40 trajectoires optimales qui forment $\mathcal{U}^{opt}$ et en exemples négatifs les 104 trajectoires non optimales qui forment $\mathcal{U}^{notOpt}$. Le programme NUKLEAR génère un classifieur D composé de 4 règles qui sont les suivantes :

$$opt \leftarrow action(12, 6), nbSmallCards(1, Player, [1, 3]). \tag{6.1}$$

$$opt \leftarrow playSmallestCard(Card, south, 3), \\ willTakeTrick(12, north, Time). \tag{6.2}$$

$$opt \leftarrow nbHonors(1, Player, [4, 5]). \tag{6.3}$$

$$opt \leftarrow nbThreats(2, Pair, 0, [7, 7]). \tag{6.4}$$

Les feuilles couvertes par chaque règle sont décrites dans le tableau 6.2. Remarquons que deux règles peuvent couvrir les mêmes feuilles, ici les règles 6.1 et 6.3 couvrent la feuille 1. On précise également que chacune de ces règles ne couvre aucune trajectoire de $\mathcal{U}^{notOpt}$. On note $\mathcal{U}_r^{opt} \in \mathcal{U}^{opt}$ l'ensemble des trajectoires couvertes par la règle r. Pour chaque ensemble $\mathcal{U}_r^{opt}$, on applique alors l'algorithme 8 de la construction de *Bottom*.

Règle	Feuilles couvertes de l'arbre	# trajectoires couvertes de $\mathcal{U}^{opt}$
6.1	1, 3, 9, 10	16
6.2	5, 6, 7	12
6.3	1, 2, 4	12
6.4	8	4

TABLE 6.2 – Feuilles couvertes par chaque règle du classifieur

Règle	Taille *Bottom*	# Locales	# subset-mces	# leq-mces	# Sélectionnées ($g\beta$)
6.1	279	50	1884	162	4
6.2	253	56	975	304	7
6.3	319	63	2162	386	9
6.4	395	63	5215	686	7

TABLE 6.3 – Statistiques sur les *Bottom* et les explications pour les groupes de trajectoires pour la donne W_{11}

Construction de la *lgg* et extraction des explications leq-minimales

Pour chaque ensemble $\mathcal{U}_r^{opt}$, on applique l'algorithme 8 qui approxime la *lgg* exacte et définit l'espace de recherche des explications leq-minimales.

Comme vu dans la section 6.4.1, le calcul de la *lgg* pour un ensemble d'observations O a un coût trop élevé et peut être très longue, en $\mathcal{O}(C^n)$ où C est la taille en nombre de littéraux de la plus grande observation et $n = |O|$. Dans notre cas, les observations ont en moyenne 257.1 littéraux, ce qui rend le calcul de la *lgg* exacte impossible. La figure 6.3 montre le nombre d'occurrences des prédicats par trajectoire pour la donne W_{11}. Les prédicats sont listés sur l'axe des abscisses, ceux qui sont grisés sont ceux qui ont été retirés de l'expérience pour des raisons variées (prédicats trop discriminants, prédicats non pertinents, prédicats dont le nombre d'occurrences est trop élevé, prédicats qui complexifient la tâche d'explication, etc.). En noir, ce sont les prédicats qui ont été gardés pour l'apprentissage du classifieur D, mais également pour la construction de *Bottom*. On remarque que certains prédicats ont un nombre d'occurrences très élevé, ce qui rend le calcul de la *lgg* exacte impossible.

Dans le tableau 6.3, on présente les statistiques sur les *Bottom* et l'extraction des explications leq-minimales pour chaque groupe de trajectoires couvertes par les règles du classifieur. La colonne *Taille Bottom* indique la taille de *Bottom* en nombre de littéraux. La colonne # Locales indique le nombre de locales de *Bottom*, la colonne # subset-minimales

Règle	Calcul de *Bottom*	Calcul des subset-minimaux	Calcul des leq-minimaux
6.1	59	444	53
6.2	110	321	149
6.3	177	1887	480
6.4	474	966	1297

TABLE 6.4 – Temps d'exécution des différents algorithmes pour la donne W_{11} (en secondes)

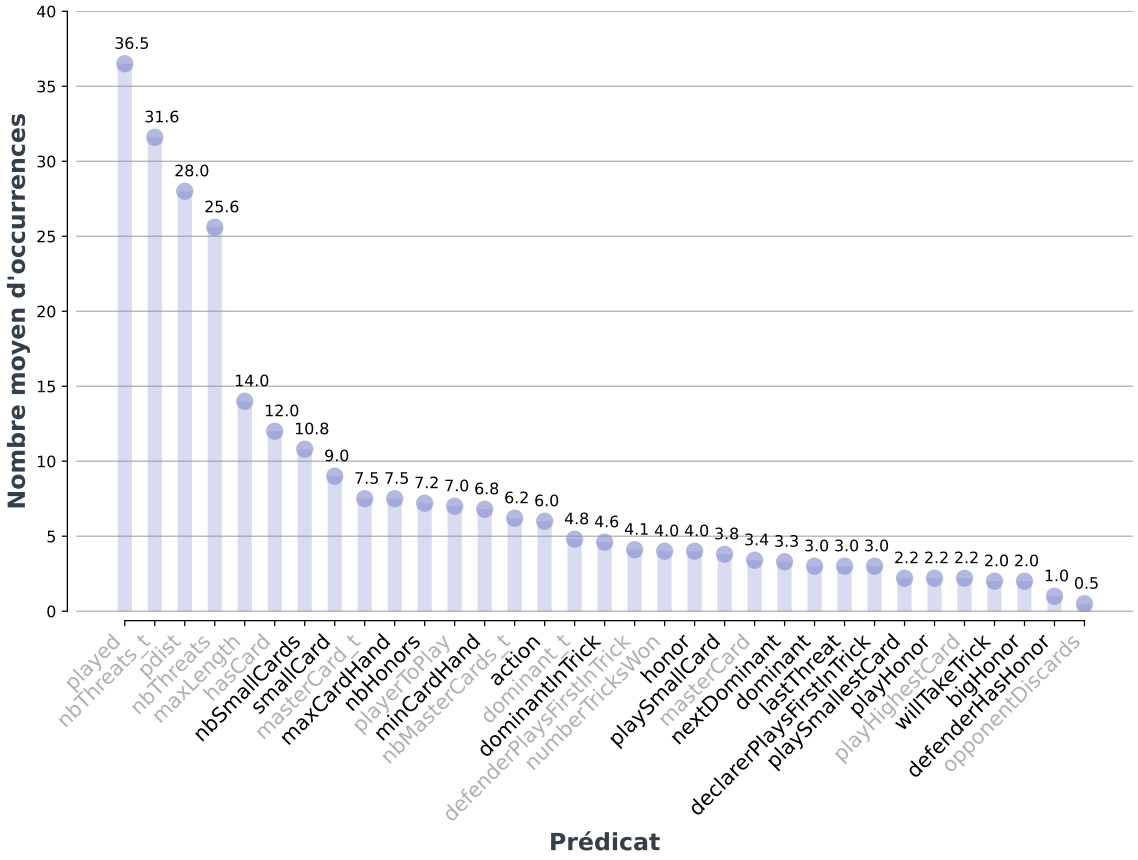


FIGURE 6.3 – Nombre moyen d'occurrences des prédicats par trajectoire pour la donne W_{11}. Les prédicats grisés ont été retirés de l'expérience.

indique le nombre d'explications subset-minimales, la colonne # leq-minimales indique le nombre d'explications leq-minimales et la colonne # Sélectionnées ($g\beta$) indique le nombre d'explications sélectionnées par l'algorithme de sélection $g\beta$, dont le détail est donné dans la section suivante. La taille des subset-mces est limitée à 3, la limite de similarité β est fixée à 0.35 et γ à 3.

Un commentaire général est que le nombre de leq-mces est bien plus petit que le nombre de subset-mces, ce qui est contre-intuitif, car par définition une subset-mce est soit une leq-mce, ou permet d'accéder à au moins une leq-mce. Par exemple pour le groupe de la règle 6.3, parmi les 386 leq-mces, 349 d'entre elles sont aussi des susbet-mces en sortie de l'algorithme *minGenRel*, c'est-à-dire qu'il y a uniquement 37 leq-mces qui ont été accédées à partir d'instanciations strictes et suppression de littéraux des 2162-349 = 1813 subset-mces restantes. Clairement, beaucoup de subset-mces accèdent aux mêmes leq-mces.

Considérons maintenant la perte de variables à l'intérieur des leq-mces par rapport aux subset-mces. Nous avons observé 567 occurrences de variables dans les 386 leq-mces alors qu'il y avait 6818 occurrences de variables dans les subset-mces. Cela veut dire que le nombre d'occurrences par motif est approximativement deux fois plus petit dans les leq-mces. Cependant, on a toujours de nombreuses variables dans les leq-mces. Finalement, dans l'ordre de préférence que nous utilisons pour la sélection des leq-mces, nous préférons les motifs instanciés, ce qui résulte en un faible nombre de variables dans les leq-mces sélectionnées.

Pour montrer le comportement de l'instanciation, nous présentons ici deux subset-mces de la règle 6.3 qui accèdent à une même leq-mces, les subset-mces :

— $nbHonors(0, east, [T_D, T_C]), nbHonors(1, south, [4, T_B]), nbSmallCards(1, south, [3, T_C])$

— $action(13, T_D), nbHonors(1, south, [4, T_B]), playHonor(14, south, T_C)$

Accèdent toutes les deux à la leq-mce $m = nbHonors(1, south, [4, 5])$.

Par ailleurs, on a 55 subset-mces qui contiennent le littéral $nbHonors(1, south, [4, T])$ rendant accessible m à partir de toutes ces mêmes subset-mce, ce qui valide le commentaire initial du nombre élevé de subset-mces qui accèdent aux mêmes leq-mces.

Dans le tableau 6.4, on présente le temps d'exécution de l'algorithme 8 de la construction de *Bottom* pour chaque sous groupe d'observations.

Sélection des explications

Nous allons présenter ici la sélection des explications leq-minimales effectuée par l'algorithme $g\beta$. Pour rappel, l'algorithme $g\beta$ sélectionne un sous-ensemble des explications leq-minimales en minimisant la similarité entre les explications sélectionnées, étant donné une mesure d'intérêt g sur les explications. Nous avons ordonné récursivement les motifs étant donné plusieurs critères, ordonnés du plus intéressant au moins intéressant pour notre expérience :

1. Les motifs ayant le moins de variables ;
2. Les motifs ayant le degré maximum de ses variables le plus élevé ;
3. Les motifs ayant le moins de littéraux ;

4. Les motifs ordonnés par ordre lexicographique.

Pour ces expérimentations, nous avons fixé $\beta = 0.35$ et $\gamma = 3$. Le choix de la valeur de ces paramètres sera expliqué ultérieurement dans cette sous-section.

Sélection $g\beta$. Nous présentons ici les 9 motifs sélectionnés après l'application de l'algorithme $g\beta$ sur l'ensemble des explications leq-minimales de la règle 6.3. Les termes commençant par une majuscule sont des variables, les variables sont locales à chaque motif :

1. $nbHonors(1, south, [4, 5])$
2. $action(10, 2), maxCardHand(2, south, [6, 7])$
3. $lastThreat(10, dec, 11, [1, 2]), minCardHand(2, south, [1, 7])$
4. $nbSmallCards(0, east, [5, 7]), nbSmallCards(1, south, [3, 7])$
5. $dominant(Card, north, [6, 6]), playSmallCard(10, south, 2)$
6. $dominantInTrick(Card, south, 6), nextDominant(2, south, [6, 7])$
7. $playHonor(Card, south, 5), defenderHasHonor(11, west, [1, 5])$
8. $willTakeTrick(10, south, 2), willTakeTrick(Card, south, 5)$
9. $declarerPlaysFirstInTrick(south, 13, T_A), declarerPlaysFirstInTrick(south, 14, T_B)$

Soit M^* l'ensemble des motifs leq-minimaux ordonnés selon l'ordre d'intérêt énoncé plus haut. Premièrement, on peut remarquer que le premier motif sélectionné est la version instanciée de la règle 6.3. Ce motif est sélectionné car il a la première place dans M^*. Le second motif sélectionné est également le second motif de M^*, il est sélectionné car il a une similarité de $0.15 < 0.35$ avec le premier motif sélectionné. Après cela, on a dans M^* 5 motifs éliminés par l'algorithme $g\beta$ car ils ont au moins un prédicat en commun avec le second motif sélectionné. Pour l'exemple, voici deux motifs qui ont été éliminés : $action(10, 2), minCardHand(2, south, [1, 7])$ et $action(10, 2), nbHonors(0, south, [6, 7])$. Les deux motifs ont un prédicat en commun avec le second motif sélectionné, leur donnant une similarité de 0.69 et 0.65 respectivement avec ce dernier, ce qui montre un bon comportement de l'algorithme.

Explications des leq-mces sélectionnées

Nous reprenons ici deux motifs sélectionnés par l'algorithme $g\beta$ dans la liste ci-dessus et nous expliquons l'optimalité des trajectoires couvertes par ces motifs :

1. $nbHonors(1, south, [4, 5])$ est une instance close du corps de la règle 6.3. Ce motif dit qu'il y a exactement un honneur dans la main de *South* entre le temps t_4 et le temps t_5 (et un nombre différent en dehors de $[t_4, t_5]$). Ce qui veut dire que (i) *South* joue un honneur (la carte 13 ou 14) en t_3, au début du deuxième pli et (ii) *South* joue son deuxième honneur en t_5, au début du troisième pli. De (i) on infère que *South* remporte le premier pli (avec les cartes 6 et 7 *East* ne peut pas le remporter) et le deuxième pli (13 et 14 sont les cartes les plus hautes du jeu). De (ii) on infère que *South* remporte le troisième pli, montrant l'optimalité des trajectoires couvertes par ce motif.

2. $action(10, 2), maxCardHand(2, south, [6, 7])$. Le second littéral dit que *South* joue sa carte 10 au premier pli. Le premier littéral dit que *South* a toujours le 2 en main à la fin de la trajectoire. Du motif entier on infère que *South* a gagné le premier pli en jouant le 10 et a gagné les plis suivants en jouant ses deux honneurs.

La valeur que l'on donne aux explications dépend de leur objectif. Clairement, si l'objectif est de savoir uniquement quelle carte jouer, alors l'explication 2 est plus simple. En revanche, si le but est d'apprendre à raisonner sur les actions et leurs conséquences, c'est-à-dire progresser en tant que joueur, alors l'explication 1 est meilleure. Dans notre cas, nous avons choisi une mesure d'intérêt g qui désavantage beaucoup les motifs contenant des variables, ce qui a provoqué le fait de n'avoir qu'un seul motif avec variable dans la sélection. Mais l'ordre que l'on choisit pour la sélection des motifs est arbitraire et dépend de l'objectif de l'explication. La présence de variables peut être également une variable d'ajustement selon l'objectif de l'explication, en effet, les variables demandent en général plus d'efforts cognitifs pour être comprises.

Pour illustrer cela, on peut analyser l'explication 9. Cette explication dit que *South* est le premier de son équipe à jouer dans le pli sa carte 13 à un temps T_A et qu'il est le premier de son équipe à jouer dans le pli sa carte 14 à un temps T_B. Si *South* joue en premier dans un pli, c'est soit qu'il a gagné le pli précédent, soit que *East* a gagné le pli précédent. Or, *East* ne peut pas gagner le premier pli car il n'a aucune carte au dessus de celles déjà posées dans le pli. Donc, *South* a gagné le premier pli, et les variables T_A et T_B nous informent que l'ordre dans lequel *South* joue ses cartes 13 et 14 n'est pas important, et peu importe l'ordre, 13 et 14 étant les cartes les plus hautes du jeu, *South* remporte le deuxième et le troisième pli.

Choix du paramètre γ. En raison de la conception de l'ensemble des caractéristiques d'un motif, si deux littéraux ont le même symbole de prédicat et un nombre élevé d'arguments dont les valeurs diffèrent entre l'un et l'autre, le point commun entre les deux littéraux qui est le symbole de prédicat sera dilué dans tout le reste et les deux littéraux seront donc considérés comme très différents. Or, il n'est pas rare d'avoir deux littéraux de même symbole de prédicat, et dont les termes sont différents, mais dont l'interprétation est rigoureusement similaire. Par exemple, on considère deux motifs de taille 1 $m_1 = nbHonors(1, west, [1, 5])$ et $m_2 = nbHonors(0, west, [6, 7])$. Ces deux motifs ont exactement la même interprétation, qui dit que *west* joue son unique carte qui soit un honneur au temps 5. Nous présentons dans le tableau 6.5 l'effet de faire varier le paramètre γ sur la similarité entre les motifs m_1 et m_2, permettant d'augmenter le résultat de la similarité de deux motifs qui ont les mêmes symboles de prédicat. Grâce à cette expérience, nous avons choisi $\gamma = 3$ pour la suite de nos expérimentations.

Choix du paramètre β. Nous montrons ici une comparaison de l'effet de faire varier le paramètre β, qui correspond à la similarité maximale entre deux motifs sélectionnés, sur le nombre d'explications sélectionnées par l'algorithme $g\beta$. On utilise pour cela les motifs leq-minimaux obtenus à partir de la règle 6.3 et on leur applique la sélection $g\beta$ avec différentes valeurs de β. Les résultats sont présentés dans le tableau 6.6.

γ	$s(m_1, m_2)$
1	0.44
2	0.58
3	0.71
4	0.79

TABLE 6.5 – Effet de γ sur la similarité entre deux motifs avec le même prédicat

Pour nos expérimentations, nous choisissons $\beta = 0.35$ pour avoir un nombre raisonnable de motifs sélectionnés. Cependant, cette valeur peut être réglée par l'utilisateur et le nombre idéal de motifs sélectionnés peut dépendre de l'application.

β	# de motifs sélectionnés
0.35	9
0.4	9
0.5	13
0.6	19
0.8	138

TABLE 6.6 – Effet de β sur le nombre de motifs sélectionnés

Effets de β sur la sélection Ici, on fournit, pour les trois premiers motifs sélectionnés m, les deux motifs n_1 et n_2 les plus éloignés tels que $s(m, n) > \beta$, avec $\beta = 0.35$. Cela peut être vu comme les deux motifs qui ont été éliminés car leur similarité avec m est plus élevée que la limite β et qui sont les moins similaires à m. Ces deux motifs auraient été sélectionnés si la valeur de β était plus élevée, et si l'ordre initial était bénéfique.

Pour $m = nbHonors(1, south, [4, 5])$:

1. $n_1 = dominant(Card, north, [6, 6]), lastThreat(10, dec, 11, [1, 2]), nbHonors(0, west, [6, 7])$
   $s(m, n_1) = 0.43$
2. $n_2 = dominant(Card, north, [6, 7]), lastThreat(10, dec, 11, [1, 2]), nbHonors(0, west, [6, 7])$
   $s(m, n_2) = 0.43$

Pour $m = action(10, 2), maxCardHand(2, south, [6, 7])$:

1. $n_1 = minCardHand(2, south, [1, 7]), playSmallCard(10, south, 2)$
   $s(m, n_1) = 0.37$
2. $n_2 = minCardHand(2, south, [1, 7]), willTakeTrick(10, south, 2)$
   $s(m, n_2) = 0.37$

Pour $m = lastThreat(10, dec, 11, [1, 2]), minCardHand(2, south, [1, 7])$:

1. $n_1 = action(Card, 5), bigHonor(Card), minCardHand(11, west, [4, 5])$
   $s(m, n_1) = 0.41$
2. $n_2 = action(Card, 5), bigHonor(Card), minCardHand(6, east, [1, 4])$
   $s(m, n_2) = 0.41$

On peut voir que les motifs éliminés sont très proches du motif sélectionné à chaque étape. On remarque, pour le deuxième motif sélectionné, que même si les littéraux ne sont pas identiques, les motifs ont une interprétation très similaire. Plus précisément, $maxCardHand(2, south, [6, 7])$ et $minCardHand(2, south, [1, 7])$ ont la même interprétation qui est que la carte 2 n'est pas jouée par *South* jusqu'à la fin du jeu, et donc que *South* joue toutes ses cartes au-dessus du 2 durant les trois plis. De même pour $action(10, 2)$ qui a exactement la même interprétation que $willTakeTrick(10, south, 2)$ et $playSmallCard(10, south, 2)$, signifiant que *South* joue la carte 10 au temps 2, lui permettant de remporter le pli.

Nous avons fait le choix expérimental de $\beta = 0.35$ et $\gamma = 3$, car dans notre expérience et plus précisément pour notre vocabulaire, il a permis l'exclusion de motifs trop ressemblants aux motifs sélectionnés. Mais si l'on change de vocabulaire, ces valeurs de paramètres peuvent avoir un effet trop restrictif ou trop permissif selon la forme des prédicats. Par exemple, si les prédicats utilisé dans le vocabulaire alternatif sont tous d'arité 1, il conviendrait de diminuer la valeur de γ car l'effet de la dilution du nombre de prédicats en commun entre deux motifs par celui de leurs arguments en commun est moindre, et à l'inverse pour la même raison si l'arité des prédicats est plus élevée, il conviendrait d'augmenter la valuer de γ.

6.6.4 Expérimentations avec la donne n°2 : le 11 est à *East*

Dans ce qui suit, on s'intéresse à la situation où le 11 est dans la main de *East* et où *North* joue la carte 3 juste après que *West* ait joué la carte 8.

Paramètres de l'expérience :

- **Nom** : E_{11}.
- **Distribution des cartes** : le 11 est dans la main de *East*.
- **Action optimale à expliquer** : jouer la carte 3 par *North* au temps t_1.
- **Nombre de trajectoires** : 144.
- **Nombre de trajectoires optimales** : 40.
- **Nombre de trajectoires non-optimales** : 104.

3 4 5 12

N

8 9 W E 6 7 11

S

2 10 13 14

L'arbre des trajectoires est représenté dans la figure 6.4. Voici les trois règles apprises par le programme NukLear :

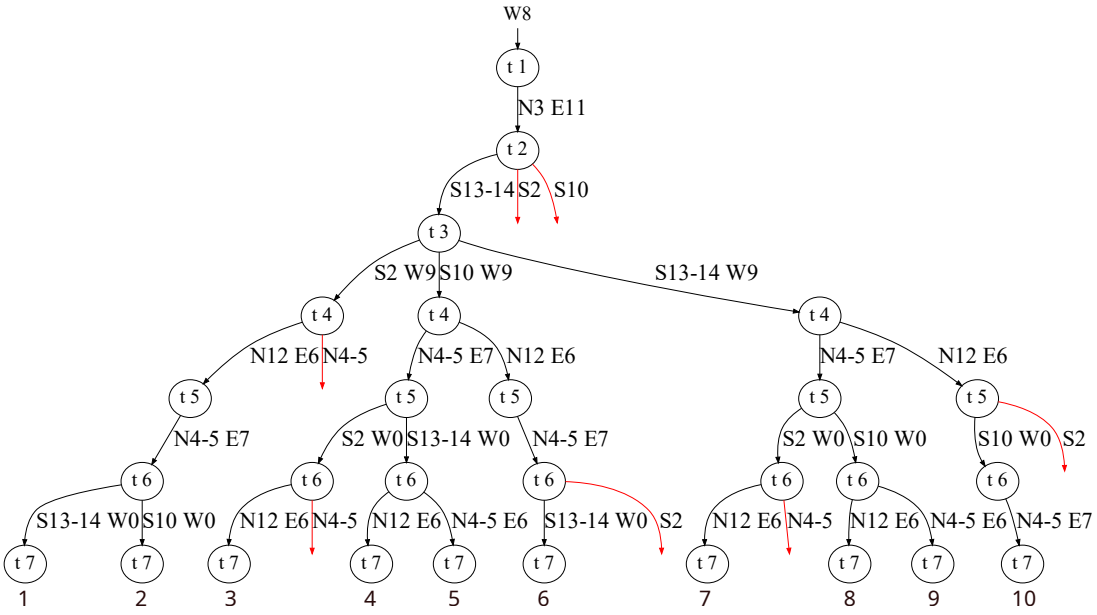


FIGURE 6.4 – Arbre de trajectoires partant de N3 pour la donne E_{11}

Règle	Feuilles couvertes de l'arbre	# trajectoires couvertes de $\mathcal{U}^{opt}$
6.5	1, 2, 10	12
6.6	3, 4, 7, 8	16
6.7	4, 5, 6, 8, 9, 10	24

TABLE 6.7 – Feuilles couvertes par chaque règle du classifieur pour la donne E_{11}

$$opt \leftarrow nextDominant(7, east, [5, 5]), dominant(10, Player, [T_A, T_B]). \quad (6.5)$$
$$opt \leftarrow maxCardHand(Card_A, east, [5, 6]), maxCardHand(Card_B, north, [7, 7]). \quad (6.6)$$
$$opt \leftarrow nextDominant(2, south, [T_A, T_B]), nbSmallCards(2, Player, [1, 4]) \quad (6.7)$$

Les trajectoires couvertes par chaque règle sont décrites dans le tableau 6.7. De la même manière que pour la donne W_{11}, on applique l'algorithme 8 pour chaque ensemble $\mathcal{U}_r^{opt}$, et on sélectionne un sous-ensemble des explications leq-minimales qui soit diversifié et intéressant en utilisant l'algorithme de sélection $g\beta$.

Le tableau 6.8 présente les statistiques sur les *Bottom* et l'extraction des explications leq-minimales pour chaque groupe de trajectoires couvertes par les règles du classifieur. Ici, nous prenons les mêmes paramètres que pour l'expérience W_{11}, c'est-à-dire une taille maximale des subset-mces fixée à 3, une limite de similarité $\beta = 0.35$ et un multiplicateur $\gamma = 3$.

Sélection $g\beta$

Ici, nous nous intéressons à la règle 6.5 en raison de son faible nombre de leq-mces (4). Nous montrons ici la mesure d'intérêt g utilisée par l'algorithme $g\beta$, en montrant

TABLE 6.8 – Statistiques sur les *Bottom* et les explications pour les groupes de trajectoires pour la donne E_{11}

Règle	Taille *Bottom*	# Locales	# subset-mces	# leq-mces	# Sélectionnées ($g\beta$)
6.5	285	29	14	4	1
6.6	321	42	1211	154	5
6.7	258	25	660	124	4

Règle	Calcul de *Bottom*	Calcul des subset-minimaux	Calcul des leq-minimaux
6.5	162	411	1
6.6	66	445	3
6.7	762	283	8

TABLE 6.9 – Temps d'exécution des différents algorithmes pour la donne W_{11} (en secondes)

les motifs leq-minimaux dans l'ordre g choisi. Cet ensemble ordonné est donné en entrée de l'algorithme $g\beta$. Les motifs sont ordonnés par ordre d'intérêt, du plus intéressant au moins intéressant, selon les critères énoncés plus haut :

1. $dominant(10, south, [5, T_A]), nextDominant(7, east, [5, 5])$
2. $dominant(10, south, [5, T_A]), nextDominant(Card_C, north, [6, T_C])$
3. $dominant(10, south, [5, T_A]), nbSmallCards(1, south, [T_B, T_C]), nextDominant(4, north, [T_B, T_B])$
4. $dominant(10, south, [5, T_A]), nbSmallCards(1, south, [T_B, T_C]), nextDominant(5, north, [T_B, T_B])$

Le seul motif sélectionné est le premier motif de la liste, car il a la première place dans la liste ordonnée. Les motifs suivants ont été éliminés, car ils ont au moins le littéral $dominant(10, south, [5, T])$ en commun avec le motif sélectionné, ce qui implique que tous sont trop similaires au premier motif pour être sélectionnés.

Explication d'une leq-mce non préférée

L'interprétation d'un motif relationnel peut parfois être complexe. Comme précisé précédemment, la présence de variables force un raisonnement plus poussé pour montrer l'optimalité des trajectoires couvertes par le motif, et cette difficulté peut être utilisée dans le cadre de l'apprentissage d'un élève. Pour montrer cette difficulté, nous allons dérouler les explications de la leq-mce la plus basse dans la liste ci-dessus, qui est le motif $dominant(10, south, [5, T_A]), nbSmallCards(1, south, [T_B, T_C]), nextDominant(5, north, [T_B, T_B])$. Afin de comprendre l'explication de ce motif, nous avons besoin de regarder plus en détail les trajectoires tombant dans les feuilles 1, 2 et 10 de l'arbre de trajectoires de la figure 6.4. Le motif décrit deux situations d'optimalité très différentes selon si on analyse les trajectoires couvertes par les feuilles 1 et 2 ou par la feuille 10, donc nous allons les traiter séparément.

1. **Feuilles 1 et 2** : Le motif couvre les cas où le déclarant joue son 2 au temps t_3. Le premier littéral $dominant(10, south, [5, T_A])$ dit que la carte 10 de *South* devient dominante au temps t_5, permettant d'inférer que *East* a joué son 11 et *North* a joué

sa carte 12 avant le temps t_4, les deux cartes empêchaient le 10 d'être dominante. Le 12 a été joué au deuxième pli, car dans notre expérience *North* a joué son 3, cela permet donc au déclarant de gagner le deuxième pli. $\mathit{nbSmallCards}(1, \mathit{south}, [T_B, T_C])$ nous indique que T_B correspond au temps t_4, car c'est le seul temps auquel *South* n'a plus qu'une petite carte (qui est 10), et nous indique donc que *South* a bien joué au temps t_3. Il reste à prouver que c'est *South* qui a joué en premier au temps t_3 et pas *East* : le fait que 5 devienne *nextDominant* à t_4 indique que *West* a joué sa dernière carte qui empêchait le 5 d'être *nextDominant*. C'est donc *West* qui joue au temps t_3, donc c'est *South* qui a joué en premier dans le deuxième pli, indiquant qu'il a gagné le premier pli. Pour le dernier pli, les cartes restantes dans la main de *South* sont toutes dominantes donc il va gagner le dernier pli.

2. **Feuille 10** : Le motif couvre le cas où *South* joue son 13 ou 14 au temps t_3. Le littéral $\mathit{dominant}(10, \mathit{south}, [5, T_A])$ indique de la même façon que pour les feuilles 1 et 2 que *East* a joué son 11 au premier pli et *North* a joué sa carte 12 au premier pli. $\mathit{nbSmallCards}(1, \mathit{south}, [T_B, T_C])$ nous indique ici que T_B correspond au temps t_6, car c'est le moment où *South* a joué juste avant son 10, lui laissant une unique petite carte en main qui est 2. On a alors l'instanciation $\mathit{nextDominant}(5, \mathit{north}, [t_6, t_6])$ avec T_B/t_6. Cela veut dire que 7 est passée dominante au temps t_6, indiquant que le 10 a été joué à t_5, permettant de gagner le troisième pli, mais cela indique également que *South* n'a plus ses cartes 13 et 14, indiquant qu'il les a jouées avant, et que donc il a emporté les deux premiers plis.

Comparaison des différents biais

Dans la section 6.4.1, nous avons présenté l'algorithme de construction de *Bottom*. Cet algorithme prend en paramètres un biais de langage, et notamment les types des arguments à l'intérieur des prédicats. Deux arguments dans deux littéraux différents de même type peuvent contenir une même variable s'il n'y a pas d'interdiction de lien sur ce type. De ce fait, des typages différents peuvent obtenir des résultats différents de *Bottom*. Dans nos expériences déroulées jusqu'ici, nous avons notamment fixé un biais où tous les arguments temporels étaient d'un unique type, il est donc possible d'avoir des bornes d'intervalle liées à des temps ponctuels par une variable : on peut avoir la leq-mce $m = \mathit{action}(2, T), \mathit{maxCardHand}(5, \mathit{north}, [5, T])$. En revanche, si l'on donne un type différent pour les bornes supérieures d'intervalle, inférieures d'intervalle et les temps ponctuels, alors la leq-mce m ne pourra pas apparaître.

Dans le tableau 6.10, nous avons comparé les tailles de *Bottom* pour deux biais différents, le biais $\mathcal{B}_{[time]}$ où les temps ponctuels et les bornes d'intervalle sont de même type, et le biais $\mathcal{B}_{[time,inf,sup]}$ où les temps ponctuels et les bornes d'intervalle sont de types différents. Ces résultats sont affichés pour chaque règle de la donne E_{11}. Dans le tableau 6.11 nous avons comparé le nombre d'apparitions des variables de temporelles dans les subset-mces pour les deux biais. Les bornes supérieures et inférieures d'intervalle sont respectivement notées *Inf* et *Sup*, et les temps ponctuels sont notés *Time*. Enfin, dans le tableau 6.12, nous avons comparé le nombre pour chaque type de variables temporelles dans les subset-mces pour les deux biais.

On remarque dans le tableau 6.10 que les tailles de *Bottom* sont à chaque fois plus

TABLE 6.10 – Tailles de *Bottom* en fonction du biais de langage pour la donne E_{11}

Règle	$\mathcal{B}_{[time]}$	$\mathcal{B}_{[time,inf,sup]}$
6.5	285	185
6.6	321	236
6.7	258	181

TABLE 6.11 – Nombre d'apparitions des variables de type temps dans les subset-mces en fonction du biais de langage pour la donne E_{11}

Règle	$Time_{[time]}$	$Time_{[time,inf,sup]}$	$Inf_{[time,inf,sup]}$	$Sup_{[time,inf,sup]}$
6.5	46	0	7	11
6.6	2548	46	801	633
6.7	1965	304	628	341

petites pour le biais $\mathcal{B}_{[time,inf,sup]}$ que pour le biais $\mathcal{B}_{[time]}$. Cela est dû au fait que typer des arguments avec un même type autorise des générations de littéraux liés par variables, là où des types différents ne le permettent pas. Le choix du biais de langage est donc crucial pour la construction de *Bottom* et peut influencer sa taille.

Concernant les apparitions des variables temporelles dans les subset-mces, on remarque dans le tableau 6.11 que les variables temporelles apparaissent beaucoup plus souvent dans les subset-mces pour le biais $\mathcal{B}_{[time]}$ que pour le biais $\mathcal{B}_{[time,inf,sup]}$. Même si l'on effectue l'addition des apparitions des variables de type *Time*, *Inf* et *Sup* pour le biais $\mathcal{B}_{[time,inf,sup]}$, on obtient un nombre d'apparitions de variables temporelles infrieur à celui du biais $\mathcal{B}_{[time]}$.

Enfin, dans le tableau 6.12, on remarque que le nombre de liens entre les variables temporelles est également plus élevé pour le biais $\mathcal{B}_{[time]}$ que pour le biais $\mathcal{B}_{[time,inf,sup]}$, de même que si l'on additionne les liens entre les variables temporelles, infrieures d'intervalle et supérieures d'intervalle pour le biais $\mathcal{B}_{[time,inf,sup]}$, on obtient un nombre de liens infrieur à celui du biais $\mathcal{B}_{[time]}$, excepté pour la règle 6.6.

6.6.5 Discussion sur la similarité entre les deux expériences

Pour faire nos expérimentations, nous avons choisi d'analyser deux variations simples d'une distribution de cartes, une où le 11 est dans la main de *West* et une où le 11 est dans la main de *East*. Ces situations peuvent mener à des issues de jeu différentes en fonction des actions du déclarant, mais ont tout de même quelques similarités. Dans cette section, nous allons étudier de façon plus détaillée ces similarités entre les deux expériences.

TABLE 6.12 – Nombre de liens entre les variables de type temps dans les subset-mces en fonction du biais de langage pour la donne E_{11}

Règle	$Time_{[time]}$	$Time_{[time,inf,sup]}$	$Inf_{[time,inf,sup]}$	$Sup_{[time,inf,sup]}$
6.5	11	0	3	0
6.6	321	14	210	138
6.7	477	20	146	24

TABLE 6.13 – Séquences d'actions du déclarant communes et leurs feuilles correspondantes pour chaque expérience.

	N3 S13-14 S2 (rouge)	N3 S13-14 S10 (vert)	N3 S13-14 S13-14 (bleu)
W_{11}	feuilles 6,7	feuille 8	feuilles 9,10
E_{11}	feuilles 1,2	feuille 6	feuilles 7,8

Similarités entre les deux expériences

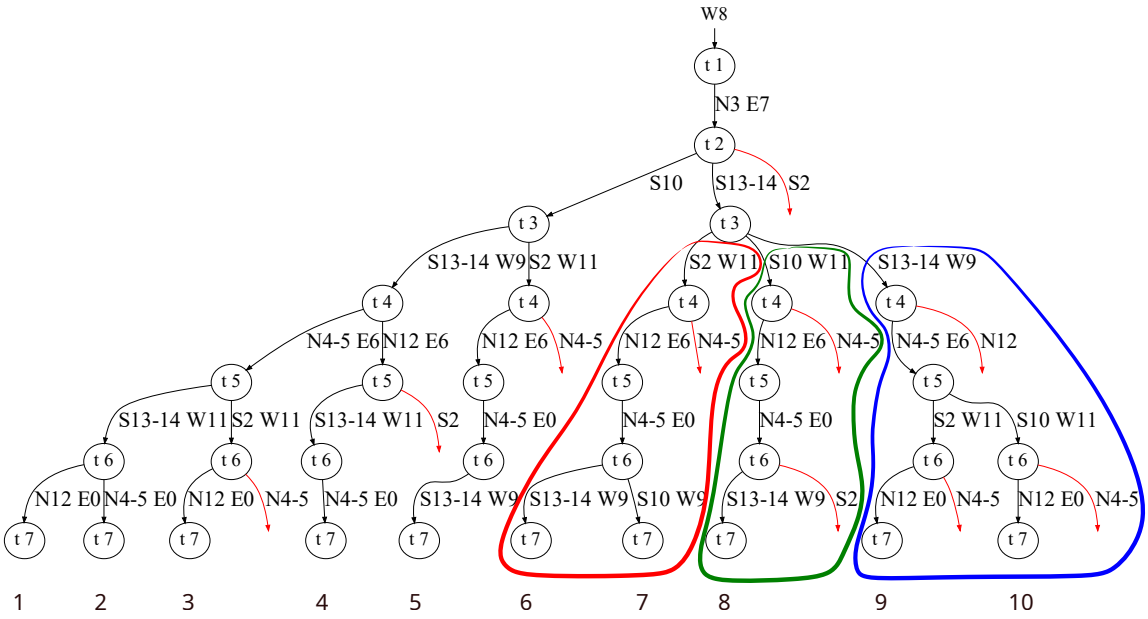
La différence principale entre les deux situations est le fait que la carte 11 soit jouée immédiatement par *East* au temps t_1 dans l'expérience E_{11}, obligeant *South* à jouer son 13 ou son 14 au temps t_2. Ce n'est pas le cas dans l'expérience W_{11}, où *South* peut jouer sa carte 10 au temps t_2. Cette différence principale implique également que le 11 soit éliminé du jeu dès le premier pli dans l'expérience E_{11}, ce qui rend plus « facile » de gagner les plis suivants pour le déclarant. On observe que quand on considère uniquement les actions du déclarant, on peut identifier 3 sous-arbres communs entre W_{11} et E_{11} où la même séquence d'actions de la part du déclarant mène à la victoire dans les deux expériences. Ces sous-arbres sont montrés dans les figures 6.5a et 6.5b. Deux sous-arbres entourés par la même couleur dans les deux figures correspondent à deux ensembles de séquences d'actions de la part du déclarant qui mènent à une récompense optimale dans les deux expériences.

Notons que toutes les trajectoires optimales de W_{11} partant de *South* qui joue son 13 ou 14 au temps t_2, c'est-à-dire les trajectoires optimales dans les sous-arbres rouge, vert et bleu de la figure 6.5a, sont également optimales dans E_{11}. Le tableau 6.13 montre les feuilles correspondant à chaque sous-arbre pour chaque expérience. Les sous-arbres sont identifiés par la séquence d'actions du déclarant menant à la racine du sous-arbre.

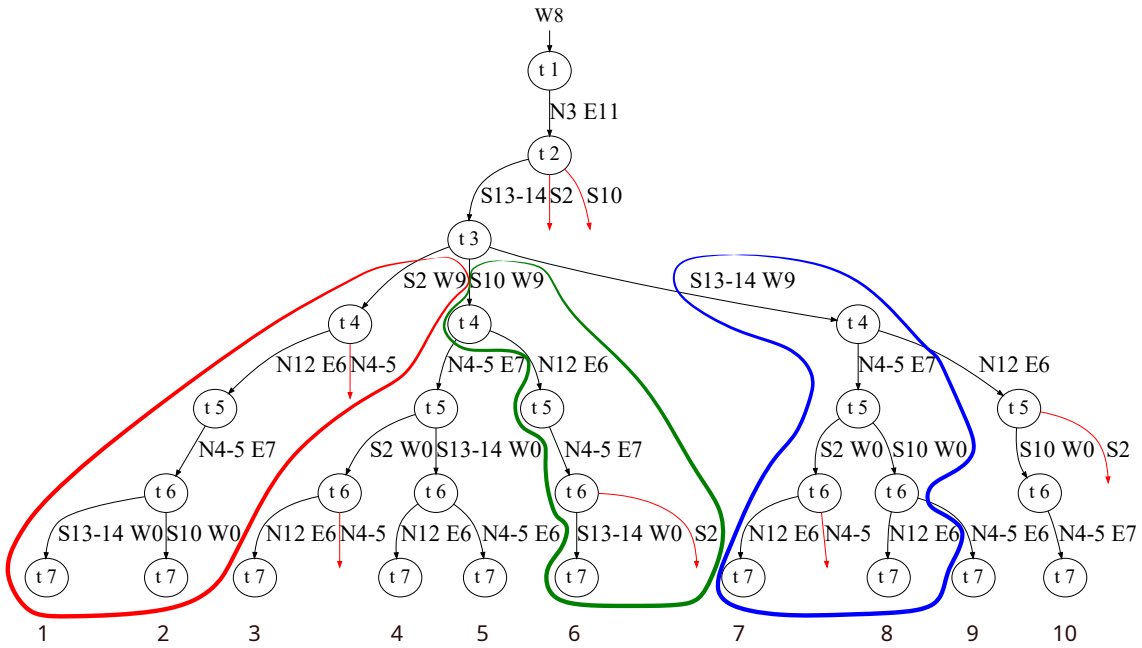
Le sous-arbre rouge, qui correspond aux trajectoires tombant dans les feuilles 6 et 7 pour W_{11} et dans les feuilles 1 et 2 pour E_{11}, est le groupe des trajectoires où *South* joue sa carte 2 au temps t_3. Notons que les séquences d'actions optimales et non optimales du déclarant dans ce sous-arbre sont exactement les mêmes pour W_{11} et E_{11}. Ceci est dû au fait que dans les deux cas, quand *South* joue sa carte 2 au temps t_3, il n'y a qu'une action possible pour *North* pour permettre au déclarant de gagner le second pli, qui est de jouer sa carte 12 au temps t_4.

Le sous-arbre vert, qui correspond aux trajectoires tombant dans la feuille 8 pour W_{11} et dans la feuille 6 pour E_{11}, est le groupe des trajectoires où *South* joue sa carte 10 au temps t_3. Dans ce cas, les actions optimales du déclarant pour W_{11} sont un sous-ensemble des actions optimales du déclarant pour E_{11}. Ceci est dû au fait que dans W_{11}, l'action de *North* au second ou au troisième pli est cruciale pour une situation qui n'arrive pas dans E_{11}. En fait, dans W_{11} *North* doit jouer son 12 au temps t_4 si *South* jour sa carte 10 au temps t_3, parce que sinon *West* gagnera le deuxième pli. Ce n'est pas le cas quand le 11 est dans la main de *East*, car dans ce cas *East* joue son 11 au temps t_1, annulant toute menace de la part de la carte 11 dans les plis suivants.

Le même raisonnement s'applique pour le sous-arbre bleu, qui correspond aux trajectoires tombant dans les feuilles 9 et 10 pour W_{11} et dans les feuilles 7 et 8 pour E_{11}. Les actions optimales du déclarant pour W_{11} sont un sous-ensemble des actions optimales du déclarant pour E_{11}. C'est également dû au fait que quand le 11 est dans la main de *West*,



(a) Identification des sous-arbres communs de la donne W_{11}



(b) Identification des sous-arbres communs de la donne E_{11}

FIGURE 6.5 – Identification des sous-arbres communs des deux expériences

l'action de *North* au temps t_4 est cruciale, il doit garder sa carte 12 jusqu'au troisième pli si *South* joue ses cartes 13 et 14 au premier et au second pli, et dans ce cas-là pour gagner le troisième pli, c'est *North* qui doit jouer sa carte 12, car *South* n'a plus de carte plus haute que le 11 chez *West*.

Explications communes

Le but de cette section est d'identifier les explications qui sont communes aux deux expériences. Nous étudions les motifs des sous-arbres rouges des deux expériences décrits plus haut. Pour W_{11}, ce sont les trajectoires tombant dans les feuilles 6 et 7, qui sont couverts par la règle 6.2, et pour E_{11}, ce sont les trajectoires tombant dans les feuilles 1 et 2, qui sont couverts par la règle 6.5. Pour cela, nous évaluons chaque leq-mce de la règle 6.2 (donc de W_{11}) sur les trajectoires tombant dans les feuilles 1 et 2 de E_{11} afin de trouver les explications leq-minimales issus de la règle 6.2 qui sont des explications communes pour les trajectoires tombant dans les feuilles 1 et 2 de E_{11}. Notons que la règle 6.5 couvre également la feuille 10 de W_{11}, mais n'est pas étudiée ici car elle n'est pas incluse dans le sous-arbre rouge. Notons également que nous ne testons pas la leq-minimalité d'une explication d'une donne par rapport aux observations de l'autre donne, nous ne testons que si l'explication est une explication commune.

Sur les 304 leq-mces de la règle 6.2, 8 sont corrects pour les trajectoires tombant dans les feuilles 1 et 2 de E_{11}. Ces motifs sont les suivants :

1. $\mathit{minCardHand}(6, \mathit{east}, [1,4]), \mathit{nbSmallCards}(1, \mathit{north}, [6,7]), \mathit{willTakeTrick}(\mathit{Card}, \mathit{south}, 6)$
2. $\mathit{minCardHand}(6, \mathit{east}, [1,4]), \mathit{nbSmallCards}(2, \mathit{north}, [2,5]), \mathit{willTakeTrick}(\mathit{Card}, \mathit{south}, 6)$
3. $\mathit{dominant}(\mathit{Card}, \mathit{south}, [T,6]), \mathit{minCardHand}(6, \mathit{east}, [1,4]), \mathit{nbSmallCards}(1, \mathit{north}, [6,7])$
4. $\mathit{dominant}(\mathit{Card}, \mathit{south}, [T,6]), \mathit{minCardHand}(6, \mathit{east}, [1,4]), \mathit{nbSmallCards}(2, \mathit{north}, [2,5])$
5. $\mathit{nbSmallCards}(1, \mathit{north}, [6,7]), \mathit{willTakeTrick}(\mathit{Card}_A, \mathit{south}, 6), \mathit{willTakeTrick}(\mathit{Card}_B, \mathit{south}, 2)$
6. $\mathit{nbSmallCards}(2, \mathit{north}, [2,5]), \mathit{willTakeTrick}(\mathit{Card}_A, \mathit{south}, 6), \mathit{willTakeTrick}(\mathit{Card}_B, \mathit{south}, 2)$
7. $\mathit{dominant}(\mathit{Card}_A, \mathit{south}, [T,6]), \mathit{nbSmallCards}(1, \mathit{north}, [6,7]), \mathit{willTakeTrick}(\mathit{Card}_B, \mathit{south}, 2)$
8. $\mathit{dominant}(\mathit{Card}_A, \mathit{south}, [T,6]), \mathit{nbSmallCards}(2, \mathit{north}, [2,5]), \mathit{willTakeTrick}(\mathit{Card}_B, \mathit{south}, 2)$

Le littéral ayant pour symbole de prédicat *nbSmallCards* de chaque motif a la même signification, qui est que *North* doit jouer une petite carte au temps t_5 et a une petite carte au temps t_6 jusqu'à la fin, ce qui veut dire que *North* a joué son 12, car ce n'est pas une petite carte et a donc gagné le second pli. Le littéral $\mathit{minCardHand}(6, \mathit{east}, [1,4])$ que l'on retrouve dans les motifs 1, 2, 3 et 4 dit que *East* joue sa plus petite carte qui est le 6 au temps t_4, et donc, que *East* n'a pas gagné le premier pli, donc c'est *South* qui a gagné le premier pli. $\mathit{dominant}(\mathit{Card}, \mathit{south}, [T,6])$ dit que *South* joue sa carte dominante au temps t_6, qui est le 10 ou le 13 ou 14, lui permettant de gagner le troisième pli.

Le tableau 6.14 montre le nombre d'explications communes entre les sous-arbres de W_{11} et E_{11}. Le tableau se lit de la façon suivante : il y a 8 explications leq-minimales de la règle couvrant les trajectoires du sous-arbre rouge de W_{11} qui sont également des explications pour les trajectoires du sous-arbre rouge de E_{11}.

Nous pouvons voir qu'il y a une variation assez importante du nombre de motifs entre deux sous-arbres communs. Ceci peut être expliqué par le fait que les règles qui

TABLE 6.14 – Nombre d'explications communes entre les sous-arbres de W_{11} et E_{11}

	N3 S13-14 S2 (rouge)	N3 S13-14 S10 (vert)	N3 S13-14 S13-14 (bleu)
W_{11}	8	149	5
E_{11}	0	1	70

couvrent les feuilles d'un sous-arbre dans une donne, peuvent couvrir des feuilles qui ne pas optimales dans le sous-arbre de l'autre donne. Par exemple, le sous-arbre vert chez W_{11} inclut la feuille 8, qui est la seule feuille couverte par la règle 6.2. Ainsi, les explications leq-minimales issues de cette règle on de grandes chances d'être des explications pour le sous-arbre vert de E_{11}, ce qui explique pourquoi il y a 149 explications issues de la règle 6.2 qui sont des explications pour le sous-arbre vert de E_{11}.

À l'inverse, pour le sous-arbre vert de E_{11}, la feuille 6 est couverte par la règle 6.7, qui couvre également les feuilles 4, 5, 8, 9 et 10 de l'arbre de E_{11}. Les feuilles 4, 5, 9 et 10 de E_{11} proviennent d'une action qui n'est pas optimale dans W_{11}. Dans E_{11} : (i) N4-5 au temps t_4 pour les feuilles 4 et 5, (ii) N12 au temps t_4 pour la feuille 9 et (iii) N12 au temps t_6 pour la feuille 10, sont toutes des actions non optimales dans W_{11}, ce qui implique que de nombreuses explications leq-minimales issues de la règle 6.7 ne sont pas des explications pour la feuille 8 de W_{11}, car elles ont de grandes chances de couvrir des actions non optimales dans W_{11}.

En conclusion de cette section, nous avons donc identifié que, malgré des différences structurelles entre les deux expériences, il est possible de trouver des explications communes entre elles. Ces explications communes représentent des façons de jouer qui sont gagnantes à la fois dans W_{11} et dans E_{11}, on peut dire qu'elles sont robustes à ces deux mondes. La présentation de telles explications peut être préférable par rapport à d'autres explications pour un élève qui apprendrait le bridge, car elles montrent des façons de jouer qui sont gagnantes dans des situations différentes.

Par ailleurs, cela donne des pistes intéressantes pour des recherches futures, où l'on veut trouver des explications dans le cas où l'on n'a pas toute l'information sur la distribution des cartes. Une description plus détaillée de ces perspectives est donnée dans la section 7.6.

6.7 Conclusion

Dans le but d'expliquer l'optimalité des décisions d'un agent artificiel jouant à un jeu de bridge simplifié, nous avons proposé dans ce chapitre une définition concept des explications communes minimales dans le cadre abductif, dont le but est d'expliquer l'étiquette attribuée par un classifieur à une observation. Ces explications sont communes à un ensemble d'observations qui partagent la même étiquette que l'observation à expliquer. Pour notre problème, ces observations sont l'ensemble des trajectoires de jeu optimales partant de l'action de l'agent artificiel à expliquer.

Nous avons proposé une formalisation de telles explications en logique du premier ordre, en exprimant tout d'abord nos observations comme étant des interprétations d'un langage du premier ordre et nos explications communes comme des motifs relationnels qui couvrent un sous-ensemble de ces explications.

Nous nous sommes placés dans le cadre des explications formelles décrites dans la section 5.3 de cette thèse. Nous avons adapté le cadre des explications abductives minimales [SCD18 ; DH20] à la logique du premier ordre en introduisant la notion d'explications communes subset-minimales, une propriété nécessaire pour la concision, étant l'une des quatre propriétés attendues d'une explication (voir section 5.1.2).

Toutefois, dans le cadre de la logique du premier ordre, des variables apparaissent dans les explications et peuvent rendre la compréhension du motif ardue dans certains cas. Par conséquent, nous avons défini les explications préférées comme étant les explications subset-minimales les plus instanciées possibles, les explications leq-minimales. Pour énumérer de telles explications, nous avons défini les explications communes leq-minimales d'un ensemble d'observations comme les sous-ensembles minimaux de la *lgg* de cet ensemble d'observations.

Tout ceci nous mène aux différentes contributions algorithmiques de ce chapitre pour l'énumération des explications leq-minimales : trois algorithmes principaux ont été présentés, (i) l'algorithme de calcul de l'approximation *Bottom* de la *lgg*, (ii) un algorithme d'extraction de motifs subset-minimaux en relationnel, et (iii) l'algorithme d'instanciation maximale des motifs subset-minimaux.

Ce travail a pour objectif principal de poser les bases d'une communication entre la machine et l'humain. Nous avons déterminé que présenter un grand nombre d'explications à un humain pouvait entraver la clarté, de ce fait la sélection d'un nombre restreint d'explications qui soient diverses a été une étape importante de ce chapitre, les différents résultats ont été montrés dans la section expérimentale.

Enfin, nous avons montré dans la section expérimentale qu'il pouvait y avoir des explications en commun entre deux mondes différents, posant les premières pierres des explications en monde partiellement observable, l'une des perspectives principales de ce travail.

Chapitre 7

Perspectives

Dans ce chapitre, nous présentons les perspectives de recherche pour les travaux développés dans les chapitres 4 et 6.

7.1 Extension des explications à un meilleur modèle du défenseur

Dans le cadre de notre travail dans le chapitre 6 sur les explications formelles abductives, nous avons montré que notre approche était adaptée à un modèle du défenseur basique que nous avons utilisé pour notre cas d'études. Cependant, les explications qui ont été analysées dans les expérimentations du chapitre 6 n'ont pas fait intervenir le modèle du défenseur en raison du faible nombre de prédicats concernant le défenseur dans notre vocabulaire. L'utilisation d'un vocabulaire alternatif plus centré sur le défenseur pourrait requérir d'utiliser le modèle du défenseur pour interpréter les explications. Par exemple, si on avait une règle dans le modèle du défenseur qui dit que le défenseur joue toujours sa plus petite carte s'il a perdu le pli précédent, sinon il joue sa plus grande carte, alors on pourrait obtenir des leq-mce contenant un littéral $playSmallestCard(Card, east, T)$ signifiant que le défenseur a joué sa plus petite carte au temps T, donc que le déclarant a gagné le pli précédent.

De ce fait, avoir un modèle du défenseur qui soit plus performant et non basé sur des règles dictées à la main paraît incontournable. Notre approche peut et doit s'appliquer à n'importe quel modèle du défenseur déterministe pour lui donner un caractère plus robuste, où les actions du défenseur sont plus complexes. Nous proposons par exemple d'étendre notre approche pour un défenseur qui utilise l'algorithme MinMax [Sha88], un algorithme de recherche de l'arbre de jeu qui permet de déterminer la meilleure action à jouer pour un joueur donné, en supposant que l'adversaire joue de manière optimale. La construction d'un MDP de la part du déclarant pour un tel défenseur est simple, il suffit que le déclarant considère que le défenseur joue de manière optimale et pour prédire le coup que le défenseur va jouer, il construit lui aussi un arbre de jeu en utilisant l'algorithme MinMax. Une fois que le déclarant a construit ce MDP, il peut utiliser notre approche pour expliquer l'optimalité de ses actions. L'arbre des trajectoires de jeu que le déclarant construit pour expliquer ses actions est alors un arbre de recherche MinMax.

7.2 Explications des trajectoires non optimales

Une autre extension possible de nos travaux est la génération d'explications abductives minimales de la non-optimalité des trajectoires de jeu. L'approche que nous avons proposée dans le chapitre 6 permet de donner des explications abductives pour les actions optimales du déclarant. Cependant, lors du processus d'apprentissage d'un élève, ou lors de l'analyse d'une partie, il est aussi important de comprendre pourquoi une action peut mener à une récompense non optimale. De nombreuses recherches s'intéressent à l'effet l'analyse d'exemples erronés sur l'apprentissage des mathématiques par les élèves [McL+12 ; Rus18 ; MAM15]. Cette analyse consiste à présenter à l'élève des fausses solution à un problème mathématique dans lesquelles une ou plusieurs étapes contiennent des erreurs. Toutes ces recherches ont montré un effet positif significatif sur les élèves pratiquant l'analyse des erreurs lors d'un test ultérieur sur un problème similaire. L'adaptation de notre approche à ce nouveau mode d'apprentissage pourrait permettre d'améliorer le processus d'apprentissage d'un élève qui utiliserait les explications que l'on génère pour apprendre à jouer au bridge.

Nos travaux sont trivialement adaptables pour donner des explications menant à une récompense non optimale. Pour ce faire, il suffit de considérer que les trajectoires que l'on cherche à expliquer sont les trajectoires menant à un état terminal qui n'est pas optimal.

Cet axe de recherche est une extension naturelle de notre approche, donnant un ensemble d'explications plus complet pour comprendre les actions du déclarant, mais fait également le lien avec les recherches sur la théorie de l'apprentissage humain.

7.3 Explications pour la planification

Les explications que nous avons proposées dans le chapitre 6 sont adaptées pour expliquer les actions d'un joueur de bridge dans le cadre d'une partie de bridge. Cependant, notre approche peut être étendue pour expliquer les actions d'un agent artificiel dans le cadre de la *planification*. En IA, la planification désigne des algorithmes qui produisent des plans d'exécution étant donné un certain objectif pour un agent dans un environnement. Formellement, une tâche de planification est définie par un tuple $\langle S, A, T, s_0, G\rangle$ où S est l'ensemble des états possibles, A est l'ensemble des actions possibles, T est la fonction de transition, s_0 est l'état initial et G est l'ensemble des états buts. Un plan est une séquence d'actions qui mène de l'état initial à un état but.

Le problème de la planification réside dans le fait qu'il peut exister un grand nombre de plans possibles pour atteindre un état but. Prenons exemple du « monde des blocs » [1], où un agent est face à une table avec des blocs empilés dans une configuration s et doit les réarranger par une suite d'actions dans une configuration objectif s^*. La figure 7.1 illustre le problème par un exemple et l'agent ne peut déplacer qu'un bloc à la fois. Naturellement, il y a un nombre élevé de plans lui permettant d'atteindre cet objectif. Le problème de planification est de trouver un plan optimal pour que l'agent atteigne la configuration objectif. On peut alors définir l'optimalité comme étant un plan tel que le nombre d'actions est inférieur à un certain seuil k. Les explications que l'on peut donner

1. `https://www.inf.ed.ac.uk/teaching/courses/ai2/module4/tutorials/tut4.pdf`

(a) Configuration initiale s (b) Configuration objectif s^*

FIGURE 7.1 – Illustration du problème du « monde des blocs »

par notre approche seront alors des explications communes aux plans optimaux, donc des séquences d'actions qui généralement mènent à l'objectif en un nombre d'actions inférieur à k.

Des travaux sur la planification ont déjà été réalisés dans le cadre des explications formelles [SCS23 ; SCS24] dont l'objectif est de trouver les actions minimales qui ont permis à un agent artificiel de résoudre un problème.

7.4 Évaluation des explications sur des utilisateurs réels

Dans le cadre de notre travail, nous avons expérimenté les explications formelles abductives sur des cas d'études de parties de bridge. Cependant, il est important de valider notre approche sur des utilisateurs réels pour évaluer la pertinence des explications que l'on génère. Pour cela, il est possible de mener des expérimentations avec des joueurs de bridge pour leur montrer les explications que l'on propose et leur demander de les évaluer. L'entreprise NUKKAI [2] qui finance cette thèse possède en son effectif des joueurs de bridge professionnels qui peuvent être sollicités pour évaluer les explications que l'on propose. L'élément clé de cette évaluation réside dans le protocole expérimental que l'on met en place pour évaluer les explications. Les différentes expériences suivantes peuvent être menées pour évaluer les explications que l'on propose :

— *Expérience 1 :* On montre à un joueur des règles qui sont fausses et des règles qui sont vraies, et on lui demande d'identifier les règles fausses. Cette expérience permet de valider la compréhension des explications que l'on propose. À cela, on peut ajouter des mesures de temps de réponse, du niveau du joueur, etc. afin de mesurer l'effet de ces paramètres sur la compréhension des explications.

— *Expérience 2 :* On montre à un joueur différentes explications d'un même ensemble de trajectoires et on lui demande de les évaluer sur un certain nombre de critères, que ce soit leur difficulté de compréhension, leur intérêt, etc.

2. `https://www.nukk.ai/`

Ce protocole, associé à une caractérisation des règles par différents attributs (nombre de littéraux, nombre de variables, prédicats utilisés, etc.) permettra d'expliciter les caractéristiques des explications qui sont les plus pertinentes pour les joueurs de bridge.

7.5 Argumentation et explications

La méthode de génération et de sélection d'explications proposée dans le chapitre 6 est une première étape vers une communication plus humaine entre l'utilisateur et le modèle, car elle prend en compte les différents points de vue possibles pour expliquer une décision. Cependant, il est possible d'aller plus loin dans cette communication en utilisant des méthodes d'argumentation formelle pour sélectionner les explications. En effet, l'utilisateur peut ne pas accepter une explication donnée par le modèle, et il est important de pouvoir argumenter sur le choix de cette explication.

L'argumentation formelle [Dun93] est un cadre théorique permettant de modéliser l'argumentation humaine et développe l'idée qu'une déclaration peut être affirmée vraie si elle est soutenue par un ensemble d'arguments qui ne peuvent être attaqués.

De nombreux travaux portent sur l'argumentation formelle et ses liens avec les explications locales [BB21 ; BB22 ; Rag+23]. Dans [KMT18] les auteurs présentent les liens entre la logique propositionnelle et l'argumentation logique.

L'intégration de l'argumentation formelle dans notre cadre d'explications permettrait de rendre les explications plus robustes et acceptables pour l'utilisateur. En fournissant des arguments supplémentaires et en montrant comment ils soutiennent l'explication, l'utilisateur peut mieux comprendre pourquoi une explication est donnée par le modèle et peut être plus enclin à l'accepter ou la rejeter. Le développement de méthodes combinant les explications abductives formelles et l'argumentation formelle constitue une direction intéressante pour de futures recherches.

7.6 Explications multimonde

Dans un contexte d'un processus de Markov partiellement observé, il n'est pas rare que le nombre de mondes possibles compatibles avec une observation soit très grand. Par exemple, dans le cas du bridge, selon le point de vue du déclarant et juste après l'entame qui est la première carte jouée dans la donne, il y a entre 1×10^6 et 5×10^6 distributions possibles des mains de *East* et *West*. Nos explications abductives communes leq-minimales présentées expliquant l'optimalité de toutes les observations d'un seul monde complètement observé. L'agent construit alors des explications spécifiques à ce monde. L'idée des *explications multimonde* est de porter les explications formelles présentées dans le chapitre 6 à un processus de Markov partiellement observé. L'intuition du travail sur les *explications multimonde* est d'affirmer que l'explication de l'optimalité d'une action peut dépendre du monde dans lequel l'agent se trouve, une explication peut être compatible avec un ou plusieurs mondes, et être incompatible avec d'autres. Dans l'approche initiale, la construction d'explications passe par l'utilisation d'un classifieur permettant de classer une observation comme étant optimale ou non. En augmentant le nombre de mondes,

le nombre de contradictions entre les différents mondes augmente et il est possible que l'apprentissage ne soit pas réalisable.

De ce fait la construction d'un classifieur par approche parcimonieuse telle que décrite dans le chapitre 4 est un atout considérable dans la construction d'explications dépendantes d'un sous-ensemble des mondes possibles. En effet, l'approche parcimonieuse permet de construire un classifieur composé de règles qui dépendent d'une classification a priori des observations. Dans le chapitre 4, la classification a priori d'une observation est le joueur qui a étiqueté l'observation. Ici, la classification a priori d'une observation est le monde auquel elle appartient. Les règles construites seront alors dépendantes à un sous-ensemble de mondes et seront utilisées afin de construire des explications pour les observations de ces mondes. Enfin, la construction d'explications multimonde permet d'introduire la notion de *probabilité d'une explication*, grâce à l'attribution d'une probabilité aux mondes compatibles avec cette explication.

Dans le cas du bridge, qui est un jeu à information incomplète, de récentes recherches ont développé un nouveau formalisme, appelé *jeu combinatoire à information incomplète* (CGII) [Li23], qui permet de modéliser la phase du jeu de la carte dans ce jeu. Dans ce formalisme, un monde est tiré aléatoirement dans un ensemble de mondes connu de tous les joueurs. Au bridge, cet ensemble est l'ensemble des distributions possibles des 52 cartes et le monde tiré est partiellement observé. L'avantage d'un tel formalisme, tel que présenté dans [Li23], est que la partialité de l'information ne tient qu'au monde tiré aléatoirement dans l'ensemble des mondes. L'utilisation de ce formalisme dans la construction des explications probabilistes étant donné l'approche présentée dans cette thèse peut s'avérer être une piste intéressante, permettant d'associer une probabilité aux explications en ne dépendant que de la distribution de probabilité sur les différents mondes.

Plusieurs travaux dans les explications formelles ont travaillé sur la notion de probabilité d'une explication abductive [Izz+23 ; BK23]. Ces travaux sont motivés par le fait que les explications abductives minimales peuvent être tout de même assez longues, rendant ainsi l'interprétation de ces explications plus compliquée, mais également, car trouver une explication abductive minimale est un processus coûteux en temps. Une explication probabiliste dans ces travaux est une explication plus courte qu'une explication abductive minimale, elle peut donc faire des erreurs de classification. La probabilité associée à un motif est alors la probabilité que les sur-ensembles de ce motif inclus dans l'observation à expliquer soient classés dans la bonne classe, ce qui est différent de la définition de la probabilité des explications multimonde pour lesquelles les explications ne font pas d'erreurs dans les mondes pour lesquels elles sont parcimonieuses.

Portage de l'apprentissage multijoueur à la logique du premier ordre. Dans le chapitre 4, nous avons présenté un algorithme pour l'approche parcimonieuse dans le cadre d'un apprentissage multijoueur. Cet algorithme prend en entrée des exemples décrits sous forme de vecteurs binaires de caractéristiques et les règles apprises sont des règles en logique propositionnelle. La proposition d'extension du travail sur les explications aux explications multimonde décrit ci-dessus nécessite de porter l'algorithme du chapitre 4 à la logique du premier ordre, car les observations utilisées dans la contribution du chapitre 6 sont décrites en logique du premier ordre et les explications générées sont également en logique du premier ordre.

Conclusion

Cette thèse a proposé deux contributions principales pour améliorer l'interaction entre l'humain et l'intelligence artificielle en prenant comme contexte le jeu de bridge.

Dans la première contribution, nous avons développé un algorithme d'apprentissage multijoueur qui prend en compte l'identité des joueurs afin de prédire leur décision face à une nouvelle situation de manière plus précise. Nous avons démontré l'importance de considérer l'identité pour améliorer la justesse des modèles prédictifs, en particulier dans les situations où les décisions peuvent varier considérablement d'un joueur à l'autre. Notre approche parcimonieuse permet de prendre en compte l'identité des joueurs sans augmenter l'espace de recherche, et les modèles appris permettent de former des groupes de joueurs se comportant de la même manière. Cette méthode s'est révelée plus performante que les approches comparatives, à l'exception de celle utilisant une forêt aléatoire, mais cette dernière est plus difficilement explicable.

Dans la deuxième contribution, nous avons abordé la production d'explications de la décision d'un agent artificiel dans un jeu de bridge simplifié. Nous avons développé une méthode qui génère des explications minimales et diverses des trajectoires de jeu optimales, en tenant compte des préférences de l'utilisateur. Cette méthode permet de produire des explications compréhensibles, évitant ainsi de submerger l'utilisateur avec des informations redondantes ou excessives. Nous pensons que cette méthode pourrait contribuer à l'amélioration de la communication entre l'humain et la machine, favorisant ainsi l'apprentissage de nouvelles stratégies par les explications de l'agent artificiel.

Ces travaux s'inscrivent dans la recherche effectuée à NukkAI, qui a utilisé le bridge comme terrain d'expérimentation pour développer des méthodes d'IA centrées sur l'humain. En mettant en lumière l'importance de l'adaptation des modèles à l'identité de l'utilisateur et de la génération d'explications adaptées aux préférences de l'utilistateur, cette thèse contribue à améliorer la communication entre l'humain et la machine.

En conclusion, cette thèse a avancé notre compréhension de la manière dont l'IA peut « garder l'humain dans la boucle » en utilisant des représentations symboliques, compréhensibles par l'humain. Cette approche prometteuse ouvre la voie à une collaboration plus efficace entre l'humain et la machine, où chacun peut apprendre de l'autre et améliorer mutuellement ses performances.

Bibliographie

[AJL21] K. Aas, M. Jullum et A. Løland. « Explaining individual predictions when features are dependent : More accurate approximations to Shapley values ». In : *Artificial Intelligence* 298 (2021), p. 103502.

[Aki+22] S. Akimoto, P. Lebreton, S. Takahashi et K. Yamagishi. « Quantitative causality analysis of viewing abandonment reasons using Shapley value ». In : *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*. 2022, p. 01-06.

[All83] J. F. Allen. « Maintaining Knowledge about Temporal Intervals ». In : *Commun. ACM* 26.11 (1983), p. 832-843.

[AJ18] D. Alvarez-Melis et T. S. Jaakkola. « On the Robustness of Interpretability Methods ». In : (2018).

[Aud+22a] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez et P. Marquis. « On Preferred Abductive Explanations for Decision Trees and Random Forests ». In : *Proceedings of IJCAI'22*. Sous la dir. de L. D. Raedt. 2022, p. 643-650.

[Aud+22b] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez et P. Marquis. « On the explanatory power of Boolean decision trees ». In : *Data Knowl. Eng.* 142 (2022), p. 102088.

[BG01] L. Bachmair et H. Ganzinger. « Resolution Theorem Proving ». In : *Handbook of Automated Reasoning (in 2 volumes)*. Sous la dir. de J. A. Robinson et A. Voronkov. Elsevier et MIT Press, 2001, p. 19-99.

[Bar94] E. Barnes. « Why P rather than Q ? The curiosities of fact and foil ». en. In : *Philosophical Studies* 73.1 (1994), p. 35-53.

[BCC57] R. Bellman, R. Corporation et K. M. R. Collection. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957.

[BR98] H. Blockeel et L. D. Raedt. « Top-Down Induction of First-Order Logical Decision Trees ». In : *Artif. Intell.* 101.1-2 (1998), p. 285-297.

[Blo+00] H. Blockeel, L. D. Raedt, N. Jacobs et B. Demoen. « Scaling Up Inductive Logic Programming by Learning from Interpretations ». In : *CoRR* cs.LG/0011044 (2000).

[Boo09] G. Boole. « Mathematical Analysis of Logic ». In : *The Mathematical Analysis of Logic : Being an Essay Towards a Calculus of Deductive Reasoning*. Cambridge Library Collection - Mathematics. Cambridge University Press, 2009, p. 3-14.

[BB21] A. Borg et F. Bex. « A Basic Framework for Explanations in Argumentation ». In : *IEEE Intell. Syst.* 36.2 (2021), p. 25-35.

[BB22] A. Borg et F. Bex. « Contrastive Explanations for Argumentation-Based Conclusions ». In : *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*. Sous la dir. de P. Faliszewski, V. Mascardi, C. Pelachaud et M. E. Taylor. International Foundation for Autonomous Agents et Multiagent Systems (IFAAMAS), 2022, p. 1551-1553.

[BK23] L. Bounia et F. Koriche. « Approximating probabilistic explanations via supermodular minimization ». In : *Uncertainty in Artificial Intelligence, UAI 2023, July 31 - 4 August 2023, Pittsburgh, PA, USA*. Sous la dir. de R. J. Evans et I. Shpitser. T. 216. Proceedings of Machine Learning Research. PMLR, 2023, p. 216-225.

[BRV20] B. Bouzy, A. Rimbaud et V. Ventos. « Recursive Monte Carlo Search for Bridge Card Play ». In : *IEEE Conference on Games, CoG 2020, Osaka, Japan, August 24-27, 2020*. IEEE, 2020, p. 229-236.

[Bru+23] E. Brusa, L. Cibrario, C. Delprete et L. Di Maggio. « Explainable AI for Machine Fault Diagnosis : Understanding Features' Contribution in Machine Learning Models for Industrial Condition Monitoring ». In : *Applied Sciences* 13 (2023), p. 2038.

[C70] R. J. C. « Transformational systems and the algebraic structure of atomic formulas ». In : *Machine Intelligence* 5 (1970), p. 135-152.

[CG05] T. Calders et B. Goethals. « Depth-First Non-Derivable Itemset Mining ». In : *SDM*. SIAM, 2005, p. 250-261.

[CLV21] T. Cazenave, S. Legras et V. Ventos. « Optimizing $\alpha\mu$ ». In : *2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021*. IEEE, 2021, p. 1-8.

[Che+22a] Y. Chen, D. M. Aleman, T. G. Purdie et C. McIntosh. « Understanding machine learning classifier decisions in automated radiotherapy quality assurance ». eng. In : *Physics in Medicine and Biology* 67.2 (2022).

[Che+22b] Z. Chen, V. Subhash, M. Havasi, W. Pan et F. Doshi-Velez. *What Makes a Good Explanation ? : A Harmonized View of Properties of Explanations*. 2022.

[CJ95] W. W. Cohen et C. D. P. Jr. « Polynomial Learnability and Inductive Logic Programming : Methods and Results ». In : *New Gener. Comput.* 13.3&4 (1995), p. 369-409.

[Coo71] S. A. Cook. « The Complexity of Theorem-Proving Procedures ». In : *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*. Sous la dir. de M. A. Harrison, R. B. Banerji et J. D. Ullman. ACM, 1971, p. 151-158.

[CRL11] D. Corapi, A. Russo et E. Lupu. « Inductive Logic Programming in Answer Set Programming ». In : *Inductive Logic Programming - 21st International Conference, ILP 2011, Windsor Great Park, UK, July 31 - August 3, 2011, Revised Selected Papers*. Sous la dir. de S. H. Muggleton, A. Tamaddoni-Nezhad et F. A. Lisi. T. 7207. Lecture Notes in Computer Science. Springer, 2011, p. 91-97.

[Cro+21] A. Cropper, S. Dumancic, R. Evans et S. H. Muggleton. « Inductive logic programming at 30 ». In : *CoRR* abs/2102.10556 (2021).

[DH20] A. Darwiche et A. Hirth. « On the Reasons Behind Decisions ». In : *ECAI'20*. Sous la dir. de L. T. 325. Frontiers in Artificial Intelligence and Applications. 2020, p. 712-720.

[DD97] L. De Raedt et L. Dehaspe. « Clausal Discovery ». In : *Machine Learning* 26 (1997), p. 99-146.

[DD98] L. De Raedt et L. Dehaspe. « Learning From Satisfiability ». In : (1998).

[Dim+20] B. Dimanov, U. Bhatt, M. Jamnik et A. Weller. « You Shouldn't Trust Me : Learning Models Which Conceal Unfairness From Multiple Explanation Methods ». en. In : *Santiago de Compostela* (2020).

[Dun93] P. M. Dung. « On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning and Logic Programming ». In : *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*. Sous la dir. de R. Bajcsy. Morgan Kaufmann, 1993, p. 852-859.

[EG17] R. Evans et E. Grefenstette. « Learning Explanatory Rules from Noisy Data ». In : *CoRR* abs/1711.04574 (2017).

[Eva+19] R. Evans, J. Hernández-Orallo, J. Welbl, P. Kohli et M. J. Sergot. « Making sense of sensory input ». In : *CoRR* abs/1910.02227 (2019).

[Fer+09] S. Ferilli, T. M. A. Basile, M. Biba, N. Di Mauro et F. Esposito. « A General Similarity Framework for Horn Clause Logic ». In : *Fundam. Inf.* 90 (2009), p. 43-66.

[Fer+02] S. Ferilli, N. Fanizzi, N. Di Mauro et T. Basile. « Efficient theta-Subsumption under Object Identity ». In : *Workshop AI, IA 2002* (2002).

[GR22] N. Gorji et S. Rubin. « Sufficient Reasons for Classifier Decisions in the Presence of Domain Constraints ». In : *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, p. 5660-5667.

[GF93] G. Gottlob et C. G. Fermüller. « Removing Redundancy from a Clause ». In : *Artif. Intell.* 61.2 (1993), p. 263-289.

[GG21] A. Gramegna et P. Giudici. « SHAP and LIME : An Evaluation of Discriminative Power in Credit Risk ». In : *Frontiers in Artificial Intelligence* 4 (2021).

[Hir03] K. Hirata. « On Condensation of a Clause ». In : *Inductive Logic Programming : 13th International Conference, ILP 2003, Szeged, Hungary, September 29-October 1, 2003, Proceedings*. Sous la dir. de T. Horváth. T. 2835. Lecture Notes in Computer Science. Springer, 2003, p. 164-179.

[HL23] L. O. Hjelkrem et P. E. d. Lange. « Explaining Deep Learning Models for Credit Scoring with SHAP : A Case Study Using Open Banking Data ». en. In : *Journal of Risk and Financial Management* 16.44 (2023), p. 221.

[How60] R. Howard. *Dynamic programming and Markov processes*. Technology Press of Massachusetts Institute of Technology, 1960.

[Hua+21] X. Huang, Y. Izza, A. Ignatiev et J. Marques-Silva. « On Efficiently Explaining Graph-Based Classifiers ». In : *Proceedings of KR'21*. 2021, p. 356-367.

[Ide95] P. Idestam-Almquist. « Generalization of Clauses under Implication ». In : *J. Artif. Intell. Res.* 3 (1995), p. 467-489.

[Ign20] A. Ignatiev. « Towards Trustable Explainable AI ». In : *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. Sous la dir. de C. Bessiere. ijcai.org, 2020, p. 5154-5158.

[Ign+20] A. Ignatiev, N. Narodytska, N. Asher et J. Marques-Silva. « On Relating "Why ?" and "Why Not ?" Explanations ». In : arXiv :2012.11067 (2020). arXiv :2012.11067 [cs].

[Ino91] K. Inoue. « Consequence-Finding Based on Ordered Linear Resolution ». In : *IJCAI'91*. Morgan Kaufmann, 1991, p. 158-164.

[Ino16] K. Inoue. « Meta-Level Abduction ». In : *FLAP* 3.1 (2016), p. 7-36.

[IRS14] K. Inoue, T. Ribeiro et C. Sakama. « Learning from interpretation transition ». In : *Mach. Learn.* 94.1 (2014), p. 51-79.

[Izz+23] Y. Izza, X. Huang, A. Ignatiev, N. Narodytska, M. C. Cooper et J. Marques-Silva. « On computing probabilistic abductive explanations ». In : *Int. J. Approx. Reason.* 159 (2023), p. 108939.

[IIM20] Y. Izza, A. Ignatiev et J. Marques-Silva. « On Explaining Decision Trees ». In : *CoRR* abs/2010.11034 (2020).

[IIM22] Y. Izza, A. Ignatiev et J. Marques-Silva. « On Tackling Explanation Redundancy in Decision Trees ». In : *J. Artif. Intell. Res.* 75 (2022), p. 261-321.

[Jac92] P. Jackson. « Computing Prime Implicates Incrementally ». In : *Automated Deduction - CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, June 15-18, 1992, Proceedings*. Sous la dir. de D. Kapur. T. 607. Lecture Notes in Computer Science. Springer, 1992, p. 253-267.

[KMT18] A. C. Kakas, P. Mancarella et F. Toni. « On Argumentation Logic and Propositional Logic ». In : *Stud Logica* 106.2 (2018), p. 237-279.

[KN86] D. Kapur et P. Narendran. « NP-Completeness of the Set Unification and Matching Problems ». In : *8th International Conference on Automated Deduction, Oxford, England, July 27 - August 1, 1986, Proceedings*. Sous la dir. de J. H. Siekmann. T. 230. Lecture Notes in Computer Science. Springer, 1986, p. 489-495.

[Kaz+24] M. Kazi Aoual, H. Soldano, C. Rouveirol et V. Ventos. « Apprentissage multijoueurs supervisé ». In : *RJCIA* (2024), p. 48.

[KT90] A. Kean et G. K. Tsiknis. « An Incremental Method for Generating Prime Implicants/Impicates ». In : *J. Symb. Comput.* 9.2 (1990), p. 185-206.

[Kel+23] G. Kelodjou, L. Rozé, V. Masson, L. Galárraga, R. Gaudel, M. Tchuenté et al. « Shaping Up SHAP : Enhancing Stability through Layer-Wise Neighbor Selection ». In : *CoRR* abs/2312.12115 (2023).

[Kle92] J. de Kleer. « An Improved Incremental Algorithm for Generating Prime Implicates ». In : *Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, July 12-16, 1992*. Sous la dir. de W. R. Swartout. AAAI Press / The MIT Press, 1992, p. 780-785.

[Kro+03] M. Krogel, S. A. Rawles, F. Železný, P. A. Flach, N. Lavrac et S. Wrobel. « Comparative Evaluation of Approaches to Propositionalization ». In : *Inductive Logic Programming : 13th International Conference, ILP 2003, Szeged, Hungary, September 29-October 1, 2003, Proceedings*. Sous la dir. de T. Horváth. T. 2835. Lecture Notes in Computer Science. Springer, 2003, p. 197-214.

[KSZ12] O. Kuzelka, A. Szabóová et F. Železný. « Bounded Least General Generalization ». In : *Inductive Logic Programming - 22nd International Conference, ILP 2012, Dubrovnik, Croatia, September 17-19, 2012, Revised Selected Papers*. Sous la dir. de F. Riguzzi et F. Železný. T. 7842. Lecture Notes in Computer Science. Springer, 2012, p. 116-129.

[LZF02] N. Lavrac, F. Železný et P. A. Flach. « RSD : Relational Subgroup Discovery through First-Order Feature Construction ». In : *Inductive Logic Programming, 12th International Conference, ILP 2002, Sydney, Australia, July 9-11, 2002. Revised Papers*. Sous la dir. de S. Matwin et C. Sammut. T. 2583. Lecture Notes in Computer Science. Springer, 2002, p. 149-165.

[LRB14] M. Law, A. Russo et K. Broda. « Inductive Learning of Answer Set Programs ». In : *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings*. Sous la dir. d'E. Fermé et J. Leite. T. 8761. Lecture Notes in Computer Science. Springer, 2014, p. 311-325.

[LRV18] S. Legras, C. Rouveirol et V. Ventos. « The Game of Bridge : A Challenge for ILP ». In : *Inductive Logic Programming - 28th International Conference, ILP 2018, Ferrara, Italy, September 2-4, 2018, Proceedings*. Sous la dir. de F. Riguzzi, E. Bellodi et R. Zese. T. 11105. Lecture Notes in Computer Science. Springer, 2018, p. 72-87.

[Li23] J. Li. « Games with incomplete information : complexity, algorithmics, reasoning ». 2023NORMC270. Thèse de doct. 2023.

[LTV20] J. Li, S. Thepaut et V. Ventos. « StarAI : Reducing incompleteness in the game of Bridge using PLP ». In : *CoRR* abs/2001.08193 (2020).

[Li+22] L. Li, X. Wu, M. Kong, D. Zhou et X. Tao. « Towards the Quantitative Interpretability Analysis of Citizens Happiness Prediction ». en. In : t. 6. 2022, p. 5094-5100.

[Lip90] P. Lipton. « Contrastive Explanation ». en. In : *Royal Institute of Philosophy Supplement* 27 (1990), p. 247-266.

[LFF02] F. A. Lisi, S. Ferilli et N. Fanizzi. « Object Identity as Search Bias for Pattern Spaces ». In : *Proceedings of the 15th European Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002*. Sous la dir. de F. van Harmelen. IOS Press, 2002, p. 375-379.

[LL17] S. M. Lundberg et S.-I. Lee. « A Unified Approach to Interpreting Model Predictions ». In : *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA : Curran Associates Inc., 2017, p. 4768-4777.

[MS04] J. Maloberti et E. Suzuki. « An Efficient Algorithm for Reducing Clauses Based on Constraint Satisfaction Techniques ». In : *Inductive Logic Programming, 14th International Conference, ILP 2004, Porto, Portugal, September 6-8, 2004, Proceedings*. Sous la dir. de R. Camacho, R. D. King et A. Srinivasan. T. 3194. Lecture Notes in Computer Science. Springer, 2004, p. 234-251.

[MI22a] J. Marques-Silva et A. Ignatiev. « Delivering Trustworthy AI through Formal XAI ». In : *Proceedings of the AAAI Conference on Artificial Intelligence* 36.11 (2022), p. 12342-12350.

[MI22b] J. Marques-Silva et A. Ignatiev. « Delivering Trustworthy AI through Formal XAI ». In : *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, p. 12342-12350.

[Mar91] P. Marquis. « Extending abduction from propositional to first-order logic ». In : *FAIR*. T. 535. Springer, 1991, p. 141-155.

[McL+12] B. M. McLaren, D. Adams, K. Durkin, G. Goguadze, R. E. Mayer, B. Rittle-Johnson et al. « To Err Is Human, to Explain and Correct Is Divine : A Study of Interactive Erroneous Examples with Middle School Math Students ». In : *21st Century Learning for 21st Century Skills*. Sous la dir. d'A. Ravenscroft, S. Lindstaedt, C. D. Kloos et D. Hernández-Leo. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 222-235.

[MAM15] B. M. McLaren, D. M. Adams et R. E. Mayer. « Delayed learning effects with erroneous examples : A study of learning decimals with a web-based tutor ». In : *International Journal of Artificial Intelligence in Education* 25.4 (2015), p. 520-542.

[MN22] O. Mey et D. Neufeld. « Explainable AI Algorithms for Vibration Data-Based Fault Detection : Use Case-Adadpted Methods and Critical Evaluation ». en. In : *Sensors* 22.2323 (2022), p. 9037.

[Mil19] T. Miller. « Explanation in artificial intelligence : Insights from the social sciences ». In : *Artificial Intelligence* 267 (2019), p. 1-38.

[Mit77] T. M. Mitchell. « Version Spaces : A Candidate Elimination Approach to Rule Learning ». In : *Proceedings of the 5th International Joint Conference on Artificial Intelligence. Cambridge, MA, USA, August 22-25, 1977*. Sous la dir. de R. Reddy. William Kaufmann, 1977, p. 305-310.

[Mit82] T. M. Mitchell. « Generalization as Search ». In : *Artif. Intell.* 18.2 (1982), p. 203-226.

[Mol18] C. Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. 2018.

[MC21] R. Morel et A. Cropper. « Learning Logic Programs by Explaining Failures ». In : *CoRR* abs/2102.12551 (2021).

[MR94] S. Muggleton et L. D. Raedt. « Inductive Logic Programming : Theory and Methods ». In : *J. Log. Program.* 19/20 (1994), p. 629-679.

[Mug95a] S. H. Muggleton. « Inverse Entailment and Progol ». In : *New Gener. Comput.* 13.3&4 (1995), p. 245-286.

[Mug95b] S. H. Muggleton. « Inverse Entailment and Progol ». In : *New Gener. Comput.* 13.3&4 (1995), p. 245-286.

[MF90] S. H. Muggleton et C. Feng. « Efficient Induction of Logic Programs ». In : *Algorithmic Learning Theory, First International Workshop, ALT '90, Tokyo, Japan, October 8-10, 1990, Proceedings*. Sous la dir. de S. Arikawa, S. Goto, S. Ohsuga et T. Yokomori. Springer/Ohmsha, 1990, p. 368-381.

[MLT15] S. H. Muggleton, D. Lin et A. Tamaddoni-Nezhad. « Meta-interpretive learning of higher-order dyadic datalog : predicate invention revisited ». In : *Mach. Learn.* 100.1 (2015), p. 49-73.

[Nab+10] H. Nabeshima, K. Iwanuma, K. Inoue et O. Ray. « SOLAR : An automated deduction system for consequence finding ». In : *AI Commun.* 23.2-3 (2010), p. 183-203.

[Néd+96] C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano et B. Tausend. « Declarative bias in ILP ». In : *Advances in Inductive Logic Programming* (1996), p. 82-103.

[NW97] S.-H. Nienhuys-Cheng et R. de Wolf. *Foundations of Inductive Logic Programming*. Sous la dir. de J. Siekmann et J. G. Carbonell. Springer-Verlag New York, Inc., 1997.

[Nie+97] S.-H. Nienhuys-Cheng, R. d. Wolf, J. Siekmann et J. G. Carbonell. *Foundations of Inductive Logic Programming*. Berlin, Heidelberg : Springer-Verlag, 1997.

[Oh+22] A. R. Oh, J. Park, J.-H. Lee, H. Kim, K. Yang, J.-H. Choi et al. « Association Between Perioperative Adverse Cardiac Events and Mortality During One-Year Follow-Up After Noncardiac Surgery ». eng. In : *Journal of the American Heart Association* 11.8 (2022), e024325.

[PS82] C. Papadimitriou et K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. T. 32. 1982.

[Plo70] G. D. Plotkin. « A Note on Inductive Generalization ». In : *Machine Intelligence* 5 (1970), p. 153-163.

[Pro90] G. M. Provan. « The Computational Complexity of Multiple-Context Truth Maintenance Systems ». In : *9th European Conference on Artificial Intelligence, ECAI 1990, Stockholm, Sweden, 1990*. 1990, p. 522-527.

[QI92] Z. Qian et K. B. Irani. « Concept Learning in Default Logic ». In : *Fourth International Conference on Tools with Artificial Intelligence, ICTAI '92, Arlington, Virginia, USA, November 10-13, 1992*. IEEE Computer Society, 1992, p. 244-250.

[Qui59] W. V. Quine. « On Cores and Prime Implicants of Truth Functions ». In : 66.9 (1959), p. 755-760.

[Qui90] J. R. Quinlan. « Learning Logical Definitions from Relations ». In : *Mach. Learn.* 5 (1990), p. 239-266.

[QC93] J. R. Quinlan et R. M. Cameron-Jones. « FOIL : A Midterm Report ». In : *ECML'93*. T. 667. Lecture Notes in Computer Science. Springer, 1993, p. 3-20.

[RSS22] J. Rabold, M. Siebers et U. Schmid. « Generating contrastive explanations for inductive logic programming based on a near miss approach ». In : *Mach. Learn.* 111.5 (2022), p. 1799-1820.

[Rae97] L. D. Raedt. « Logical Settings for Concept-Learning ». In : *Artif. Intell.* 95.1 (1997), p. 187-201.

[RD94] L. D. Raedt et S. Džeroski. « First-Order Jk-Clausal Theories Are Pac-Learnable ». In : *Artificial Intelligence* 70.1-2 (1994), p. 375-392.

[RL95] L. D. Raedt et W. V. Laer. « Inductive Constraint Logic ». In : *Algorithmic Learning Theory, 6th International Conference, ALT '95, Fukuoka, Japan, October 18-20, 1995, Proceedings*. Sous la dir. de K. P. Jantke, T. Shinohara et T. Zeugmann. T. 997. Lecture Notes in Computer Science. Springer, 1995, p. 80-94.

[Rag+23] A. Rago, F. Russo, E. Albini, F. Toni et P. Baroni. « Explaining Classifiers' Outputs with Causal Models and Argumentation ». In : *FLAP* 10.3 (2023), p. 421-509.

[Ray09] O. Ray. « Nonmonotonic abductive inductive learning ». In : *J. Appl. Log.* 7.3 (2009), p. 329-340.

[RK87] R. Reiter et J. de Kleer. « Foundations of Assumption-based Truth Maintenance Systems : Preliminary Report ». In : *Proceedings of the 6th National Conference on Artificial Intelligence. Seattle, WA, USA, July 1987*. Sous la dir. de K. D. Forbus et H. E. Shrobe. Morgan Kaufmann, 1987, p. 183-189.

[RSG18] M. T. Ribeiro, S. Singh et C. Guestrin. « Anchors : High-Precision Model-Agnostic Explanations ». In : *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018).

[RSG16] M. T. Ribeiro, S. Singh et C. Guestrin. « "Why Should I Trust You ?" : Explaining the Predictions of Any Classifier ». In : *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, p. 1135-1144.

[Rib+21] T. Ribeiro, M. Folschette, M. Magnin et K. Inoue. « Learning any semantics for dynamical systems represented by logic programs ». working paper or preprint. 2021.

[Rob65] J. A. Robinson. « A Machine-Oriented Logic Based on the Resolution Principle ». In : *J. ACM* 12.1 (1965), p. 23-41.

[RB18] M. Robnik-Šikonja et M. Bohanec. « Perturbation-Based Explanations of Prediction Models ». en. In : *Human and Machine Learning : Visible, Explainable, Trustworthy and Transparent*. Sous la dir. de J. Zhou et F. Chen. Cham : Springer International Publishing, 2018, p. 159-175.

[Rou94] C. Rouveirol. « Flattening and Saturation : Two Representation Changes for Generalization ». In : *Mach. Learn.* 14.1 (1994), p. 219-232.

[Rou+23] C. Rouveirol, M. Kazi Aoual, H. Soldano et V. Ventos. « Explaining the optimal trajectories ». In : *RuleML+RR*. T. To appear. LNCS. Springer, 2023.

[Rus18] S. J. Rushton. « Teaching and learning mathematics through error analysis ». In : *Fields Mathematics Education Journal* 3.1 (2018), p. 4.

[Sai+20] S. Saito, E. Chua, N. Capel et R. Hu. « Improving LIME Robustness with Smarter Locality Sampling ». In : *CoRR* abs/2006.12302 (2020).

[SCS23] L. Saulières, M. Cooper et F. D. de Saint-Cyr. « Predicate-based explanation of a Reinforcement Learning agent via action importance evaluation ». In : *4th workshop on Advances in Interpretable Machine Learning and Artificial Intelligence AIMLAI 2023 ECML/PKDD conference*. 2023.

[SCS24] L. Saulières, M. Cooper et F. D. de Saint-Cyr. « Backward explanation via redefinition of predicates ». In : *JIAF-JFPDA* (2024), p. 79.

[Sha88] C. E. Shannon. « Programming a Computer for Playing Chess ». In : *Computer Chess Compendium*. Sous la dir. de D. Levy. New York, NY : Springer New York, 1988, p. 2-13.

[Sha53] L. S. Shapley. « 17. A Value for n-Person Games ». In : *Contributions to the Theory of Games (AM-28), Volume II*. Sous la dir. de H. W. Kuhn et A. W. Tucker. Princeton : Princeton University Press, 1953, p. 307-318.

[SL13] A. Sheshadri et M. Lease. « SQUARE : A Benchmark for Research on Computing Crowd Consensus ». In : *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2013, November 7-9, 2013, Palm Springs, CA, USA*. Sous la dir. de B. Hartman et E. Horvitz. AAAI, 2013.

[SCD18] A. Shih, A. Choi et A. Darwiche. « A Symbolic Approach to Explaining Bayesian Network Classifiers ». In : *IJCAI'18*. ijcai.org, 2018, p. 5103-5111.

[Sil+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche et al. « Mastering the game of Go with deep neural networks and tree search ». In : *Nature* 529 (2016), p. 484-503.

[SV01] L. Simon et A. del Val. « Efficient Consequence Finding ». In : *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*. Sous la dir. de B. Nebel. Morgan Kaufmann, 2001, p. 359-370.

[Sla+20] D. Slack, S. Hilgard, E. Jia, S. Singh et H. Lakkaraju. « Fooling LIME and SHAP : Adversarial Attacks on Post hoc Explanation Methods ». In : arXiv :1911.02508 (2020). arXiv :1911.02508 [cs, stat].

[Sol11] H. Soldano. « A Modal View on Abstract Learning and Reasoning ». In : *Proceedings of the Ninth Symposium on Abstraction, Reformulation, and Approximation, SARA 2011*. Sous la dir. de M. R. Genesereth et P. Z. Revesz. AAAI, 2011.

[SSB20] H. Soldano, G. Santini et D. Bouthinon. « Attributed Graph Pattern Set Selection Under a Distance Constraint ». In : *Complex Networks and Their Applications VIII*. 2020, p. 228-241.

[SR17] A. Soulet et F. Rioult. « Exact and Approximate Minimal Pattern Mining ». In : *Advances in Knowledge Discovery and Management*. T. 6. Springer, 2017, p. 61-81.

[Sri99] A. Srinivasan. *The Aleph Manual*. 1999.

[ŠK10] E. Štrumbelj et I. Kononenko. « An Efficient Explanation of Individual Classifications Using Game Theory ». In : *J. Mach. Learn. Res.* 11 (2010), p. 1-18.

[ŠK13] E. Štrumbelj et I. Kononenko. « Explaining prediction models and individual predictions with feature contributions ». In : *Knowledge and Information Systems* 41 (2013), p. 647-665.

[Sza+09] L. Szathmary, P. Valtchev, A. Napoli et R. Godin. « Efficient Vertical Mining of Frequent Closures and Generators ». In : *IDA*. T. 5772. Lecture Notes in Computer Science. Springer, 2009, p. 393-404.

[Tao+24] Y. Tao, O. Viberg, R. S. Baker et R. F. Kizilcec. *Cultural Bias and Cultural Alignment of Large Language Models*. 2024.

[Tis67] P. L. Tison. « Generalization of Consensus Theory and Application to the Minimization of Boolean Functions ». In : *IEEE Trans. Electron. Comput.* 16.4 (1967), p. 446-456.

[Tou16] S. Tourret. « Abduction in first order logic with equality. (Prime implicate generation in equational logic) ». Thèse de doct. Grenoble Alpes University, France, 2016.

[Val84] L. G. Valiant. « A Theory of the Learnable ». In : *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*. Sous la dir. de R. A. DeMillo. ACM, 1984, p. 436-445.

[VT17] V. Ventos et O. Teytaud. « Le bridge, nouveau défi de l'intelligence artificielle ? » In : *Rev. d'Intelligence Artif.* 31 (2017), p. 249-279.

[WBL16] A. W Flores, K. Bechtel et C. Lowenkamp. « False Positives, False Negatives, and False Analyses : A Rejoinder to "Machine Bias : There's Software Used Across the Country to Predict Future Criminals. And it's Biased Against Blacks." » In : *Federal probation* 80 (2016).

[Wan+23] C. Wang, Y. Gao, C. Fan, J. Hu, T. L. Lam, N. D. Lane et al. « Learn2Agree : Fitting with Multiple Annotators Without Objective Ground Truth ». In : *Trustworthy Machine Learning for Healthcare - First International Workshop, TML4H 2023, Proceedings*. Sous la dir. de H. Chen et L. Luo. T. 13932. Lecture Notes in Computer Science. Springer, 2023.

[Win70] P. H. Winston. « Learning structural descriptions from examples ». Thèse de doct. Massachusetts Institute of Technology, USA, 1970.

[Yan+14] Y. Yan, R. Rosales, G. Fung, S. Ramanathan et J. G. Dy. « Learning from multiple annotators with varying expertise ». In : *Mach. Learn.* 95 (2014).

[ZPP14] Q. Zeng, J. M. Patel et D. Page. « QuickFOIL : Scalable Inductive Logic Programming ». In : *Proc. VLDB Endow.* 8.3 (2014), p. 197-208.

Annexes

Annexe A

Enchères pour le chapitre 4

Dans cette annexe, nous présentons la séquence d'enchères pour l'étude de cas du chapitre 4. Cette annexe n'est pas importante pour la compréhension du chapitre, mais elle peut être utile pour les lecteurs qui souhaitent approfondir leur compréhension des enchères au bridge ainsi que l'enjeu de l'enchère considérée dans l'étude de cas.

La séquence d'enchères considérée dans l'étude de cas du chapitre 4 est donnée dans le tableau A.1. Elle se lit case par case de gauche à droite, chaque case correspondant à la décision d'un joueur. Nous allons lister les informations que les joueurs font passer à propos de leur main à chaque décision :

- *South* **1SA**
  - Entre 15 et 17 points d'honneurs ;
  - Pas de 17 points d'honneurs combiné avec 5 cartes dans une couleur majeure ;
  - Main équilibrée ou semi-équilibrée ;
  - Pas plus de 5 cartes dans une couleur majeure ;
  - Pas de distribution des couleurs avec 5 cartes dans une majeure et 4 cartes dans une autre couleur.
- *West* **P**
  - Pas plus de 5 cartes dans une couleur avec plus d'un honneur parmi *As*/*Roi*/*Dame* et 9 points d'honneurs ou plus ;
  - Pas 5 cartes dans une couleur majeure avec plus d'un honneur parmi *As*/*Roi*/*Dame*, combiné avec 4 cartes dans une autre couleur avec plus d'un honneur avec 10+ points d'honneurs ;
  - Pas 4 cartes dans une couleur majeure avec plus d'un honneur parmi *As*/*Roi*/*Dame*, combiné avec 4 cartes dans une couleur mineure avec plus d'un honneur avec 10+ points d'honneurs.
- *North* **3SA**
  - Entre 8 et 15 points d'honneurs ;
  - Pas plus de 3 cartes dans une couleur majeure excepté si la distribution de la main est (4,3,3,3).
- *East* **P** : Pas plus de 4 cartes dans une couleur avec 3 honneurs.

South	West	North	East
1SA	P	3SA	P
P	P		

TABLE A.1 – Séquence d'enchères pour l'étude de cas du chapitre 4

— *South* **P** : Aucune information n'est passée.

— *West* **P** : Aucune information n'est passée.

Grâce à ces contraintes d'enchères et dans la situation particulière étudiée dans le chapitre 4, le joueur qui entame (*West*) sait que les adversaires devraient avoir moins de cartes majeures que de cartes mineures. Comme vu dans le chapitre 4, *West* possède une main très équilibrée avec 4 cartes dans une couleur mineure et 3 cartes dans les autres couleurs. Ainsi, *West* a deux possibilités : (i) soit il joue dans une couleur majeure en espérant trouver la couleur majeure que son partenaire possède en surnombre par rapport aux adversaires ; (ii) soit il joue dans la couleur la plus longue qu'il possède, lui donnant plus d'assurance de gagner le plus de plis.

Annexe B

Données pour le chapitre 4

Dans cette annexe, nous présentons le prétraitement des données pour le chapitre 4. Nous commençons par présenter les données brutes, puis nous expliquons comment nous les avons pré-traitées pour les rendre utilisables par notre algorithme d'apprentissage multijoueur.

Nous disposons d'un ensemble de 7263 exemples, étiquetés par 10 joueurs différents. Les fichiers d'entrée sont au format Aleph. On a alors l'ensemble des exemples E sous la forme d'une liste de faits dont le prédicat est le prédicat du concept cible. On a les exemples positifs E^+ pour lesquels le concept cible est vrai et les exemples négatifs E^- pour lesquels le concept cible est faux. On a également une théorie du domaine B composée de faits et de règles, les règles inclues dans la théorie du domaine sont tirées de la théorie du domaine utilisée dans [LRV18].

Le programme d'apprentissage multijoueur est écrit en Java et prend en entrée des données sous forme propositionnelle, c'est-à-dire que les exemples sont représentés par des vecteurs de caractéristiques binaires. Pour transformer les données brutes, écrites en logique du premier ordre en données écrites en logique propositionnelle, on applique une opération de propositionnalisation.

Après avoir consulté une étude comparative sur différentes techniques de propositionnalisation [Kro+03], nous avons choisi d'appliquer le programme RSD [LZF02], car le format des données d'entrée du programme se rapproche beaucoup de celle en entrée d'Aleph, ce qui correspond parfaitement aux données que l'on a. Le programme RSD fonctionne de la façon suivante :

1. Il génère des règles dont le corps est une conjonction de littéraux légale étant donné un biais de langage et un ensemble de modes (voir section 3.2.2).
2. Il instancie les variables existentielles des règles en utilisant les faits instanciés de la base de données.
3. Il construit une représentation propositionnelle des données, avec en attribut de chaque exemple, une représentation binaire des règles construites par RSD avec un 1 au i-ème attribut si l'exemple vérifie la i-ème règle, et un 0 sinon.

Annexe C

Algorithmes annexes pour le chapitre 6

Algorithme 13 $minSubsets(m, Cm, Res, E, firstVisit, isSubsetMin)$

Entrée: m un motif, Res un ensemble de littéraux t.q. $m \cap Res = \emptyset$, Cm la couverture de m dans E, *isSubsetMin* booléen, *True* si m est subset-minimal par rapport à E, *firstVisit* : booléen, *True* si premier appel pour m.

Sortie: Retourne l'ensemble des motifs subset-minimaux par rapport à E à partir de $m \cup Res$, incluant m si *isSubsetMin* et *firstVisit* sont *True* et ordonnant les littéraux dans *Res* dans l'ordre croissant de la couverture si *firstVisit* est *True*

1: $MG \leftarrow \emptyset$
2: **si** *firstVisit* et *isSubsetMin* **alors** $MG \leftarrow \{m\}$ ▷ m subset-minimal et à retourner
3: **fin si**
4: **si** $Cm = \emptyset$ **ou** $Res = \emptyset$ **alors retourne** MG
5: **fin si**
6: **si** *firstVisit* **alors** $sort(Res, Cm)$ ▷ l de Res dans l'ordre croissant de $cov([m, l], E)$
7: **fin si**
8: $l \leftarrow head(Res)$
9: $TL \leftarrow tail(Res)$
10: $Cml \leftarrow coverage([m, \{l\}], Cm)$ ▷ $cov([m, l], E)$
11: $isSubsetMinl \leftarrow$ False
12: **si** $subsetMinimal(m, \{l\}, Cm, Cml, E)$ **alors**
13: $isSubsetMinl \leftarrow True$
14: **fin si**
15: $MGl \leftarrow minSubsets([m, l], Cml, Res \setminus \{l\}, True, isSubsetMinl)$ ▷ *l'ensemble des motifs subset-minimaux par rapport à E à partir de $m \cup Res$ et chacun contenant m et $\{l\}$*
16: $MGnl \leftarrow minSubsets(m, Cm, Res \setminus \{l\}, False, isSubsetMin)$ ▷ *l'ensemble des motifs subset-minimaux par rapport à E à partir de $m \cup Res$, excluant m, et chacun contenant m mais pas $\{l\}$*
17: **retourne** $MG \cup MGl \cup MGnl$

L'algorithme 15 implémente la fonction $maxInst$ impliquée dans le calcul de la borne M contenant les leq-mces.

Algorithme 14 *subsetMinimal*(p, Cp, E)

Entrée: p un motif, Cp la couverture de p dans E, E un ensemble d'observations
Sortie: retourne *True* ssi p est subset-minimal par rapport à E

```
1: pour e ∈ p faire
2:     si |cov([p \ e], E)| = |Cp| alors
3:         retourne False
4:     fin si
5: fin pour
6: retourne True
```

Algorithme 15 *maxInst*$(m, Vars, O)$

Entrée: m une explication commune pour O, *Vars* sous-ensemble de $vars(m)$
Sortie: retourne les instances maximales de m quand on substitue les variables de *Vars* par des constantes apparaissant dans une observation o ou variables dans *Vars*, qui sont des explications pour O

```
si Vars = ∅ alors retourne {m}
fin si
V ← head(Vars) ; Tvars ← tail(Vars) ; o ← head(O)
Max ← ∅
pour toutes les substitutions {V/t_i} t.q. m.{V/t_i} ⪯_θ o avec t_i une constante dans o
ou une variable dans Tvars faire
    si cov(m{V/c_j}, O) = O alors
        m ← m{V/c_j}
        Max ← Max ∪ maxInst(m, Tvars, O)
    fin si
fin pour
Max ← Max ∪ maxInst(m, Tvars, O)
retourne selectMaxInst(Max)
```

Annexe D

Modèle du défenseur pour le chapitre 6

Algorithme 16 Modèle du défenseur

Entrée: Main du défenseur M, position du défenseur Pos, historique des cartes jouées $Hist$

Sortie: $Card$: Carte à jouer par le $defenseur$

si $defenseur$ est le premier à jouer de la donne **alors** ▷ Modèle du défenseur pour l'entame
    si une carte parmi $\{13, 14\}$ est dans M et qu'elle est la plus grande d'une séquence de taille au moins 3 **alors**
        $Card$ est la plus grande de la séquence
    sinon si 12 est dans M et qu'elle est la plus grande d'une séquence de taille au moins 2 et qu'au moins une carte parmi $\{10, 9\}$ est dans M **alors**
        $Card$ est 12
    sinon $Card$ est la plus petite carte de M
    fin si
sinon si $defenseur$ est premier à jouer dans le pli ($Pos = 1$) **alors** ▷ Modèle du défenseur hors entame (jusqu'au dernier *fin si*)
    si $|M|$ pair **alors**
        $Card$ est la plus grande carte de M
    sinon
        $Card$ est la plus petite carte de M
    fin si
sinon
    si La plus grande carte dans M est plus haute que la dernière carte de $Hist$ **alors**
        $Card$ est la plus grande carte de M
    sinon
        $Card$ est la plus petite carte de M
    fin si
fin si

Annexe E

Vademecum pour le chapitre 6

Dans cette annexe, nous listons les prédicats utilisés dans les expérimentations du chapitre 6 pour décrire les observations, et nous illustrons des exemples d'instanciations de ces prédicats sur les trajectoires 1 et 2 issues de la donne W_{11} et la trajectoire 3 issue de la donne E_{11} qui sont les suivantes :

1. **Trajectoire 1** :

```
W8    N3    E7   S2
W11   N5    E6   S10
W9    N12   E0   S13
```

2. **Trajectoire 2** :

```
W8    N3    E7   S13
S14   W9    N5   E6
S2    W11   N12  E0
```

3. **Trajectoire 3** :

```
W8    N3    E11  S13
S2    W9    N12  E6
N4    E7    S10  W0
```

Lorsqu'il y a un grand nombre d'instanciations d'un prédicat, nous n'affichons que quelques exemples de ces instanciations.

Dans ce vadémécum, nous donnons le type d'arguments dans chaque prédicat à la place de l'argument dans le prédicat, ici les arguments temporels, c'est-à-dire les instants ou les intervalles de temps, sont représentés par des types différents pour plus de clarté. Pour clarifier, l'argument *Player* peut être instancié dans certains prédicats avec un sous-ensemble de l'ensemble {*north*, *east*, *south*, *west*} représentant les joueurs. Les valeurs que cet argument peut prendre seront spécifiées pour chaque prédicat.

Prédicats ponctuels

Ce sont des prédicats liés à une observation ou une action qui se passe à un temps précis.

- *dominantInTrick*(*Card*, *Player*, *Time*) : une carte *Card* d'un joueur *Player* (parmi tous les joueurs) est dominante dans le pli courant si elle est plus forte que toutes les cartes jouées dans le pli courant au temps *Time*. On a les instanciations suivantes pour la trajectoire 2 : *dominantInTrick*(*8*, *west*, *1*), *dominantInTrick*(*8*, *west*, *2*), *dominantInTrick*(*14*, *south*, *4*), *dominantInTrick*(*11*, *west*, *6*).
- *willTakeTrick*(*Card*, *Player*, *Time*) : la carte *Card* que le joueur *Player* (*north* ou *south*) joue au temps *Time* est plus grande que toutes les cartes des autres joueurs dans leur main et dans le pli courant. Le joueur va ainsi emporter le pli courant. Dans la trajectoire 1, on a l'instanciation suivante : *willTakeTrick*(*13*, *south*, *6*).
- *action*(*Card*, *Time*) : la carte *Card* est jouée par le déclarant (*north* ou *south*) au temps *Time*. On a les instanciations suivantes pour la trajectoire 1 : *action*(*3*, *1*), *action*(*2*, *2*), *action*(*5*, *3*), ...
- *playSmallestCard*(*Card*, *Player*, *Time*) : le joueur *Player* (*north* ou *south*) joue la plus petite carte dans sa main *Card* au temps *Time*. On a les instanciations suivantes pour la trajectoire 1 : *playSmallestCard*(*3*, *north*, *1*), *playSmallestCard*(*2*, *south*, *2*), *playSmallestCard*(*10*, *south*, *4*), ...
- *declarerPlaysFirstInTrick*(*Player*, *Card*, *Time*) : le joueur *Player* qui est déclarant (*north* ou *south*) joue la carte *Card* au temps *Time* et est le premier de son équipe à jouer dans le pli courant. On a les instanciations suivantes pour la trajectoire 2 : *declarerPlaysFirstInTrick*(*north*, *3*, *1*), *declarerPlaysFirstInTrick*(*south*, *14*, *3*), *declarerPlaysFirstInTrick*(*south*, *2*, *5*).
- *playSmallCard*(*Card*, *Player*, *Time*) : le joueur *Player* (*north* ou *south*) joue une carte *Card* qui est une petite carte (2 à 10) au temps *Time*. On a les instanciations suivantes pour la trajectoire 2 : *playSmallCard*(*3*, *north*, *1*), *playSmallCard*(*5*, *north*, *4*), *playSmallCard*(*2*, *south*, *5*).
- *playHonor*(*Card*, *Player*, *Time*) : le joueur *Player* (*north* ou *south*) joue une carte *Card* qui est un honneur (11 à 14) au temps *Time*. On a les instanciations suivantes pour la trajectoire 2 : *playHonor*(*13*, *south*, *2*), *playHonor*(*14*, *south*, *3*), *playHonor*(*12*, *north*, *6*).

Prédicats où le temps est un intervalle

Ce sont des prédicats qui sont vrais dans un intervalle de temps donné et faux à l'extérieur de l'intervalle.

- *dominant*(*Card*, *Player*, [*Inf*, *Sup*]) : une carte *Card* dans la main d'un joueur *Player* (parmi tous les joueurs) est plus grande que toutes les cartes dans les mains des autres joueurs dans l'intervalle de temps [*Inf*, *Sup*]. Dans la trajectoire 2, on a les instanciations suivantes : *dominant*(*13*, *south*, [*1*, *2*]), *dominant*(*14*, *south*, [*1*, *3*]), *dominant*(*4*, *north*, [*6*, *6*]), *dominant*(*12*, *north*, [*4*, *6*]), *dominant*(*10*, *south*, [*7*, *7*])

— *nextDominant*(*Card*, *Player*, [*Inf*, *Sup*]) : une carte *Card* est *nextDominant* dans la main de *Player* (parmi tous les joueurs) si les seules cartes plus hautes que *Card* dans les mains des autres joueurs sont les cartes qui sont *dominant*. Exception faite si le joueur *Player*′ ayant la carte la plus élevée en dessous des cartes *nextDominant* joue une carte, alors les cartes des *Player* qui n'ont aucune carte de *Player*′ entre elles et les *nextDominant* deviennent, elles aussi, *nextDominant*. Dans la trajectoire 3, on a les instanciations suivantes :

nextDominant(4, *north*, [4, 4]), *nextDominant*(5, *north*, [4, 4]),
nextDominant(12, *north*, [1, 4]), *nextDominant*(7, *east*, [5, 5]),
nextDominant(5, *north*, [6, 7])

Remarquons que le 4 et le 5 deviennent *nextDominant* brièvement au temps 4 car (i) elles sont dans la main de *north* et (ii) *south*, qui avait la carte la plus élevée en dessous des cartes *nextDominant*, a joué une carte, et il ne reste plus de cartes de *south* entre les cartes *nextDominant* et les cartes 4 et 5. Par la suite, comme *north* joue son 12, le 4 et le 5 ne sont plus *nextDominant*, car c'est *east* qui possède la *nextDominant*.

— *nbThreats*(*Card*, *Pp*, *Dn*, [*Inf*, *Sup*]) : le nombre de cartes *Dn* dans la main de l'adversaire plus grandes que la carte *Card* dans la main d'un joueur de la paire de partenaires *Pp* (*dec* ou *def*) dans l'intervalle de temps [*Inf*, *Sup*]. Dans la trajectoire 1, on a les instanciations suivantes : *nbThreats*(10, *dec*, 1, [1, 2]), *nbThreats*(10, *dec*, 1, [3, 4]), ...

— *lastThreat*(*Card1*, *Pp*, *Card2*, [*Inf*, *Sup*]) : l'adversaire de la paire de partenaires *pp* (*dec* ou *def*) n'a plus qu'une seule carte *Card*2 plus haute que la carte *Card*1 dans la main d'un joueur de *Pp*, dans l'intervalle de temps [*Inf*, *Sup*]. On a les instanciations suivantes pour la trajectoire 2 : *lastThreat*(*11*, *def*, *12*, [*4*, *5*]), *lastThreat*(*10*, *dec*, *11*, [*1*, *5*]), ...

— *nbSmallCards*(*Dn*, *Player*, [*Inf*, *Sup*]) le joueur *Player* (parmi tous les joueurs) possède un nombre *Dn* de petites cartes dans sa main dans l'intervalle de temps [*Inf*, *Sup*]. *Dn* peut prendre la valeur 0. On a les instanciations suivantes pour la trajectoire 2 : *nbSmallCards*(*3*, *north*, [*1*, *1*]), *nbSmallCards*(*2*, *east*, [*1*, *1*]), *nbSmallCards*(*1*, *west*, [*1*, *3*]), *nbSmallCards*(*2*, *south*, [*1*, *5*]), ...

— *nbHonors*(*Dn*, *Player*, [*Inf*, *Sup*]) le joueur *Player* (parmi tous les joueurs) possède un nombre *Dn* d'honneurs dans sa main dans l'intervalle de temps [*Inf*, *Sup*]. *Dn* peut prendre la valeur 0. On a les instanciations suivantes pour la trajectoire 2 : *nbHonors*(*2*, *south*, [*1*, *2*]),*nbHonors*(*1*, *west*, [*1*, *5*]),*nbHonors*(*1*, *north*, [*1*, *6*]), ...

— *minCardHand*(*Card*, *Player*, [*Inf*, *Sup*]) : la plus petite carte de la main du joueur *Player* (parmi tous les joueurs) est la carte *Card* dans l'intervalle de temps [*Inf*, *Sup*]. On a les instanciations suivantes pour la trajectoire 2 : *minCardHand*(*3*, *north*, [*1*, *1*]), *minCardHand*(*9*, *west*, [*1*, *3*]), *minCardHand*(*6*, *east*, [*1*, *4*]), *minCardHand*(*2*, *south*, [*1*, *5*]), ...

— *maxCardHand*(*Card*, *Player*, [*Inf*, *Sup*]) : la plus grande carte de la main du joueur *Player* (parmi tous les joueurs) est la carte *Card* dans l'intervalle de temps [*Inf*, *Sup*]. On a les instanciations suivantes pour la

trajectoire 2 : $maxCardHand(7, east, [1, 1])$, $maxCardHand(14, south, [1, 3])$, $maxCardHand(6, east, [2, 4])$, $maxCardHand(11, west, [1, 5])$, ...

— $defenderHasHonor(Card, Player, [Inf, Sup])$: la carte *Card* est un honneur chez un défenseur *Player* (parmi *east* et *west*) dans l'intervalle de temps $[Inf, Sup]$. On a l'instanciation suivante pour la trajectoire 2 : $defenderHasHonor(11, west, [1, 5])$.

Prédicats qui ne sont pas liés au temps

Ce sont les prédicats caractérisant les différents objets du jeu et qui sont vrais tout le long de la partie.

— $smallCard(Card)$: la carte *Card* est comprise entre 2 et 10.

— $honor(Card)$: la carte *Card* est comprise entre 11 et 14.

— $bigHonor(Card)$: la carte *Card* est comprise entre 13 et 14.

Les différents types d'arguments

Ici nous listons les types des arguments des prédicats et leurs instanciations possibles.

— *Card* : les cartes numérotées de 2 à 14 (avec le 11 = Valet, 12 = Dame, 13 = Roi, 14 = As).

— *Player* : les joueurs *north, South, East, West*

— *Pp* : paire de partenaires : *decl* (North/South) ou *def* (East/West)

— *Inf* : entier, borne inférieure de l'intervalle de vérité du prédicat.

— *Sup* : entier, borne supérieure de l'intervalle de vérité du prédicat.

— *Time* : entier, numéro du temps découpé par les actions du déclarant.

— *Dn* : entier, nombre de cartes.