HAL open science

Actes des 16es Journées d'Intelligence Artificielle Fondamentale

Zied Bouraoui, Anaelle Wilczynski

To cite this version:

Zied Bouraoui, Anaelle Wilczynski. Actes des 16es Journées d'Intelligence Artificielle Fondamentale : JIAF 2022. Plate-Forme Intelligence Artificielle, Association Française pour l'Intelligence Artificielle, 2022. hal-03781382

HAL Id: hal-03781382

https://hal.science/hal-03781382

Submitted on 4 Oct 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

AfIA

Association française
pour l'Intelligence Artificielle

JIAF

Journées d'Intelligence Artificielle Fondamentale

PFIA 2022

Table des matières

Zied Bouraoui, Anaëlle Wilczynski
Éditorial 5

Comité de programme 6

Session 1 : Choix social 7

Laurent Gourvès, Julien Lesca and Anaëlle Wilczynski
Sur l'Équité via la Sélection en Séquence pour l'Allocation de Biens Indivisibles 8

Chouaib Fellah, Sébastien Konieczny, Patricia Everaere and Ramon Pino Perez.
Borda, Annulation et Fusion de Croyances 10

Session 2 : IA explicable 22

Manuel Amoussou, Khaled Belahcene, Nicolas Maudet, Vincent Mousseau and Wassila Ouerdane.
Des explications par étapes pour le modèle additif 23

Khaled Belahcene, Jérome Gaigne and Sylvain Lagrue.
Fragment explicable par schémas d'arguments des affectations nécessaires dans un tri non compensatoire 35

Hénoïk Willot, Khaled Belahcene and Sébastien Destercke.
Explications de recommandations fondées sur des principes d'équité à l'aide de transferts 47

Session 3 : Intelligence ou bêtise artificielle ? 49

Jean Lieber, Jean-Guy Mailly, Pierre Marquis, Henri Prade and François Rollin.
Quelques réflexions autour de la notion de bêtise artificielle 50

Session 4 : Argumentation 61

Jean-Guy Mailly.
Sémantiques à base d'extensions pour les systèmes d'argumentation incomplets 62

Yohann Bacquey, Jean-Guy Mailly, Pavlos Moraitis and Julien Rossit.
Admissibility in Strength-based Argumentation : Complexity and Algorithms 64

Vivien Beuselinck, Jérôme Delobelle and Srdjan Vesic.
Restreindre l'Impact des Arguments Contradictoires dans les Sémantiques Graduées en Argumentation Abstraite 75

Christopher Leturc and Grégory Bonnet.
Raisonner sur l'éthique avec un cadre d'argumentation fondé sur une logique modale normale 77

Session 5 : Optimisation et planification 88

Junkang Li, Bruno Zanuttini, Véronique Ventos and Tristan Cazenave.
Generalisation of alpha-beta search for AND-OR graphs with partially ordered values 89

Sergej Scheck, Alexandre Niveau and Bruno Zanuttini.
A KC Map for Variants of Nondeterministic PDDL 100

Vincent Barichard and Igor Stéphan.
Une sémantique opérationnelle pour les QCHR 110

Session 6 : Représentation logique 120

Arnaud Kohler
Modelling a faillibilistic and perspectivistic reasoning 121

Sabine Frittella, Marta Bilkova, Ondrej Majer, Sajad Nazari and Daniil Kozhemiachenko.
Fonctions de croyances interprétées sur la logique de Belnap–Dunn 133

Philippe Balbiani and Cigdem Gencer.
Advanced languages of terms for ontologies 135

Éditorial

Journées d'Intelligence Artificielle Fondamentale

Les Journées d'Intelligence Artificielle Fondamentale (JIAF) constituent un rendez-vous annuel de la communauté francophone travaillant sur l'Intelligence Artificielle Fondamentale. Les thématiques de recherche sont relatives aux méthodes et outils fondamentaux de l'Intelligence Artificielle. Les journées sont composées d'exposés de synthèse, permettant à la communauté de découvrir des thématiques connexes au travers d'exposés de spécialistes, et de communications sélectionnées par le comité de programme.

Les thématiques de recherche des JIAF sont relatives aux méthodes et outils fondamentaux de l'Intelligence Artificielle :

- Définition de modèles de représentation des informations (croyances, connaissances, préférences, obligations et permissions, actions, incertitude, confiance, réputation) : langages des logiques classiques ou non classiques, modèles possibilistes, ontologies, langages à base de contraintes, représentations graphiques, etc.
- Définition et automatisation de raisonnements sur ces informations : raisonnement spatio-temporel, dynamique des informations, révision de croyances, fusion d'informations symboliques, raisonnement par argumentation, raisonnement causal, raisonnement abductif, raisonnement à partir de cas, etc.
- Mise au point de méthodes de codage des informations et d'algorithmes de traitement efficaces : compilation de connaissances, SAT, contraintes, ASP, etc.
- Modélisation formelle de l'interaction : entre utilisateurs et systèmes informatiques, entre entités informatiques autonomes (agents), intégration de ces deux aspects dans les divers agents conversationnels, agents de recherche, assistants personnels.
- Choix social, théorie des jeux, algorithmes pour les jeux.
- Pour des objectifs de décision, planification, ordonnancement, diagnostic, apprentissage et dans différents contextes d'application, comme par exemple le Web sémantique.

Ces 16èmes Journées d'Intelligence Artificielle Fondamentale (JIAF 2021) ont eu lieu du 30 juin au 1 juillet 2022, dans le cadre de la Plate-Forme Intelligence Artificielle (PFIA). Les éditions précédentes se sont déroulées à Bordeaux (en ligne - 2022), Angers (en ligne - 2020), Toulouse (2019), Amiens (2018), Caen (2017), Montpellier (2016), Rennes (2015), Angers (2014), Aix-en-Provence (2013), Toulouse (2012), Lyon (2011), Strasbourg (2010), Marseille (2009), Paris (2008) et Grenoble (2007).

Zied Bouraoui, Anaëlle Wilczynski

Comité de programme

Président

— Zied Bouraoui (CRIL, Univ Artois & CNRS) ;
— Anaëlle Wilczynski (MICS, CentraleSupélec, Université Paris-Saclay).

Membres

— Khaled Belahcène (Heudiasyc, Université de Technologie de Compiègne) ;
— Francesco Belardinelli (IBISC, Université d'Évry) ;
— Nawal Benabbou (LIP6, Sorbonne Université) ;
— Elise Bonzon (LIPADE, Université Paris Descartes) ;
— Tristan Cazenave (LAMSADE, Université Paris Dauphine) ;
— Nadia Creignou (LIS, Aix-Marseille Université) ;
— Tiago de Lima (CRIL, Univ Artois & CNRS) ;
— Sylvie Doutre (IRIT, Université Toulouse 1 Capitole) ;
— Jérôme Euzenat (LIG, INRIA) ;
— Hugo Gilbert (LAMSADE, Université Paris-Dauphine) ;
— George Katsirelos (MIAT, INRA) ;
— Sébastien Konieczny (CRIL, CNRS) ;
— Marie-Laure Mugnier (LIRMM, Université de Montpellier) ;
— Jean Lieber (LORIA, INRIA) ;
— Pierre Marquis (CRIL, IUF, Univ Artois & CNRS) ;
— Meltem Öztürk (LAMSADE, Université Paris Dauphine) ;
— Odile Papini (LIS, Aix-Marseille Université) ;
— Célia da Costa Pereira (I3S, Université Nice Sophia Antipolis) ;
— Laurent Perrussel (IRIT, Université Toulouse 1 Capitole) ;
— Sophie Pinchinat (IRISA, INRIA) ;
— Stéphanie Roussel (ONERA) ;
— Julien Rossit (LIPADE, Université Paris Decartes) ;
— Serena Villata (I3S, CNRS) ;
— Bruno Zanuttini (GREYC, UNICAEN).

Session 1 : Choix social

Actes JIAF 2022

Sur l'Équité via la Sélection en Séquence pour l'Allocation de Biens Indivisibles *

Laurent Gourvès[1] **Julien Lesca**[2] **Anaëlle Wilczynski**[3]

[1] Université Paris Dauphine-PSL, LAMSADE, 75016, Paris, France
[2] Huawei Technologies, Paris Research Center, Paris, France
[3] MICS, CentraleSupélec, Université Paris-Saclay, France
laurent.gourves@dauphine.fr julien.lesca@huawei.com
anaelle.wilczynski@centralesupelec.fr

Résumé

Parmi les critères d'équité pour l'allocation de ressources indivisibles à un groupe d'agents, certains sont basés sur des niveaux d'utilité minimaux à garantir à tous les agents. Ces niveaux peuvent provenir d'une méthode d'allocation spécifique, telle que le critère *maximin fair-share* issu du protocole *je coupe et tu choisis*. Nous proposons d'analyser des critères dont les niveaux d'utilité minimaux s'inspirent de la sélection en séquence, un protocole bien établi pour l'attribution de biens indivisibles. Nous étudions ces critères et leurs connexions avec d'autres critères d'équité connus, permettant ainsi d'enrichir l'état des connaissances sur la répartition équitable de biens indivisibles.

Abstract

Among the fairness criteria for allocating indivisible resources to a group of agents, some are based on minimum utility levels. These levels can come from a specific allocation method, such as maximin fair-share criterion which is based on the cut-and-choose protocol. We propose to analyze criteria whose minimum utility levels are inspired by picking sequences, a well-established protocol for allocating indivisible resources. We study these criteria and investigate their connections with known fairness criteria, enriching the understanding of fair allocation of indivisible goods.

1 Introduction

Le partage équitable de biens indivisibles est une question fondamentale, difficile, et largement étudiée dans le domaine de la prise de décision collective [2, 4]. De nombreux critères ont été proposés afin d'évaluer l'équité d'une allocation lorsque les agents expriment leurs préférences sur les ensembles de biens via des utilités additives. Citons par exemple l'absence d'envie (EF) [7], critère basé sur la comparaison qui demande qu'aucun agent ne préfère le lot assigné à un autre agent au sien. De nombreux autres critères d'équité imposent, sans besoin de comparaison, que chaque agent ait une utilité pour sa part qui est supérieure ou égale à un niveau d'utilité prédéfini, appelé *garantie équitable* [1]. Dans cette veine, on peut citer la proportionnalité (Prop) [9] où chaque agent doit avoir une utilité au moins égale à sa valeur pour l'ensemble des biens divisé par le nombre d'agents n. Une garantie équitable peut aussi être définie selon une procédure d'allocation donnée comme, par exemple, la *maximin-fair-share* (MMS) [6] ou la *min-max-fair-share* (mFS) [3] calculées grâce au protocole *je coupe et tu choisis*.

Dans cet article, nous définissons des critères d'équité dont la garantie équitable est calculée grâce au protocole d'allocation bien établi dit de sélection en séquence (*picking sequences (PS)*) [5]. Pour ce protocole, tous les biens sont initialement disponibles et, étant donné une séquence d'agents appelée *politique*, les agents sélectionnent tour à tour un objet parmi ceux qui restent. La séquence est dite sincère si le choix se porte invariablement sur l'objet préféré. La sélection en séquence est un bon candidat pour le partage de ressources car facile à comprendre et à mettre en œuvre. Même si l'allocation finale n'est pas construite avec un tel protocole, tout agent peut néanmoins prétendre que son utilité devrait être aussi bonne que celle résultant de la sélection en séquence munie de la politique qu'il a en tête.

Pour un entier p compris entre 1 et n, nous proposons un critère simple nommé PS_p dans lequel la garantie équitable de chaque agent est son utilité pour le sous-ensemble d'objets construit comme suit. Classez les objets du meilleur

*Ce résumé est issu de l'article [8] paru dans les actes de ADT 2021.

au pire selon la préférence de l'agent et conservez les éléments dont les rangs sont les multiples de p. Un agent serait doté d'un tel ensemble dans une sélection en séquence si ses positions dans la politique étaient des multiples de p, et si les autres agents avaient la même préférence que lui. En effet, sans connaître les préférences des autres, un agent peut supposer que, dans le pire des cas, tout le monde a le même classement d'objets que le sien. Le fait que l'agent apparaisse récursivement dans la politique s'inspire de la méthode *round robin*. Round robin appartient à la classe des politiques récursivement équilibrées (*recursively balanced* (RB)), où chaque séquence d'agents peut être divisée en tours au cours desquels tous les agents choisissent un objet exactement une fois. Dans PS_p, le paramètre p permet de passer progressivement d'un scénario très optimiste où tous les agents choisiraient en premier ($p = 1$) à un scénario plus pessimiste où tous choisiraient en dernier ($p = n$).

2 Contribution

Une allocation PS_p n'existe pas toujours si $p < n$. De plus, le problème de décision lié à l'existence d'une allocation PS_p est computationnellement difficile.

Théorème 1 *Déterminer si une allocation PS_p existe est* NP*-complet, même si :*
- *$m = 2n$ et $p < n$ est une constante, ou*
- *$p = 1$ et $n = 2$, impliquant la difficulté quand $p = n-1$.*

Néanmoins, lorsque $p = n$, l'existence d'une allocation PS_p est garantie, tout comme l'existence d'une allocation satisfaisant la relaxation à un bien près du concept PS_1.

Pour certains critères d'équité qui s'y prêtent, on note *crit*1 (resp., *crit*X) la relaxation à un bien près (resp., à tout bien près) d'un critère *crit*. Une allocation est *crit*1 (resp., *crit*X) si elle est *crit* en retirant un élément (resp., n'importe quel élément) de l'ensemble auquel on se compare.

Proposition 1 *Toute allocation résultant d'une sélection en séquence sincère avec politique RB est PS_n et $PS_1$1.*

Notons que la réciproque de la Proposition 1 n'est pas vraie, ainsi les allocations satisfaisant les critères PS n'émergent pas forcément d'une sélection en séquence. Par la Proposition 1, une allocation PS_p1 existe toujours pour tout $p \in [n]$. Ceci n'est pas le cas pour $\mathrm{PS}_p\mathrm{X}$, même si $p = n - 1$, pour tout nombre n d'agents. Puisqu'une allocation $\mathrm{PS}_1 1$ existe toujours et qu'une allocation $\mathrm{PS}_p\mathrm{X}$ n'existe pas toujours même pour $p = n - 1$, on peut se concentrer sur les relaxations qui rendent les critères les plus forts possibles, et donc considérer seulement $\mathrm{PS}_1 1$ et $\mathrm{PS}_1\mathrm{X}$.

Nous avons établi les relations d'implication entre nos nouveaux critères et des critères d'équité connus, afin d'enrichir l'échelle d'équité suivante [3] : EF ⇒ mFS ⇒ Prop ⇒ MMS. Ces relations sont résumées en Figure 1.

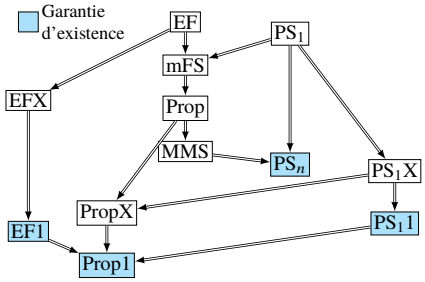


FIGURE 1 – Récapitulatif des relations entre critères d'équité ainsi que leur garantie d'existence. Un arc partant d'un critère A vers un critère B signifie que A implique B (A est plus fort que B). S'il n'y a pas de chemin orienté de A vers B alors A n'est pas plus fort que B.

Nous prouvons également que le critère PS_p fournit une $\frac{n-p+1}{n}$-approximation pour MMS, amenant PS_n au même rapport d'approximation $\frac{1}{n}$ qu'EF1.

Références

[1] Bogomolnaia, A., H. Moulin et R. Stong: *Guarantees in Fair Division : general or monotone preferences*. arXiv preprint arXiv :1911.10009, 2019.

[2] Bouveret, S., Y. Chevaleyre et N. Maudet: *Fair Allocation of Indivisible Goods*. Dans *Handbook of Computational Social Choice*, pages 284–310. 2016.

[3] Bouveret, S. et M. Lemaître: *Characterizing conflicts in fair division of indivisible goods using a scale of criteria*. Auton. Agents Multi-Agent Syst., 30(2) :259–290, 2016.

[4] Brams, S. J., P. H. Edelman et P. C. Fishburn: *Fair Division of Indivisible Items*. Theory and Decision, 55(2) :147–180, 2003.

[5] Brams, S. J. et A. D. Taylor: *The win-win solution : guaranteeing fair shares to everybody*. WW Norton & Company, 2000.

[6] Budish, E.: *The Combinatorial Assignment Problem : Approximate Competitive Equilibrium from Equal Incomes*. Journal of Political Economy, 119(6) :1061–1103, 2011.

[7] Foley, D. K.: *Resource Allocation and the Public Sector*. Yale Economic Essays, 7(1) :45–98, 1967.

[8] Gourvès, L., J. Lesca et A. Wilczynski: *On Fairness via Picking Sequences in Allocation of Indivisible Goods*. Dans *Proc. of ADT'21*, pages 258–272, 2021.

[9] Steinhaus, H.: *The Problem of Fair Division*. Econometrica, 16(1) :101–104, 1948.

Actes JIAF 2022

Borda, Annulation et Fusion de Croyances

Patricia Everaere[1] **Chouaib Fellah**[2] **Sébastien Konieczny**[2] **Ramón Pino Pérez**[2]

[1] CRIStAL - CNRS - Université de Lille - Centrale Lille - UMR 9189 - France
[2] CRIL - CNRS - Université d'Artois - France

patricia.everaere-caillier@univ-lille.fr
{fellah,konieczny,pinoperez}@cril.fr

Résumé

Dans ce travail, nous explorons les liens entre la règle de vote de Borda et les opérateurs de fusion de croyances en logique propositionnelle. Plus précisément, nous définissons deux familles d'opérateurs de fusion inspirés de la définition de la règle de vote de Borda. Nous introduisons également une notion d'annulation (*cancellation*) dans la fusion de croyances, inspirée de l'axiomatisation de la règle de vote de Borda proposée par Young. Cela nous permet de fournir une caractérisation de l'opérateur de fusion drastique.

Abstract

In this work, we explore the links between the Borda voting rule and belief merging operators. More precisely, we define two families of merging operators inspired by the definition of the Borda voting rule. We also introduce a notion of cancellation in belief merging, inspired by the axiomatization of the Borda voting rule proposed by Young. This allows us to provide a characterization of the drastic merging operator.

1 Introduction

Les opérateurs de fusion de croyances visent à synthétiser un ensemble de bases de croyances contradictoires afin de former une vision cohérente du monde. Pour cela, ces opérateurs bénéficient de la complémentarité des bases de croyances, qui permet de générer de nouvelles informations qui sont réparties dans les bases de croyances, tout en résolvant les conflits logiques entre ces bases.

La fusion de croyances [16, 17, 19, 21, 22, 28] peut être considérée comme l'intersection de deux thèmes de recherche. Le premier est la révision de croyances [1, 11, 12, 14], où le problème est de corriger les croyances de l'agent par une information (plus fiable). Dans la révision et la fusion, le problème est de trouver les informations les plus plausibles compte tenu de l'entrée. La différence est que pour la révision, l'entrée est une base de croyances unique, alors que pour la fusion, il s'agit d'un ensemble de bases. Les principes régissant ces deux problèmes sont donc étroitement liés. Le deuxième sujet de recherche est la décision du groupe telle qu'étudié dans la théorie du choix social [2, 3], et notamment dans les méthodes de vote. Le but d'une méthode de vote est de définir une préférence sociale (une préférence pour l'ensemble du groupe) à partir d'un ensemble de préférences individuelles données par les individus. De même, la fusion des croyances vise à définir les "croyances du groupe" à partir des croyances fournies par plusieurs sources. Ainsi, le choix social (en particulier les méthodes de vote) et la fusion des croyances partagent la préoccupation de savoir comment prendre en compte fidèlement l'ensemble des entrées (préférences pour le vote, croyances pour la fusion).

Les liens formels entre la fusion de croyances et la révision de croyances sont bien connus. Par exemple, il est facile de montrer que la révision de croyances peut être considérée comme un cas particulier de fusion de croyances lorsqu'il existe une seule base de croyances dans le profil. [17].

Les liens entre la fusion de croyances et la théorie du choix social ont également été étudiés. Par exemple dans [18] il y a une comparaison entre les fonctions de choix social et les opérateurs de fusion de croyances. Certaines questions issues du choix social et des méthodes de vote ont également été investiguées dans le contexte de la fusion de croyances. Par exemple, Everaere, Konieczny et Marquis étudient le problème de manipulation pour les opérateurs de fusion [6], ils étudient aussi le problème de la découverte de la vérité (*truth-tracking*) et une généralisation du théorème du jury de Condorcet pour les opérateurs de fusion de croyances est donnée [8]. Certains opérateurs de fusion de croyances égalitaires ont été proposés en utilisant des techniques standard issues du choix social [9]. Une

généralisation du théorème d'impossibilité d'Arrow dans le cadre de la fusion de croyances est donnée dans [25]. Enfin, des travaux sur l'agrégation de jugements et son lien avec la fusion de croyances ont été réalisés [10]. L'agrégation des jugements peut être considérée comme un problème intermédiaire entre le vote et la fusion de croyances. Comme dans la fusion de croyances, l'agrégation de jugements [5, 20, 23, 24, 27] fonctionne à partir de formules logiques, et comme en vote (et contrairement à la fusion), il y a un agenda (un ensemble de formules) sur lequel le groupe doit décider.

Dans ce travail, nous voulons étudier un lien plus direct et plus technique entre certains opérateurs de fusion en logique propositionnelle et la méthode de vote de Borda. En fait, une large classe d'opérateurs de fusion de croyance est la classe des opérateurs de fusion basés sur une distance entre interprétations [15, 19]. Ces opérateurs utilisent une distance afin de générer une évaluation de la plausibilité de chaque monde possible par rapport à chaque base de croyances à partir du profil d'entrée, puis ils utilisent une fonction d'agrégation afin d'obtenir une évaluation globale de la plausibilité de chaque monde possible.

Plusieurs fonctions d'agrégation peuvent être considérées afin de donner lieu à des opérateurs au comportement rationnel, comme la somme [22, 28], le leximax [17], la somme des puissances n^{ième} [18], le leximin [7], etc.

Lorsque la somme est utilisée comme fonction d'agrégation, les opérateurs correspondants sont assez proches de la règle de vote de Borda [4, 26, 29]. Dans les deux cas, les informations fournies par les sources individuelles génèrent un score (une évaluation numérique) pour chaque alternative, qui sont ensuite agrégés à l'aide de la somme.

Il existe quelques différences entre les deux processus. Pour la règle de vote de Borda, la source fournit un ordre (linéaire). Pour la fusion de croyances, la source fournit une base de croyances. Une distance entre interprétations permet d'obtenir un ordre à partir de la base de croyances, qui est en fait un préordre total (nous verrons que cela a un impact sur les résultats formels).

Mais la similitude est suffisamment grande pour justifier des investigations plus approfondies. En particulier, il existe des résultats formels intéressants sur la règle de vote de Borda que nous voulons étudier dans le contexte de la fusion de croyances. Cela nous permet de définir deux nouvelles familles d'opérateurs de fusion de croyances et un résultat de caractérisation.

Essentiellement, il existe deux définitions de la règle de vote de Borda. Ces deux définitions sont équivalentes lorsque les ordres linéaires sont considérés. Mais ce n'est plus le cas sur les préordres totaux. Nous obtenons ainsi deux définitions différentes, appliquées à la fusion de croyances et qui nous donnent de nouveaux opérateurs de fusion.

Un autre résultat formel intéressant est que la règle de vote de Borda est caractérisée par une propriété d'annulation [29]. Cette propriété concerne vraiment les ordres et ne peut pas être directement traduite en fusion de croyances. Mais cette idée peut nous donner des contreparties intéressantes. Nous étudions cette classe de propriétés d'annulation. Et nous donnons une caractérisation d'un opérateur de fusion de croyances utilisant l'une de ces propriétés.

Le plan du papier est le suivant. La section suivante fournit les notions et notations requises pour l'article. Puis dans la section 3, on rappelle la définition des opérateurs de fusion IC, le théorème de représentation en termes de préordres de plausibilité, et la définition des opérateurs de fusion basés sur la distance. Dans la section 4, nous donnons les deux définitions de la règle de vote de Borda, nous montrons qu'elles ne sont pas équivalentes lorsque les préordres sont utilisés, et nous donnons la propriété d'annulation. Dans la section 5, nous donnons la définition des deux familles d'opérateurs de fusion à la Borda. Ces opérateurs ont besoin d'une fonction qui associe un préordre à chaque base de croyances. Cela peut être fourni, par exemple, par une distance, par un opérateur de fusion IC ou par un opérateur de révision de croyance. Ensuite, dans la section 7 nous étudions les propriétés de ces deux familles d'opérateurs. Et dans la section 8 nous étudions les propriétés d'annulation et nous fournissons une caractérisation de l'opérateur $\Delta^{d_D,f}$. Nous concluons dans la section 9.

2 Préliminaires

On considère un langage propositionnel $\mathcal{L}$ défini à partir d'un ensemble fini de variables propositionnelles $\mathcal{P}$ et les connecteurs standards. Une interprétation ω est une fonction totale de $\mathcal{P}$ dans $\{0, 1\}$. L'ensemble de toutes les interprétations est noté Ω. Une interprétation ω est un modèle de la formule (base de croyances) $\varphi \in \mathcal{L}$ si et seulement si elle la rend vraie au sens usuel. $[[\varphi]]$ représente l'ensemble des modèles de la formule φ, i.e., $[[\varphi]] = \{\omega \in \Omega : \omega \models \varphi\}$. Lorsque M est un ensemble de modèles, On note φ_M une formule telle que $[[\varphi_M]] = M$.

Un *profil* E est un vecteur de formules $E = (\varphi_{i_1}, \ldots, \varphi_{i_n})$ où chaque i_j représente un agent/individu/source (par conséquent, différents agents/individus/sources sont autorisés à présenter des formules identiques), il représente donc un groupe de n agents. On note $\mathcal{E}$ l'ensemble des profils. $\bigwedge E$ représente la conjonction des formules de $E = (\varphi_{i_1}, \ldots, \varphi_{i_n})$, i.e., $\bigwedge E = \varphi_{i_1} \wedge \ldots \wedge \varphi_{i_n}$. Un profil E est dit cohérent si et seulement si $\bigwedge E$ est cohérent. Deux profils $E_1 = (\varphi_{i_1}, \ldots, \varphi_{i_n})$, $E_2 = (\varphi_{j_1}, \ldots, \varphi_{j_n})$ sont équivalents, dénoté $E_1 \equiv E_2$, s'il existe une fonction bijective σ de $\{i_1, \ldots, i_n\}$ sur $\{j_1, \ldots, j_n\}$ telle que pour chaque $i_k \in \{i_1, \ldots, i_n\}$, $\varphi_{i_k} \equiv \varphi_{\sigma(i_k)}$. La concaténation des profils est notée $\sqcup$. Par abus de notation on écrira $\varphi \sqcup E$ au lieu de $(\varphi) \sqcup E$. On note par E^n le profil obtenu en concaténant E n fois. Plus précisément, $E^n = E_1 \sqcup \ldots \sqcup E_n$ où chaque

E_i est équivalent à E. La notation $\neg E$ représente le profil composé des négations des bases de croyances du profil $E = (\varphi_{i_1}, \ldots, \varphi_{i_n})$, i.e. $\neg E = (\neg\varphi_{i_1}, \ldots, \neg\varphi_{i_n})$.

Si A est un ensemble, on dénote $|A|$ la cardinalité de A. Le symbole $\subseteq$ dénotera l'inclusion et $\subset$ l'inclusion stricte, i.e., $A \subset B$ si et seulement si $A \subseteq B$ et $A \neq B$.

Un préordre total sur un ensemble X est une relation binaire $\preceq$ qui est réflexive, transitive et totale, $\prec$ dénote la relation stricte associée définie par $\omega \prec \omega'$ si et seulement si $\omega \preceq \omega'$ et $\omega' \not\preceq \omega$, $\simeq$ dénote la relation d'indifférence définie par $\omega \simeq \omega'$ si et seulement si $\omega \preceq \omega'$ et $\omega' \preceq \omega$. Soit $\preceq$ un préordre total sur X, et $B \subseteq X$, alors $\min(B, \preceq) = \{b \in B : \nexists x \in B,\ x \prec b\}$.

Lorsque $\preceq$ est un préordre total sur X, le *rang canonique* $r_{\preceq} : X \to \mathbb{N}$ associé à $\preceq$ est obtenu en posant $r_{\preceq}(x)$ comme l'entier maximal k tel qu'il existe $x_0, x_1, \ldots, x_k$ dans X avec $x_k = x$ et $x_i \prec x_{i+1}$ pour $i = 0, \ldots, k-1$.

Soit $X = \{x_1, \ldots, x_m\}$ et un vecteur $P = (\preceq_1, \ldots, \preceq_n)$ de préordres totaux sur X, nous définissons $\pi_{ij}(P)$, pour tout $i, j \in \{1, \ldots, m\}$, par $\pi_{ij}(P) = |\{1 \leqslant k \leqslant n : x_i \prec_k x_j\}|$. $\pi_{ij}(P)$ représente le nombre de sources dans le vecteur P qui préfèrent x_i à x_j. Notez que $\pi_{ii}(P) = 0$ pour tout $i \in \{1, \ldots, m\}$, et que $\forall i, j \in \{1, \ldots, m\}, \pi_{ij}(P) \geqslant 0$.

Les opérateurs de fusion que nous allons considérer sont des fonctions de l'ensemble des profils et de l'ensemble des formules propositionnelles (qui représentent les contraintes d'intégrité, notées μ) dans un ensemble de formules, i.e. $\Delta : \mathcal{E} \times \mathcal{L} \to \mathcal{L}$. Nous utiliserons la notation $\Delta_\mu(E)$ au lieu de $\Delta(E, \mu)$. Nous écrivons aussi $\Delta(E)$ au lieu de $\Delta_\top(E)$ où $\top$ est une formule tautologique arbitraire.

3 Fusion IC

Rappelons dans cette section les postulats pour la fusion IC, le théorème de représentation et la définition des opérateurs basés sur une distance [17].

Définition 1 *Un opérateur de fusion Δ est appelé un* opérateur de fusion IC *s'il satisfait les postulats (IC0-IC8).*

- **(IC0)** $\Delta_\mu(E) \models \mu$
- **(IC1)** *Si μ est cohérent, alors $\Delta_\mu(E)$ est cohérent*
- **(IC2)** *Si $\bigwedge E \wedge \mu$ est cohérent, alors $\Delta_\mu(E) \equiv \bigwedge E \wedge \mu$*
- **(IC3)** *Si $E_1 \equiv E_2$ et $\mu_1 \equiv \mu_2$, alors $\Delta_{\mu_1}(E_1) \equiv \Delta_{\mu_2}(E_2)$*
- **(IC4)** *Si $\varphi_1 \models \mu$, $\varphi_2 \models \mu$ et $\Delta_\mu((\varphi_1, \varphi_2)) \wedge \varphi_1$ est cohérent, alors $\Delta_\mu((\varphi_1, \varphi_2)) \wedge \varphi_2$ est cohérent*
- **(IC5)** $\Delta_\mu(E_1) \wedge \Delta_\mu(E_2) \models \Delta_\mu(E_1 \sqcup E_2)$
- **(IC6)** *Si $\Delta_\mu(E_1) \wedge \Delta_\mu(E_2)$ est cohérent, alors $\Delta_\mu(E_1 \sqcup E_2) \models \Delta_\mu(E_1) \wedge \Delta_\mu(E_2)$*
- **(IC7)** $\Delta_{\mu_1}(E) \wedge \mu_2 \models \Delta_{\mu_1 \wedge \mu_2}(E)$
- **(IC8)** *Si $\Delta_{\mu_1}(E) \wedge \mu_2$ est cohérent, alors $\Delta_{\mu_1 \wedge \mu_2}(E) \models \Delta_{\mu_1}(E) \wedge \mu_2$*

Ces postulats capturent le comportement souhaitable pour les opérateurs de fusion (voir [17, 19] pour plus de détails).

Définition 2 *Une fonction $E \mapsto \preceq_E$ qui associe à chaque profil E un préordre total sur les mondes $\preceq_E$ est appelé un assignement syncrétique si :*

1. *Si $\omega, \omega' \models \bigwedge E$, alors $\omega \simeq_E \omega'$*
2. *Si $\omega \models \bigwedge E$ et $\omega' \not\models \bigwedge E$, alors $\omega \prec_E \omega'$*
3. *Si $E_1 \equiv E_2$, alors $\preceq_{E_1} = \preceq_{E_2}$*
4. *$\forall \omega \models \varphi_1$, $\exists \omega' \models \varphi_2$, $\omega' \preceq_{\varphi_1 \sqcup \varphi_2} \omega$*
5. *Si $\omega \preceq_{E_1} \omega'$ et $\omega \preceq_{E_2} \omega'$, alors $\omega \preceq_{E_1 \sqcup E_2} \omega'$*
6. *Si $\omega \preceq_{E_1} \omega'$ et $\omega \prec_{E_2} \omega'$, alors $\omega \prec_{E_1 \sqcup E_2} \omega'$*

Tout opérateur de fusion IC peut être représenté par un assignement syncrétique :

Théorème 1 *[17] Un opérateur de fusion Δ est un opérateur de fusion IC si et seulement si il existe un assignement syncrétique qui associe à chaque profil E un préordre total $\preceq_E$ tel que pour chaque formule μ, $[[\Delta_\mu(E)]] = \min([[\mu]], \preceq_E)$.*

Observation 1 Une analyse de la preuve de ce théorème révèle que :

• Les postulats (IC0), (IC1), (IC7), (IC8) suffisent pour obtenir la représentation par un assignement $E \mapsto \preceq_E$ où $\preceq_E$ est un préordre total sur les interprétations.

• Modulo les postulats mentionnés au point précédent (qui permettent la représentation) on a : (IC2) est équivalent aux propriétés [1] 1 et 2 ; (IC3) est équivalent à la propriété 3 ; (IC4) est équivalent à la propriété 4 ; (IC5) est équivalent à la propriété 5 ; (IC6) est équivalent à la propriété 6.

Une façon intéressante de définir des opérateurs de fusion IC concrets est d'utiliser une distance d entre les interprétations et une fonction d'agrégation f (voir [19]), afin de construire un assignement syncrétique, puis d'utiliser l'équation du Théorème 1 pour définir l'opérateur. Plus précisément :

Définition 3 *Soit $d : \Omega \times \Omega \to \mathbb{R}$ et $f : \bigcup_n \mathbb{R}^n \to \mathbb{R}$ respectivement une distance [2] entre les interprétations et une fonction d'agrégation. Soit $E = (\varphi_1, \ldots, \varphi_n)$ un profil, on pose :*

— $d(\omega, \varphi_i) = \min_{\omega' \models \varphi_i} d(\omega, \omega')$

— $d^f(\omega, E) = f_{\varphi_i \in E}\{d(\omega, \varphi_i)\}$

— $\omega \preceq_E^{d,f} \omega'$ *ssi* $d^f(\omega, E) \leqslant d^f(\omega', E)$

*Étant donné un profil E et une formule μ, l'*opérateur de fusion basé sur la distance $\Delta_\mu^{d,f}(E)$ *est défini comme* $[[\Delta_\mu^{d,f}(E)]] = \min([[\mu]], \preceq_E^{d,f})$.

1. Les propriétés sont celles des assignements syncrétiques (Définition 2).

2. Formellement, seule une pseudo-distance est nécessaire (l'inégalité triangulaire n'est pas nécessaire).

Les distances usuelles considérées sont la distance drastique ($d_D(\omega, \omega') = 0$ si $\omega = \omega'$ et 1 sinon), et la distance de Hamming ($d_H(\omega, \omega') = k$ si ω et ω' diffèrent sur k variables). Les fonctions d'agrégation habituelles sont la somme, le leximax, la somme des puissances n^{ième}, le leximin etc. Toutes ces fonctions d'agrégation satisfont les propriétés qui permettent d'obtenir des opérateurs de fusion IC [15, 19]. Quand la distance drastique d_D est utilisée [3], nous désignons les opérateurs de fusion IC $\Delta^{d_D,f}$ par l'*opérateur drastique de fusion* (ces opérateurs sont tous équivalents à $\Delta^{d_D,\Sigma}$, voir le corollaire 1).

4 Règles de Borda

Nous avons délibérément choisi le pluriel dans le titre de cette section car dans la théorie du choix social, il existe deux façons de définir la règle de Borda. Elles sont équivalentes lorsque les préférences des électeurs sont des ordres linéaires. Mais elles ne sont pas nécessairement équivalentes si les préférences des électeurs sont des préordres totaux comme nous le verrons ensuite.

Nous rappelons quelques notions de base sur la théorie du choix social avant d'introduire les règles de Borda. On désigne par $N = \{1, 2, \ldots, n\}$ l'ensemble fini d'électeurs et $X = \{x_1, x_2, \ldots, x_m\}$ l'ensemble fini d'alternatives. Le bulletin de vote (ou la préférence) d'un électeur i sera donné par un préordre total $\preceq_i$ sur X. $x_1 \preceq_i x_2$ signifie que l'agent i préfère l'alternative x_1 à l'alternative x_2. [4] Un *profil* est un n-uplet $P = (\preceq_1, \preceq_2, \ldots, \preceq_n)$, qui recueille les préférences de tous les électeurs sur X, de manière ordonnée. Nous dirons que P est un *profil linéaire* si chaque $\preceq_i$ dans P est un ordre linéaire sur X.

Une *fonction de choix de bien-être* est une fonction W associant à un profil P un préordre total $\preceq_P$ sur X, la préférence sociale associée au profil P.

Définissons maintenant la règle de score de Borda :

Définition 4 *Étant donné un profil $P = (\preceq_1, \ldots, \preceq_n)$ sur un ensemble d'alternatives $X = \{x_1, \ldots, x_m\}$ et une alternative $a \in X$, le score Borda de a est l'entier $r^s_P(a) = \sum_{i=1}^n r_{\preceq_i}(a)$. Ensuite, $\preceq^s_P$ est défini en posant $a \preceq^s_P b$ si $r^s_P(a) \leqslant r^s_P(b)$. Enfin, la règle de score de Borda, B^s,* est la fonction de choix de bien-être $B^s(P) = \preceq^s_P$.

Exemple 1 *Considérons l'ensemble des alternatives $X = \{x_1, x_2, x_3\}$ et le profil suivant $P = (\preceq_1, \preceq_2)$ où $x_1 \prec_1 x_2 \simeq_1 x_3$ et $x_2 \prec_2 x_1 \prec_2 x_3$. Il est facile de voir que $r^s_P(x_1) = 1 = r^s_P(x_2)$ et $r^s_P(x_3) = 3$. Donc, $x_1 \simeq^s_P x_2 \prec^s_P x_3$.*

3. En fait les résultats sont vrais en définissant $d_D(\omega, \omega') = 0$ si $\omega = \omega'$ et $a \in \mathbb{R}^{+*}$ sinon. Dans la suite on appellera distance drastique n'importe laquelle de ces distances.

4. Nous faisons ce choix, à l'inverse de ce qui se fait habituellement dans le choix social, afin d'être cohérent avec les ordres utilisés pour la révision des croyances.

Nous définissons maintenant la règle de beta-Borda (ou règle de Borda par comparaison par paires).

Définition 5 *Soit X l'ensemble d'alternatives $\{x_1, \ldots, x_m\}$. Le score par comparaison par paires (β) de l'alternative x_i dans le profil $P = (\preceq_1, \ldots, \preceq_n)$ est défini comme suit : $\beta_i(P) = \sum_{j=1}^m (\pi_{ij}(P) - \pi_{ji}(P))$.*
Lorsque P est clair selon le contexte, nous écrivons simplement β_i au lieu de $\beta_i(P)$.
Maintenant, on pose $x_i \preceq^\beta_P x_j$ ssi $\beta_i \geqslant \beta_j$. Finalement, la règle de beta-Borda B^β est définie par $B^\beta(P) = \preceq^\beta_P$.

Exemple 2 *Considérons $X = \{x_1, x_2, x_3\}$ et P comme dans l'exemple 1. Il est facile de voir que $\beta_1(P) = 2$, $\beta_2(P) = 1$ et $\beta_3(P) = -3$, donc $x_1 \prec^\beta_P x_2 \prec^\beta_P x_3$.*

Nous pouvons maintenant affirmer que les deux définitions sont équivalentes pour les ordres linéaires, mais pas pour les préordres totaux. Ce résultat est connu dans le choix social, il est mentionné dans [29], mais sans la preuve. Nous avons dons ajouté cette preuve pour être complet.

Proposition 1 *Pour chaque profil linéaire P on a $B^s(P) = B^\beta(P)$. Cette égalité n'est pas vraie en général,* i.e. *il existe un profil P tel que $B^s(P) \neq B^\beta(P)$.*

Preuve : Nous observons qu'une autre façon de calculer $\sum_{j=1}^m \pi_{ij}(P)$ est, pour chaque $k \in \{1, \ldots, n\}$, de compter le nombre d'alternatives y t.q. $x_i \prec_k y$ puis faire la somme de ces nombres pour $k \in \{1, \ldots, n\}$. De même, une autre façon de calculer $\sum_{j=1}^m \pi_{ji}(P)$ est, pour chaque $k \in \{1, \ldots, n\}$, de compter le nombre d'alternatives y t.q. $y \prec_k x_i$ puis de faire la somme de ces nombres pour $k \in \{1, \ldots, n\}$. Notez que si $\preceq_k$ est un ordre linéaire, le nombre d'alternatives y t.q. $x_i \prec_k y$ est exactement $m - 1 - r_{\preceq_k}(x_i)$ et le nombre d'alternatives y t.q. $y \prec_k x_i$ est exactement $r_{\preceq_k}(x_i) - 1$. Donc,
$\sum_{j=1}^m \pi_{ij}(P) = \sum_{k=1}^n m - 1 - r_{\preceq_k}(x_i) = nm - n - r^s_P(x_i)$,
$\sum_{j=1}^m \pi_{ji}(P) = \sum_{k=1}^n r_{\preceq_k}(x_i) - 1 = r^s_P(x_i) - n$ et
$\beta_i = \sum_{j=1}^m \pi_{ij}(P) - \sum_{j=1}^m \pi_{ji}(P) = nm - 2r^s_P(x_i)$. Donc, $x_i \preceq^\beta_P x_j$ ssi $\beta_i \geqslant \beta_j$ ssi $nm - 2r^s_P(x_i) \geqslant nm - 2r^s_P(x_j)$ ssi $r^s_P(x_i) \leqslant r^s_P(x_j)$ ssi $x_i \preceq^s_P x_j$.

Les calculs des exemples 1 et 2 montrent que $B^s(P) \neq B^\beta(P)$. ∎

Un résultat intéressant de Young ([29]) est une caractérisation axiomatique de la règle de Borda pour les profils linéaires. Introduisons quatre axiomes pour une fonction de bien-être social W :
Neutralité : Pour chaque permutation σ sur les alternatives, $W(\sigma(P)) = \sigma(W(P))$.
Fidélité : Pour chaque profil singleton $(\preceq)$, $W((\preceq)) = \preceq$.
Cohérence : Supposons que P_1 et P_2 sont des profils sur des ensembles d'électeurs disjoints. Si $\min(W(P_1)) \cap \min(W(P_2)) \neq \emptyset$, alors $\min(W(P_1 \sqcup P_2)) = \min(W(P_1)) \cap \min(W(P_2))$.

Annulation : Si pour chaque paire d'alternatives x_i, x_j on a $\pi_{ij}(P) = \pi_{ji}(P)$ alors $W(P)$ est le préordre total plat [5].

Théorème 2 *[29] Restreint à des profils linéaires, la seule fonction de bien-être social satisfaisant la neutralité, la fidélité, la cohérence et l'annulation est la règle de Borda B^s.*

Ainsi, sur les profils linéaires, les deux définitions des règles de Borda coïncident, et cette règle est caractérisée exactement par ces quatre axiomes.

5 Fusion à la Borda

Dans cette section, nous utiliserons les deux définitions de la précédente section (qui diffèrent dans le cas général d'après la proposition 1) afin de définir des opérateurs de fusion. Pour cela, nous avons besoin d'une fonction génératrice qui associe un préordre total sur les interprétations à n'importe quelle base φ :

Définition 6 *On appelle* fonction génératrice *toute fonction $\triangleleft$ qui associe à chaque base de croyances φ un préordre total sur les interprétations $\preceq_\varphi$ tel que* $\min(\Omega, \preceq_\varphi) = [[\varphi]]$ *et $\preceq_\varphi = \preceq_{\varphi'}$ quand $\varphi \equiv \varphi'$.*

Notez que $\triangleleft$ peut être fourni par un opérateur de révision $\circ$ ou un opérateur de fusion Δ. En effet des théorèmes de représentation en termes d'assignements fidèles/syncrétiques permettent d'associer un préordre à chaque base de croyances φ. Notez qu'une distance d entre interprétations est aussi une manière de définir un préordre pour chaque base de croyances φ, avec $\omega \preceq_\varphi \omega'$ si et seulement si $d(\omega, \varphi) \leqslant d(\omega', \varphi)$.

Une fois qu'une fonction génératrice $\triangleleft$ est considérée, il est facile de définir un profil de préordres sur Ω (les éléments de Ω sont considérés comme des alternatives) $P_E = (\preceq_{\varphi_1}, \ldots, \preceq_{\varphi_n})$ à partir d'un profil de bases de croyances $E = (\varphi_1, \ldots, \varphi_n)$.

Définition 7 *Étant donné une fonction génératrice $\triangleleft$ et un profil $P_E^{\triangleleft} = (\preceq_{\varphi_1}, \ldots, \preceq_{\varphi_n})$ sur Ω associé à $E = (\varphi_1, \ldots, \varphi_n)$ avec $\triangleleft$, nous considérons l'assignement $P_E^{\triangleleft} \mapsto \preceq^{\beta}_{P_E}$ (voir Définition 5). L'*opérateur de fusion β-Borda $\Delta^{\beta_\triangleleft}$ *est défini par :*

$$[[\Delta^{\beta_\triangleleft}_\mu(E)]] = \min([[\mu]], \preceq^{\beta}_{P_E})$$

On dit qu'un opérateur de fusion est basé sur la comparaison par paires s'il est identique à sa version de fusion β-Borda :

Définition 8 *Soit Δ un opérateur de fusion et $\triangleleft_\Delta$ la fonction génératrice associée à Δ. On dit que Δ est* basé sur des comparaisons par paires *ssi $\Delta = \Delta^{\beta_{\triangleleft_\Delta}}$.*

5. Le préordre total plat sur X, est le préordre total unique $\preceq$ tel que pour toutes x, y in X, on a $x \preceq y$.

De manière analogue, en utilisant la fonction de score de bien-être de Borda B^s de la Définition 4, on peut définir un autre opérateur de fusion, qui transforme chaque préordre en une fonction de score pour chaque base, et le score d'une interprétation est la somme de tous ses scores. Cette façon de procéder peut être liée aux opérateurs $\Delta^{d,\Sigma}$ usuels, mais au lieu d'utiliser une distance, nous utilisons une information plus qualitative fournie par le préordre :

Définition 9 *Étant donné une fonction génératrice $\triangleleft$, un profil $P_E = (\preceq_{\varphi_1}, \ldots, \preceq_{\varphi_n})$ sur Ω associé à $E = (\varphi_1, \ldots, \varphi_n)$ avec $\triangleleft$, on considère l'assignement $E \mapsto \preceq^{s}_{P_E}$ (voir Définition 4). L'*opérateur de fusion s-Borda $\Delta^{s_\triangleleft}$ *est défini par :*

$$[[\Delta^{s_\triangleleft}_\mu(E)]] = \min([[\mu]], \preceq^{s}_{P_E})$$

6 Exemple

Dans cette section, nous donnons un exemple pour illustrer la différence de comportement entre les opérateurs β-Borda, les opérateurs s-Borda et les opérateurs de fusion classiques basés sur la distance.

Considérez le profil $E = (\varphi_1, \varphi_2, \varphi_3)$, avec $[[\varphi_1]] = \{111, 001\}$, $[[\varphi_2]] = \{011, 110\}$, et $[[\varphi_3]] = \{000\}$. Nous ne considérons aucune contrainte d'intégrité, alors $\mu = \top$.

Nous considérerons l'opérateur de fusion basé sur la distance $\Delta^{d^\rho_H,\Sigma}$,c'est-à-dire l'opérateur pour lequel la somme est la fonction d'agrégation, et la distance est la distance de Hamming pondérée d^ρ_H, définie ci-dessous :

Définition 10 *Soit $\rho : \mathcal{P} \to \mathbb{R}^*$ une fonction qui associe à toute variable propositionnelle du langage un poids strictement positif. Alors la distance de Hamming pondérée d^ρ_H entre deux interprétations ω et ω' est défini comme :*

$$d^\rho_H(\omega, \omega') = \sum_{x \in \mathcal{P}, \omega(x) \neq \omega'(x)} \rho(x)$$

Utiliser une telle distance est très naturel lorsque toutes les variables propositionnelles n'ont pas la même importance. C'est le cas si la base peut être découpée en plusieurs thèmes (encodés par des sous-ensembles de variables propositionnelles), certains thèmes étant plus importants que d'autres.

Notez que la distance de Hamming standard est obtenue dans le cas particulier où tous les poids sont égaux à 1.

Dans cet exemple, nous utilisons la fonction de poids $\rho = (1, 3, 6)$. Les résultats de la fusion du profil E sont donnés pour l'opérateur $\Delta^{d^\rho_H,\Sigma}$, pour ses versions s-Borda et β-Borda, où la fonction génératrice $\triangleleft$ est donnée par $\Delta^{d^\rho_H,\Sigma}$.

Les tables 1, 2, 3 résument les calculs respectifs. La première colonne donne les mondes possibles. Pour les tables 1 et 2 les colonnes des φ_i (i = 1. . .3) donnent pour chaque interprétation ω la valeur $d^\rho_H(\omega, \varphi_i)$(pour s-Borda les valeurs exactes de la distance ne sont pas utilisées, seul l'ordre

de plausibilité de chaque base est considéré), la colonne la plus à droite donne pour chaque interprétation ω la valeur de $d_H^{\rho,\Sigma}(\omega, E)$. Pour la table 3 la colonne π_{ij} donne la valeur $\pi_{ij}(E)$(Le nombre de sources parmi φ_1, φ_2 et $\varphi_3 \in E$ qui préfèrent l'interprétation courante ω aux autres interprétations), la colonne π_{ji} donne la valeur $\pi_{ji}(E)$(Le nombre de sources parmi φ_1, φ_2 et $\varphi_3 \in E$ qui préfèrent les autres interprétations à l'interprétation courante ω),

On peut alors vérifier que le résultat pour l'opérateur basé sur la distance est $[[\Delta^{d_H^{\rho},\Sigma}(E)]] = \{001\}$. Alors que pour sa version s-Borda le résultat est : $[[\Delta^{s_\triangleleft}(E)]] = \{000, 110\}$. Finalement, en utilisant la version β-Borda de cet opérateur, on calcule les positions relatives de tous les couples d'interprétations via le β_i, et on obtient comme résultat : $[[\Delta^{\beta_\triangleleft}(E)]] = \{110\}$.

Notons enfin qu'il n'y a pas, en général, de corrélation logique entre les opérateurs $\Delta^{d_H^{\rho},\Sigma}$, $\Delta^{s_\triangleleft}$ et $\Delta^{\beta_\triangleleft}$. Dans cet exemple on voit déjà que c'est le cas entre $\Delta^{d_H^{\rho},\Sigma}$ et les deux autres opérateurs, mais on a $\Delta^{\beta_\triangleleft}(E) \vdash \Delta^{s_\triangleleft}(E)$. Pour voir que ce n'est pas toujours le cas faisons la fusion sous les contraintes $[[\mu]] = \{010, 111\}$. D'après les tables 2 et 3, nous pouvons facilement voir que $[[\Delta_\mu^{s_\triangleleft}(E)]] = \{010\}$ et $[[\Delta_\mu^{\beta_\triangleleft}(E)]] = \{111\}$ Donc, en général, $\Delta_\mu^{s_\triangleleft}(E) \nvdash \Delta_\mu^{\beta_\triangleleft}(E)$ et $\Delta_\mu^{\beta_\triangleleft}(E) \nvdash \Delta_\mu^{s_\triangleleft}(E)$.

ω	φ_1	φ_2	φ_3	$d_{\rho,\Sigma}$
000	6	4	0	10
001	0	3	6	9
010	7	1	3	11
011	1	0	9	10
100	7	3	1	11
101	1	4	7	12
110	6	0	4	10
111	0	1	10	11

Table 1 – $\Delta^{d_H^{\rho},\Sigma}(E)$

ω	φ_1	φ_2	φ_3	$d_{\rho,s}$
000	2	3	0	5
001	0	2	4	6
010	3	1	2	6
011	1	0	6	7
100	3	2	1	6
101	1	3	5	9
110	2	0	3	5
111	0	1	7	8

Table 2 – $\Delta^{s_\triangleleft}(E)$

ω	π_{ij}	π_{ji}	β_i
000	9	10	-1
001	11	8	3
010	9	10	-1
011	11	8	3
100	8	11	-3
101	6	13	-7
110	12	7	5
111	10	9	1

Table 3 – $\Delta^{\beta_\triangleleft}(E)$

7 Propriétés des opérateurs à la Borda

Nous allons maintenant étudier les propriétés logiques des deux familles d'opérateurs définies ci-dessus.

Premièrement, nous pouvons montrer que les opérateurs de fusion s-Borda satisfont tous les postulats de fusion IC sauf (IC4). Ce fait est une conséquence directe d'un résultat de [17] :

Proposition 2 *Un opérateur de fusion $\Delta^{s_\triangleleft}$ satisfait (IC0-IC3) et (IC5-IC8).*

On obtient donc tous les postulats usuels de rationalité sauf (IC4) qui demande une certaine symétrie entre les bases, que nous ne pouvons assurer ici.

Notez maintenant que nous obtenons le même résultat pour les opérateurs β-Borda :

Proposition 3 *Un opérateur de fusion $\Delta^{\beta_\triangleleft}$ satisfait (IC0-IC3) et (IC5-IC8).*

Preuve : Étant donné que l'opérateur $\Delta^{\beta_\triangleleft}$ est défini par l'assignement $E \mapsto \preceq^{\beta}_{P_E}$ (voir Définition 7), l'observation 1 assure que (IC0), (IC1), (IC7) et (IC8) sont satisfaits. En plus, d'après l'observation 1, afin de prouver (IC2), il faut prouver que l'assignement satisfait les propriétés 1 et 2 ; afin de prouver (IC3) il faut prouver que l'assignement satisfait la propriété 3 ; afin de prouver (IC5) il faut prouver que l'assignement satisfait la propriété 5 et, finalement, afin de prouver (IC6) il faut prouver que l'assignement satisfait la propriété 6. Ensuite, nous procédons à la démonstration des propriétés 1-3, 5 et 6.

Propriété 1 : On veut montrer que si $\omega_1 \models \bigwedge E$ et $\omega_2 \models \bigwedge E$, alors $\omega_1 \simeq^{\beta}_{P_E} \omega_2$ où $P_E = (\preceq_{\varphi_1}, \ldots, \preceq_{\varphi_n})$. Donc, on doit prouver $\beta_1(P_E) = \beta_2(P_E)$. calculons $\beta_1(P_E)$ et $\beta_2(P_E)$:

$$\beta_1(P_E) = \sum_{1 \leqslant j \leqslant m} (\pi_{1j}(P_E) - \pi_{j1}(P_E))$$

et

$$\beta_2(P_E) = \sum_{1 \leqslant j \leqslant m} (\pi_{2j}(P_E) - \pi_{j2}(P_E))$$

Puisque $\forall i, \pi_{ii}(P_E) = 0$, on peut facilement en déduire : $\beta_1(P_E) = (\pi_{12}(P_E) - \pi_{21}(P_E)) + \sum_{3 \leqslant j \leqslant m}(\pi_{1j}(P_E) - \pi_{j1}(P_E))$ et $\beta_2(P_E) = (\pi_{21}(P_E) - \pi_{12}(P_E)) + \sum_{3 \leqslant j \leqslant m}(\pi_{2j}(P_E) - \pi_{j2}(P_E))$. On a $\omega_1 \models \bigwedge E$, alors, $\forall \varphi_i \in E$, $\omega_1 \models \varphi_i$, donc $\forall \omega_j \in \Omega$, $\omega_1 \preceq_{\varphi_i} \omega_j$ (aucune interprétation $\omega_j \in \Omega$ ne peut être préférée à ω_1), alors $\pi_{j1}(P_E) = 0$ (1).
Le même raisonnement fonctionne pour ω_2, et $\forall j$ $\pi_{j2}(P_E) = 0$ (2).
De (1) et (2) on déduit que :

$$\beta_1(P_E) = \sum_{3 \leqslant j \leqslant m} \pi_{1j}(P_E) \text{ et } \beta_2(P_E) = \sum_{3 \leqslant j \leqslant m} \pi_{2j}(P_E)$$

On a $\forall \varphi_i \in E$, $\omega_1 \models \varphi_i$ et $\omega_2 \models \varphi_i$. Donc, $\forall \omega_j \in \Omega$, $\omega_1 \prec_{\varphi_i} \omega_j$ si et seulement si $\omega_2 \prec_{\varphi_i} \omega_j$. De cela, on déduit que $\pi_{1j}(P_E) = \pi_{2j}(P_E)$, et $\sum_{3 \leqslant j \leqslant m} \pi_{1j}(P_E) = \sum_{3 \leqslant j \leqslant m} \pi_{2j}(P_E)$. Cela nous donne $\beta_1(P_E) = \beta_2(P_E)$, donc $\omega_1 \simeq^{\beta}_{P_E} \omega_2$.

Propriété 2 : On veut montrer que si $\omega_1 \models \bigwedge E$ et $\omega_2 \not\models \bigwedge E$, alors $\omega_1 \prec^{\beta}_{P_E} \omega_2$. Donc, on doit prouver $\beta_1(P_E) > \beta_2(P_E)$, ce qui revient à prouver :

$$\sum_{1 \leqslant j \leqslant m} \pi_{1j}(P_E) - \pi_{j1}(P_E) > \sum_{1 \leqslant j \leqslant m} \pi_{2j}(P_E) - \pi_{j2}(P_E) \quad (1)$$

Puisque $\omega_1 \models \bigwedge E, \forall \varphi_i \in E, \forall \omega_j \in \Omega$, $\omega_1 \preceq_{\varphi_i} \omega_j$, ainsi $\pi_{j1}(P_E) = 0$, donc l'équation 1 est équivalente à :

$$\sum_{1 \leqslant j \leqslant m} \pi_{1j}(P_E) > \sum_{1 \leqslant j \leqslant m} \pi_{2j}(P_E) - \pi_{j2}(P_E)$$

ou à :

$$\pi_{12}(P_E) + \sum_{3 \leqslant j \leqslant m} \pi_{1j}(P_E) >$$

$$(\pi_{21}(P_E) - \pi_{12}(P_E)) + \sum_{3 \leqslant j \leqslant m} \pi_{2j}(P_E) - \pi_{j2}(P_E)$$

Pour montrer l'inégalité ci-dessous, il suffit de montrer :

$$\pi_{12}(P_E) > (\pi_{21}(P_E) - \pi_{12}(P_E)) \quad (2)$$

et

$$\sum_{3 \leqslant j \leqslant m} \pi_{1j}(P_E) \geqslant \sum_{3 \leqslant j \leqslant m} \pi_{2j}(P_E) - \pi_{j2}(P_E) \quad (3)$$

Pour (2) on sait déjà que $\pi_{21}(P_E) = 0$. On sait aussi que $\forall \varphi_i \in E$, $\omega_1 \preceq_{\varphi_i} \omega_2$ et $\omega_2 \not\models E$, alors $\exists \varphi_i \in E$, t.q. $\omega_2 \not\models \varphi_i$, donc $\omega_1 \prec_{\varphi_i} \omega_2$, ce qui implique que $\pi_{12}(P_E) > 0$, alors $\pi_{12}(P_E) > -\pi_{12}(P_E)$ et (2) est montrée.
Pour (3), $\forall \varphi_i \in E$, $\omega_1 \models \varphi_i$, alors $\omega_1 \preceq_{\varphi_i} \omega_2$, et $(\omega_2 \preceq_{\varphi_i} \omega_j) \implies (\omega_1 \preceq_{\varphi_i} \omega_j)$ (les bases qui préfèrent ω_2 à ω_j préfèrent également ω_1 à ω_j), donc

$$\pi_{1j}(P_E) \geqslant \pi_{2j}(P_E)$$

et aussi

$$\sum_{3 \leqslant j \leqslant m} \pi_{1j}(P_E) \geqslant \sum_{3 \leqslant j \leqslant m} \pi_{2j}(P_E)$$

On sait que $\pi_{j2}(P_E) \geqslant 0$, alors $\sum_{3 \leqslant j \leqslant m} \pi_{j2}(P_E) \geqslant 0$, donc

$$\sum_{3 \leqslant j \leqslant m} \pi_{1j}(P_E) \geqslant \sum_{3 \leqslant j \leqslant m} \pi_{2j}(P_E) - \pi_{j2}(P_E)$$

ce qui montre que (3) est vraie. En conséquence, $\omega_1 \prec^{\beta}_{P_E} \omega_2$.

Propriété 3 : On veut montrer que si $E_1 \equiv E_2$, alors $\preceq^{\beta}_{P_{E_1}} = \preceq^{\beta}_{P_{E_2}}$. On suppose que $E_1 = (\varphi_1, \ldots, \varphi_n)$ et $E_2 = (\varphi'_1, \ldots, \varphi'_n)$ et pour chaque $i \in \{1, \ldots, n\}$, $\varphi_i \equiv \varphi'_i$. Puisque $\triangleleft$ est une fonction génératrice, pour chaque $i \in \{1, \ldots, n\}$, $\preceq_{\varphi_i} = \preceq_{\varphi'_i}$. Donc, $P_{E_1} = P_{E_2}$ et évidemment $\preceq^{\beta}_{P_{E_1}} = \preceq^{\beta}_{P_{E_2}}$.

Avant de procéder à la preuve des conditions 5 et 6, on observe le fait suivant. Si E_1 et E_2 sont deux profils, nous avons pour chaque $k \in \{1, \ldots, m\}$

$$\beta_k(P_{E_1 \sqcup E_2}) = \beta_k(P_{E_1}) + \beta_k(P_{E_2}) \quad (2)$$

Propriété 5 : On veut montrer que si $\omega_1 \preceq^{\beta}_{P_{E_1}} \omega_2$ et $\omega_1 \preceq^{\beta}_{P_{E_2}} \omega_2$, alors $\omega_1 \preceq^{\beta}_{P_{E_1 \sqcup E_2}} \omega_2$. Par supposition on a $\omega_1 \preceq^{\beta}_{E_1} \omega_2$, *i.e.* $\beta_1(P_{E_1}) \geq \beta_2(P_{E_1})$ $(*)$. On a aussi $\omega_1 \preceq^{\beta}_{E_2} \omega_2$, *i.e.* $\beta_1(P_{E_2}) \geq \beta_2(P_{E_2})$ $(**)$. De $(*)$ et $(**)$ on a, $\beta_1(P_{E_1}) + \beta_1(P_{E_2}) \geqslant \beta_2(P_{E_1}) + \beta_2(P_{E_2})$. De cela et grâce à l'équation 2 on obtient, $\beta_1(P_{E_1 \sqcup E_2}) \geqslant \beta_2(P_{E_1 \sqcup E_2})$. Donc, $\omega_1 \preceq^{\beta}_{P_{E_1 \sqcup E_2}} \omega_2$.

Propriété 6 : On veut montrer que si $\omega_1 \preceq^{\beta}_{P_{E_1}} \omega_2$ et $\omega_1 \prec^{\beta}_{P_{E_2}} \omega_2$, alors $\omega_1 \prec^{\beta}_{P_{E_1 \sqcup E_2}} \omega_2$. Comme au point précédent, à partir des hypothèses, on obtient $\beta_1(P_{E_1}) \geq \beta_2(P_{E_1})$ (1) et $\beta_1(P_{E_2}) > \beta_2(P_{E_2})$ (2). De (1) et (2) on a, $\beta_1(P_{E_1}) + \beta_1(P_{E_2}) > \beta_2(P_{E_1}) + \beta_2(P_{E_2})$. De cela et grâce à l'équation 2 on obtient $\beta_1(P_{E_1 \sqcup E_2}) > \beta_2(P_{E_1 \sqcup E_2})$. Donc, $\omega_1 \prec^{\beta}_{P_{E_1 \sqcup E_2}} \omega_2$. ∎

Cette définition des opérateurs β-Borda (comparaison par paires) nous permet de fournir une caractérisation de l'opérateur de fusion drastique.

Théorème 3 *Soit $\Delta^{d,f}$ un opérateur basé sur la distance. $\Delta^{d,f}$ est basé sur des comparaisons par paires si et seulement si c'est l'opérateur drastique.*

La preuve est organisée en trois lemmes.

Lemme 1 *Si une distance d n'est pas la distance drastique alors il existe x, y et z tels que $0 < d(x, y) < d(x, z)$.*

Preuve : Si d n'est pas la distance drastique, alors on peut trouver ω_1, ω_2, ω_3 et ω_4 tels que $d(\omega_1, \omega_2) \neq 0$, $d(\omega_3, \omega_4) \neq 0$, et $d(\omega_1, \omega_2) \neq d(\omega_3, \omega_4)$.

Supposons, sans perte de généralité, que $0 < d(\omega_1, \omega_2) < d(\omega_3, \omega_4)$.

On pose $a = d(\omega_1, \omega_2)$ et $b = d(\omega_1, \omega_3)$.

Si $b = 0$, alors $\omega_1 = \omega_3$ et en posant $x = \omega_1$, $y = \omega_2$ et $z = \omega_4$, on a $0 < d(x, y) < d(x, z)$.

Si $b \neq 0$, on considère deux cas : $b = a$ ou $b \neq a$. Dans le premier cas où $b = a$, on pose $x = \omega_3$, $y = \omega_1$ et $z = \omega_4$ et on a $0 < d(x, y) < d(x, z)$. Dans le deuxième cas où $b \neq a$:

— Si $b < a$, on pose $x = \omega_3$, $y = \omega_1$ et $z = \omega_4$, et on a $0 < d(x, y) < d(x, z)$.

— Si $a < b$, on pose $x = \omega_1$, $y = \omega_2$ et $z = \omega_3$, on a de nouveau $0 < d(x, y) < d(x, z)$, ce qui prouve ce lemme.

∎

Lemme 2 *Si une distance d n'est pas la distance drastique d_D, alors $\Delta^{d,f}$ n'est pas basé sur des comparaisons par paires.*

Preuve :

Soit une distance non drastique d et l'ensemble des interprétations $\Omega = \{\omega_1, \ldots, \omega_m\}$ avec $m \geqslant 4$. D'après le Lemme 1, il y a ω_i, ω_j, et ω_k tel que $d(\omega_i, \omega_j) < d(\omega_i, \omega_k)$. On considère φ_{ω_i}, la base telle que $[[\varphi_{\omega_i}]] = \{\omega_i\}$. Alors on a $0 = d(\omega_i, \varphi_{\omega_i}) < d(\omega_j, \varphi_{\omega_i}) < d(\omega_k, \varphi_{\omega_i})$. On considère $\varphi_{\neg\omega_i}$ la base définie par $\varphi_{\neg\omega_i} = \neg\varphi_{\omega_i}$, en particulier, $[[\varphi_{\neg\omega_i}]] = \Omega \setminus \{\omega_i\}$. Alors $\forall \omega \neq \omega_i$, $d(\omega, \varphi_{\neg\omega_i}) = 0$, tandis que $d(\omega_i, \varphi_{\neg\omega_i}) > 0$. Soit $\omega_h \in [[\varphi_{\neg\omega_i}]]$ t.q. $d(\omega_i, \omega_h) = min_{\omega \in [[\varphi_{\neg\omega_i}]]} d(\omega_i, \omega) = d(\omega_i, \varphi_{\neg\omega_i})$.

On remarque que nécessairement $d(\omega_i, \omega_h) \leqslant d(\omega_i, \omega_j)$. Donc, $d(\omega_i, \omega_h) \leqslant d(\omega_i, \omega_k)$.

On a $d(\omega_i, \varphi_{\omega_i}) < d(\omega_h, \varphi_{\omega_i}) < d(\omega_k, \varphi_{\omega_i})$ et $d(\omega_h, \varphi_{\neg\omega_i}) = d(\omega_k, \varphi_{\neg\omega_i}) < d(\omega_i, \varphi_{\neg\omega_i})$.

On considère $E = (\varphi_{\omega_i}, \varphi_{\neg\omega_i})$. Calculons β_i et β_h. On sait que β_i compte le nombre d'interprétations les moins bien classées que ω_i dans toutes les bases, moins le nombre d'interprétations les mieux classées que ω_i dans toutes les bases. Ici on a 2 bases φ_{ω_i} et $\varphi_{\neg\omega_i}$, alors pour φ_{ω_i}, ω_i est mieux que toutes les autres interprétations, et c'est le contraire pour $\varphi_{\neg\omega_i}$. Alors $\beta_i = \Sigma_{p=1, p\neq i}^{p=m} \pi_{ip} - \Sigma_{p=1, p\neq i}^{p=m} \pi_{pi}$ $= (m-1) - (m-1) = 0$.

On va maintenant calculer $\beta_h = \Sigma_{p=1, p\neq h}^{p=m} \pi_{hp} - \Sigma_{p=1, p\neq h}^{p=m} \pi_{ph}$.

Soient $l \in \{1, \ldots, m\}$, $l \neq i$, $l \neq k$ et $l \neq h$. Comme $\omega_h \equiv_{\varphi_{\neg\omega_i}} \omega_l$, $\pi_{hl} - \pi_{lh} = a_{h,l}$, avec $a_{h,l} = 0$ si $d(\omega_h, \varphi_{\omega_i}) = d(\omega_l, \varphi_{\omega_i})$ ou $a_{h,l} = 1$ si $d(\omega_h, \varphi_{\omega_i}) < d(\omega_l, \varphi_{\omega_i})$. Le cas $a_{h,l} = -1$ n'est pas possible car cela correspond à la situation où $d(\omega_h, \varphi_{\omega_i}) > d(\omega_l, \varphi_{\omega_i})$, i.e. $d(\omega_h, \omega_i) > d(\omega_l, \omega_i)$, impossible par supposition $(d(\omega_i, \omega_h) = min_{\omega \in \Omega} d(\omega_i, \omega))$. Alors $\pi_{hl} - \pi_{lh} \geqslant 0$.

De plus, $\pi_{hi} - \pi_{ih} = 0$ (puisque $\omega_i \models \varphi_{\omega_i}$ et $\omega_h \not\models \varphi_{\omega_i}$) et $\pi_{hk} - \pi_{kh} = 1$ (puisque par supposition $d(\omega_h, \varphi_{\omega_i}) < d(\omega_k, \varphi_{\omega_i})$). Alors, $\beta_h > 0$. De plus, $\beta_h > \beta_i$, *i.e.* $\omega_h <_E^{\beta} \omega_i$.

D'autre part, par supposition $d(\omega_i, \varphi_{\neg\omega_i}) = d(\omega_i, \omega_h)$ et comme ω_i est le modèle unique de φ_{ω_i}, alors $d(\omega_h, \varphi_{\omega_i}) = d(\omega_i, \omega_h)$. De plus $(d(\omega_i, \varphi_{\omega_i}),\ d(\omega_i, \varphi_{\neg\omega_i})) = (d(\omega_h, \varphi_{\neg\omega_i}),\ d(\omega_h, \varphi_{\omega_i}))$. Donc, pour toute fonction d'agrégation anonyme f utilisée pour calculer la distance de ω_i et ω_h à E, nous avons la même sortie, *i.e.* $\omega_i \simeq_{\Delta^{d,f}} \omega_h$. Cela prouve que les préordres générés par l'opérateur β-Borda et par les opérateurs $\Delta^{d,f}$ peuvent être différents, et ainsi le lemme.

∎

Lemme 3 *L'opérateur drastique $\Delta^{d_D,f}$ est basé sur des comparaisons par paires.*

Preuve : On considère le profil $E = (\varphi_1, \ldots \varphi_n)$ de n bases et un ensemble de m interprétations $\{\omega_1, \ldots, \omega_m\}$. On note par $E_{\leq^{d_D}} = (\leq_1^{d_D}, \ldots \leq_n^{d_D})$ le profil des préordres obtenus à partir de E avec la distance drastique d_D.

$\forall j, k \in \{1, \ldots, m\}$, $\pi_{jk}(E_{\leq^{d_D}})$ représente le nombre de préordres dans $E_{\leq^{d_D}}$ qui préfèrent (strictement) ω_j à ω_k. Comme la distance drastique d_D est utilisée, on sait que $\omega_j <_\varphi \omega_k$ si et seulement si $d_D(\omega_j, \varphi) < d_D(\omega_k, \varphi)$ i.e. si et seulement si $\omega_j \models \varphi$ et $\omega_k \not\models \varphi$. En conséquence :

$\pi_{kj}(E_{\leq^{d_D}}) = |\{i \in \{1, \ldots n\} : \omega_k \models \varphi \wedge \omega_j \not\models \varphi\}|$.

$\beta_k(E_{\leq^{d_D}}) = \Sigma_{j=1, j\neq k}^{m} \pi_{kj}(E_{\leq^{d_D}}) - \pi_{jk}(E_{\leq^{d_D}})$

$= \Sigma_{j=1, j\neq k}^{m} |\{i \in \{1, \ldots n\} : \omega_k \models \varphi_i \wedge \omega_j \not\models \varphi_i\}| - |\{i \in \{1, \ldots n\} : \omega_j \models \varphi_i \wedge \omega_k \not\models \varphi_i\}|$

$= \Sigma_{j=1, j\neq k}^{m} |\{i \in \{1, \ldots n\} : \omega_k \models \varphi_i \wedge \omega_j \not\models \varphi_i\}| - \Sigma_{j=1, j\neq k}^{m} |\{i \in \{1, \ldots n\} : \omega_j \models \varphi_i \wedge \omega_k \not\models \varphi_i\}|$

$= \Sigma_{i=1, \omega_k \models \varphi_i}^{n} \Sigma_{j=1, j\neq k, \omega_j \not\models \varphi_i}^{m} 1 - \Sigma_{i=1, \omega_k \not\models \varphi_i}^{n} \Sigma_{j=1, j\neq k, \omega_j \models \varphi_i}^{m} 1$

$= \Sigma_{i=1, \omega_k \models \varphi_i}^{n} |[[\neg\varphi_i]]| - \Sigma_{i=1, \omega_k \not\models \varphi_i}^{n} |\varphi_i|$

On a $\forall i, |\varphi_i| = m - |[[\neg\varphi_i]]|$, alors on obtient :

$\beta_k(E_{\leq^{d_D}}) = \Sigma_{i=1, \omega_k \models \varphi_i}^{n} |[[\neg\varphi_i]]| - \Sigma_{i=1, \omega_k \not\models \varphi_i}^{n} (m - |[[\neg\varphi_i]]|)$

$= \Sigma_{i=1, \omega_k \models \varphi_i}^{n} |[[\neg\varphi_i]]| - \Sigma_{i=1, \omega_k \not\models \varphi_i}^{n} m + \Sigma_{i=1, \omega_k \not\models \varphi_i}^{n} |[[\neg\varphi_i]]|$

$= (\Sigma_{i=1, \omega_k \models \varphi_i}^{n} |[[\neg\varphi_i]]| + \Sigma_{i=1, \omega_k \not\models \varphi_i}^{n} |[[\neg\varphi_i]]|) - \Sigma_{i=1, \omega_k \not\models \varphi_i}^{n} m$

$= \Sigma_{i=1}^{n} |[[\neg\varphi_i]]| - \Sigma_{i=1, \omega_k \not\models \varphi_i}^{n} m$

Et $|\{i \in \{1, \ldots, n\} : \omega_k \not\models \varphi_i\}| = \Sigma_{i=1}^{n} d_D(\omega_k, \varphi_i) = d_{d_D,\Sigma}(\omega_k, E)$.

Finalement, on obtient :

$\beta_k(E_{\leq^{d_D}}) = \Sigma_{i=1}^{n} |[[\neg\varphi_i]]| - m * d_{d_D,\Sigma}(\omega_k, E)$

On note que le terme $\Sigma_{i=1}^{n} |[[\neg\varphi_i]]|$ est le même pour toutes les interprétations ω_k, et que la multiplication par $-m$ inverse l'échelle, mais ne change pas la relation donnée par la comparaison des distances $d_{d_D,\Sigma}(\omega_k, E)$.

Alors les préordres obtenus par $\leq_{d_D,\Sigma}$ et par le calcul des scores Borda β (les comparaisons par paires) sont les mêmes. Ceci achève la preuve du Théorème 3.

∎

Ainsi, l'opérateur de fusion drastique est le seul opérateur de la classe des opérateurs basés sur la distance à être identique à sa version β-Borda (i.e. à être un opérateur de comparaison par paires). Dans la section suivante, nous montrerons une caractérisation plus générale de cet opérateur de fusion drastique.

8 Annulation

Dans cette section, nous allons introduire et discuter la notion d'annulation dans la fusion des croyances.

Dans la caractérisation de la règle de Borda, la propriété d'annulation peut être interprétée comme exigeant qu'une relation $a < b$ entre deux alternatives puisse être annulée par une relation inverse $b < a$ (donnée par un autre individu). La définition formelle de l'annulation est donnée dans la section 4.

Nous allons maintenant explorer cette idée générale afin de définir les propriétés d'annulation dans le cadre de la fusion de croyances. L'idée est de permettre l'annulation d'une base dans le profil par une autre entrée. La définition exacte de cette "autre entrée" nous conduit à différentes propriétés d'annulation (les noms des postulats viennent du mot anglais *cancellation*).

Définition 11 *Un opérateur de fusion Δ satisfait la propriété ($\neg\varphi$-**Can**) si pour chaque formule φ, et chaque contrainte d'intégrité μ,*

($\neg\varphi$-Can) $\Delta_\mu(\varphi \sqcup \neg\varphi) \equiv \mu$

On peut vérifier ce qui suit :

Proposition 4 *Soit Δ un opérateur de fusion IC. Alors Δ satisfait ($\neg\varphi$-**Can**) si et seulement si il satisfait ($\neg E$-**Can**)*

($\neg E$-Can) $\Delta_\mu(E \sqcup \neg E) \equiv \mu$

Nous pouvons maintenant fournir une caractérisation de l'opérateur de fusion drastique $\Delta^{d_D,f}$ en termes d'annulation :

Théorème 4 *Un opérateur de fusion IC Δ satisfait la propriété d'annulation ($\neg\varphi$-**Can**) si et seulement si $\Delta = \Delta^{d_D,f}$.*

Preuve : Pour prouver ce théorème, on démontre d'abord la proposition suivante. On ne présentera pas ce résultat comme un lemme, car cette proposition est intéressante en elle-même et pas seulement comme outil technique dans une preuve.

Proposition 5 *La fonction génératrice $\triangleleft$ associée à un opérateur de fusion IC Δ ne génère que des préordres à 2 strates si et seulement si $\Delta = \Delta^{d_D,\Sigma}$.*

Preuve : ($\Rightarrow$) On suppose que la fonction génératrice $\triangleleft$ associée à un opérateur de fusion IC Δ ne génère que des préordres à 2 strates.

On considère un profil E, et deux interprétations ω_1 et ω_2. On suppose, sans perte de généralité, que le nombre de bases dans E impliqué par ω_1 est supérieur au nombre de bases dans E impliqué par ω_2 (c'est-à-dire que $d_{d_D,\Sigma}(\omega_1, E) < d_{d_D,\Sigma}(\omega_2, E)$). Pour comparer les deux interprétations, on peut distinguer et séparer les bases en 4 cas :

Cas 1 : Les bases φ_i^1 telles que $\omega_1 \models \varphi_i^1$ et $\omega_2 \models \varphi_i^1$

Cas 2 : Les bases φ_i^2 telles que $\omega_1 \not\models \varphi_i^2$ et $\omega_2 \not\models \varphi_i^2$

Cas 3 : Les bases $\varphi_{i_1}^3$ et $\varphi_{i_2}^3$, associées par paires, telles que $\omega_1 \models \varphi_{i_1}^3$, $\omega_2 \not\models \varphi_{i_1}^3$ et $\omega_1 \not\models \varphi_{i_2}^3$ et $\omega_2 \models \varphi_{i_2}^3$.

Cas 4 : Les bases restantes φ_i^4, telles que $\omega_1 \models \varphi_i^4$ et $\omega_2 \not\models \varphi_i^4$.

De (IC2), on sait que $\omega_1 \simeq_{\varphi_i^1} \omega_2$.

Il y a exactement deux niveaux dans les préordres associés à φ_i^2. Comme $\omega_1 \not\models \varphi_i^2$ et $\omega_2 \not\models \varphi_i^2$, ω_1 et ω_2 appartiennent au même niveau pour chaque $\preceq_{\varphi_i^2}$. En d'autres termes, $\forall \varphi_i^2, \omega_1 \simeq_{\varphi_i^2} \omega_2$. Donc $\omega_1 \simeq_{\varphi_i^2} \omega_2$.

Pour le cas 3, si $\omega_1 \prec_{\varphi_{i_1}^3 \sqcup \varphi_{i_2}^3} \omega_2$ ou $\omega_1 \succ_{\varphi_{i_1}^3 \sqcup \varphi_{i_2}^3} \omega_2$, cela contredit (IC4). Alors $\omega_1 \simeq_{\varphi_{i_1}^3 \sqcup \varphi_{i_2}^3} \omega_2$.

Pour le cas 4, il est clair que $\omega_1 \prec_{\varphi_i^4} \omega_2$.

Par (IC5) et (IC6), on obtient $\omega_1 \simeq_{\varphi_i^1 \sqcup \varphi_i^2 \sqcup \varphi_{i_1}^3 \sqcup \varphi_{i_2}^3} \omega_2$. Comme par ailleurs $\omega_1 \prec_{\varphi_i^4} \omega_2$, on obtient $\omega_1 \prec_E \omega_2$.

On vient de montrer que si $d_{d_D,\Sigma}(\omega_1, E) < d_{d_D,\Sigma}(\omega_2, E))$, alors $\omega_1 \prec_E \omega_2$.

Par le même raisonnement, on peut montrer que si $d_{d_D,\Sigma}(\omega_1, E) = d_{d_D,\Sigma}(\omega_2, E))$, alors $\omega_1 \simeq_E \omega_2$. Donc le préordre obtenu de Δ pour E coïncide avec le préordre obtenu de $\Delta^{d_D,\Sigma}$.

($\Leftarrow$) Cette partie est claire : si $\Delta = \Delta^{d_D,\Sigma}$, alors la fonction génératrice $\triangleleft$ associée à $\Delta^{d_D,\Sigma}$ ne contient que les préordres à 2 strates. ∎

Un corollaire intéressant de la Proposition 5 est le suivant :

Corollaire 1 *Si $\Delta^{d_D,f}$ est un opérateur de fusion alors $\Delta^{d_D,f} \equiv \Delta^{d_D,\Sigma}$.*

On passe maintenant à la preuve du théorème principal.

$\Rightarrow$: On suppose que $\Delta \not\equiv \Delta^{d_D,f}$.

Comme $\Delta \not\equiv \Delta^{d_D,f}$, d'après la proposition 5, on sait qu'il existe au moins une base φ telle que l'assignement $\preceq_\varphi$ définit un préordre avec plus de deux strates. Il y a donc trois interprétations ω_1, ω_2 et ω_3 telles que $\omega_1 \models \varphi$, $\omega_2 \not\models \varphi$, $\omega_3 \not\models \varphi$ et $\omega_2 \prec_\varphi \omega_3$.

On considère $\varphi' = \neg\varphi$, et comme Δ satisfait ($\neg\varphi$-**Can**), $\Delta(\varphi \sqcup \varphi') \equiv \top$.

Comme $\omega_2 \not\models \varphi$ et $\omega_3 \not\models \varphi$, on a $\omega_2 \models \varphi'$ et $\omega_3 \models \varphi'$, alors $\omega_2 \simeq_{\varphi'} \omega_3$.

De la condition 6 de l'assignement syncrétique, comme $\omega_2 \prec_\varphi \omega_3$ et $\omega_2 \simeq_{\varphi'} \omega_3$, alors $\omega_2 \prec_{\{\varphi,\varphi'\}} \omega_3$: cela contredit le fait que $\Delta(\varphi \sqcup \varphi') \equiv \top$. Alors $\Delta \equiv \Delta^{d_D,f}$.

$\Leftarrow$ **:** On doit prouver que $\Delta^{d_D,f}$ satisfait la propriété **($\neg\varphi$-Can)**. Premièrement, d'après le corollaire 1 que $\Delta^{d_D,f} \equiv \Delta^{d_D,\Sigma}$. Maintenant, on doit prouver que $\Delta^{d_D,\Sigma}$ satisfait la propriété **($\neg\varphi$-Can)**. On considère le profil $E = \{\varphi_1, \ldots, \varphi_n\}$, sa négation $\neg E = \{\neg\varphi_1, \ldots, \neg\varphi_n\}$ et une contrainte d'intégrité μ.

On considère une interprétation ω.

$d^{d_D,\Sigma}(\omega, E) = \Sigma_{i=1}^{i=n} d_D(\omega, \varphi_i)) = k$, où $0 \leqslant k \leqslant n$.

$d^{d_D,\Sigma}(\omega, \neg E) = \Sigma_{i=1}^{i=n} d_D(\omega, \neg\varphi_i)) = n - k$.

$d^{d_D,\Sigma}(\omega, E \sqcup \neg E) = k + (n - k) = n$.

Comme $\forall\omega, d^{d_D,\Sigma}(\omega, E \sqcup \neg E) = n$, toutes les interprétations de μ sont équivalents et $\Delta_\mu(E \sqcup \neg E) \equiv \mu$.

∎

En fait, après une lecture attentive de la preuve, on s'aperçoit que le Théorème 4 peut s'exprimer de manière plus générale. Pour rendre ce théorème vrai, on a besoin de :

— une fonction génératrice $\triangleleft$

— un assignement associé au processus de fusion, satisfaisant la condition 6 de l'assignement syncrétique.

On sait qu'une distance d entre interprétations est un moyen de définir une fonction génératrice, car elle définit naturellement un préordre sur les interprétations. Ainsi une question intéressante se pose sur l'expression du Théorème 4 lorsqu'une distance d et une fonction d'agrégation f sont disponibles.

Proposition 6 *Si la fonction d'agrégation f est anonyme et non décroissante, l'opérateur de fusion $\Delta^{d,f}$ satisfait la propriété **($\neg\varphi$-Can)** propriété d'annulation si et seulement si $d = d_D$.*

Preuve : On suppose que l'opérateur de fusion $\Delta^{d,f}$ satisfait la propriété **($\neg\varphi$-Can)**, et que $d \neq d_D$.

Puisque $d \neq d_D$, il y a au moins une base k et trois interprétations $\omega_1, \omega_2, \omega_3$ telles que $\omega_1 \models k$, $\omega_2 \not\models \varphi$, $d(\omega_1, \omega_2)$ est minimale parmi toutes les interprétations de $\neg\varphi$ (i.e. $d(\omega_1, \omega_2) = \min_{\omega_k \models \neg\varphi}(d(\omega_1, \omega_k))$) et $d(\omega_2, \varphi) < d(\omega_3, \varphi)$.

On considère $E = \langle \varphi_{\omega_1}, \varphi_{\neg\omega_1} \rangle$. Par la propriété **($\neg\varphi$-Can)**, $\Delta(\langle \varphi_{\omega_1}, \varphi_{\neg\omega_1} \rangle) \equiv \top$, alors $\omega_3 \simeq_E \omega_1$.

Mais comme $0 = d(\omega_1, \varphi) < d(\omega_2, \varphi) < d(\omega_3, \varphi)$

et $0 = d(\omega_2, \varphi_{\neg\omega_1}) = d(\omega_3, \varphi_{\neg\omega_1}) < d(\omega_1, \varphi)$.

Puisque $d(\omega_1, \omega_2) = \min_{\omega_k \models \neg\varphi}(d(\omega_1, \omega_k)) = d(\omega_1, \varphi_{\neg\omega_1})$, et $d(\omega_1, \omega_2) = d(\omega_2, \varphi_{\omega_1})$, on a $d(\omega_1, \varphi_{\neg\omega_1}) = d(\omega_2, \varphi_{\omega_1})$.

Puisque la fonction d'agrégation f est non décroissante et anonyme, $d(\omega_1, \{\varphi_{\omega_1}, \varphi_{\neg\omega_1}\}) = d(\omega_2, \{\varphi_{\omega_1}, \varphi_{\neg\omega_1}\}) < d(\omega_3, \{\varphi_{\omega_1}, \varphi_{\neg\omega_1}\})$

Cela contredit $\Delta(\varphi_{\omega_1} \sqcup \varphi_{\neg\omega_1}) \equiv \top$.

∎

On peut souligner que dans la proposition 6, aucune hypothèse n'est faite sur l'assignement associé à $\Delta^{d,f}$. En particulier, on ne suppose pas que $\Delta^{d,f}$ soit un opérateur de fusion IC.

Le postulat d'annulation **($\neg\varphi$-Can)** est intéressant pour obtenir la caractérisation ci-dessus, mais c'est un exemple très particulier d'une famille plus large de postulats d'annulation que nous allons introduire et discuter maintenant, et qui pourraient conduire à d'autres résultats de caractérisation.

Définition 12 (Annulation 1)

($\neg\varphi$-Can) $\forall\varphi \Delta_\mu(\varphi \sqcup \neg\varphi) \equiv \mu$

(φ-Can) $\forall\varphi\ \exists\varphi' \Delta_\mu(\varphi \sqcup \varphi') \equiv \mu$

($\overline{\varphi}$-Can) $\forall\varphi \Delta_\mu(\varphi \sqcup \overline{\varphi}) \equiv \mu$

On regroupe ici trois définitions liées, qui partagent le fait qu'une base peut être annulée par une autre base. On commence par **($\neg\varphi$-Can)**, qu'on a déjà défini plus haut. Notez qu'il s'agit d'un cas particulier, utilisant la négation, du postulat **(φ-Can)**, qui stipule que toute base peut être annulée par une autre base. Un autre cas particulier potentiellement intéressant est **($\overline{\varphi}$-Can)**, où l'on utilise le dual de φ, tel que défini dans [13], où $\overline{\varphi}$ s'obtient en remplaçant chaque littéral apparaissant dans φ par sa négation.

Mais on peut être un peu plus général, et accepter qu'une seule base ne suffira peut-être pas à annuler n'importe quelle base, et qu'un ensemble de bases sera nécessaire. Ceci conduit à d'autres postulats possibles :

Définition 13 (Annulation 2)

(E-Can) $\forall\varphi\ \exists E\ \Delta_\mu(\varphi \sqcup E) \equiv \mu$

(fE-Can) $\forall\varphi\ \exists E\ t.q.\ f(E)\ et\ \Delta_\mu(\varphi \sqcup E) \equiv \mu$

($\neg\omega$-Can) $\forall\varphi\ \exists E\ t.q.\ (\forall\psi \in E\ \exists\omega_i\ \psi = \varphi_{\neg\omega_i})$ $et\ \Delta_\mu(\varphi \sqcup E) \equiv \mu$

($\neg\omega^n$-Can) $\forall\varphi\ \ if\ E = \bigsqcup_i (\varphi_{\neg\omega_i})^{r_{\leq\varphi}(\omega_i)}$ $alors\ \Delta_\mu(\varphi \sqcup E) \equiv \mu$

La forme la plus générale de ces postulats d'annulation est **(E-Can)** qui stipule qu'un ensemble de bases peut annuler une base choisie. **(fE-Can)** est une restriction de **(E-Can)**, où E ne peut pas être choisi librement, mais doit satisfaire une condition $f()$. Une telle restriction intéressante est donnée par **($\neg\omega$-Can)** où toutes les bases de E doivent être des négations d'une base complète φ_ω. On spécialise encore ce dernier postulat et on obtient **($\neg\omega^n$-Can)** qui donne la définition exacte du profil E correspondant à une base φ donnée. Cette dernière forme est liée aux opérateurs s-Borda.

Notons que tous ces postulats parlent de l'annulation d'une formule φ i.e. un profil composé d'une seule base, mais ceci peut être généralisé à des profils plus généraux. Donnons juste le résultat pour **(E-Can)** (on peut prouver un résultat similaire pour tous les autres postulats d'annulation) :

Proposition 7 *Soit Δ un opérateur de fusion IC. Alors Δ satisfait **(E-Can)** si et seulement si Δ satisfait **(E-CanE)***

(E-CanE) $\forall E' \, \forall \varphi \, \exists E \; \Delta_\mu(E' \sqcup \varphi \sqcup E) \equiv \Delta_\mu(E')$

Il est clair que certains des postulats ci-dessus sont plus généraux que d'autres :

Proposition 8 *On a la relation de la figure 1, où $X \rightarrow Y$ signifie que Y est plus général que X (i.e. si Δ satisfait X alors il satisfait Y).*

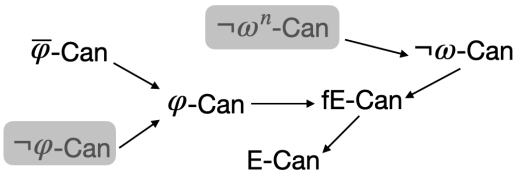


Figure 1 – Postulats d'annulation

Nous laissons l'exploration des différentes formes d'annulation pour des travaux futurs. Nous donnons un dernier résultat, ainsi qu'une conjecture finale.

Proposition 9 *Soit Δ un opérateur qui satisfait (IC1-IC3) et (IC5-IC8). Si Δ est un opérateur de fusion s-Borda alors Δ satisfait **($\neg\omega^n$-Can)**.*

Notre conjecture est que nous croyons que **($\neg\omega^n$-Can)** encode précisément le comportement des opérateurs de fusion s-Borda, mais nous n'avons pas encore la preuve formelle.

On a donc mis en gris sur la figure 1 les deux postulats d'annulation pour lesquels on a des liens avec des opérateurs de fusion concrets. Nous sommes convaincus qu'une exploration plus poussée de ces liens peut nous fournir d'autres caractérisations intéressantes des opérateurs de fusion.

9 Conclusion

Dans cet article, nous avons défini deux familles d'opérateurs de fusion inspirés de la définition de la règle de vote de Borda. Nous avons également introduit la notion d'annulation dans la fusion des croyances, inspirée de l'axiomatisation de la règle de vote de Borda proposée par Young [29]. Ceci nous a permis de montrer que l'opérateur de fusion drastique est le seul opérateur de fusion IC satisfaisant une variante d'annulation, ce qui nous a donné une caractérisation de cet opérateur. Nous sommes convaincus qu'une exploration plus systématique des autres variantes d'annulation que nous proposons (ou d'autres) peut nous conduire à d'autres caractérisations intéressantes des opérateurs de fusion de croyances.

Remerciements

Ce travail a bénéficié du support de la chaire IA BE4musIA (ANR-20-CHIA-0028) de l'Agence Nationale de Recherche (ANR). Le quatrième auteur est partiellement financé par le programme PAUSE du Collège de France.

Les auteurs remercient les relecteurs pour leur lecture attentive de cet article et pour leurs remarques pertinentes.

Références

[1] Alchourrón, C. E., P. Gärdenfors et D. Makinson: *On the logic of theory change : Partial meet contraction and revision functions*. Journal of Symbolic Logic, 50 :510–530, 1985.

[2] Arrow, K. J.: *Social choice and individual values*. Wiley, New York, second édition, 1963.

[3] Arrow, K.J., A. K. Sen et K. Suzumura (rédacteurs): *Handbook of Social Choice and Welfare*, tome 1. North-Holland, 2002.

[4] Borda, J. C.: *Mémoire sur les élections au scrutin*. Histoire de l'Académie Royale des Sciences, 1781.

[5] Everaere, Patricia, Sébastien Konieczny et Pierre Marquis: *Counting votes for aggregating judgments*. Dans *International conference on Autonomous Agents and Multi-Agent Systems, (AAMAS'14), Paris, France, May 5-9, 2014*, pages 1177–1184, 2014.

[6] Everaere, Patricia, Sébastien Konieczny et Pierre Marquis: *The strategy-proofness landscape of merging*. Journal of Artificial Intelligence Research, 28 :49–105, 2007.

[7] Everaere, Patricia, Sébastien Konieczny et Pierre Marquis: *Disjunctive merging : Quota and Gmin merging operators*. Artificial Intelligence, 174(12-13) :824–849, 2010.

[8] Everaere, Patricia, Sébastien Konieczny et Pierre Marquis: *The epistemic view of belief merging : can we track the truth ?* Dans *Nineteenth European Conference on Artificial Intelligence (ECAI'10)*, pages 621–626, 2010.

[9] Everaere, Patricia, Sébastien Konieczny et Pierre Marquis: *On egalitarian belief merging*. Dans *Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*, 2014.

[10] Everaere, Patricia, Sébastien Konieczny et Pierre Marquis: *Belief Merging versus Judgment Aggregation*. Dans *Fourteenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*, pages 999–1007, 2015.

[11] Fermé, E. et S. O. Hansson: *AGM 25 Years : Twenty-Five Years of Research in Belief Change*. Journal of Philosophical Logic, 40(2) :295–331, 2011.

[12] Gärdenfors, P.: *Knowledge in flux*. MIT Press, 1988.

[13] Haret, Adrian et Stefan Woltran: *Belief Revision Operators with Varying Attitudes Towards Initial Beliefs*. Dans *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI'19)*, pages 1726–1733, 2019.

[14] Katsuno, H. et A. O. Mendelzon: *Propositional knowledge base revision and minimal change*. Artificial Intelligence, 52 :263–294, 1991.

[15] Konieczny, Sébastien, Jérôme Lang et Pierre Marquis: *DA^2 Merging Operators*. Artificial Intelligence, 157 :49–79, 2004.

[16] Konieczny, Sébastien et Ramón Pino Pérez: *On the logic of merging*. Dans *Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 488–498, 1998.

[17] Konieczny, Sébastien et Ramón Pino Pérez: *Merging information under constraints : a logical framework*. Journal of Logic and Computation, 12(5) :773–808, 2002.

[18] Konieczny, Sébastien et Ramón Pino Pérez: *Propositional belief base merging or how to merge beliefs/goals coming from several sources and some links with social choice theory*. European Journal of Operational Research, 160(3) :785–802, 2005.

[19] Konieczny, Sébastien et Ramón Pino Pérez: *Logic Based Merging*. Journal of Philosophical Logic, 40(2) :239–270, 2011.

[20] Lang, Jérôme, Gabriella Pigozzi, Marija Slavkovik, Leon van der Torre et Srdjan Vesic: *A partial taxonomy of judgment aggregation rules and their properties*. Social Choice and Welfare, 48(2) :327–356, 2017.

[21] Liberatore, P. et M. Schaerf: *Arbitration (or how to merge knowledge bases)*. IEEE Transactions on Knowledge and Data Engineering, 10(1) :76–90, 1998.

[22] Lin, J. et A. O. Mendelzon: *Merging databases under constraints*. International Journal of Cooperative Information System, 7(1) :55–76, 1998.

[23] List, C. et P. Pettit: *Aggregating sets of judgments. An impossibility result*. Economics and Philosophy, 18 :89–110, 2002.

[24] List, C. et C. Puppe: *Judgment Aggregation : A Survey*. Dans *The Handbook of Rational and Social Choice*. Oxford University Press, 2009.

[25] Mata Díaz, Amílcar et Ramón Pino Pérez: *Impossibility in belief merging*. Artificial Intelligence, 251 :1 – 34, 2017.

[26] Merlin, Vincent: *The axiomatic characterizations of majority voting and scoring rules*. Mathématiques et sciences humaines, 41 :87 – 109, 2003.

[27] Pigozzi, Gabriella: *Belief merging and the discursive dilemma : an argument-based account to paradoxes of judgment aggregation*. Synthese, 152(2) :285–298, 2006.

[28] Revesz, P. Z.: *On the semantics of arbitration*. International Journal of Algebra and Computation, 7(2) :133–160, 1997.

[29] Young, H. P.: *An Axiomatization of Borda's Rule*. Journal of Economic Theory, 9 :43–52, 1974.

Session 2 : IA explicable

Actes JIAF 2022

Des explications par étapes pour le modèle additif

Manuel Amoussou[1] **Khaled Belahcene**[2] **Nicolas Maudet**[3] **Vincent Mousseau**[1] **Wassila Ouerdane**[1]

[1]MICS, CentraleSupélec, Université Paris-Saclay, France
[2]Heudiasyc, Université de Technologie de Compiègne, CNRS, France
[3]Lip6, Sorbonne Université, CNRS, France

{manuel.amoussou, vincent.mousseau, wassila.ouerdane}@centralesupelec.fr, khaled.belahcene@hds.utc.fr, nicolas.maudet@lip6.fr

Résumé

Nous nous intéressons au problème du calcul d'explications de comparaisons par paires issues d'un modèle de préférence additif. La structure de ces explications s'appuie sur une décomposition de la paires à expliquer sous la forme d'une séquence de jugements de préférence. Pour que cette explication soit comprise et acceptée par celui à qui elle est destinée, chaque élément de la séquence construite doit être aussi intelligible et cognitivement simple que possible. Ainsi, nous proposons plusieurs schémas d'arguments permettant d'inférer de nouvelles connaissances, sous forme de comparaisons par paires, à partir de celles précédemment validées. Ces schémas d'arguments dont nous garantissons la correction, exploitent un certain nombre de propriétés du modèle additif. Nous préconisons spécifiquement l'utilisation du schéma couverture car il répond à certaines propriétés souhaitables pour les explications. Imposer que la décomposition soit faite en éléments cognitivement simples se fait au prix de l'exhaustivité. Cependant, les résultats expérimentaux montrent que nous sommes capables de fournir des explications dans une large proportion de cas.

Abstract

We explore the problem of providing explanations for pairwise comparisons based on an underlying additive model. We follow a step-wise approach and provide explanations that take the form of a sequence of preference statements. Each statement should be as meaningful, relevant and cognitively simple as possible for the explanation to be accepted by an explainee. More specifically, we describe several schemes allowing to derive new knowledge, in the form of comparative statements, from previously accepted ones. These schemes exploit a number of well-understood properties of the additive model, and we ensure the correctness of the overall explanatory sequences. While these different schemes may correspond to alternative explanation strategies, we specifically advocate the use of the covering scheme because it meets some desirable properties for explanations. Imposing cognitively simple steps comes at the price of completeness. However, experimental results show that we are able to provide insightful explanations in many cases.

1 Introduction

Dans cet article, nous abordons le problème de la production d'explications progressives pour les comparaisons par pairess d'alternatives basées sur des modèles de décision bien établis. Les alternatives sont caractérisées par un certain nombre de critères. L'idée principale est de décomposer la recommandation en affirmations simples présentées au destinataire de l'explication. L'ensemble de la séquence de comparaisons doit soutenir formellement la recommandation. Les explications que nous visons sont donc *contrastives*, dans le sens où la décision à expliquer compare deux alternatives, et *exactes* (par opposition à *heuristiques*) dans le sens où nous fournissons des garanties que l'explication produite est correcte par rapport au modèle sous-jacent. Il est également courant de distinguer entre les explications *locales* (lorsqu'elles se concentrent sur une recommandation spécifique) et les explications *globales* (lorsqu'elles traitent du modèle en général) : notre approche est globalement fidèle au modèle, et localement pertinente pour la comparaison par paires à expliquer. Enfin, même si cet aspect n'est pas détaillé dans ce travail, la perspective est de donner à celui qui re**c**coit l'explication la possibilité d'accepter ou de contredire ces affirmations. Pour rendre les choses concrètes, nous commen**c**cons par l'exemple suivant.

Exemple et motivations Nous considérons sept critères ($\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}, \mathsf{g}$), chacun décrit sur une échelle à

deux niveaux, ce qui facilite la représentation symbolique des alternatives (*e.g* des hôtels). Chaque alternative peut être représentée par son vecteur d'évaluation ($s_1 = (✗, ✗, ✓, ✓, ✓, ✓)$) ou plus succinctement par le sous-ensemble de critères sur lesquels elle est évaluée positivement ($s_1 = \{\mathbf{cdefg}\}$). De plus, pour chaque critère, la valeur symbolisée par ✓ est plus désirable que la valeur symbolisée par ✗ (*e.g* un hôtel avec un le petit déjeuner inclus est mieux que sans).

Table 1 – Évaluation de deux alternatives

	a	b	c	d	e	f	g
s_1	✗	✗	✓	✓	✓	✓	✓
s_2	✓	✗	✗	✓	✗	✗	✗

L'agrégation des critères se fait à l'aide d'une fonction de score additif, attribuant des poids aux différents critères. Cette fonction est comme suit :

$$w = \langle 128, 126, 77, 59, 52, 41, 37 \rangle$$

Par exemple, le score de s_1 correspond à $score(s_1) = 77 + 59 + 52 + 41 + 37 = 276$ tant dis que celui de s_2 is : $score(s_2) = 128 + 59 = 187$. Il est également utile d'encoder la comparaison de deux alternatives sous la forme d'un vecteur $\{-1, 0, +1\}^n$ d'arguments en faveur (PRO) ou contre (CON) s_1, ou neutres (NEU). Dans notre exemple, PRO = $\{\mathbf{c}, \mathbf{e}, \mathbf{f}, \mathbf{g}\}$, CON = $\{\mathbf{a}\}$, tandis que NEU = $\{\mathbf{b}, \mathbf{d}\}$:

Les explications peuvent prendre plusieurs formes. Nous listons différentes explications possibles au fait que s_1 est préféré à s_2 :

(i) La première approche que nous nommerons par (*divulgation du modèle*) consisterait à fournir le calcul complet du score pour les deux options, comme illustré ci-dessus. Mais en remarquant que **d**, un argument neutre, satisfait à la fois par s_1 et s_2, nous pourrions l'omettre et fournir simplement la somme des arguments PRO contre les arguments CON.

(ii) l'approche *contre-factuelle* recherche la modification minimale de l'entrée qui changerait le résultat. Par exemple, nous pourrions affirmer que, si s_2 avait satisfait **b**, s_2 aurait été recommandé plutôt que s_1. Ou (en affectant l'autre alternative cette fois), si s_1 n'avait pas satisfait **cd**.

(iii) En suivant une approche *prime implicant*, nous pourrions produire les arguments suffisants pour expliquer la décision. Dans notre cas, deux explications possibles pourraient être données : (1) étant donné que **bd** sont des arguments neutres, les arguments PRO **cef** sont suffisants pour contrer tout ensemble de CON arguments. En particulier, cela montre que la décision resterait la même même si **g** était un argument CON. Et (2) étant donné que **b** est un argument neutre, les arguments PRO **cefg** sont suffisants pour contrer tout ensemble d'arguments CON. En particulier, cela montre que la décision resterait la même même si **d** était un argument CON.

(iv) En suivant une approche *pas à pas ou progressive*, nous pourrions exposer une collection d'affirmations visant à prouver la décision. Par exemple, nous pourrions affirmer que **cdefg** est préféré à **ac**, et que **ac** est préféré à **ad**, de sorte que notre conclusion devrait tenir, suivant un raisonnement transitif. Ou, en utilisant une logique différente, nous pourrions affirmer que **cd** est préféré à **a**, tandis que **efg** est préféré à **d**, ce qui justifie entièrement notre décision.

Cet exemple nous permet d'illustrer certains principes clés de l'explication (voir par *e.g.* [18, 6]). :

— *L'intelligibilité du langage*— nous souhaitons que les explications soient transmises dans un langage qui soit significatif pour la personne qui re**c**coit cette explication. Dans notre exemple, les poids des critères utilisés dans les calculs peuvent ne pas être facilement interprétés par le destinataire de l'explication [1]

— *pertinence*— nous souhaitons que les explications se concentrent sur les informations pertinentes. Dans notre exemple, comme nous l'avons remarqué, la mention d'arguments neutres peut sembler non pertinente et doit être évitée si possible.

— *simplicité cognitive*— nous souhaitons que les explications soient "faciles à traiter" par le destinataire de l'explication. Ceci peut être instancié de différentes manières : les explications de type "prime implicante" cherchent des (sous-ensemble) raisons suffisantes minimales, tandis que les explications pas à pas (progressives) font appel à des comparaisons intermédiaires impliquant un nombre limité de critères.

Notre ambition dans cet article est de développer un modèle d'explications pas à pas (progressif) basé sur des principes et limité sur le plan cognitif. Comme l'illustre notre exemple, il peut y avoir différentes "logiques" en jeu lors de la combinaison des affirmations. Pour en tenir compte, nous décrivons un certain nombre de *schémas* pour de telles explications dans le contexte d'une comparaison basée sur un modèle de somme pondérée (Section 3). Par approche basée sur des principes, nous entendons que chaque schéma est attaché à un certain nombre de propriétés bien comprises du modèle de décision sous-jacent, que nous rendons explicites et discutons dans cet article. Le calcul résultant est correct (Section 4). Par "cognitivement limité", nous entendons que nos affirmations seront contraintes de manière à rester faciles à comprendre par le destinataire de l'explication. Cela a pour conséquence de rendre le calcul résultant *non* complet, mais nous explorons cette question en détail et fournissons plusieurs éléments montrant que notre ap-

1. Un autre aspect, non étudié ici, est qu'il peut ne pas être adéquat de divulguer entièrement le modèle pour des questions de confidentialité ou de manipulation.

proche est satisfaisante en termes de complétude empirique (Section 5).

2 Notre modèle

Nous considérons un ensemble d'éléments ou d'items, noté $[m]$. Une *comparaison par paires* est une paires d'alternatives $(A, B) \in 2^{[m]} \times 2^{[m]}$, interprétée comme une déclaration de préférence 'A est préféré à B'.

Les schémas Notre objectif est de fournir un langage formel et un mécanisme de raisonnement permettant de soutenir (expliquer) de tels comparaisons par pairess. Nous nous appuyons sur la notion de *schéma d'arguments*, c'est-à-dire un opérateur liant une séquence de comparaisons, appelée prémisse, satisfaisant certaines conditions, à une autre comparaison appelé conclusion [21]. Comme nous traitons des préférences, les schémas d'arguments sont des moyens de dériver de nouvelles préférences à partir de celles qui ont été établies précédemment. Il est à noter que tous nos schémas opèrent sur le même ensemble de prémisses – des séquences finies de comparaisons par pairess, représentés par des listes entre crochets – et le même ensemble de conclusions – de comparaisons par pairess dans des $2^{[m]} \times 2^{[m]}$. Nous désignerons un schéma arbitraire s comme suit :

$$[(A_1, B_1), \ldots, (A_k, B_k)] \xrightarrow{s} (A, B)$$

Correction Le fait que nos schémas d'arguments nous permettent uniquement de dériver des conclusions cohérentes avec la relation de préférence est capturé par la notion de correction :

Définition 1 *Un schéma d'argument est* correct *par rapport à une relation de préférence $\succsim$ si, lorsque toutes les prémisses appartiennent à $\succsim$, alors la conclusion appartient également à $\succsim$.*

A ce stade, nous laissons la relation de préférence non spécifiée, mais dans la section 3 nous approfondirons ce lien entre les propriétés des relations de préférence et les schémas.

Des affirmations simples L'approche *par étapes* est formalisée par l'exigence de *simplification* par rapport à la difficulté relative d'un énoncé.

Définition 2 *Une paires composée d'une prémisse $[(A_1, B_1), \ldots (A_k, B_k)]$ et d'une conclusion (A, B) est* simplifiante *lorsque la prémisse est moins difficile que la conclusion.*

Nous pensons que cette définition est très générale, car elle capture l'un des objectifs de l'explication. Pour être applicable, cependant, elle nécessite de spécifier la difficulté relative d'une prémisse et d'une conclusion.

Nous introduisons un modèle spécifique permettant de dériver la difficulté relative des énoncés, où cette difficulté est purement syntaxique et résulte directement du nombre d'items impliqués dans la comparaison par paires.

Définition 3 (Difficultés des affirmations) *La* difficulté *d'une comparaison par paires $(A, B) \in 2^{[m]} \times 2^{[m]}$ est la paires ordonnée d'entiers $(|A|, |B|)$. Par conséquent, on dit qu'une comparaison (A, B) est* moins difficile que *qu'une autre comparaison (A', B') lorsque $|A| \leq |A'|$, $|B| \leq |B'|$ et au moins une comparaison est stricte. Une séquence de comparaisons par paires $[(A_1, B_1), \ldots (A_k, B_k)]$ est* moins difficile qu'une *comparaison (A, B) lorsque toutes les comparaisons (A_i, B_i) sont moins difficiles que (A, B). Enfin, nous définissons des* classes de difficulté *de comparaisons par paires en fixant des limites supérieures à la difficulté : pour tous les entiers p, q de 0 à m, soit $\Delta(p, q) = \{(A, B) \in 2^{[m]} \times 2^{[m]} : |A| \leq p, |B| \leq q\}$.*

Notons $\mathcal{A}$ l'ensemble des éléments syntaxiquement atomiques, ceux qui sont considérés comme évidents et légitimes pour être utilisés comme étapes d'une explication pour le destinataire de l'explication considéré. Nous utiliserons les classes de difficulté $\Delta(p, q)$ pour spécifier cet ensemble. Dans le contexte de l'explication des préférences entre un sous-ensemble d'items désirés, certaines valeurs de la paires (p, q) présentent un intérêt particulier : les $\Delta(m, m)$ sont des énoncés non restreints ; les comparaisons dans $\Delta(m, 0)$ représentent des énoncés de dominance Pareto ; les comparaisons dans $\Delta(1, 1)$ peuvent être interprétées comme des *swaps* [12], représentant l'échange d'un critère contre un autre ; ceux en $\Delta(1, m)$ ou en $\Delta(m, 1)$ représentent un seul élément plus fort ou plus faible qu'un sous-ensemble d'autres, respectivement considérés comme un argument pour ou contre.

Par exemple, dans le contexte de comparaisons d'hôtels, un argument dans $\Delta(1, 1)$ pourrait être *"nous préférons avoir un petit-déjeuner gratuit qu'un accès wifi gratuit"*. Un argument dans $\Delta(1, 2)$ pourrait être *"Nous préférons avoir une piscine que le petit-déjeuner et le wifi gratuits"*. Pour comprendre à quel point il peut être difficile d'interpréter des arguments d'ordre supérieur, considérons des arguments dans $\Delta(2, 2)$. Ceux-ci pourraient correspondre à *"Un petit-déjeuner gratuit et un accès wifi sont préférables à avoir une piscine et à être proche du centre ville"*.

Dans la section 5, nous étudierons comment le fait de limiter l'explication à l'utilisation de ces classes d'énoncés simples affecte la capacité à produire des explications.

Explication basée sur des schémas. Les énoncés atomiques évidents limitent la difficulté de chaque étape d'une explication. Comme une explication est une séquence de

tels énoncés, nous cherchons également à produire des explications correctes de longueur minimale :

Définition 4 (Le problème d'explication) *Étant donné une comparaison par paires* $(A, B) \in 2^{[m]} \times 2^{[m]}$*, une relation de préférence* $\succsim$*, un ensemble d'énoncés* $\mathcal{A}$ *appartenant à* $\succsim$*, un ensemble de schémas* $\mathcal{S}$*, et un entier positif* k *: existe-t-il un entier positif* $k' \leq k$*, une liste de longueur* k' *d'énoncés* $[(A_1, B_1), \ldots, (A_{k'}, B_{k'})]$ *appartenant tous à* $\mathcal{A}$ *et un schéma* $s \in \mathcal{S}$ *tel que* $[(A_1, B_1), \ldots, (A_k, B_k)] \xrightarrow{s} (A, B)$ *?*

Notez que cette définition reste agnostique quant à la fa**c**con dont la relation de préférence est représentée dans les entrées.

Nous allons maintenant présenter dans la section suivante les différents schémas d'argument que nous pouvons trouver dans $\mathcal{S}$ et considérés pour le raisonnement sur les préférences.

3 Schémas de raisonnement sur les préférences

Cette section est consacrée à la construction de règles de dérivation adéquates pour raisonner sur les préférences. Nous formalisons ces règles comme des opérateurs liant une liste de prémisses à une conclusion, où prémisses et conclusions sont des comparaisons par paires. En faisant l'hypothèse d'un modèle additif, les schémas "*cancellation-based*" [2] ou une version simplifiée de ceux-ci, le schéma "*décomposition*", fournissent des règles qui sont correctes, mais la vérification de leur validité nécessite un calcul arithmétique qui peut sembler sans rapport avec la requête. Nous proposons d'éviter cette étape cognitivement insatisfaisante en identifiant les propriétés clés du modèle additif - la transitivité et la cancellation - et en formalisant des règles de dérivation tirant parti de chacune d'entre elles, à partir de la base : les schémas *transitif* et *ceteris paribus*. Nous introduisons ensuite le schéma *transitif réduit*, qui permet de dériver directement toute conclusion qui peut être prouvée à l'aide des deux schémas précédents. Ensuite, nous spécifions des exigences supplémentaires - *indépendance des éléments non pertinents (III)* [2] et commutation - qui donnent lieu à de nouveaux schémas, pour finalement aboutir au schéma *couverture*, qui particularise à la fois les schémas *transitif réduit* et *décomposition*.

3.1 Propriétés des préférences

Nous nous intéressons à la relation de préférence $\succsim$ qui pourrait exister entre les alternatives $A, B \in 2^{[m]}$, avec $A \succsim B$ signifiant que A est considéré comme au moins aussi bon que B.

2. Independance from Irrelevant Items

Nous rappelons quelques caractéristiques utiles que peuvent posséder les relations de préférence.

Définition 5 (Propriétés des préférences) *Soit* $\succsim \subset 2^{[m]} \times 2^{[m]}$ *une relation binaire entre alternatives. On dit :*

- $\succsim$ *est transitive lorsque, pour toute alternatives* $A, B, C \in 2^{[m]}$*, si* $A \succsim B$ *et* $B \succsim C$ *alors* $A \succsim C$ *;*
- $\succsim$ *satisfait cancellation (de premier ordre) si la préférence entre les alternatives ne dépend pas des éléments communs, c'est-à-dire* $\forall A, B \in 2^{[m]}$ $A \succsim B \iff (A \setminus B) \succsim (B \setminus A)$ *;*
- $\succsim$ *est additif quand il existe un m-tuple de nombres réels* $\langle \omega_i \rangle_{i \in [m]} \in \mathbb{R}^{[m]}$ *tel que* $A \succsim B \iff \sum_{i \in A} \omega_i \geq \sum_{i \in B} \omega_i$.
- $\succsim$ *est un ordre linéaire additif lorsqu'il est additif et qu'il n'y a pas d'indifférence, c'est-à-dire que si* $A \neq B$ *donc soit* $A \not\succsim B$ *ou* $B \not\succsim A$ *[10].*

De toute évidence, une préférence additive satisfait à la fois aux propriétés de transitivité et de cancellation.

3.2 Le schéma transitif

Comme nous nous effor**c**cons d'expliquer les recommandations dérivant d'un modèle de somme pondérée, nous pouvons mécaniser les propriétés de transitivité et d'annulation sous la forme de règles de dérivation. Par exemple, nous définissons le *schéma transitif binaire* $(2 - tr)$, permettant de chaîner les affirmations de préférence comme suit :

$$[(A, B), (B, C)] \xrightarrow{2-tr} (A, C)$$

En suivant l'approche que nous décrivons dans la section 2, étant donné un *explanandum* sous la forme d'une comparaison par paires (A, B) appartenant à $\succsim$, un *explanans* est une preuve consistant en des applications récursives d'une règle de dérivation – par exemple, *2-tr* – permettant de dériver la conclusion (A, B) à partir de prémisses acceptables (voir Section 2). Néanmoins, les preuves sont des objets récursifs qui peuvent être lourds à calculer ou à présenter au destinataire de l'explication, et nous proposons de pallier ce problème en introduisant des dispositifs de raisonnement plus puissants. En effet, considérons le cas d'un raisonnement purement transitif : enchaîner des lemmes transitifs revient à considérer des chaînes de prémisses transitives. Par exemple, si nous savons que $A \succsim B$, $B \succsim C$, $C \succsim D$ et $D \succsim E$, nous pouvons déduire que $A \succsim E$, en utilisant n'importe lequel des arbres de preuve décrits dans la Figure 1, et nous désignons $(A, B), (B, C), (C, D), (D, E) \vdash_{2-tr} (A, E)$. Nous pensons que cette abondance de preuves syntaxiques n'est pas pertinente pour la question du calcul des explications, et nous proposons donc de considérer le *schéma transitif* (tr) suivant.

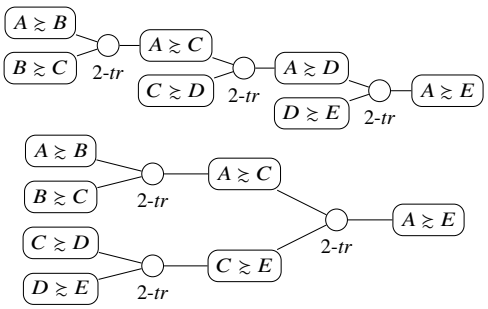


FIGURE 1 – Deux structures de preuve enchaînant des schémas transitifs binaires permettant de dériver la même conclusion à partir des mêmes axiomes.

Définition 6 (Schéma transitif (tr)) *La prémisse* $[(A_1, B_1), \ldots (A_k, B_k)]$ *et la conclusion* (A, B) *satisfont le* schéma transitif *quand, pour tout* $2 \leq j \leq k$, $A_j = B_{j-1}$, $A_1 = A$ *et* $B_k = B$.

Formellement, $\vdash_{2-tr} = \xrightarrow{tr}$ – *ce qui peut être prouvé en utilisant le schéma* $2 - tr$ *est exactement ce qui peut être dérivé en une seule application du schéma* tr. Nous avons échangé la nature récursive de la preuve utilisant un schéma binaire pour une dérivation *one-shot* utilisant un schéma opérant sur une liste de prémisses de longueur non limitée.

Exemple 1 *La prémisse* $[(\mathbf{acg}, \mathbf{bef}), (\mathbf{bef}, \mathbf{bfg})]$ *satisfait syntaxiquement le schéma transitif pour la conclusion* $(\mathbf{acg}, \mathbf{bfg})$ *:*

$$[(\mathbf{acg}, \mathbf{bef}), (\mathbf{bef}, \mathbf{bfg})] \xrightarrow{tr} (\mathbf{acg}, \mathbf{bfg})$$

Qui peut être exprimé comme suit : "Dès lors que $\mathbf{acg} \succ \mathbf{bef}$ *et* $\mathbf{bef} \succ \mathbf{bfg}$, $\mathbf{acg}$ *devrait être préféré à* $\mathbf{bfg}$*". Notons toutefois que la première comparaison est complexe puisqu'il fait intervenir six critères différents.*

Notez que les comparaisons par paires composant la prémisse d'un schéma transitif sont ordonnées, donc la séquence d'alternatives $A \equiv A_0 \gtrsim B_0 \equiv A_1 \gtrsim \cdots \gtrsim B_{k-1} \equiv A_k \gtrsim B_k \equiv B$ est non croissante par rapport à la préférence.

Observation 1 *Si la préférence* $\gtrsim$ *est transitive, alors le schéma transitif est correct w.r.t.* $\gtrsim$ *: si toutes les comparaisons par pairess des prémisses sont dans* $\gtrsim$, *alors la conclusion est également une comparaison dans* $\gtrsim$.

Exemple 2 *Dans l'exemple 1, la prémisse appartient à* $\gtrsim$, *car pour* $(\mathbf{acg}, \mathbf{bef})$ *nous avons :* $242 = w_a + w_c + w_g > w_b + w_e + w_f = 219$, *et pour* $(\mathbf{bef}, \mathbf{bfg})$ *:* $219 = w_b + w_e + w_f > w_b + w_f + w_g = 204$. *Cependant, si nous utilisons la prémisse* $[(\mathbf{acg}, \mathbf{abc}), (\mathbf{abc}, \mathbf{bfg})]$, *cette dernière satisfait le schéma transitif mais n'appartient pas à* $\gtrsim$, *puisque pour* $(\mathbf{acg}, \mathbf{abc})$, *nous avons :* $242 = w_a + w_c + w_g < w_a + w_b + w_c = 331$ *(voir la fonction de score* w *dans la Section 2).*

3.3 Le schéma *ceteris paribus*

La propriété de cancellation permet de raisonner *ceteris paribus* – tout le reste étant égal – indépendamment du contexte, et représente une grande opportunité en termes d'explication, puisque la préférence peut être déduite de comparaisons par paires où les éléments communs ne sont pas mentionnés. Elle motive la définition suivante du schéma argumentatif *ceteris paribus*.

Définition 7 (Le schéma Ceteris paribus (cp)) *La prémisse* $[(A_1, B_1), \ldots (A_k, B_k)]$ *et la conclusion* (A, B) *satisfont le* schéma ceteris paribus *quand* $k = 1$, $A_1 \backslash B_1 = A \backslash B$ *et* $B_1 \setminus A_1 = B \setminus A$. *Dans ce cas, les comparaisons* (A_1, B_1) *et* (A, B) *sont dites* congruents.

De toute évidence, la congruence est une relation d'équivalence entre les comparaisons par paires. Dans la classe de congruence d'une comparaison donnée $(A, B) \in 2^{[m]} \times 2^{[m]}$, la comparaison $(A \setminus B, B \setminus A)$, où les alternatives sont disjointes par paires et obtenues à partir de (A, B) en soustrayant les éléments communs $A \cap B$ respectivement de A et B, présente un intérêt particulier. Lorsque (A, B) et (A_1, B_1) sont congruents, la préférence de A sur B se traduit par la préférence de $A \cap B$ sur $B \cap A$ en considérant que "tout le reste" – dans ce cas, les éléments de $A_c apB$ – "est égal" (ceteris paribus), puis à la préférence de A_1 sur B_1 en considérant $A_1 \cap B_1$ non pertinent.

Exemple 3 *En comparant* $\mathbf{acg}$ *à* $\mathbf{bfg}$, *il pourrait être justifié de considérer que, puisque* $\mathbf{g}$ *apparaît dans les deux alternatives, ce critère peut être omis : "la première option est meilleure que la seconde car, toutes choses égales par ailleurs,* $\mathbf{ac}$ *est préférable à* $\mathbf{bf}$*".*

Formellement, nous écrivons $[(\mathbf{ac}, \mathbf{bf})] \xrightarrow{cp} (\mathbf{acg}, \mathbf{bfg})$

Observation 2 *Si la préférence* $\gtrsim$ *satisfait la cancellation, alors le schéma* ceteris paribus *est correct en ce qui concerne* $\gtrsim$.

Nous serons souvent confrontés au problème inverse : étant donné un état initial, existe-t-il un état final tel que la comparaison par paires de l'état initial à l'état final est congruent à une comparaison par paires donnée ?

Lemme 1 (Quatrième problème de congruence) *Étant donnée une alternative* $A \in 2^{[m]}$ *et une comparaison par paires* $(A', B') \in 2^{[m]} \times 2^{[m]}$, *si* $A \supseteq (A' \setminus B')$ *et* $A \cap (B' \setminus A') = \emptyset$ *il existe exactement une alternative* $B \in 2^{[m]}$ *tel que* (A, B) *et* (A', B') *sont congruents, avec* $B = A \setminus (A' \setminus B') \cup (B' \setminus A')$, *sinon il n'en existe pas.*

Exemple 4 *Considérons la paire* $(\mathbf{ade}, \mathbf{bce})$. *L'ensemble des comparaisons par paires congruentes à celle-ci est :* $\{(\mathbf{ad}, \mathbf{bc}), (\mathbf{adf}, \mathbf{bcf}), (\mathbf{adg}, \mathbf{bcg}), (\mathbf{adef}, \mathbf{bcef}), (\mathbf{adeg}, \mathbf{bceg}), (\mathbf{adfg}, \mathbf{bcfg}), (\mathbf{adefg}, \mathbf{bcefg})\}$. *Les alternatives*

de départ {**ad**, **ade**, **adf**, **adg**, **adef**, **adeg**, **adfg**, **adefg**} *sont exactement les sur-ensembles de* **ad** *qui ne contiennent pas* **bc**, *et pour chacun d'entre eux, il n'existe qu'un seul état correspondant. Cela a du sens lorsque la comparaison pas paires* (**ade**, **bce**) *est comprise comme 'donner* **ade**, *prendre* **bce**', *dont* **e** *peut être omis. Ensuite,* **ad** *ne peut être effectivement pris que dans un état qui le contient déjà, et* **bc** *ne peut être effectivement ajouté que dans un état qui ne le contient pas encore.*

3.4 Le schéma transitif réduit

Lorsque la préférence satisfait à la fois aux propriétés de transitivité et de cancellation, il est correct d'utiliser à la fois les schémas *tr* et *cp* pour dériver de nouvelles comparaisons par paires.

La Figure 2 illustre une telle preuve. Pour des raisons pratiques, nous souhaitons rationaliser et mécaniser ce modèle de raisonnement, en formalisant un schéma - appelé *transitif réduit (rt)* - liant directement la prémisse $[(\mathtt{a},\mathtt{b}),(\mathtt{c},\mathtt{f})]$ à la conclusion $(\mathtt{aceg},\mathtt{befg})$ et en laissant les étapes intermédiaires non spécifiées [3].

$$\left.\begin{array}{l}(\mathtt{a},\mathtt{b}) \xrightarrow{cp} (\mathtt{acg},\mathtt{bcg})\\ (\mathtt{c},\mathtt{f}) \xrightarrow{cp} (\mathtt{bcg},\mathtt{bfg})\end{array}\right\} \xrightarrow{tr} (\mathtt{acg},\mathtt{bfg}) \xrightarrow{cp} (\mathtt{aceg},\mathtt{befg})$$

FIGURE 2 – Une preuve de $[(\mathtt{a},\mathtt{b}),(\mathtt{c},\mathtt{f})] \vdash_{cp,tr} (\mathtt{aceg},\mathtt{befg})$

Définition 8 (Le schéma transitif réduit (*rt*)) $[(A_1,B_1),\ldots,(A_k,B_k)] \xrightarrow{rt} (A,B)$ *lorsqu'il existe* $(A'_1,B'_1),\ldots,(A'_k,B'_k)$ *and* (A',B') *tel que :*

$$\begin{cases}\forall i \in [k]\ [(A'_i,B'_i)] \xrightarrow{cp} (A_i,B_i);\\ [(A'_1,B'_1),\ldots,(A'_k,B'_k)] \xrightarrow{tr} (A',B');\ \textit{and}\\ [(A',B')] \xrightarrow{cp} (A,B).\end{cases}$$

Exemple 5 *La preuve représentée par la Figure 2 peut être lue comme suit :*

"On doit préférer **acg** *à* **bcg**, *puisque toutes choses étant égales par ailleurs,* **a** *est préféré à* **b**. *Alors* **bcg** *doit être préféré à* **bfg**, *puisque, toutes choses égales par ailleurs,* **c** *est préféré à* **f**. *Donc* **acg** *devrait être préféré à* **bfg**."

Nous notons :

$$[(\mathtt{a},\mathtt{b}),(\mathtt{c},\mathtt{f})] \xrightarrow{rt} (\mathtt{aceg},\mathtt{befg})$$

3. Cela revient à permettre au schéma *tr* d'opérer sur l'ensemble quotient des classes congruentes de comparaisons par paires, au lieu d'être représentatif de ces classes.

Nous pouvons noter que la définition 8 est inefficace, car l'espace de recherche des séquences de comparaisons de longueur k est fini, mais intractablement grand. Le manque de spécification peut être surmonté en reconstruisant les étapes intermédiaires manquantes, en résolvant itérativement k quatrièmes problèmes congruents (voir Lemme 1), comme résumé par Algorithme 1 qui s'exécute en $O(km^2)$.

Le schéma *rt* remplit son rôle en permettant la dérivation en une seule étape de preuves combinant les dérivations *tr* et *cp*.

Proposition 1 *Si les prémisses* $\mathcal{P} = \bigcup_{j=1}^{n}(A_j,B_j)$ *permettent de prouver que la conclusion* (A,B) *via les schémas cp et tr, alors il existe une liste* $[P'_1,\ldots,P'_k]$ *de comparaisons par paires* $\mathcal{P}$ *tel que* $[P'_1,\ldots,P'_k] \xrightarrow{rt} (A,B)$.

Démonstration 1 *Par construction, le schémas rt subsume les schémas tr et cp (donc* $\vdash_{tr,cp} \subset \vdash_{rt}$*), et particularise une preuve combinant des dérivations de tr et cp (donc* $\vdash_{tr,cp} \supset \vdash_{rt}$*). Moreover, chaining rt derivations is useless, because if* $L_1 \xrightarrow{rt} (A,B)$ *et* $L_2 \xrightarrow{rt} (B,C)$*, alors* $L_1 \& L_2 \xrightarrow{rt} (A,C)$*, où* & *est la concaténation de liste. Du coup,* $\vdash_{rt} = \xrightarrow{rt}$.

Algorithm 1 checking an instance of the *rt* scheme

Require: a list of comparative statements $[(A_1,B_1),\ldots,(A_k,B_k)]$, a comparative statement $(A,\ B)$
Ensure: **true**, if the premise and conclusion satisfy the *rt* scheme ;
  false, else
  $A'_1 \leftarrow A \setminus B$
  for $i = 1$ **to** k **do**
    if $A'_i \not\supseteq A_i$ **or** $A'_i \cap B_i \neq \emptyset$ **then**
      return False
    else
      $B'_i \leftarrow A'_i \setminus A_i \cup B_i$
      $A'_{i+1} \leftarrow B'_i$
    end if
  end for
  return $(B \setminus A = B'_k)$

3.5 Le schéma III - transitif réduit

Lors d'un raisonnement sur la préférence d'une alternative A par rapport à une autre alternative B, la mention d'un élément *neutre* j qui n'est ni dans A ni dans B, ou à la fois dans A et B, pourrait être considérée comme préjudiciable à l'explication. En effet, en raison de la propriété de cancellation, il pourrait être jugé inutile ; il pourrait être considéré comme hors sujet et vu comme un dispositif purement rhétorique ; il révèle des informations qui restent latentes lorsqu'on suit la stratégie *divulgation du modèle*. Nous appelons cette propriété *Indépendance des éléments non pertinents (Independence of Irrelevant Items –III)*, et nous l'appliquons de manière normative en considérant une

restriction du schéma transitif réduit aux instances qui la satisfont.

Définition 9 *Le schéma III- transitive réduit (III-rt)] Une prémisee* $[(A_1, B_1), \ldots, (A_k, B_k)]$ *et une conclusion* (A, B) *satisfont le* schéma III-transitive réduit *lorsqu'elle satisfont le schéma transitive réduit et* $\left(\bigcup_{i=1}^{k} A_i \cup \bigcup_{i=1}^{k} B_i\right) \subseteq (A \cup B) \setminus (A \cap B)$.

Exemple 6 *Le schéma dans l'Exemple 5 est un schéma III-transitive réduit. Cependant, si pour la même conclusion, nous utilisons la prémisse :* $[(\mathbf{acg}, \mathbf{bef}), (\mathbf{e}, \mathbf{g})]$, *le schéma ne l'est plus. En effet, les comparaisons par paires font intervenir le critère* $\mathbf{e}$ *qui n'apparaît pas dans la conclusion et peut sembler non pertinent.*

3.6 Le schéma décomposition

Introduit dans [2] et mettant en oeuvre des propriétés de cancellation d'ordre supérieur, le schéma décomposition vise à tirer parti de la propriété additive supposée de la relation de préférence [4]. Lorsque la préférence est additive, les affirmation de préférence se traduisent par des comparaisons linéaires, qui peuvent être additionnées. Ensuite, les scores des éléments apparaissant de part et d'autre s'annulent, permettant parfois de dériver de nouvelles comparaisons.

Définition 10 (Le schéma décomposition (*dec*)) *Une prémisse* $[(A_1, B_1), \ldots, (A_k, B_k)]$ *et une conclusion* (A, B) *satisfont le* schéma décomposition *lorsque chaque comparaison par paires* (A_i, B_i) *est disjointe et, pour touts les items* $j \in A \setminus B$, *il y a autant d'occurrences de* j *in the sets* $A_1, \ldots, A_k$ *qu'il y a dans les ensembles* $B_1, \ldots, B_k$ *plus un; pour tous les items* $j \in B \setminus A$, *il y a autant d'occurrences de* j *dans les ensembles* $B_1, \ldots, B_k$ *qu'il y a dans les ensembles* $A_1, \ldots, A_k$ *plus un; et pour tout élément* j *qui n'est ni dans* A *ni dans* B, *ou à la fois dans* A *et* B, *il y a autant d'occurrences de* j *dans les ensembles* $A_1, \ldots, A_k$ *que dans les ensembles* $B_1, \ldots, B_k$, *soit* $\forall j \in [m]$

$$\sum_{i=1}^{k} |A_i \cap \{j\}| + |B \cap \{j\}| = \sum_{i=1}^{k} |B_i \cap \{j\}| + |A \cap \{j\}|$$

Proposition 2 *Si la préférence* $\succsim$ *est additive, alors le schéma décomposition est correct par rapport à* $\succsim$.

Exemple 7 *Considérons le schéma décomposition suivant :*

$$[(\mathbf{bc}, \mathbf{de}), (\mathbf{efg}, \mathbf{ac})] \xrightarrow{dec} (\mathbf{bfg}, \mathbf{ad})$$

En supposant que la préférence $\succsim$ *est additive, et que* $\mathbf{bc} \succsim \mathbf{de}$ *et* $\mathbf{efg} \succsim \mathbf{ac}$. *De la première comparaison, nous déduisons que* $\omega_{\mathbf{b}} + \omega_{\mathbf{c}} \geq \omega_{\mathbf{d}} + \omega_{\mathbf{e}}$; *de la seconde que* $\omega_{\mathbf{e}} + \omega_{\mathbf{f}} + \omega_{\mathbf{g}} \geq \omega_{\mathbf{a}} + \omega_{\mathbf{c}}$. *Par addition, nous obtenons* $\omega_{\mathbf{e}} + \omega_{\mathbf{f}} + \omega_{\mathbf{g}} + \omega_{\mathbf{b}} + \omega_{\mathbf{c}} \geq \omega_{\mathbf{d}} + \omega_{\mathbf{e}} + \omega_{\mathbf{a}} + \omega_{\mathbf{c}}$.

Ensuite, comme l'illustre la Figure 3 en annulant $\omega_{\mathbf{e}}$ *et* $\omega_{\mathbf{c}}$ *des deux côtés (il s'agit en fait d'une instance de* cacellation du second ordre*, car elle est effectuée sur deux comparaisons par paires), nous obtenons* $\omega_{\mathbf{f}} + \omega_{\mathbf{g}} + \omega_{\mathbf{b}} \geq \omega_{\mathbf{d}} + \omega_{\mathbf{a}}$, *alors* $\mathbf{bfg} \succ_{\omega} \mathbf{ad}$.

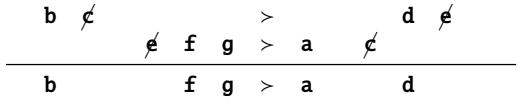


FIGURE 3 – Schéma décomposition : représentation graphique

Le schéma décomposition généralise strictement les schémas introduits précédemment.

Proposition 3 *Si la prémisse* $[(A_1, B_1), \ldots, (A_k, B_k)]$ *et la conclusion* (A, B) *satisfont le schéma transitif ou le schéma transitif réduit, alors elles satisfont le schéma décomposition.*

Nous notons que le schéma décomposition est commutatif par construction. Néanmoins, il ne semble pas très satisfaisant en tant que dispositif d'explication, car les propriétés de cancellation d'ordre supérieur qu'il met en oeuvre sont complexes et de faible intérêt normatif - en fait, même si elles sont syntaxiquement transductives, leur justification dérive de la forme additive que nous nous effor**c**cons de contourner.

En général, les instances du schéma décomposition ne satisfont pas le schéma transitif réduit. La séquence de comparaisons par paires de la prémisse ne peut pas être interprétée comme des justifications *ceteris paribus* de comparaisons par paires entre alternatives, car à un moment donné, ils nécessitent soit d'ajouter un élément à une alternative où il est déjà présent, soit de retirer un élément d'une alternative où il est absent.

Exemple 8 *La prémisse* $[(\mathbf{bc}, \mathbf{de}), (\mathbf{efg}, \mathbf{ac})]$ *et la conclusion* $(\mathbf{bfg}, \mathbf{ad})$ *satisfont le schéma décomposition, mais ne peuvent être interprétées comme une séquence de comparaisons par paires parce que par exemple l'alternative initiale* $\mathbf{bfg}$ *de la conclusion ne contient pas* $\mathbf{c}$.

Cette situation peut être évitée lorsque les alternatives mentionnées dans la prémisse sont toutes disjointes par pairess.

4. Ce schéma de décomposition est moins général que le schéma dit *syntactic cancellative* décrit dans [2], car il ne permet pas la répétition de la conclusion. Il a été démontré que cela réduit l'expressivité

Proposition 4 *Soient* $[(A_1, B_1), \ldots, (A_k, B_k)]$ *une prémisse et une conclusion* (A, B) *satisfaisant le schéma décomposition. Chaque permutation* σ *des indices* $[k]$ *pemet à la prémisse* $[(A_{\sigma(1)}, B_{\sigma(1)}), \ldots, (A_{\sigma(k)}, B_{\sigma(k)})]$ *et la conclusion* (A, B) *de satisfaire le schéma transitif réduit si et seulement si les alternatives* $A_1, \ldots, A_k, B_1, \ldots, B_k$ *sont disjointes par paires.*

Démonstration 2 *Supposons sans perte de généralité que* $A \cap B = \emptyset$*, et soit* j *apparaît dans au moins dans deux alternatives* $A_1, \ldots, A_k, B_1, \ldots, B_k$*.*

— *Supposons que* $j \notin A$*. S'il apparaît dans* A_i*, alors laissez* $\sigma(1) := i$*, sinon il apparaît dans* B_i *et* $B_{i'}$ *et donc* $\sigma(k-1) := i$ *et* $\sigma(k) := i'$*. Cela met en échec soit le premier ou le dernier quatrième problème congruent de Alg. 1.*

— *Supposons que* $j \notin B$*. S'il apparaît dans* B_i*, alors laissez* $\sigma(k) := i$*, sinon il apparaît dans* A_i *et* $A_{i'}$ *et donc* $\sigma(1) := i$ *et* $\sigma(2) := i'$*. Cela met en échec soir la vérification finale, soit le deuxième quatrième problème congruent de Alg. 1.*

Réciproquement, lorsque toutes les alternatives $A_1, \ldots, A_k, B_1, \ldots, B_k$ *sont disjointes par pairess, chaque élément de* $A \setminus B$ *apparaît exactement une fois dans le* B_i *et jamais dans le* A_i*, chaque élément de* $B \setminus A$ *apparaît exactement une fois dans le* A_i *et jamais dans le* B_i*, et les éléments des deux ensembles ou aucun n'apparaît jamais (ce qui rend incidemment la transaction III). Par conséquent, les éléments de l'ensemble* $\bigcup_{i=1}^{k} A_i$ *peuvent être retirés dans n'importe quel ordre de A et ceux de l'ensemble* $\bigcup_{i=1}^{k} B_i$ *peuvent être ajoutés dans n'importe quel ordre, de fa**c**con à s'accumuler dans B.*

Nous définissons en conséquence le *schéma couverture*.

3.7 Le schéma couverture

Dans ce schéma, une liste de comparaisons $[(A_1, B_1), \ldots, (A_k, B_k)]$ soutient une conclusion (A, B) si, et seulement si, les *pros* $A_1, \ldots, A_k$ partitionnent $A \setminus B$ et les *cons* $B_1, \ldots, B_k$ partitionnent $B \setminus A$.

Définition 11 (Le schéma couverture (cov)**)** *Une instance du* schéma couverture *est une instance du schéma décomposition où toutes les alternatives* $A_1, \ldots, A_k, B_1, \ldots, B_k$ *sont disjointes par paires.*

Le schéma couverture est commutatif et indépendant des éléments non pertinents par construction. Il particularise les schémas transitifs réduits – tout en contournant l'ordonnancement fastidieux des comparaisons par paires – et, comme corollaire, il est correct dans des conditions beaucoup plus souples que le schéma décomposition.

Proposition 5 *Si la préférence* $\succsim$ *est transitive et satisfait la cancellation, alors le schéma couverture est correct en ce qui concerne* $\succsim$*.*

Le schéma couverture décrit exactement l'algèbre morale introduite par Benjamin Franklin (voir [22]) pour déduire les préférences.

Exemple 9 *Considérons la conclusion :* $(\mathbf{bfg}, \mathbf{cde})$*. La prémisse* $[(\mathbf{fg}, \mathbf{e}), (\mathbf{b}, \mathbf{cd})]$ *constitue un schéma couverture*

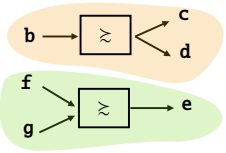
$$[(\mathbf{fg}, \mathbf{e}), (\mathbf{b}, \mathbf{cd})] \xrightarrow{cov} (\mathbf{bfg}, \mathbf{cde})$$

Le schéma couverture particularise à la fois le schéma transitif réduit et le schéma décomposition. En tant que tel, il nous donne le meilleur des deux mondes, en un sens. D'une part, il formalise une preuve, articulant des dérivations transitives et *ceteris paribus*, qui peut être présentée au destinataire de l'explication sous forme de diagramme, comme dans la Figure 4a, ou de manière narrative comme dans la Figure 4c (pour des comparaisons d'hôtels par exemple). D'un autre côté, les prémisses peuvent être comprises comme regroupant des contre avec des pour plus forts, de manière à "couvrir" les contre, et peuvent être présentées visuellement au destinataire de l'explication, comme dans la Figure 4b.

(a) Schéma couverture : diagramme de preuve de Ex. 9

$$\left.\begin{array}{l} \mathtt{fg} \succ \mathtt{e} \xrightarrow{cp} \mathtt{bfg} \succ \mathtt{be} \\ \mathtt{b} \succ \mathtt{cd} \xrightarrow{cp} \mathtt{be} \succ \mathtt{cde} \end{array}\right\} \xrightarrow{tr} \mathtt{bfg} \succ \mathtt{cde}$$

(b) Schéma couverture : une représentation visuelle de Ex.9



(c) Schéma couverture : une représentation narrative de Ex.9

*"As, all other things being equal, having free breakfast and wifi access is preferred to having a swimming pool (**fg**, **e**), and being close to the city is preferred than having a sports hall and a low tourist tax (**b**, **cd**), we get that (**bfg**, **cde**)"*

FIGURE 4 – Trois représentations du schéma couverture

Notez également qu'un schéma couverture unique de longueur k peut être interprété comme $k!$ schémas transitifs, puisque la validité de ses prémisses ne dépend pas de leur ordre. Ainsi, par exemple pour notre Ex. 9, deux schémas transitifs correspondraient à ce schéma couverture : $[(\mathtt{bfg}, \mathtt{cdfg}), (\mathtt{cdfg}, \mathtt{cde})]$ or $[(\mathtt{bfg}, \mathtt{be}), (\mathtt{be}, \mathtt{cde})]$.

Résumé. Le Tableau 2 et la Figure 5 résument les schémas d'arguments introduits dans cette section, leurs propriétés, les exigences de la relation de préférence pour leur exactitude, et leurs relations.

Schéma	Propriétés	Exigences pour correction
décomposition	commutative	additivité
transitif réduit		transitivité + cancellation
III-transitif red.	III	transitivité + cancellation
covering	commutative, III	transitivité + cancellation
transitif.		transitivité
ceteris paribus		cancellation

Table 2 – Propriétés structurelles des schémas de raisonnement.

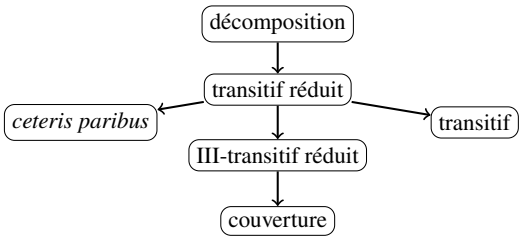


Figure 5 – Relations entre les schémas d'arguments. Une flèche allant de $scheme_1$ à $scheme_2$ indique que toutes les instances satisfaisant $scheme_2$ satisfont également $scheme_1$, mais pas l'inverse.

4 Expliquer avec des schémas

La Section 3 portait sur la conception d'un outil déductif permettant de dériver des préférences complexes à partir de préférences plus simples, d'une manière *correcte* par rapport au modèle de préférence latent. Étant donné un *explanans*–une comparaison par paires qui doit être expliquée–il nous permet de présenter le *problème d'explication* comme un problème d'*abduction* consistant à trouver des prémisses qui satisfont une certaine exigence de minimalité étant donné une conclusion et un ensemble de règles. Dans cette section, nous étudions l'expressivité relative et la complexité de calcul de l'explication avec les schémas *transitif réduite* (rt)–see Def. 8–et *couverture* (cov)–see Def. 11– ainsi que l'influence du choix des énoncés atomiquement simples.

À partir de maintenant, nous appellerons $\mathcal{E}(s, \mathcal{A}_{\succsim},)$ l'ensemble des pairess (A, B) qui peuvent être dérivées en utilisant le schéma s à partir de comparaison respectant les contraintes syntaxiques de $\mathcal{A}$, la contrainte sémantique d'être cohérent avec $\succsim$, et impliquant au plus k prémisses. Pour une paires donnée (A, B), le problème de l'existence de l'explication demande si $(A, B) \in \mathcal{E}(s, \mathcal{A}_{\succsim}, k)$. Par convention, cette paires (A, B) est considérée comme non évidente.

4.1 Résoudre les problèmes d'explication à l'aide de schémas

La figure 5 indique les dépendances logiques existant entre les prémisses et les conclusions satisfaisant les différents schémas. Cela a une conséquence évidente sur les relations d'explicabilité : plus général implique plus explicatif. De plus, les conclusions du schéma transitif peuvent toutes être obtenues via le schéma transitif réduit, et celles du *ceteris paribus* via les schémas décomposition, transitif réduit ou couverture.

Lorsque la préférence est additive, tous nos schémas de raisonnement sont corrects. Cependant, nous ne pouvons pas nous attendre à ce qu'ils soient *complets*, même sans aucune restriction syntaxique sur $\mathcal{A}$: lorsque $A \succ B$ sont adjacents dans la relation de préférence $\succsim$, c'est-à-dire lorsqu'il n'y a pas une autre alternative X s.t. $A \succ X \succ B$, (A, B) est appelée une *paires critique*, voir [10]. Mais cela signifie à son tour que la conclusion (A, B) ne peut pas être obtenue avec le schéma rt - encore moins par les schémas III-rt, cov et tr.

Ces pairess critiques ne sont donc pas explicables avec ces schémas (voir Ex.10).

Exemple 10 *La conclusion* $(\mathbf{bcd}, \mathbf{aefg})$ *constitue une paires critique. En effet,* $\omega_{\mathbf{b}} + \omega_{\mathbf{c}} + \omega_{\mathbf{d}} = 262$, $\omega_{\mathbf{a}} + \omega_{\mathbf{e}} + \omega_{\mathbf{f}} + \omega_{\mathbf{g}} = 258$, *et il n'est pas possible d'exhiber une autre alternative parmi les* $2^{[m]}$ *avec une note* $\in]258, 262[$.

En revanche, dès qu'une paires *n'est* pas critique, et à condition que $\mathcal{A}$ ne mette aucune contrainte syntaxique sur les comparaisons utilisées, il doit exister au moins une explication avec les schémas tr et rt. Cela signifie que la complexité de décider de l'existence d'une explication pour ces schémas est directement liée à celle de décider si une paires est critique. Nous montrons que ce problème est difficile.

Théorème 1 *Étant donné* $\omega \in \mathbb{N}_0^m$, *et* $A, B \in 2^{[m]}$ *tels que* $A \succsim B$ *où* $\succsim$ *est la relation de préférence additive induite de* ω. *Décider si* (A, B) *est une paires critique est Co-NP-complete.*

Démonstration 3 *Réduction de* subset-sum *[11].*

Dans subset-sum, *on nous donne un ensemble* A *de taille* m, *un poids positif* $w(a)$, *pour chaque* a *dans*A, *et un entier positif* B. *On se demande s'il existe un sous-ensemble* $A' \subseteq A$ *tel que la somme des poids soit exactement de* K. *Ce problème est connu pour être NP-complet. Nous construisons une instance du problème de la paires critique comme suit. On prend un ensemble* C *de* $m + 2$ *critères : pour chaque élément* $a_i \in A$, *on prend un critère* c_i, *de poids* $2s(a_i)$. *On ajoute deux autres critères :* a_{n+1} *de poids* $2K-1$, *et* a_{n+2}, *de poids* $2K+1$. *Nous demandons si la paires* (X, Y) *est critique, où* $X = \langle 0, \ldots, 1, 0\rangle$ *et* $Y = \langle 0, \ldots 1\rangle$. *Notons que* X *et* Y *ont des poids respectifs de* $2K - 1$ *et*

$2K+1$, on cherche donc une alternative intermédiaire de poids exactement $2K$, un nombre pair. Nous affirmons que la réponse à cette question est non si le problème original de SUBSET-SUM *est une instance de oui. Pour s'en convaincre, il suffit d'observer que tous les poids de notre instance de paires critique sont pairs, sauf ceux de c_{n+1} et c_{n+2} qui sont impairs. Comme $s(c_{n+2}) > 2K$, il ne peut certainement pas faire partie de la solution. De plus, la solution ne peut pas inclure c_{n+1}, car dans ce cas, il serait le seul poids impair, et la somme serait alors impaires. Nous nous retrouvons avec les critères $c_1, \ldots c_n$, dont les poids sont précisément deux fois plus élevés que ceux du problème original de la somme des sous-ensembles.*

En corollaire, il est difficile de décider si une explication existe avec ces schémas.

Corollaire 1 *Étant donné $\omega \in \mathbb{N}_0^m$ $A, B \in 2^{[m]}$ tel que $A \gtrsim B$, où $\gtrsim$ est la relation de préférence additive induite par ω, et $\mathcal{A}$ l'ensemble des affirmations $\Delta(m,m)$. Décider si $(A,B) \in \mathcal{E}(s, \mathcal{A}_{\gtrsim}, +\infty)$ est NP-complete pour $s \in \{rt, tr\}$.*

Pour notre schéma de choix cov, nous avons le résultat suivant par une preuve indépendante.

Théorème 2 *Étant donné $\omega \in \mathbb{N}_0^m$, $A, B \in 2^{[m]}$ tel que $A \gtrsim B$ where $\gtrsim$ est une relation de préférence additive induite par ω et $\mathcal{A}$ l'ensemble des affirmations $\Delta(m,m)$. Décider si $(A,B) \in \mathcal{E}(cov, \mathcal{A}_{\gtrsim}, +\infty)$ est NP-complete.*

Démonstration 4 *(sketch) Membership is obvious, as the scheme itself is a polynomial certificate. Hardness results from reduction from* BIN-PACKING *[11].*

En ce qui concerne les autres schémas, alors que cp est facile, nous conjecturons que la complexité de dec et de III-rt est intractable.

4.2 Expliquer avec des affirmations atomiques

Nous abordons maintenant les explications qui imposent des restrictions syntaxiques aux ensembles d'éléments atomiques utilisés, $\Delta(1,1)$, $\Delta(1,m)$, and $\Delta(m,1)$ (voir Sect.2).

Théorème 3 *Lorsque la relation $\gtrsim$ est additive, $\mathcal{E}(cov, \mathcal{A}_{\gtrsim}, \infty)$ est transitive lorsque $\mathcal{A}_{\gtrsim} \in \{\Delta(1,1), \Delta(1,m), \Delta(m,1)\}$.*

Démonstration 5 *Supposons $[(A_1,B_1),\ldots,(A_k,B_k)] \xrightarrow{cov} (A,B)$ et $[(A'_1,B'_1),\ldots,(A'_k,B'_k)] \xrightarrow{cov} (B,C)$. Nous montrons que la conclusion (A,C) est obtenu en appliquant le schéma couverture à une certaine prémisse. Il est facile de vérifier que $[(A_1,B_1),\ldots,(A_k,B_k),(A'_1,B'_1),\ldots,(A'_k,B'_k)] \xrightarrow{dec} (A,C)$, et nous avons seulement besoin de prouver que les ensembles $A_1,\ldots, A_k$, $A'_1, \ldots, A'_{k'}$, $B_1,\ldots, B_k$, $B'_1, \ldots, B'_{k'}$ sont disjoints par paires. Nous savos déjà que $\langle A_i, B_{i'}\rangle$ et que $\langle A'_i, B'_{i'}\rangle$ sont disjoint par paires. De plus, A_i sont inclus dans $A \setminus B$ tandis que $A'_{i'}$ sont dans $B \setminus C$, pour qu'ils ne se croisent pas (pareil pour B_i and $B'_{i'}$). Les seules intersections qu'il reste à considérer sont $A_i \cap B'_{i'}$ and $B_i \cap A'_{i'}$. Supposons sans perte de généralité $A_i \cap B'_{i'} \neq \emptyset$. En raison des contraintes syntaxiques $\mathcal{A}$, nous considérons cette intersection comme un singleton $\{j\}$. Nous supprimons les pairess (A_i, B_i) et $(A'_{i'}, B'_{i'})$ de la premisse, et les rempla**c**cons par les pairess $(A_i \cup A'_{i'} \setminus \{j\}, B_i \cup B'_{i'} \setminus \{j\})$. Cette comparaison appartient à $\mathcal{A}$, et aussi à $\gtrsim$ car elle est additive (par addition des inégalités caractérisant $A_i \gtrsim B_i$ et $A'_{i'} \gtrsim B'_{i'}$ et annulation des termes ω_j apparaissant de part et d'autre). En itérant cette opération, on obtient un schéma couverture supportant (A,C) de taille non supérieure à $k+k'$.*

En corollaire, lorsque l'on restreint les atomes à la mention d'un seul pro contre un nombre quelconque de con (resp. un nombre quelconque de pro contre un con), le schéma transitif réduit n'est pas plus expressif que le schéma couverture. Ce n'est pas le cas lorsque nous permettons de mélanger ces atomes (comme illustré par Ex. 9).

Proposition 6 *Pour tout nombre entier positif k, lorsque $\mathcal{A} \in \{\Delta(1,1), \Delta(1,m), \Delta(m,1)\}$ et $\gtrsim$ est additive, $\mathcal{E}(rt, \mathcal{A}_{\gtrsim}, k) = \mathcal{E}(cov, \mathcal{A}_{\gtrsim}, k)$.*

On peut se demander si ces affirmations atomiques rendent le problème computationnelement plus simple à traiter. Bien que l'on sache que la réponse est positive pour $\mathcal{A} = \Delta(1,1)$ [1], nous montrons qu'à partir de $k \geq 2$ le problème est difficile.

Théorème 4 *Étant donné $\omega \in \mathbb{N}_0^m$ $A, B \in 2^{[m]}$ tel que $A \gtrsim B$ où $\gtrsim$ est une relation de préférence additive induite par ω, et $\mathcal{A} = \Delta(1,k)$. Lorsque $k \geq 2$, décider si $(A,B) \in \mathcal{E}(cov, \mathcal{A}_{\gtrsim}, +\infty)$ est NP-complete.*

Démonstration 6 *(sketch) Réduction de* 3D-MATCHING *[11].*

Lorsqu'on utilise le schéma couverture, la longueur des explications est limitée par le nombre m d'éléments. Par conséquent, il existe une gamme de valeurs de m pour lesquelles trouver une explication pourrait s'avérer trop difficile pour un humain, mais peut être facilement réalisé par une machine, soit avec un solveur ou même par force brute.

5 Complétude empirique du schéma couverture

Les résultats de la Section 4 établissent que les ensembles de comparaisons atomiques $\Delta(1,m)$ ou $\Delta(m,1)$ utilisant le schéma cov nous libèrent de la tâche de séquencer les

explications. Bien sûr, cela a un prix, car certaines pairess qui peuvent être expliquées autrement peuvent ne pas l'être avec ces comparaisons. Notre objectif est de donner un aper**c**cu de la "complétude empirique" des comparaisons atomiques en utilisant le schéma cov.

Étant donné une relation de préférence additive spécifique $\mathcal{R}$, l'ensemble $\mathcal{A}_{\mathcal{R}}$ de comparaisons par paires qui peut être utilisé pour expliquer la paires (A, B) tel que $(A, B) \in \mathcal{R}$ et $(A, B) \notin \mathcal{A}_{\mathcal{R}}$ est le suivant :

$$\mathcal{A}_{\mathcal{R}} = [\Delta(1, m) \cup \Delta(m, 1)] \cap \mathcal{R}$$

lus précisément, nous considérons les relations de préférence sur les alternatives qui sont représentables par *un ordre linéaire additif sur l'algèbre des sous-ensembles d'un ensemble fini* (voir [10]). Nous supposons l'ordre suivant sur les alternatives singletons : $\mathbf{a} \succ \mathbf{b} \succ \mathbf{c} \succ \mathbf{d} \succ \mathbf{e} \succ \mathbf{f} \succ \ldots$. Alors, nous notons par $T^m_{\succ} = \{(A, B) \in 2^{[m]} \times 2^{[m]} : A \succ B \text{ and } A \cap B = \emptyset\}$, et nous avons $\mathcal{A}_{\succ} \subseteq T^m_{\succ}$. Le nombre d'ordres linéaires additifs sur $2^{[m]}$ croît très rapidement [16, 10] : il est de 14 pour $m = 4$, mais pour $m = 7$ nous avons déjà plus de 200 millions d'ordres.

Techniquement, nous avons été en mesure de générer touts les ordres linéaires additives de $m \in [[4; 6]]$. De plus, pour décider si $(A, B) \in \mathcal{E}(cov, \mathcal{A}_{\succ}, \infty)$, nous avons utilisé un programme linéaire mixte en nombres entiers. Enfin, pour évaluer les proportions de pairess $T^m_{\succ} \backslash \mathcal{A}_{\succ}$ qui sont explicables, on calcule pour chaque *ordre linéaire additif* $\succ$ donné m, la valeur suivante

$$\mathfrak{M}_{m,\succ} = \frac{|\mathcal{E}(cov, \mathcal{A}_{\succ}, \infty) \cap T^m_{\succ} \backslash \mathcal{A}_{\succ}|}{|T^m_{\succ} \backslash \mathcal{A}_{\succ}|}$$

m	Minimum	Médian	Maximum	$\lvert T^m_{\succ} \backslash \mathcal{A}_{\succ} \rvert$
4	66.7%	66.7%	100%	3
5	72.0%	80.0%	100%	25
6	78.46%	84.62%	100%	130

Table 3 – $\mathfrak{M}_{m,\succ}$ for $m \in [[4; 6]] \quad \forall \succ$

Le Tableau 3 résume les valeurs minimale, médiane et maximale obtenues sur les ordres linéaires additifs respectivement de 14, 516 et 124187 (pour $m = 4, 5, 6$).

Nous remarquons que les valeurs minimale et médiane de $\mathfrak{M}_{m,\succ}$ augmentent avec m. En ce qui concerne les valeurs maximales, nous constatons qu'elles sont toutes égales à 100%, ce qui signifie que pour tous les $m \in [[4; 6]]$, il existe au moins un ordre linéaire additif $\succ$ pour lequel toutes les pairess sont explicables. En regardant plus globalement l'ensemble des valeurs du tableau 3, on peut dire qu'une majorité significative des pairess de $T^m_{\succ} \backslash \mathcal{A}_{\succ}$ sont explicables. Par exemple, pour $m = 6$, plus de 3 pairess sur 4 sont explicables quel que soit l'ordre linéaire additif considéré.

Bien sûr, l'explicabilité d'une paires arbitraire (A, B) dépend de ses caractéristiques par rapport au classement des critères qui les composent dans l'ordre sur les singletons. Par exemple, les pairess tels que (**ac**, **bd**) seront toujours explicables puisque $\mathbf{a} \succ \mathbf{b}$ et $\mathbf{c} \succ \mathbf{d}$. Cependant, il sera plus difficile de trancher pour des pairess comme (**ad**, **bc**) ou (**ae**, **bcd**) ou (**bde**, **acf**).

6 Travaux connexes

Récemment, [17] ont exploré les explications dans le contexte de la décision des classifieurs linéaires. Ils se concentrent sur les PI-explications (ou raisons suffisantes), c'est-à-dire les explications fournissant des raisons suffisantes pour expliquer une décision donnée, quelle que soit la valeur des autres critères [8, 7], une stratégie d'explication différente de la nôtre comme mentionné dans l'introduction. Notre vision des explications comme des preuves déductives cognitivement limitées rappelle les *systèmes de preuves limitées* proposés dans le contexte de la logique des descriptions [13, 9]. De même, une approche progressive similaire a été étudiée dans le contexte des problèmes de satisfaction de contraintes [3]. Enfin, les explications basées sur des axiomes ont été préconisées dans le domaine du choix social computationnel [5, 20]. En particulier, les travaux récents de [4] exploitent également les axiomes étudiés dans la théorie du vote pour produire des explications pour les décisions collectives, mais appliquées à un cadre différent (le vote), et en utilisant des techniques de preuve différentes (méthodes de tableau).

7 Conclusion

Nous proposons un cadre pour expliquer les comparaisons issues d'un modèle additif. Ce cadre s'accompagne de différents schémas pour raisonner sur les préférences, et nous nous concentrons sur un schéma spécifique : le schéma de couverture. De plus, à des fins cognitives, nous proposons de restreindre les explications aux ensembles d'éléments atomiques qui préfèrent un pour à un groupe de contre, ou un groupe de pour à un seul contre. Le moteur d'explication basé sur le recouvrement avec des ensembles restreints d'éléments atomiques n'est pas complet, mais des études empiriques montrent que des explications peuvent être calculées dans une grande proportion de cas.

De plus, le fait de fournir un schéma d'argumentation avec le résultat d'une comparaison par paires ouvre la possibilité de discuter ou de contester ce résultat. Ceci est rendu possible par ce qu'on appelle les questions critiques [21], un outil associé aux schémas d'argumentation représentant des attaques ou des critiques qui, si l'on n'y répond pas de manière adéquate, falsifient l'argument correspondant au schéma. Cela conduit naturellement à la perspective à long terme de la nature interactive du processus d'explication : ces schémas devraient être intégrés dans un processus dialectique, dans lequel l'utilisateur final devrait pouvoir

contester [19], tandis que d'autre part le système devrait acquérir des connaissances sur les préférences de l'utilisateur à travers un processus d'élicitation indirect [15]. L'imbrication harmonieuse de l'explication et de la recommandation appelle à la conception de systèmes d'initiative mixte [14] dans lesquels l'utilisateur peut être actif en défiant le système, et le système adaptatif dans ses réponses.

Références

[1] Belahcene, Khaled, Christophe Labreuche, Nicolas Maudet, Vincent Mousseau, and Wassila Ouerdane: *Explaining robust additive utility models by sequences of preference swaps*. Theory and Decision, 82(2) :151–183, 2017.

[2] Belahcene, Khaled, Christophe Labreuche, Nicolas Maudet, Vincent Mousseau, and Wassila Ouerdane: *Comparing options with argument schemes powered by cancellation*. In *Proc. 28th IJCAI*, pages 1537–1543, Macao, Macau SAR China, 2019.

[3] Bogaerts, Bart, Emilio Gamba, and Tias Guns: *A framework for step-wise explaining how to solve constraint satisfaction problems*. Artif. Intell., 300 :103–550, 2021.

[4] Boixel, Arthur, Ulle Endriss, and Ronald de Haan: *A Calculus for Computing Structured Justifications for Election Outcomes*. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI-2022)*, February 2022.

[5] Cailloux, Olivier and Ulle Endriss: *Arguing about Voting Rules*. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 287–295. ACM, 2016.

[6] Coste-Marquis, Sylvie and Pierre Marquis: *From Explanations to Intelligible Explanations*. In *1st International Workshop on Explainable Logic-Based Knowledge Representation (XLoKR'20)*, Rhodes, Greece, 2020. Workshop at KR'20.

[7] Darwiche, A. and P. Marquis: *On Quantifying Literals in Boolean Logic and its Applications to Explainable AI*. Journal of Artificial Intelligence Research, 72 :285–328, 2021.

[8] Darwiche, Adnan and Auguste Hirth: *On The Reasons Behind Decisions*. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI)*, 2020.

[9] Engström, Fredrik and Claes Strannegård Abdul Rahim Nizamani: *Generating Comprehensible Explanations in Description Logic*. In *Informal Proceedings of the 27th International Workshop on Description Logics*, Vienna, 2014.

[10] Fishburn, Peter C., Aleksandar Pekec, and James A. Reeds: *subset comparisons for additive linear orders*. Mathematics of Operations Research, 27 :227–243, 2002.

[11] Garey, Michael R. and David S. Johnson: *Computers and Intractability, a Guide to the Theory of NP-completeness*. Freeman, 1979.

[12] Hammond, J, Ralph Keeney, and H Raiffa: *Even Swaps : A Rational Method for Making Trade-offs*. Harvard business review, 76 :137–8, 143, March 1998.

[13] Horridge, Matthew, Samantha Bail, Bijan Parsia, and Uli Sattler: *Toward Cognitive Support for OWL Justifications*. Know.-Based Syst., 53 :66–79, nov 2013, ISSN 0950-7051.

[14] Horvitz, Eric: *Uncertainty, Action, and Interaction : In Pursuit of Mixed-Initiative Computing*. Intelligent Systems, pages 17–20, 2000.

[15] Labreuche, Christophe, Nicolas Maudet, Wassila Ouerdane, and Simon Parsons: *A Dialogue Game for Recommendation with Adaptive Preference Models*. In *Proceedings of the 14th International Conference on Autonomous Agent and MultiAgent systems (AAMAS).*, pages 959–967, 2015.

[16] Maclagan, D.: *Boolean Term Orders and the Root System B_n*. Order, 15 :279–295, 1998.

[17] Marques-Silva, Joao, Thomas Gerspacher, Martin Cooper, Alexey Ignatiev, and Nina Narodytska: *Explaining Naive Bayes and Other Linear Classifiers with Polynomial Time and Delay*. In Larochelle, H., M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (editors) : *Advances in Neural Information Processing Systems*, volume 33, pages 20590–20600. Curran Associates, Inc., 2020.

[18] Miller, Tim: *Explanation in artificial intelligence : Insights from the social sciences*. Artif. Intell., 267 :1–38, 2019.

[19] Mulligan, Deirdre K., Daniel Kluttz, and Nitin Kohli: *Shaping Our Tools : Contestability as a Means to Promote Responsible Algorithmic Decision Making in the Professions*. In Werbach, Kevin (editor) : *After the Digital Tornado*. Cambridge University Press, 2020.

[20] Procaccia, Ariel D.: *Axioms Should Explain Solutions*. The Future of Economic Design, 2019.

[21] Walton, Douglas: *Argumentation schemes for Presumptive Reasoning*. Mahwah, N. J.,Erlbaum, 1996.

[22] Willcox, William B. (editor): *The Papers of Benjamin Franklin*, pages 299–300. New Haven and London, Yale University Press, 1975.

Actes JIAF 2022

Fragment explicable par schémas d'arguments des affectations nécessaires dans un tri non compensatoire

Khaled Belahcene **Jérome Gaigne** **Sylvain Lagrue**

Heudiasyc, Université de Technologie de Compiègne et CNRS, France

{khaled.belahcene; jerome.gaigne; sylvain.lagrue}@hds.utc.fr

Résumé

Nous étudions un schéma d'arguments visant à expliquer une décision nécessaire étant donnée une jurisprudence dans le cadre du tri non compensatoire. Dans ce modèle un ensemble de candidats est classé dans deux catégories ordonnées par un ensemble de jurés disposant d'un avis personnel sur l'ensemble des candidats. Un candidat est rangé dans la catégorie supérieure s'il est accepté par un ensemble suffisant des jurés. Nous montrons que le schéma étudié est une approximation de ce modèle et partageons des résultats contre-intuitifs concernant les performances relatives de solveurs bien connus.

Abstract

We focus on an argument scheme explaining a necessary decision with respect to a jurisprudence considering the non compensatory sorting model. In this model, a set of candidate is sorted in two ordered categories by a set of point of view expressing an individual preference over the set of candidates. Candidates are ranked in the upper category if they are accepted by a sufficient subset of points of view. We show that the argument scheme is an approximation of this model and share some counterintuitive results concerning the relative performance of some well known solvers.

1 Introduction

Le problème du tri par approbation justifiable a été introduit dans [3]. Il correspond à la situation dans laquelle : un jury, composé de plusieurs membres, doit classer des candidats dans deux catégories ordonnées. Les auteurs nomment ces catégories BON et MAUVAIS, qui pourraient être ACCEPTÉS et REFUSÉS dans le cas d'un recrutement par exemple. Chacun des jurés dispose d'un avis sur l'ensemble des candidats sous la forme d'une préférence et en accepte un sous-ensemble. Enfin, ces ensembles d'approbation sont agrégés suivant des coalitions suffisantes de jurés. Un candidat est donc rangé dans la catégorie supérieure s'il a été accepté par un sous-ensemble suffisant des jurés.

Dans le cadre du tri par approbation, les auteurs ont identifié deux situations d'intérêt pour l'explicabilité :

— justifier que l'ensemble des décision prises est conforme à la procédure précédente ;
— justifier son classement à un candidat.

Nous nous intéressons ici à la seconde situation dans laquelle un candidat souhaite recevoir une justification de son classement. Il existe dans ce cas deux possibilités :

— le candidat aurait pu être classé autrement.
— le candidat n'aurait pas pu être classé autrement.

Dans le second cas, pour justifier la décision du jury au candidat, il s'agit de montrer que la décision prise était nécessaire, donc qu'une autre décision concernant son classement n'aurait pas respecté la procédure de tri par approbation. Afin de fournir une explication du caractère nécessaire de la décision qui soit humainement intelligible, les auteurs ont proposé un schéma d'argumentation. Celui-ci a été montré suffisant pour prouver la nécessité d'une décision, cependant, à notre connaissance, il n'a pas été prouvé que ce schéma est aussi nécessaire pour prouver la nécessité d'une décision. Afin de limiter la divulgation d'informations privées concernant les décisions des jurés, une *jurisprudence* peut servir de cas de référence pour former le schéma d'argumentation.

Dans la section 2, nous introduisons ou rappelons les définitions concernant le tri non compensatoire sous-jacent au processus du tri par approbation et son problème inversé. Ensuite la section 3 reprend le schéma d'argumentation justifiant du caractère nécessaire d'une décision. Dans cette section, nous avons étudié le caractère nécessaire de ce schéma et montrons que celui-ci couvre un fragment explicable de la décision nécessaire. Lors de cette étude, certains résultats contre-intuitifs concernant les performances relatives de solveurs sur l'exhibition du schéma d'argumentation sont apparus. Nous les abordons dans la section 4. Enfin, la section 5 revient sur nos résultats et propose de nouvelles pistes d'études.

2 Tri non compensatoire et problème NCS inversé

Dans cette section nous définissons le tri non compensatoire ainsi que certaines de ses propriétés. Nous rappelons son problème inverse ainsi qu'un théorème de représentation important mis en avant par les auteurs de [3].

2.1 Tri non compensatoire

Le tri par approbation est aussi appelé *tri non compensatoire* [4] (aussi appelé *NCS* pour *non compensatory sorting*).

Afin de simplifier la définition du *tri non compensatoire*, nous rappelons la définition d'*upset* (« section finissante »en français).

Définition 1 *Soit A un ensemble et R une relation binaire sur A. Le sous-ensemble $B \subseteq A$ est un* upset *de (A, R) ssi $\forall a \in A, \forall b \in B$, si bRa, alors $a \in B$.*

Définition 2 *Tri non compensatoire*
Soient un ensemble d'alternatives $\mathbb{X}$ et un ensemble de points de vue $\mathcal{N}$. Soient deux catégories ordonnées ACCEPTÉ *et* REFUSÉ. *Chaque point de vue $i \in \mathcal{N}$ dispose d'une préférence sur l'ensemble des alternatives sous la forme d'un préordre complet noté $\succsim_i$. Dans un soucis de cohérence sémantique et syntaxique, nous utiliserons parfois le préodre $\precsim_i$ défini tel que $\forall a, b \in \mathbb{X}, a \precsim_i b \quad ssi \quad b \succsim_i a$. Chaque point de vue $i \in \mathcal{N}$ dispose aussi d'un ensemble d'approbation $\mathcal{A}_i$ tel que $\mathcal{A}_i$ est un* upset *de $(\mathbb{X}, \precsim_i)$. Enfin, soit S l'ensemble des coalitions suffisantes tel que S est un* upset *de $\left(2^{\mathcal{N}}, \subseteq\right)$. Un processus de tri est un tri non compensatoire si et seulement si chaque alternative acceptée est approuvée par une coalition suffisante de points de vue,* i.e. *soit $\alpha : \mathbb{X} \mapsto \left\{ACCEPTÉ, REFUSÉ\right\}$ la fonction caractérisant les décisions du processus, α correspond à un tri non compensatoire si et seulement si*

$$\forall x \in \mathbb{X}, \alpha\left(x\right) = \begin{cases} ACCEPTÉ, \text{ si } \left\{i \in \mathcal{N} : x \in \mathcal{A}_i\right\} \in S \\ REFUSÉ, \text{ sinon} \end{cases}$$

De cette définition peuvent être tirées deux propriétés sur la rationalité du processus de tri. Le processus de tri est rationnel sur le plan individuel, car il assure qu'il n'existe pas d'alternative non approuvée par un point de vue $i \in \mathcal{N}$ qui soit préférée à une alternative acceptée par i, *i.e.* $\mathcal{A}_i$ est un *upset* de $(\mathbb{X}, \precsim_i)$. De plus, le processus de tri est rationnel sur le plan collectif, car il assure que si un sous-ensemble des points de vue est suffisant pour classer une alternative dans la catégorie supérieure, alors tout sur-ensemble de celui-ci est aussi suffisant, *i.e.* S est un *upset* de $\left(2^{\mathcal{N}}, \subseteq\right)$.

Exemple 1 *Soient $\mathbb{X} = \{a, b, c, d, e\}$ un ensemble d'alternatives, $\mathcal{N} = \{1, 2, 3\}$ un ensemble de points de vue. Les préférences individuelles des points de vue sont les suivantes :*

$$e \succ_1 a \succ_1 c \succ_1 b \succ_1 d$$
$$b \succ_2 a \succ_2 c \succ_2 d \succ_2 e$$
$$c \succ_3 b \succ_3 a \succ_3 d \succ_3 e$$

Les points de vue approuvent respectivement leurs deux alternatives préférées :

$$\mathcal{A}_1 = \{a, e\}$$
$$\mathcal{A}_2 = \{a, b\}$$
$$\mathcal{A}_3 = \{b, c\}$$

Enfin, soient les coalitions suffisantes suivantes :

$$S = \{\{1, 2\}, \{2, 3\}, \{1, 3\}, \{1, 2, 3\}\}$$

Suite à la procédure de tri non compensatoire, la catégorisation obtenue est la suivante $\alpha = \left\{\left(a, ACCEPTÉ\right), \left(b, ACCEPTÉ\right), \left(c, REFUSÉ\right), \left(d, REFUSÉ\right), \left(e, REFUSÉ\right)\right\}$.

2.2 Problème NCS inversé

Dans le cas où la procédure est remise en question, il faut pouvoir à partir de la sortie α et de certaines données, à savoir les préférences individuelles de chaque point de vue, vérifier qu'il est possible de trouver un ensemble d'approbation pour chaque point de vue ainsi qu'un ensemble de coalitions suffisantes permettant d'obtenir α en respectant le processus de tri non compensatoire. Dans le problème NCS inversé (aussi appelé *inv-NCS*), seule l'existence ou l'inexistence de ces paramètres est importante (*i.e.* il n'est pas nécessaire de trouver ces paramètres).

Les auteurs de [3] mettent en avant dans le théorème 1, un théorème de représentation important permettant de répondre au problème inv-NCS en supprimant la notion de coalitions suffisantes ce qui permet notamment de ne pas divulguer l'information sur ces coalitions. Ce théorème est le suivant :

Théorème 1 *Formulation par paires du modèle de tri non compensatoire*
Une affectation α d'alternatives à des catégories peut être représentée dans le modèle de tri non compensatoire si et seulement si il existe $\langle \mathcal{A}_i \rangle_{i \in \mathcal{N}} \subseteq (2^{\mathbb{X}})^{\mathcal{N}}$ tel que :

1. *$\forall i \in \mathcal{N}, \mathcal{A}_i$ est un* upset *de $(\mathbb{X}, \precsim_i)$.*
2. *$\forall \langle g, b \rangle \in \alpha^{-1}\left(ACCEPTÉ\right) \times \alpha^{-1}\left(REFUSÉ\right)$, $\exists i \in \mathcal{N}$ tel que $g \in \mathcal{A}_i$ et $b \notin \mathcal{A}_i$.*

Ce théorème permet de réduire le problème inv-NCS vers SAT ([3], section 3.4). Cette formulation a été étendue à plusieurs catégories, et relâchée en une formulation Max-SAT pour pouvoir tenir compte de présence de bruit dans les données, dans [7].

3 Étude du schéma d'argumentation

Dans [3] les auteurs exhibent un schéma d'argumentation permettant de donner une explication humainement compréhensible de la nécessité qu'une certaine alternative soit classée dans une des catégories. Dans cette section nous rappelons ce schéma puis nous répondons à la question restée ouverte sur le caractère nécessaire de l'existence d'un tel schéma lorsqu'une décision est impossible.

3.1 Rappel du schéma d'argumentation expliquant une décision nécessaire étant donnée une jurisprudence

Tout d'abord, revenons sur la notion de décision nécessaire. Nous supposons que nous disposons d'un cas de référence (une jurisprudence) $\alpha^\star : \mathbb{X}^\star \mapsto \{\text{ACCEPTÉ}, \text{REFUSÉ}\}$ avec $\mathbb{X}^\star \subseteq \mathbb{X}$. Étant donné que le problème inv-NCS nous permet de savoir si une décision est possible, pour montrer qu'une décision est nécessaire, il suffit de dire que la décision est possible (*i.e.* constitue une instance positive au problème inv-NCS) et que la décision opposée ne l'est pas (*i.e.* constitue une instance négative au problème inv-NCS).

Définition 3 *Décision nécessaire étant donnée une jurisprudence*
Soit $\alpha^\star$ une instance positive au problème inv-NCS, on dit qu'une alternative $x \in \mathbb{X}$ est nécessairement assignée à la catégorie $C \in \{ACCEPTÉ, REFUSÉ\}$ étant donnée $\alpha^\star$ si $\alpha^\star \cup \{(x, \bar{C})\}$ est une instance négative au problème inv-NCS, avec $\bar{C}$ est la catégorie opposée à C.

Le schéma d'argumentation est quant à lui une condition suffisante pour montrer qu'une instance est négative au problème inv-NCS. La question étant de savoir s'il constitue aussi une condition nécessaire lorsque nous avons une instance négative à ce problème.

Définition 4 *Schéma d'argumentation de la décision nécessaire étant donnée une jurisprudence*
On dit qu'un ensemble $\{\langle g_1, b_1\rangle, ..., \langle g_k, b_k\rangle\}$ instancie le schéma d'argumentation refusant l'assignation α par rapport à l'assignation $\alpha^\star$, avec $\forall j = 1, ..., \left|\alpha^{-1}(ACCEPTÉ)\right|, \alpha(g_j) = ACCEPTÉ$ et $\forall j = 1, ..., \left|\alpha^{-1}(REFUSÉ)\right|, \alpha(b_j) = REFUSÉ$ et $\forall x \in \mathbb{X}^\star, \alpha(x) = \alpha^\star(x)$, si :

$$\exists B \subseteq \mathcal{N}, |B| \leq \min(|\mathcal{N}|, k-1),$$

$$\forall i \notin B, \forall u \in \{1, ..., k\}, g_u \gtrsim_i b_u \text{ est fausse} \quad (1)$$

et

$$\forall j \in B, \forall p, q \in \{1, ..., k\}, \text{si } p \neq q \text{ alors}$$
$$\exists \langle g, b\rangle \in \{g_p, g_q\} \times \{b_p, b_q\}, g \gtrsim_j b \text{ est fausse} \quad (2)$$

Cette définition se base sur l'existence d'un sous-ensemble B des points de vue de taille plus petite que le nombre de paires d'alternatives que nous voulons séparer (*cf.* théorème 1 sur la formulation par paires du problème inv-NCS). La conditions 1, nous dit alors qu'aucun point de vue hors de B n'est capable de séparer ne serait-ce qu'une des paires qui nous intéressent. La conditions 2, nous dit quant à elle, que chaque point de vue de B n'est capable de séparer qu'une seule des paires d'intérêts à la fois.

La preuve que ce schéma constitue bel et bien une condition suffisante pour avoir une instance négative au problème inv-NCS est assez directe puisque nous avons k paires à séparer et seulement au mieux $k-1$ points de vue ne pouvant en séparer qu'une seule à la fois. De ce fait, quoi qu'il arrive, il y aura toujours au moins une paire qui sera inséparable confirmant que l'instance est donc négative au problème inv-NCS.

L'avantage de ce schéma est qu'il permet de ne pas divulguer les ensembles approuvés par chaque point de vue tout en donnant une justification intelligible.

Exemple 2 *Soient $\mathbb{X} = \{a, b, c, d, e\}$, $\mathcal{N} = \{1, 2\}$ et $\alpha = \{(a, ACCEPTÉ), (b, REFUSÉ), (c, REFUSÉ), (d, REFUSÉ), (e, REFUSÉ)\}$.*

$$a \succ_1 c \succ_1 d \succ_1 b \succ_1 e$$
$$e \succ_2 b \succ_2 d \succ_2 a \succ_2 c$$

Cette configuration est une instance positive au problème inv-NCS, car en prenant $\mathcal{A}_1 = \{a\}$ et $S = \{\{1\}, \{1, 2\}\}$, il est possible d'obtenir α.

Maintenant supposons que l'on souhaite obtenir une justification de la classification de b comme REFUSÉ. Si l'on prend l'assignation $\alpha' = \alpha \setminus \{(b, REFUSÉ)\} \cup \{(b, ACCEPTÉ)\}$, on obtient cette fois que l'instance est négative au problème inv-NCS. En recherchant des schémas d'argumentation, on peut par exemple trouver $B = \{1\}$ avec l'ensemble de paires $\{\langle a, d\rangle, \langle b, e\rangle\}$. Autrement dit, selon le point de vue 1, il n'est possible de séparer qu'une seule de ses deux paires et selon le point de vue 2, il est impossible de séparer une de ses paires, car l'alternative REFUSÉE est préférée à sa paire ACCEPTÉE.

Ce schéma n'est pas unique, nous pouvons par exemple prendre $B = \{1, 2\}$ avec l'ensemble de paires $\{\langle a, d\rangle, \langle b, d\rangle, \langle b, e\rangle\}$.

3.2 Caractère nécessaire du schéma d'argumentation

Afin de voir si l'existence d'un schéma est une condition nécessaire au fait que l'instance soit négative au problème inv-NCS, nous avons décidé d'automatiser la recherche d'un contre-exemple. Pour cette recherche, il faut pour chaque instance vérifier que

1. L'instance est négative au problème inv-NCS

2. et qu'il n'existe pas de schéma pour cette instance.

Pour la première propriété, nous utilisons la formulation SAT du problème inv-NCS proposée dans [3]. Pour la seconde, nous avons représenté le schéma en SAT de la façon suivante.

3.2.1 Modélisation SAT du schéma d'argumentation

Dans le but de simplifier l'expérimentation, nous avons considéré des profils de préférences sous la forme d'ordres totaux stricts. Dans ce cas, le schéma se réécrit comme suit :

Définition 5 *Schéma d'argumentation de la décision nécessaire étant donnée une jurisprudence lorsque les préférences sont des ordres totaux stricts*
On dit qu'un ensemble $\{\langle g_1, b_1\rangle, ..., \langle g_k, b_k\rangle\}$ *instancie le schéma d'argumentation refusant l'assignation* α *par rapport à la l'assignation* $\alpha^\star$*, avec* $\forall j = 1, ..., \left|\alpha^{-1}\left(\textit{ACCEPTÉ}\right)\right|, \alpha\left(g_j\right) = \textit{ACCEPTÉ}$ *et* $\forall j = 1, ..., \left|\alpha^{-1}\left(\textit{REFUSÉ}\right)\right|, \alpha\left(b_j\right) = \textit{REFUSÉ}$ *et* $\forall x \in \mathbb{X}^\star, \alpha(x) = \alpha^\star(x)$*, si :*

$$\exists B \subseteq \mathcal{N}, |B| \leq \min\left(|\mathcal{N}|, k-1\right),$$

$$\forall i \notin B, \forall u \in \{1, ..., k\},\ b_u \succ_i g_u \tag{3}$$

et

$$\forall j \in B, \forall p, q \in \{1, ..., k\}, \textit{si } p \neq q \textit{ alors}$$

$$\exists \langle g, b\rangle \in \left\{g_p, g_q\right\} \times \left\{b_p, b_q\right\}, b \succ_j g \tag{4}$$

Nous noterons dans cette sous-section $\left|\alpha^{-1}(\text{ACCEPTÉ})\right| = m$, $\left|\alpha^{-1}(\text{REFUSÉ})\right| = n$ et $\mathbb{A} = \alpha^{-1}(\text{ACCEPTÉ}) \times \alpha^{-1}(\text{REFUSÉ})$.

On introduit deux familles de variables propositionnelles.

$$\begin{aligned} \beta_i &= i \in B \\ \lambda_{gb} &= \langle g, b\rangle \in \{\langle g_1, b_1\rangle, ..., \langle g_k, b_k\rangle\} \end{aligned}$$

Afin de représenter des contraintes de cardinalité en SAT, nous utilisons l'encodage de Bailleux et Boufkhad [2] qui nous permettra notamment de comparer les cardinalités de deux ensembles. Nous pouvons alors comparer directement les deux cardinalités au centre du schéma :

$$|\{i \in \mathcal{N} : \beta_i \text{ est vrai }\}| < \left|\left\{\langle g, b\rangle \in \mathbb{A} : \lambda_{gb} \text{ est vrai }\right\}\right|$$

Pour ce faire, dans l'encodage proposé par Bailleux et Boufkhad pour les contraintes de cardinalité de ces deux ensemble, nous avons gardé le *totalizer* (qui permet d'obtenir la cardinalité de l'ensemble) mais nous avons modifié le *comparator* (qui, dans la version proposée par les auteurs, permet de comparer cette cardinalité a une borne inférieure et une borne supérieure fixées préalablement). L'idée derrière cette modification est de réaliser un *comparator* entre les cardinalités des deux ensembles. Pour faire cela, nous ajoutons la règle suivante sur les variables de sorties des deux *totalizers* (ces variables sont une représentation unaire de la cardinalité de chaque ensemble) :

$$\exists i \in \{1, ..., n\}, S_i^\lambda \wedge \neg S_i^\beta$$

Cette règle nous dit qu'il existe au moins une variable de plus dans la représentation unaire de la cardinalité de l'ensemble $\left\{(g, b) \in \mathbb{A} : \lambda_{gb} \text{ est vrai }\right\}$ de vraie que dans la représentation unaire de la cardinalité de l'ensemble $\{i \in \mathcal{N} : \beta_i \text{ est vrai }\}$. Autrement dit, la cardinalité de $\left\{\langle g, b\rangle \in \mathbb{A} : \lambda_{gb} \text{ est vrai }\right\}$ doit valoir au moins $|\{i \in \mathcal{N} : \beta_i \text{ est vrai }\}| + 1$.

Notons $n_\lambda = \left|\alpha^{-1}(\text{ACCEPTÉ})\right| \times \left|\alpha^{-1}(\text{REFUSÉ})\right|$ le nombre maximum d'éléments dans l'ensemble $\left\{\langle g, b\rangle \in \mathbb{A} : \lambda_{gb} \text{ est vrai }\right\}$, et $n_\beta = |\mathcal{N}|$ le nombre maximum d'éléments dans l'ensemble $\{i \in \mathcal{N} : \beta_i \text{ est vrai }\}$.

Pour réaliser cela, nous proposons de créer $l = \max\left(n_\beta, n_\lambda\right)$ nouvelles variables W_i, $i \in \{1, ..., l\}$ qui représentent l'information précédente de la manière suivante :

$$S_i^\lambda \wedge \neg S_i^\beta \leftrightarrow W_i$$

$$\bigvee_{i=1...l} W_i$$

La représentation unaire comportant le moins de variables est alors complétée de nouvelles variables pour obtenir un nombre de variables identique à la représentation en comportant le plus, *i.e.* l. Toutes ces nouvelles variables sont alors forcées comme fausses, c'est-à-dire pour $S^\star$ avec $\star = \text{argmin}_{x \in \{\lambda, \beta\}}\left(n_x\right)$:

$$\text{Aug :} \quad \bigwedge_{i=n_\star+1...l} \neg S_i^\star$$

La modélisation des cardinalités est donc la suivante :
— *Totalizers* : Totalizer $\left(\lambda_{gb}\right) \wedge$ Totalizer $\left(\beta_i\right)$
— Comparaison des cardinaux :

$$\begin{aligned} \text{Comp}_1 &: \bigwedge_{i=1...l} \neg S_i^\lambda \vee S_i^\beta \vee W_i \\ \text{Comp}_2 &: \bigwedge_{i=1...l} S_i^\lambda \vee \neg W_i \\ \text{Comp}_3 &: \bigwedge_{i=1...l} \neg S_i^\beta \vee \neg W_i \\ \text{Comp}_4 &: \bigvee_{i=1...l} W_i \end{aligned}$$

Enfin les autres contraintes du schéma d'argumentation se représentent assez facilement en SAT de la façon suivante :

$$\text{C}_1 : \bigwedge_{\substack{\forall i \in \mathcal{N}, \\ \forall \langle g, b\rangle \in \mathbb{A}, \\ g \succ_i b}} \beta_i \vee \neg \lambda_{gb}$$

$$\text{C}_2 : \bigwedge_{\substack{\forall i \in \mathcal{N}, \\ \forall p \neq q = 1, ..., mn, \\ \forall \langle g, b\rangle \in \left\{b_p, b_q\right\} \times \left\{g_p, g_q\right\}, \\ g \succ_i b}} \neg \beta_i \vee \neg \lambda_{g_p b_p} \vee \neg \lambda_{g_q b_q}$$

L'utilisation de cet encodage permet de comparer les cardinalités des deux ensembles entre eux et ainsi de n'appeler le solveur qu'une seule fois pour les diverses tailles de schéma possible.

La formule finale est donc :

$$\phi : \quad \text{Aug} \wedge \text{Totalizers} \wedge \bigwedge_{i=1}^{4} \text{Comp}_i \wedge \text{C}_1 \wedge \text{C}_2$$

3.2.2 Recherche de contre-exemple

En utilisant cette modélisation ainsi que la modélisation SAT du problème inv-NCS, nous sommes arrivés à la conclusion suivante : il existe des instances du problème pour lesquelles le problème est une instance négative à inv-NCS mais ne possède pas de schéma d'argumentation. Pour faire cela, nous avons parcouru en exhaustivité l'ensemble des instances pour différentes valeurs des paramètres : nombre d'alternatives, nombre de points de vue et nombre d'alternatives classées comme ACCEPTÉ (et donc REFUSÉ).

Afin de limiter l'espace de recherche et éviter un certain nombre de symétries, voici la procédure que nous avons suivie pour générer l'ensemble des instances :

- Choisir les paramètres suivants : nombre d'alternatives ($n_{\text{alternatives}}$), nombre de points de vue (n_{POVs}), nombre d'alternatives classées comme ACCEPTÉ ($n_{\text{ACCEPTÉ}}$ et donc $n_{\text{REFUSÉ}} = n_{\text{alternatives}} - n_{\text{ACCEPTÉ}}$).
- Générer un profil de préférence :
  - Pour le premier point de vue, en calculant une des permutations avec $n_{\text{ACCEPTÉ}}$, $n_{\text{REFUSÉ}}$ répétitions sur les alternatives. C'est ce point de vue qui va fixer le nom de chaque alternative *i.e.* $g_1, ..., g_{n_{\text{ACCEPTÉ}}}$ dans l'ordre de la plus préférée à la moins préférée et $b_1, ..., b_{n_{\text{REFUSÉ}}}$ de la même façon.
  - Pour le reste des points de vue, en calculant une des permutations sans répétition sur les alternatives.
- L'assignation α est générée très facilement en prenant $\alpha^{-1}(\text{ACCEPTÉ}) = \{g_1, ..., g_{n_{\text{ACCEPTÉ}}}\}$ et $\alpha^{-1}(\text{REFUSÉ}) = \{b_1, ..., b_{n_{\text{REFUSÉ}}}\}$.
- Pour chaque point de vue, calculer un hash.
- Ranger ces hashs du plus petit au plus grand et stocker ce résultat dans une table de rappel.
- Si ce n-uplet de hashs est déjà dans la table alors reprendre à la génération d'un nouveau profil.
- Sinon, résoudre le problème inv-NCS via la formulation SAT disponible sous-section 3.4 de [3].
- Si le problème inv-NCS est insatisfiable, alors interroger l'existence d'un schéma d'argumentation via la formulation SAT détaillée précédemment.
- Si le schéma d'argumentation est aussi insatisfiable, alors enregistrer l'instance actuelle comme un contre-exemple.

Comme dans les deux problèmes étudiés, inv-NCS et le schéma d'argumentation, nous ne nous intéressons qu'aux paires où une alternative est classée comme ACCEPTÉ et une comme REFUSÉ et à la position de ces alternatives dans les préférences, pour éviter les symétries du type « permuter g_i et g_j », nous générons le premier point de vue en utilisant des permutations avec répétition (qui dans notre cas retourne l'ensemble des bipartitions de taille $n_{\text{ACCEPTÉ}}$ et $n_{\text{REFUSÉ}}$) et en fixant les noms des alternatives depuis ce point de vue.

Pour éviter dans la suite d'avoir des symétries de type « permuter le point de vue i et le point de vue j », nous stockons un n-uplet trié de hashs (un hash par point de vue) dans une table et vérifions via cette table que notre profil de préférence n'a pas déjà été calculé.

Malgré ces deux astuces, il reste encore des symétries, mais cela a tout de même permis la recherche exhaustive pour les valeur des paramètres disponibles tables 1 et 2.

# alternatives	# points de vue	# contre-exemples
3	2	0
4	2	0
4	3	0
5	2	0
5	3	0
5	4	0
6	2	0
6	3	898

Table 1 – Nombre de contre-exemples en fonction du nombre d'alternatives et du nombre de points de vue

Le nombre de contre-exemples trouvés pour 6 alternatives et 3 points de vue est très faible par rapport au nombre de configurations parcourues. Sans table de rappel, nous avons $\sum_{k=1}^{5} \left(C_6^k\right) \times 6! \times 6! = 32\,140\,800$ configurations qui descendent à seulement 15 785 210 avec la table. Dans ces configurations, 8 387 380 sont des instance négatives au problème inv-NCS. Cela nous amène à une fréquence des contre-exemples égale à $\frac{898}{15\,785\,210} \simeq 5.69 \times 10^{-5}$ sur l'ensemble des configurations (positives ou négatives au problème inv-NCS) et une proportion des instances négatives non couvertes par le schéma de $\frac{898}{8\,387\,380} \simeq 1.07 \times 10^{-4}$

# alternatives ACCEPTÉ	# contre-exemples
1	0
2	12
3	874
4	12
5	0

Table 2 – Nombre de contre-exemples en fonction du nombre d'alternatives classées comme ACCEPTÉ lorsqu'il y a 6 alternatives et 3 points de vue

Une des symétries restantes est l'inversion des rôles entre les alternatives classées comme ACCEPTÉ et comme REFUSÉ ce qui explique l'aspect symétrique de la table 2.

Bien qu'il y ait des contre-exemples ceux-ci sont peu nombreux par rapport au nombre total d'instances parcourues mais aussi par rapport au nombre total d'instances négatives parcourues.

Voici un exemple de contre-exemple avec 6 alternatives et 3 points de vue :

Exemple 3 *Cet exemple reprend l'exemple 2 en ajoutant un point de vue et une alternative. Prenons* $\mathbb{X} = \{a, b, c, d, e, x\}$, $\mathcal{N} = \{1, 2, 3\}$ *et* $\alpha = \{(a, ACCEPTÉ), (b, ACCEPTÉ), (x, ACCEPTÉ), (c, REFUSÉ), (d, REFUSÉ), (e, REFUSÉ)\}$

$$x \succ_1 c \succ_1 a \succ_1 b \succ_1 d \succ_1 e$$
$$a \succ_2 c \succ_2 x \succ_2 d \succ_2 b \succ_2 e$$
$$e \succ_3 b \succ_3 d \succ_3 a \succ_3 c \succ_3 x$$

Cette configuration ne satisfait pas le problème inv-NCS et ne contient pas de schéma d'argumentation. Toutefois, dans le cas où nous considérons cette fois-ci la catégorisation $\alpha' = \{(a, ACCEPTÉ), (b, ACCEPTÉ), (x, REFUSÉ), (c, REFUSÉ), (d, REFUSÉ), (e, REFUSÉ)\}$ *où* x *est maintenant dans la catégorie REFUSÉ, alors le problème est une instance positive au problème inv-NCS car nous pouvons prendre :*

$$\mathcal{A}_1 = \{a, b, c, x\}$$
$$\mathcal{A}_2 = \{a\}$$
$$\mathcal{A}_3 = \{b, e\}$$

et

$$S = \{\{2\}, \{1, 3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$$

afin d'obtenir cette catégorisation α'*. Nous avons alors une décision nécessaire sur la catégorie de* x *(*i.e. *la décision,* $\alpha(x) = ACCEPTÉ$*, n'est pas possible alors que sa décision opposée,* $\alpha(x) = REFUSÉ$*, l'est) qui n'est pas couverte par le schéma d'argumentation.*

3.2.3 Fragments explicables

La présence de contre-exemples témoigne de l'incomplétude du schéma à rendre compte de la nécessité d'une décision. Cependant, celui-ci ne perd pas son intérêt qui est de fournir une explication intelligible humainement à un problème complexe. Se pose alors la question de savoir à quel point cette approximation est proche du modèle original.

Nous pouvons encore élargir cette question, car actuellement nous considérons des schémas de toute taille, néanmoins son objectif est de fournir une explication compréhensible de la nécessité de la décision c'est pourquoi nous pouvons nous intéresser à la question des schémas dont la taille maximale est fixée. Ce nouveau modèle permet d'expliquer des fragments de plus en plus grands de la décision nécessaire à mesure que l'on augmente cette limite. En effet, si nous considérons uniquement les schémas de taille 1, nous obtenons une caractérisation de la dominance au sens de Pareto. Si nous augmentons la taille du maximale schéma du schéma, nous conservons les fragments expliqués par les modèles de taille plus petite tout en repoussant les limites de ces fragments. Cette approche est intéressante à plusieurs titres :

— Dans un premier temps, lorsque l'on fixe la taille du schéma, celui-ci devient calculable en temps polynomial dans l'espace des paires. Se pose alors la question de l'efficacité d'un algorithme polynomial face à la modélisation et l'utilisation d'un solveur tel que présenté précédemment.
— L'objectif de fournir une explication compréhensible peut être contrecarré par la taille du schéma. La simplification qu'apporte le schéma par rapport à la complexité du modèle original perd de l'intérêt à mesure que la taille du schéma augmente. Dans un sens plus celle-ci augmente plus le fragment explicable devient grand mais moins il est facile de comprendre le schéma. On peut donc envisager de limiter l'espace de recherche pour garantir l'intelligibilité des explications.

Cette imbrication des modèles ouvre beaucoup de pistes de recherche notamment en ce qui concerne les portions que chacun de ces modèles peut expliquer (schéma de taille au plus $k, \forall k \in \{1..|n_{\text{paires}}|\}$, avec n_{paires} le nombre de paires).

Sachant que le schéma complet ne permet d'expliquer qu'un fragment incomplet des décisions nécessaires, il peut aussi être intéressant de se demander s'il existe des structures explicables couvrant cet espace manquant. Nous pouvons notamment voir la construction du contre-exemple disponible exemple 3 reprenant la configuration de l'exemple 2 en ajoutant simplement une alternative et un point de vue.

4 Comportement du schéma d'argumentation

Durant la recherche de contre-exemples, nous avons comparé l'efficacité de différentes formulations du schéma d'argumentation ainsi que celle de différents solveurs. Ces expérimentations ont été réalisées sur un ordinateur ASUS G551J disposant d'un processeur Intel® Core™ i7-4750HQ CPU @ 2.00GHz × 8 et 8 Gio de mémoire DDR3 cadencé à 1600 MHz. Tous les temps d'exécution utilisés dans la suite ont été obtenus en utilisant le temps utilisateur de la commande `/usr/bin/time` sur un système d'exploitation Ubuntu 20.04.4 LTS.

Comme le schéma d'argumentation possède des contraintes de cardinalités, nous avons essayé de représenter le problème avec une formulation pseudo-booléenne. Pour résoudre à la fois les formulations SAT et pseudo-booléennes, nous avons utilisé le solveur Gophersat dans

sa version 1.2 [5] qui fait les deux et qui nous a permis via son API de réaliser des tests rapidement. Nous avons comparé ses performances avec le solveur Glucose dans sa version 4.0 [1] qui est basé sur le solveur Minisat [6] dans la résolution de problèmes formulés en SAT.

Après avoir présenté le protocole utilisé pour réaliser les comparaisons entre formalismes et entre solveurs, nous comparons les performances de Gophersat pour résoudre des problèmes SAT et pseudo-booléens équivalents. Puis nous concluons cette section en comparant les performances de Glucose et de Gophersat sur le problème de décision (*i.e* étant donné une configuration, existe-t-il un schéma d'arguments ?).

Sur cette dernière étude, nous nous attendions à avoir de meilleurs performances en utilisant Glucose, pourtant l'expérience nous montre que sur ce problème, Gophersat semble être plus efficace dans la grande majorité des cas.

4.1 Protocole

Cette sous-section présente le protocole ayant pour but d'étudier l'influence des différents paramètres sur le temps de résolution. Nous nous plaçons toujours dans le cas où les préférences des points de vue sont des ordres totaux stricts.

Les différents paramètres du schéma pouvant influer sur les performances sont les suivants :

— Le nombre d'alternatives : n
— Le nombre de points de vue : m
— Le taux d'alternatives catégorisées comme « ACCEPTÉ » : p
— La taille minimale du schéma d'argumentation (le nombre minimum de couples appartenant au multiplet) : s

L'échantillon de test est constitué d'exemples générés suivant l'algorithme du listing 1.

Afin de tester l'influence des différents paramètres sur les performances, l'ensemble des configurations d'exemples qui ont été générées est disponible ci-dessous.

Les configurations ont par défaut les valeurs suivantes :

— nombre d'alternatives : 20
— nombre de points de vue : 5
— pourcentage d'alternatives acceptées : 35%
— taille minimale du schéma : 2

Pour tester l'influence des différents paramètres, voici les différentes valeurs essayées pour chacun d'entre eux (un seul paramètre est modifié à la fois, les autres restent à leur valeur par défaut) :

— nombre d'alternatives : 10, 20, 30, 40, 50
— nombre de points de vues : 3, 4, 5, 7, 10
— pourcentage d'alternatives acceptées : 10%, 35%, 50%, 75%, 90%
— taille minimale du schéma : −1 (pas de schéma), 1, 2, 3, 4, 5, 6

```
# ENTRÉES :
#   n : nombre d'alternatives
#   m : nombre de points de vue
#   p : taux d'alternatives catégorisées
#                          comme "ACCEPTÉ"
#   s : taille minimale du schéma
# SORTIES :
#   - Les préférences
#   - L'assignation
GÉNÉRER(n,m,p,s):
    alternatives := [1..n]
    acceptes := [1..floor(n*p)]
    refuses := [floor(n*p)+1..n]
    do
      preferences := []
      for i from 0 to m-1
        preferences[i] := copy(alternatives)
        preferences[i] = melanger(preferences[i])
      done
      formule := modeliser(preferences,
                           acceptes, refuses)
      taille_minimale := resoudreInc(formule)
    while taille_minimale != s
    return (preferences, (acceptes, refuses))
```

Listing 1: Protocole de génération d'exemples. **melanger()** utilise le mélange de Fisher-Yates pour assurer une équiprobabilité à toutes les permutations. **modeliser()** utilise la formulation SAT expliquée section 4.2. **resoudreInc(formule)** résout incrémentalement la formule via Gophersat en augmentant la taille du schéma à chaque itération

Pour chacune des configurations, 100 instances ont été générées.

4.2 Comparaison entre représentations (pseudo-booléen contre SAT)

La nature du schéma d'argumentation se prête naturellement à une modélisation pseudo-booléenne, car le schéma possède des contraintes qui sont directement représentables en SAT mais aussi des contraintes de cardinalité.

$$\text{Card1PB :} \quad \sum_{(g,b)\in\mathbb{A}} \lambda_{gb} = k$$
$$\text{Card2PB :} \quad \sum_{i\in\mathcal{N}} \beta_i = k-1$$

$$\text{C1PB :} \quad \forall i \in \mathcal{N}, \forall \langle g,b\rangle \in \mathbb{A}, g \succ_i b, \quad \beta_i + \neg\lambda_{gb} \geq 1$$

$$\text{C2PB :} \quad \forall i \in \mathcal{N}, \forall p \neq q = 1,\ldots,mn,$$
$$\forall \langle g,b\rangle \in \{g_p, g_q\} \times \{b_p, b_q\}, g \succ_i b,$$
$$\neg\beta_i + \neg\lambda_{g_p b_p} + \neg\lambda_{g_q b_q} \geq 1$$

Cette formulation n'est pas une représentation parfaite du schéma, car ici nous fixons le nombre de paires et le

nombre de points de vue dans l'ensemble B. Nous avons fait ce choix, car cela n'impacte pas le résultat « existe-t-il un schéma d'argumentation ? », en effet dans le cas où nous avons $k + n$ paires, dont une alternative est acceptée et une est refusée, à séparer avec $n > 0$ et seulement $k - 1$ points de vue pouvant n'en séparer qu'une seule à la fois et le reste ne pouvant en séparer aucune, retirer n paires ne change pas le résultat, nous avons toujours plus de paires à séparer que de point de vue ne pouvant en séparer qu'une seule à la fois. Ceci est possible car ces paires sont toutes deux à deux disjointes sur chacun des $k - 1$ points de vue pouvant les séparer, c'est-à-dire que pour chaque couple de paires, il existe une des alternatives refusées qui est préférée à une des alternatives acceptées, cela implique qu'il est possible de séparer au plus une paire par point de vue. En retirant n paires, on conserve k paires toutes deux à deux disjointes sur les $k - 1$ points de vue pouvant les séparer. Via cette disjonction, nous avons $k - 1$ points de vue ne pouvant séparer au plus qu'une seule paire à la fois, le reste des points de vue ne pouvant séparer aucune paire et k paires à séparer. Il restera donc une paire inséparable. L'énumération des schémas n'est alors plus exhaustive mais tous les schémas dont le nombre de paires est largement supérieur au nombre de points de vue dans l'ensemble B sont sous-sommés par des schémas avec k paires et $k - 1$ points de vue dans B. La réponse au problème de décision n'est donc pas impactée par ce choix.

Afin de comparer l'efficacité de la formulation pseudo-booléenne à une formulation SAT, nous avons transformé la formulation précédente en SAT en utilisant le codage de Bailleux et Boufkhad pour les contraintes de cardinalités, le reste des contraintes pseudo-booléenne étant déjà sous forme normale conjonctive :

$$\begin{aligned}
\text{Card1SAT :} \quad & \left|\left\{\langle g, b\rangle \in \mathbb{A} : \lambda_{gb} \text{ est vrai } \right\}\right| = k \\
\text{Card2SAT :} \quad & \left|\{i \in \mathcal{N} : \beta_i \text{ est vrai } \}\right| = k - 1 \\
\text{C1SAT :} \quad & \bigwedge_{\substack{\forall i \in \mathcal{N} \\ \forall \langle g,b\rangle \in \mathbb{A}, \\ g \succ_i b}} \beta_i \vee \neg\lambda_{gb}
\end{aligned}$$

C2SAT :

$$\bigwedge_{\substack{\forall i \in \mathcal{N} \\ \forall p \neq q = 1,\ldots,mn \\ \forall \langle g,b\rangle \in \{g_p, g_q\} \times \{b_p, b_q\}, \\ g \succ_i b}} \neg\beta_i \vee \neg\lambda_{g_p b_p} \vee \neg\lambda_{g_q b_q}$$

La formule finale étant donc :

$$\Phi_2 : \quad \text{Card1SAT} \wedge \text{Card2SAT} \wedge \text{C1SAT} \wedge \text{C2SAT}$$

En utilisant le solveur Gophersat sur ces deux formulations équivalentes avec la totalité des instances générées, nous obtenons les résultats figure 1. Ces résultats comptent uniquement la lecture du problème soit SAT soit pseudo-booléen ainsi que sa résolution. Dans ce cas, nous pouvons

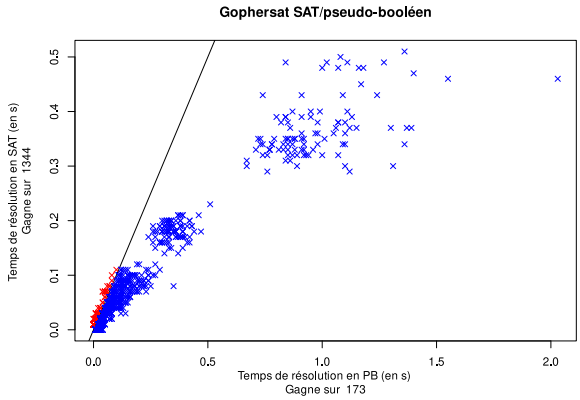


FIGURE 1 – Diagrammes du temps de résolution en secondes d'une instance SAT contre son équivalent pseudo-booléen via le solveur Gophersat

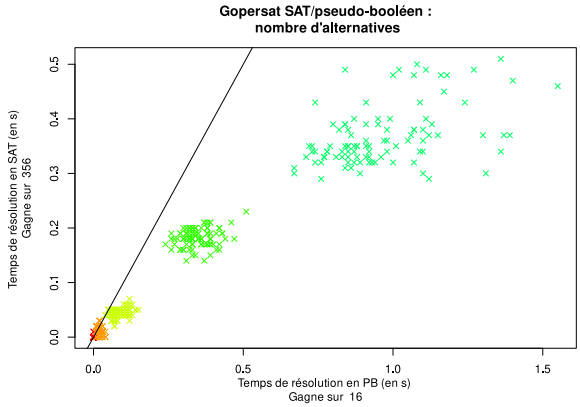


FIGURE 2 – Diagrammes du temps de résolution en secondes d'une instance SAT contre son équivalent pseudo-booléen via le solveur Gophersat en fonction du nombre d'alternatives : rouge = 10, orange = 20, jaune = 30, vert = 40, vert printemps = 50

voir que la résolution de la formalisation SAT semble plus rapide que son équivalent pseudo-booléen.

Nous pouvons maintenant nous intéresser aux effets des différents paramètres sur ces formalismes, la figure 2 montre l'influence du nombre d'alternatives sur les performances. Nous pouvons voir que si les deux sont impactés par l'augmentation du nombre d'alternatives, cette augmentation est plus forte avec la modélisation pseudo-booléenne.

D'un autre côté si nous considérons cette fois l'impact du nombre de points de vue, nous obtenons les résultats disponibles tableau 3. Il semblerait que le nombre de points de vue ait un impact plus fort sur la résolution en pseudo-booléen. Du côté de la résolution SAT, nous pouvons noter que la parité de ce nombre semble avoir un impact plus fort.

#POVs	PB	SAT
3	10	61
4	8	52
5	8	62
7	8	53
10	5	62

TABLE 3 – Récapitulatif des performances des formalismes pseudo-booléens et SAT avec le solveur Gophersat. #POVs indique le nombre de points de vue testé, PB (resp. SAT) indique le nombre d'instances plus rapides avec le formalisme pseudo-booléen (resp. SAT).

%acceptées	PB	SAT
10	7	29
35	13	50
50	14	52
75	6	54
90	10	18

TABLE 4 – Récapitulatif des performances des formalismes pseudo-booléens et SAT avec le solveur Gophersat. %acceptées indique le pourcentage d'alternatives acceptées testé, PB (resp. SAT) indique le nombre d'instances plus rapide avec le formalisme pseudo-booléen (resp. SAT).

La table 4 montre que le taux d'alternatives classées comme ACCEPTÉ semble avoir un impact plus fort sur la modélisation SAT.

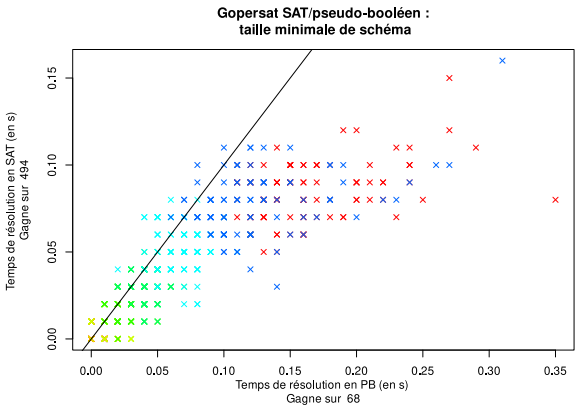


FIGURE 3 – Diagrammes du temps de résolution en secondes d'une instance SAT contre son équivalent pseudo-booléen via le solveur Gophersat en fonction de la taille minimale de schéma : rouge = pas de schéma, orange = 1, jaune = 2, vert = 3, vert printemps = 4, cyan = 5, bleu = 6

La taille minimale du schéma a très logiquement un impact sur les deux représentations, car celle-ci correspond au nombre d'appels au solveur nécessaire pour trouver une solution. Toutefois, d'après la figure 3, cet impact semble encore une fois plus important pour la représentation pseudo-booléenne.

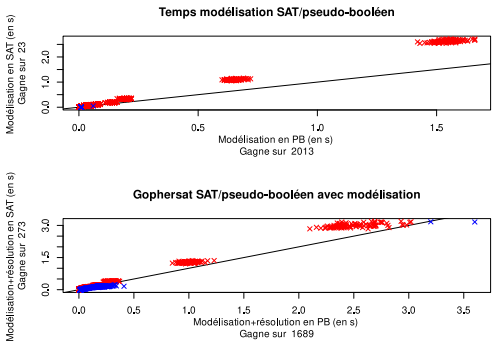


FIGURE 4 – Ligne 1 : Diagrammes du temps de modélisation en secondes d'une instance en SAT contre son équivalent pseudo-booléen. Ligne 2 : Diagrammes du temps de résolution en secondes d'une instance SAT contre son équivalent pseudo-booléen via le solveur Gophersat en prenant en compte le temps de modélisation

Bien que la modélisation SAT soit plus rapide à résoudre, tout cela reste à contre-balancer, car jusque là nous ne prenions pas en compte le temps qu'il a fallu pour modéliser ces instances. La figure 4 montre le temps nécessaire à la modélisation et à l'écriture sur le disque des modélisations. Cette fois, nous voyons que le temps nécessaire pour modéliser en SAT est plus important que celui pour modéliser en pseudo-booléen. Dans cette situation, si nous nous intéressons au temps de résolution avec modélisation, la figure 4 montre que, c'est le formalisme pseudo-booléen qui est meilleur en moyenne même si la différence n'est pas aussi importante que plutôt lorsque nous ne considérions que le temps de résolution.

4.3 Comparaison entre solveurs (Gophersat contre Glucose)

Dans cette sous-section, nous présentons quelques résultats intéressants obtenus en utilisant deux solveurs différents pour les problèmes SAT. Comme mentionné en introduction de cette section nous avons utilisé pour cela Glucose 4.0 et Gophersat 1.2.

Nous avons comparé les deux modélisations SAT présentées précédemment sur ces solveurs. Nous appellerons SAT la modélisation utilisant une constante pour k expliquée sous-section 4.2 et SATNoK la modélisation sans constante présentée sous-section 3.2.1.

A priori, nous nous attendons à ce que Glucose soit plus performant que Gophersat, cependant la figure 5 montre que quelle que soit la modélisation (SAT ou SATNoK),

Gophersat est au mieux un bon concurrent au pire surclasse Glucose.

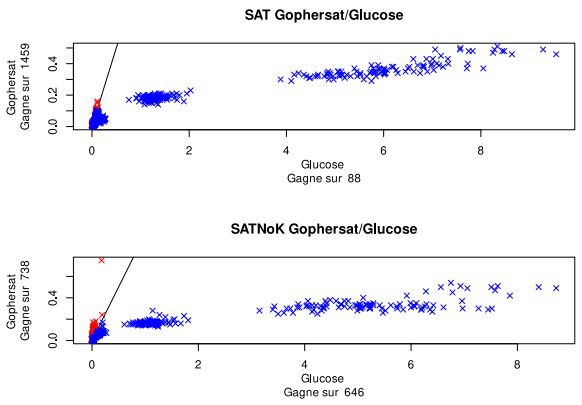


FIGURE 5 – Ligne 1 : Diagramme du temps de résolution d'une instance SAT sur Gophersat en comparaison avec Glucose. Ligne 2 : Diagramme du temps de résolution d'une instance SATNoK sur Gophersat en comparaison avec Glucose.

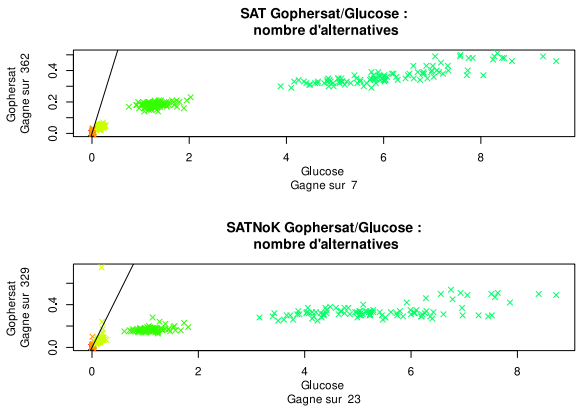


FIGURE 6 – Ligne 1 : Diagramme du temps de résolution d'une instance SAT sur Gophersat en comparaison avec Glucose en fonction du nombre d'alternatives. Ligne 2 : Diagramme du temps de résolution d'une instance SATNoK sur Gophersat en comparaison avec Glucose en fonction du nombre d'alternatives. Nous avons : rouge = 10, orange = 20, jaune = 30, vert = 40, vert printemps = 50

Pour essayer de comprendre ce qui cause cet écart de performance contre-intuitif, nous pouvons comme précédemment nous intéresser aux différents paramètres du schéma et voir ce qui influe sur les performances relatives de chaque solveur. Comme le suggère la forme des nuages de points sur la figure 5, le nombre d'alternatives a un plus gros impact sur les performances de Glucose que sur celle de Gophersat, la figure 6 confirme cette suggestion. Si nous détaillons ces résultats, pour un nombre d'alternatives égal à 10, les performances de Glucose et de Gophersat sont quasiment identiques. Lorsque nous montons à 20 alternatives, avec la modélisation SAT, Gophersat commence à avoir de meilleures performances (ce qui n'est pas aussi vrai avec SATNoK). Enfin à partir de 30 alternatives, presque 100% des instances sont plus rapides sur Gophersat que sur Glucose. Nous pouvons même noter, sur la figure, un impact exponentiellement plus important sur Glucose que sur Gophersat.

#POVs	Gl/SAT	Go/SAT	Gl/NoK	Go/NoK
3	0	74	15	35
4	1	73	23	28
5	7	66	27	29
7	4	63	37	32
10	6	57	64	22

TABLE 5 – Récapitulatif des performances des formalismes SAT et SATNoK avec les solveurs Glucose et Gophersat. #POVs indique le nombre de points de vue testé, Gl (resp. Go) indique le nombre d'instances SAT ou SATNoK plus rapides avec Glucose (resp. Gophersat).

Le nombre de points de vue a lui aussi un impact. Avec SATNoK, Glucose est moins pénalisé par l'augmentation du nombre de points de vue que Gophersat, cependant cela semble moins évident avec la modélisation SAT comme nous pouvons le voir dans la table 5.

%acceptées	Gl/SAT	Go/SAT	Gl/NoK	Go/NoK
10	10	0	44	0
35	5	70	24	31
50	1	60	11	32
75	2	60	26	16
90	12	0	40	1

TABLE 6 – Récapitulatif des performances des formalismes SAT et SATNoK avec les solveurs Glucose et Gophersat. %acceptées indique le pourcentage d'alternatives acceptées testé, Gl (resp. Go) indique le nombre d'instances SAT ou SATNoK plus rapides avec Glucose (resp. Gophersat).

La table 6 montre que Gophersat prend l'avantage sur Glucose lorsque la répartition de la jurisprudence entre les catégories ACCEPTÉ et REFUSÉ est équilibrée, mais que Glucose est plus rapide sur les répartitions fortement déséquilibrées.

Enfin, en regardant cette fois la taille minimale de schéma présente dans les instances, nous obtenons la figure 7. Nous pouvons noter que les performances de Gophersat diminuent significativement plus vite que celle de Glucose en augmentant la taille minimale de schéma avec la modélisation SATNoK pour laquelle lorsqu'il n'y a pas de schéma 96

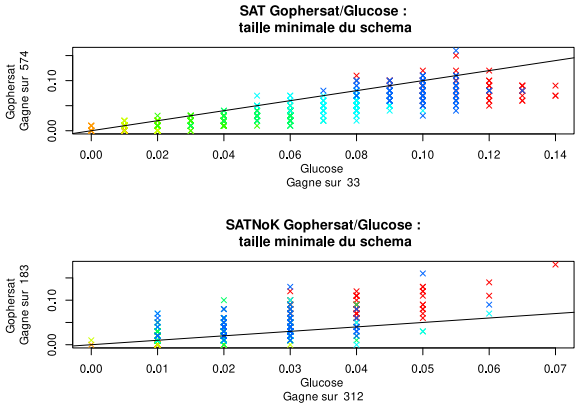


FIGURE 7 – Ligne 1 : Diagramme du temps de résolution d'une instance SAT sur Gophersat en comparaison avec Glucose en fonction de la taille minimale de schéma. Ligne 2 : Diagramme du temps de résolution d'une instance SATNoK sur Gophersat en comparaison avec Glucose en fonction de la taille minimale de schéma. Nous avons : rouge = pas de schéma, orange = 1, jaune = 2, vert = 3, vert printemps = 4, cyan = 5, bleu = 6

instances sur les 100 ont été plus rapides sur Glucose. Avec la modélisation SAT, il est plus difficile de noter un changement dans la relation des performances entre les deux solveurs.

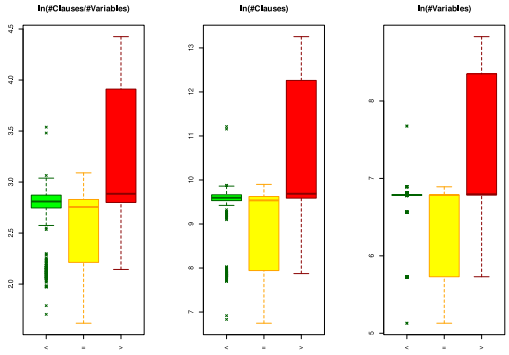


FIGURE 8 – Diagrammes en boîtes montrant la répartition de la quantité logarithme du nombre de clauses sur le nombre de variables, du logarithme du nombre de clauses et du logarithme du nombre de variables avec la modélisation SATNoK lorsque Glucose est plus rapide que Gophersat (<), est aussi rapide (=) et est plus lent (>).

La figure 8 montre que Glucose est aussi efficace voire plus efficace lorsqu'il y a peu de variables ou de clauses. Cependant, il est impossible de conclure sur le fait que ce soit le nombre de clauses, le nombre de variables ou le rapport entre ces deux quantités qui influe sur les performances.

Nous n'avons pas poussé plus loin l'étude permettant d'identifier cet écart de performances, néanmoins nous pensons que cela pourrait être dû à la configuration de la machine ne disposant pas de beaucoup de ressources ou encore à des stratégies différentes lors de la résolution. Toutefois, n'ayant pas exploré ces aspects, nous n'avons pas de réponse concrète sur ce point.

5 Conclusion

Nous avons montré que le schéma d'arguments visant à donner une explication d'une décision nécessaire étant donnée une jurisprudence dans le modèle du tri non compensatoire que nous avons étudié ne permet pas de justifier l'ensemble des décisions nécessaires. Cette constatation ouvre la question de la proportion du fragment explicable que recouvre ce schéma. À cette question, nous avons trouvé que pour des préférences sous forme d'ordres totaux stricts avec de faibles paramètres (6 alternatives et 3 points de vue) cette portion non couverte est très faible, toutefois cette recherche bien qu'exhaustive possède des symétries. De plus, comme ce schéma a pour objectif de fournir une explication, plus celui-ci est grand, plus il est difficile de le comprendre. Dans cette optique, ce schéma étant paramétrable en fonction de sa taille, il peut alors être intéressant de se pencher sur les différents niveaux de fragments explicables que l'on obtient en fixant la taille maximale du schéma. Enfin, lors de la recherche de décision nécessaire non couverte par ce schéma, nous avons pu mettre en avant une construction intéressante de contre-exemple en ajoutant sur une configuration possédant un schéma, un point de vue et une alternative. L'axe de recherche sur la caractérisation de cet espace non couvert par le schéma s'ouvre aussi.

Enfin, sur le calcul de ce schéma via des modélisations SAT et pseudo-booléenne nous avons constaté qu'en termes de résolution pure sur notre problème, le solveur Gophersat est moins efficace sur une formulation pseudo-booléenne que sur son équivalent SAT. Toutefois, ce résultat s'inverse lorsque l'on prend en compte le temps de modélisation *a priori* de la résolution avec Gophersat. Quant à la comparaison des performances sur notre problème entre Gophersat et Glucose pour des formulations SAT, nous avons constaté que Gophersat était souvent plus efficace que Glucose.

Références

[1] G. Audemard, L. Simon, Glucose, `https://www.labri.fr/perso/lsimon/glucose/`, 2022/03/17

[2] O. Bailleux, Y. Boufkhad : Efficient CNF Encoding of Boolean Cardinality Constraints, *CP2003*, pp. 108–122, 2003

[3] K. Belahcene, Y. Chevaleyre, C. Labreuche, N. Maudet, V. Mousseau, W. Ouerdane : Accountable Approval Sorting, IJCAI 2018, pp. 70–76.

[4] D. Bouyssou, T. Marchant : An axiomatic approach to noncompensatory sorting methods in MCDM, I : The case of two categories. Eur. J. Oper. Res. 178(1) : 217-245 (2007)

[5] Centre de Recherche en informatique de Lens, Gophersat, `https://github.com/crillab/gophersat`, 2022/03/17

[6] N. Eén, N. Sörensson, Minisat, `http://minisat.se/`, 2022/03/17

[7] A. Tlili, K. Belahcène, O. Khaled, V. Mousseau, W. Ouerdane : Learning non-compensatory sorting models using efficient SAT/MaxSAT formulations. Eur. J. Oper. Res. 298(3) : 979-1006 (2022)

Actes JIAF 2022

Explications de recommandations fondées sur des principes d'équité à l'aide de transferts

Hénoïk Willot Khaled Belahcene Sébastien Destercke

Heudiasyc, Université de Technologie de Compiègne, France

{henoik.willot, khaled.belahcene, sebastien.destercke}@hds.utc.fr

Résumé

Nous explorons la génération d'explication pour des préférences basées sur un modèle de somme pondérée ordonnée (*OWA*) favorisant une répartition équilibrée des performances relatives à différents points de vue. Nous proposons des explications, correctes vis-à-vis du modèle, fondées sur la composition par transitivité d'arguments élémentaires fondés sur la dominance de Pareto, des transferts de Pigou-Dalton, et l'information préférentielle fournie par le décideur. Nous proposons plusieurs approches heuristiques permettant de calculer ces explications, validées par une campagne expérimentale montrant que les explications obtenues sont souvent de longueur optimale.

1 Introduction

La somme pondérée ordonnée (*OWA*) a été introduite en décision multi-critères par Yager [6] et nous intéresse pour sa ressemblance avec la somme pondérée classique. En effet, la seule différence porte sur la non linéarité introduite avec l'ordre croissant appliqué aux critères avant d'y appliquer une somme pondérée. Ainsi les valeurs des poids appliqués deviennent dépendantes du rang et non plus du critère en lui même, ce qui permet de représenter de nouveaux opérateurs impossibles pour la somme pondérée classique comme le minimum ou le maximum.

Definition 1 (Permutation croissante)**.** *Nous définissons la permutation croissante* $^\uparrow$ *comme la permutation sur* $\mathcal{X}^n$*, l'espace des candidats, tel que* $x \mapsto x^\uparrow$ *avec* $x_1^\uparrow \leq x_2^\uparrow \leq \cdots \leq x_n^\uparrow$.

Definition 2 (Somme pondérée ordonnée)**.** *est une fonction* $\mathcal{X}^n \to \mathbb{R}^+$ *définie par un vecteur de poids* $w \in \mathcal{W}$ *tel que* $\forall\, i\ w_i \in [0,1]$ *et* $\sum_{i=1}^n w_i = 1$ *:*

$$OWA_w(x) = \sum_{i=1}^n w_i x_i^\uparrow$$

Les vecteurs de poids que nous étudions dans ce papier sont ceux possédant une contrainte de valeurs décroissantes, *i.e.* $w_1 \geq \ldots w_n$, ainsi le poids le plus grand est appliqué au premier critère, celui étant le plus petit, ce qui permet de favoriser une répartition équilibrée des critères. Nous appelons ces opérateurs FOWA [1].

En s'appuyant sur la similarité avec la somme pondérée, nous avons appliqué une méthode similaire à Belahcene et al. [1], inspirée des *even swaps* [3], pour construire des explications des résultats. L'idée est de décomposer la préférence à expliquer par un enchaînement d'arguments, plus simples et que l'on sait faire du sens pour le demandeur, permettant *in fine* de relier les deux extrémités de la préférence.

2 Explications de la dominance de Lorenz

Parmi les préférences possibles issues d'un FOWA, un sous-ensemble apparaît distinctement. Il s'agit des candidats qui sont Lorenz-dominés.

Definition 3 (Lorenz-dominance)**.** *On dit de deux candidats* $a, b \in \mathcal{X}^n$ *que a Lorenz-domine b s'ils vérifient :*

$$\forall i \in \{1, \ldots, n\} \sum_{j=1}^i a_j^\uparrow \geq \sum_{j=1}^i b_j^\uparrow$$

Si, et seulement si, les candidats vérifient la définition 3, alors peut importe les poids du FOWA, la préférence issue de ce dernier sera dans le même sens que la dominance. Expliquer ces préférences représente un intérêt car elles constituent le socle des préférences nécessaires du

1. *Fairness-oriented Ordered Weighted Average*

modèle. Il existe dans la littérature un lien entre la Lorenz-dominance et un principe intitulé le principe de Pigou-Dalton (PDP)[5].

Definition 4 (Principe de Pigou-Dalton)**.** *Notons $a, b \in \mathcal{X}^n$ Le vecteur $a^{\uparrow}$ est préféré au vecteur $b^{\uparrow}$ par le principe de Pigou-Dalton s'il existe des indices des critères $i, j \in \{1, \ldots, n\}$, $i < j$ et une quantité $\epsilon > 0$ tels que :*

$$\begin{cases} \forall k \in \{1, \ldots, n\},\ k \neq i,\ k \neq j,\ a \uparrow_k = b \uparrow_k; \\ a_i^{\uparrow} = b_i^{\uparrow} + \epsilon \ \leq a_{i+1}^{\uparrow};\ \textit{and} \\ a_j^{\uparrow} = b_j^{\uparrow} - \epsilon \geq a_{j-1}^{\uparrow}. \end{cases}$$

Ce principe est fondamentalement un principe de redistribution des richesses en faveur des plus pauvres, ce qui favorise donc les répartitions équilibrées parmi les critères, tout comme nos FOWA. Cela constitue un premier argument à utiliser pour les expliquer. Dans la littérature, un lien d'équivalence existe entre la Lorenz-dominance, et l'existence d'une suite d'utilisations du PDP. Une heuristique pour calculer cette dernière en temps polynomial existe dans le contexte connexe à la Lorenz-dominance qu'est la majorisation [4].

Cependant cet algorithme ne prend pas en compte plusieurs choses dans sa conception, comme le besoin qu'à chaque étape de la suite les candidats soient ordonnés, suppose que $\sum_{j=1}^{n} a_j^{\uparrow} = \sum_{j=1}^{n} a_j^{\uparrow}$, et ne cherche pas à optimiser la longueur de la suite. Pour palier ces problèmes nous proposons une nouvelle heuristique et la comparons à une recherche A* pour étudier sa distance par rapport à la longueur minimale.

3 Explications de préférences nécessaires

Comme nous l'avons énoncé précédemment, les préférences issues de la Lorenz-dominance constituent des préférences nécessaires au FOWA, nécessaires dans le sens où tout FOWA produit forcément la même préférence. Dans la recherche de robustesse par rapport aux préférences de l'utilisateur, au lieu de nous concentrer sur la détermination d'un unique vecteur de poids de FOWA w, nous préférons appliquer une méthode de décision robuste, décrite sous le nom d'UTA-GMS dans le cadre d'un modèle additif [2]. Cette méthode détermine un ensemble d'opérateurs W qui sont compatibles avec les informations préférentielles (PI) que nous avons récoltées auprès de l'utilisateur, sous la forme de questions sur des couples d'alternatives. Ainsi, nous sommes sûrs de ne pas faire d'erreur de généralisation lorsque nous trouvons qu'une préférence est nécessaire sur cet ensemble, elle l'est aussi pour l'utilisateur vu qu'elle résulte des informations qu'il nous a donné.

Grâce à l'équivalence entre être Lorenz-dominé et les préférences nécessaires des FOWA, toute préférence qui devient nécessaire dans W et qui n'est pas issue de la Lorenz-dominance, l'est forcément à cause de la PI récoltée. A l'aide de la programmation linéaire, nous pouvons donc calculer des coefficients à appliquer aux différentes préférences récoltées pour quantifier leur implication dans la préférence, et ensuite les utiliser comme arguments, accompagnés par le PDP et la dominance de Pareto dans l'explication.

L'existence de tels coefficients est assurée par le lemme de Farkas, nous n'avons donc ensuite qu'à reconstruire un ordonnancement à appliquer à nos arguments afin d'avoir une explication reliant les deux candidats. Pour ce faire nous avons appliqué plusieurs méthodes de complexité et de généralité croissantes pour reconstruire l'explication. Nous essayons tout d'abord de trouver la bonne permutation parmi les arguments. Nous avons ensuite relâchons notre hypothèse d'indépendance des arguments et fusionnons les contributions de la PI afin d'obtenir des explications non obtenues. Cependant, encore un trop gros nombre de préférences ne sont pas recomposées (environ 60% dans notre campagne expérimentale).

Pour palier ce problème nous avons commencé à mettre en place une démarche hybride entre la programmation linéaire et l'ordonnancement et les résultats sont encourageants pour réduire drastiquement la proportion sans explication, mais avec une complexité plus élevée. Nous envisageons donc de continuer vers de l'ordonnancement non contraint afin de pouvoir trouver une explication à pratiquement toutes les préférences.

Références

[1] Belahcène, Khaled, Christophe Labreuche, Nicolas Maudet, Vincent Mousseau et Wassila Ouerdane: *Comparing Options with Argument Schemes Powered by Cancellation*. Dans Kraus, Sarit (rédacteur) : *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1537–1543. ijcai.org, 2019.

[2] Greco, Salvatore, Vincent Mousseau et Roman Slowinski: *Ordinal regression revisited : Multiple criteria ranking using a set of additive value functions*. Eur. J. Oper. Res., 191(2) :416–436, 2008.

[3] Hammond, John S., Ralph L. Keeney et Howard Raiffa: *The Even-Swap Method for Multiple Objective Decisions*. Dans Haimes, Yacov Y. et Ralph E. Steuer (rédacteurs) : *Research and Practice in Multiple Criteria Decision Making*, pages 1–14, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[4] Marshall, A.W., I. Olkin et B.C. Arnold: *Inequalities : Theory of Majorization and Its Applications*. Springer Series in Statistics. Springer New York, 2010.

[5] Shorrocks, Anthony F.: *Ranking Income Distributions*. Economica, 50(197) :3–17, 1983.

[6] Yager, Ronald R.: *On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking*. Dans Dubois, Didier, Henri Prade et Ronald R. Yager (rédacteurs) : *Readings in Fuzzy Sets for Intelligent Systems*, pages 80–87. Morgan Kaufmann, 1993.

Session 3 : Intelligence ou bêtise artificielle ?

Actes JIAF 2022

Quelques réflexions autour de la notion de bêtise artificielle *

Jean Lieber[1] **Jean-Guy Mailly**[2] **Pierre Marquis**[3,4]
Henri Prade[5] **François Rollin**

[1] Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy
[2] Université Paris Cité, LIPADE, F-75006 Paris
[3] Univ. Artois, CNRS, CRIL, 62300 Lens
[4] Institut Universitaire de France
[5] IRIT, Toulouse
jean.lieber@loria.fr jean-guy.mailly@u-paris.fr
marquis@cril.univ-artois.fr prade@irit.fr francoisrollin3@icloud.com

La bêtise est souvent l'ornement de la beauté ; c'est elle qui donne aux yeux cette limpidité morne des étangs noirâtres et ce calme huileux des heures tropicales.

Charles Baudelaire, *Journaux intimes* (1887)

Résumé

Le professeur Rollin a écrit au sujet de la bêtise artificielle. Pouvions-nous rester indifférent à cela ? Cet article montre le contraire.

Abstract

Professor Rollin wrote about artificial stupidity. Could we remain indifferent to this ? This article shows the contrary.

1 Introduction

Dans un ouvrage paru récemment et consacré à la notion de bêtise, François Rollin consacre un chapitre intitulé « BA ? » et consacré à la *bêtise artificielle* [24]. Si on voit la bêtise comme une « intelligence en creux », cela suggère des liens forts entre BA et IA, d'où l'intérêt pour la communauté de l'IA de se pencher sur cette question.

Mais comment pourrait-on définir la BA ?

La thèse que François Rollin développe est, en substance, qu'alors qu'un système d'IA doit effectuer des opérations de tri dans les données et d'adaptation sur ces données, un système de BA s'abstiendra d'un de ces deux types d'opérations. Il en conclut qu'un système de BA pourrait être un système associant à une question une réponse sans lien avec elle et que la mise en place d'un système de BA serait facile. Il illustre cette idée par des exemples, dont ceux-ci :

> Comment résoudre une équation du second degré ? Réponse de la machine : en faisant manger à tous les centenaires de la région Hauts-de-France six kilos de cœur d'artichaut par jour pendant huit semaines.
>
> Combien d'étoiles y a-t-il au total dans l'Univers à 20% près ? Réponse : 8.
>
> *(extraits cités avec l'aimable autorisation de l'auteur)*

Cette thèse nous a interpellés et cet article présente quelques réflexions à ce sujet. La bêtise est un sujet peut-être encore plus fascinant que l'intelligence, au vu du nombre considérable d'ouvrages célébrant, interrogeant, dénonçant la bêtise. Citons-en tout de même quelques-uns dans différents registres [14, 19, 16, 3, 12, 7].

Cet article est organisé comme suit. La section 2 tente de cerner la notion de bêtise. Puis, quelques bêtises faites par des êtres humains ou des machines seront discutées (section 3). On s'intéressera ensuite à la notion de système de BA. À son sujet, les deux questions suivantes peuvent se poser à des spécialistes de l'IA :

*Les auteurs remercient les rapporteurs anonymes de cet article pour leurs retours encourageants ou critiques (ou les deux). Ils se sont efforcés d'en tenir compte pour la version finale, en espérant n'avoir pas mal interprété ces remarques et tout en sachant que le thème de l'article leur fournissait une excuse toute trouvée le cas échéant. Ils ont également apprécié les suggestions faites par ces rapporteurs anonymes, même quand ils n'ont pas pu les intégrer pleinement dans cet article.

— Pourquoi construire (sciemment) un système de BA ?
— Comment construire un tel système ?

Les sections 4 et 5 abordent ces deux questions. Après une conclusion provisoire (section 6), les quatre premiers auteurs laissent la parole au cinquième : François Rollin *himself*.

2 Cerner la notion de bêtise

Une approche possible pour cerner la « bêtise artificielle » est de procéder par une mise en contraste avec « l'intelligence artificielle (IA) ».

2.1 Un point de vue anthropocentré

Il s'agit alors de commencer par préciser ce qu'il faut comprendre par « intelligence artificielle ». Et dans une telle tentative, on se heurte immédiatement à la question épineuse de devoir définir ce que recouvre « l'intelligence ». Pour ce faire, on contourne le plus souvent la difficulté en faisant *référence à l'humain*, qui présente la forme d'intelligence sans doute la plus aboutie, en tout cas la plus polymorphe.

Le test de Turing [28] va dans le sens d'un tel jeu d'imitation : un processus est vu comme « intelligent » quand celui qui l'observe ne peut pas prédire s'il est le fait d'un être humain ou d'une machine. Ainsi, l'attribut « artificiel » ne pose pas de souci spécifique dans l'élaboration d'une définition (« artificiel » signifie seulement que le processus « intelligent » qui est analysé est produit par une machine). Seule « l'intelligence » pose question.

On notera au passage que la comparaison à l'humain pour tenter de s'en sortir et caractériser « l'intelligence » par une opposition pré-existait au test de Turing. Ainsi, pour René Descartes [8, cinquième partie], « l'intelligence » (au sens de la raison) est l'apanage de l'humain, c'est ce qui sépare l'Homme de l'animal. Le bêtise serait donc *a contrario* le propre de l'animal (ce qui explique l'étymologie du mot « bêtise » et les nombreuses références à l'animal quand il s'agit de parler d'âneries...). La thèse de Descartes est évidemment discutable : la raison n'est qu'une forme d'intelligence parmi d'autres et, d'un point de vue biologique, l'être humain est un animal parmi d'autres. Aujourd'hui, dans le champ disciplinaire qu'est l'IA, quand on met en avant des algorithmes dits bio-inspirés mettant en œuvre une forme d'intelligence collective, on mentionne le plus souvent des sociétés animales (comme c'est le cas pour les colonies de fourmis), mais pas des sociétés humaines. Est-ce parce qu'elles présentent des comportements trop complexes pour être modélisées de façon satisfaisante ou parce que la valeur ajoutée par le collectif est finalement trop limitée pour être notable ? Chacun peut avoir un point de vue sur la question.

2.2 Être bête

La bêtise (au sens de la capacité à être bête ou plutôt de l'incapacité à ne pas être bête) a souvent été définie en creux comme une déficience des capacités intellectuelles (en particulier, comme le veulent cette fois l'étymologie de l'« intelligence » et les tests de QI, le manque d'aptitudes à établir des liens). Sans surprise, la référence au développement intellectuel chez l'Homme a servi de repère pour établir au XIXe siècle une gradation de la bêtise [13], où l'idiot est celui qui ne peut pas communiquer par la parole, l'imbécile celui qui est incapable de lire et d'écrire et le sot, celui qui ne raisonne ni ne se comporte de manière normale. Comme l'écrivait Eugène Marbeau : « La bêtise ne comprend pas ; la sottise comprend de travers » (*in* Livre d'or de la Comtesse Diane, 1886).

Pour qu'un agent (qu'il soit humain ou artificiel) puisse être qualifié de bête, il ne suffit évidemment pas qu'il produise une bêtise de temps en temps (à ce compte-là, tous les humains seraient stupides), mais que l'agent concerné *persiste dans ses erreurs*, même quand une interaction soutenue avec son environnement lui fournit suffisamment d'arguments valables pour qu'en principe, les erreurs ne soient pas reproduites. Les agents artificiels ont à ce titre une prédisposition plus grande à la bêtise que les agents humains, simplement parce que leur interaction avec leur environnement est typiquement beaucoup plus réduite, voire inexistante (les agents artificiels ne sont pas tous munis de capteurs et d'effecteurs) mais aussi parce que leur force est précisément de pouvoir répéter sans se lasser (et très rapidement) les mêmes traitements et en conséquence de répéter les mêmes erreurs. On peut noter que cette idée de la bêtise liée à la persistance de l'erreur fait partie des points étudiés en détail dans l'ouvrage de François Rollin cité en introduction [24], même s'il n'est guère mis en avant dans le chapitre sur la BA.

La bêtise prenant des formes très variées, il est sans doute illusoire de vouloir caractériser précisément ce qu'est être bête. On peut néanmoins retenir cette idée de persistance dans l'erreur comme un de ses traits marquants. L'immutabilité des croyances nous sort de la condition humaine, et de sa capacité à savoir en interrogeant ses savoirs antérieurs pour les remettre en question chaque fois que nécessaire. Elle nous rapproche une fois de plus de l'animal, en particulier, de l'autruche (dont la politique pourrait être de ne rien voir pour ne rien savoir) voire du sur-humain comme le laisse entendre la célèbre citation latine *errare humanum est, perseverare diabolicum*. Pour laisser le dernier mot à Eugène Marbeau : « La sottise se croit très habile, et ne doute de rien. » (*Pensées et maximes diverses*, 1906).

2.3 Faire des bêtises

Définir ce qu'est une bêtise est sans doute beaucoup plus simple que de définir ce qu'est être bête. Lançons-nous donc dans une tentative de cerner les bêtises.

Une bêtise peut être vue comme le résultat d'un processus « intelligent » qui a mal tourné, qu'il s'agisse d'un processus de raisonnement ou de prise de décision ; ce résultat est simplement faux ou inadapté, parce qu'il s'appuie sur des informations qui le sont tout autant ou parce qu'il est engendré en utilisant des règles de raisonnement fallacieuses ou en passant par des chemins de traverses, ou encore parce qu'il s'appuie sur un modèle décisionnel irrationnel. Supposer au départ l'existence d'un processus « intelligent » qui aurait pu convenir permet de séparer les bêtises des absurdités, qui, elles, seraient produites par des processus totalement non pertinents. Ainsi, à la question « Combien font 12345 fois 67890 ? », une réponse comme « Stéphanie de Monaco » ne serait pas juste bête, mais bien absurde. Et un agent qui répondrait systématiquement « Stéphanie de Monaco » à toute question posée (sauf éventuellement quand la bonne réponse pourrait être « Stéphanie de Monaco »...) [1] serait assurément jugé stupide.

Pour illustrer la production de bêtises comme résultat d'un processus « intelligent » inadapté, considérons les deux exemples suivants (l'un lié au raisonnement, l'autre à la prise de décision) :

> *Tous les chats sont mortels, Socrate est mortel, donc Socrate est un chat.*
>
> Eugène Ionesco, *Rhinocéros*.

Ici, les informations utilisées sont correctes, c'est l'inférence qui ne l'est pas : en l'espèce, elle est abductive, pas déductive, et donc peut aboutir à des conclusions erronées (comme c'est le cas ici).

Voici un autre exemple (un dialogue entre deux amis) :

— *J'ai lu dans un magazine que ma consommation d'alcool (une bouteille de Bordeaux par jour) est excessive et pourrait nuire gravement à ma santé.*
— *Sans doute. Qu'as-tu fait alors ?*
— *J'ai arrêté de lire.*

Dans ce cas, la décision prise est inattendue et sans doute inadaptée.

Décider du caractère erroné ou inadéquat du résultat requiert une *comparaison à une norme*. Selon une certaine « distance » à un point de référence, le résultat en question pourra être jugé approprié ou non, et le processus l'ayant produit sera considéré comme (plus ou moins) « intelligent » ou « bête ». La distance en question peut alors susciter chez celui ou celle qui la mesure des sentiments variés, du courroux, du rire, voire de l'émerveillement.

Notons que la norme employée pour mesurer l'adéquation du résultat produit par le processus de raisonnement ou de prise de décision considéré peut être *universelle* (la logique mathématique permet, par exemple, de séparer les raisonnements valides de ceux qui ne le sont pas) ou ne pas l'être (il n'existe pas de modèle unique et consensuel pour caractériser ce qu'est une prise de décision raisonnable). Cette norme est surtout souvent *relative à un contexte*, en particulier à la personne qui reçoit la bêtise et décide de la classer comme telle ; et cela est fonction de ses aptitudes propres, ce qui fait que ce qui est bêtise pour un individu donné ne l'est pas forcément pour un autre. Prenons un exemple pour lequel une norme universelle existe : il s'agit encore de multiplier 12345 par 67890. Il n'y a qu'un seul vrai résultat possible : le produit recherché vaut 838102050. Le résultat -1 produit par l'individu A pourra être considéré comme stupide par toute personne ayant un petit bagage mathématique, le résultat 838102051 produit par l'individu B, tout aussi erroné, sera peut-être considéré comme une erreur de calcul par certains et comme une bêtise par d'autres (on pourra observer que comme 67890 finit par un 0, le produit recherché doit lui aussi finir par un 0), le résultat 838012050 produit par l'individu C, tout aussi faux, sera sans doute perçu comme moins bête. C'est cette « distance » au résultat escompté (peut-être incorrect lui aussi) qui fait qu'une simple erreur acquiert le statut de bêtise.

On notera au passage que, pour ce genre de tâches, les humains, à cause de leurs limitations cognitives, sont souvent beaucoup plus bêtes que des machines élémentaires comme des calculettes. La capacité des humains à raisonner correctement avec des nombres (et, en particulier, avec des probabilités) est réduite et connue comme telle depuis longtemps. Dans son ouvrage [20], le mathématicien américain John Allen Paulos a introduit le néologisme « *innumeracy* » pour désigner cette incapacité (l'« *innumeracy* » est aux nombres ce que l'« *illiteracy* » – en français, l'analphabétisme – est aux lettres). La période pandémique que nous traversons depuis deux ans a conduit à un florilège d'erreurs de raisonnement, comme celle de refuser la vaccination par le fait (avéré) qu'il y a plus de cas graves de la COVID parmi les vaccinés que parmi les non-vaccinés, mais vu comme un argument (fallacieux) qui soutiendrait le fait que la probabilité de faire une COVID grave en étant vacciné serait supérieur à celui de faire un COVID grave en ne l'étant pas (évidemment, l'erreur vient du fait que le nombre de personnes vaccinées et de personnes non vaccinées n'est pas le même !). De telles erreurs de raisonnement peuvent malheureusement conduire à des issues tragiques, comme cela se produit dans d'autres cadres tel le cadre juridique (voir le passionnant ouvrage [26]).

Plus généralement, il est bien connu en théorie de la décision que des systèmes de postulats de *rationalité* qui pouvaient sembler raisonnables pour décrire le comportement d'un décideur, ont rencontré des situations où l'application du critère de décision qui en découlait ne correspondait pas

1. Comme le personnage de Véronique dans le sketch « Télémagouilles » des Inconnus.

à la conduite de la majorité des décideurs [15].

Evidemment, si les machines dépassent largement les humains quand il s'agit de faire des calculs, leur intelligence est beaucoup moins polymorphe : elles ne savent rien faire d'autre ! Si intelligence il y a, c'est surtout celle des humains qui les conçoivent et les programment qu'il faut mettre en avant. En particulier, une calculette n'a pas plus conscience des opérations qu'elle réalise et de ce qu'est un nombre, qu'une horloge comtoise n'a du temps qui s'égrène, ou qu'un programme de reconnaissance faciale (voir à ce sujet la section 3.3) n'a d'un visage et de ce qui le constitue.

On pourra enfin garder en tête que les humains peuvent parfaitement percevoir comme bêtes des processus de raisonnement qui ne le sont pas du tout mais au contraire collent parfaitement à la norme universelle quand elle existe. Ainsi, la validité d'un raisonnement ne dépend que de la capacité de sa conclusion à être vraie dans tous les cas de figure où ses prémisses le sont, et absolument pas de la vérité des prémisses et conclusion en présence. En particulier, quand les prémisses sont fausses, toutes les conclusions s'ensuivent. Pour reprendre un exemple célèbre et paraphraser le grand logicien Bertrand Russell : « Si $2+2 = 5$ alors je suis le Pape. » En effet, on sait que l'on peut retirer des deux membres d'une équation comme $2 + 2 = 5$ la même quantité, disons 3, en préservant la validité de l'équation. Ainsi, de l'hypothèse $2+2 = 5$, on déduit $1 = 2$. Or, le Pape et moi sommes deux. Mais si $1 = 2$, alors nous sommes la même personne : je suis le Pape. Ce raisonnement est valide, il respecte les canons de la logique classique et pour autant, il apparaît comme dénué de sens (en particulier, parce que ses prémisses et sa conclusion n'ont aucun lien).

3 La bêtise est la chose du monde la mieux partagée [2]

3.1 Un bel exemple de bêtise humaine

Une histoire qui circule concerne un problème de mathématiques simple qu'un étudiant aurait résolu de façon surprenante, créative, mais fausse (une belle bêtise). Le problème (un peu reformulé) était celui du calcul de

$$\lim_{x \to 0} \frac{1}{5x^2}$$

Il semblerait que l'étudiant se soit rappelé de la solution d'un exercice similaire :

$$\lim_{x \to 0} \frac{1}{8x^2} = +\infty$$

Afin de laisser à la lectrice / au lecteur la possibilité d'imaginer la solution de l'étudiant, nous l'avons mise en fin d'article, en annexe A : la suite de la section fait l'hypothèse que vous êtes allé(e) lire cette annexe.

Maintenant, on peut imaginer ce que cela supposerait comme compétences pour une IA de générer des réponses telles que celle-là (une classe de bêtises, plutôt qu'une bêtise individuelle qu'il est toujours possible d'afficher simplement) et de façon involontaire (i.e., un système d'IA qui ne soit pas conçu pour générer des bêtises, mais qui en génère malgré le concepteur). On imagine qu'un système capable de générer la bêtise de l'étudiant doit à la fois travailler sur une représentation d'expressions mathématiques (par une structure arborescente, par exemple) et sur une représentation visuelle, avec des liens entre ces représentations. De notre point de vue, la construction d'une telle IA n'aurait rien de triviale !

3.2 Des IA bêtes

Un programme d'IA – une IA – ne fait que ce pourquoi il – elle – est programmée. On conçoit facilement qu'un programme sophistiqué puisse faire des choses qu'on jugera remarquables, en matière de résolution de problèmes, ou de reconnaissance de formes, faisant par là montre d'« intelligence » (artificielle). Des programmes très simples peuvent cependant donner le change. C'est le cas en particulier du programme ELIZA [29] qui dès le milieu des années 1960, simulait un entretien avec un thérapeute en reformulant ce que disait le « patient » sous forme de questions et relançait le dialogue avec des phrases toutes faites faisant écho à ce que venait de dire le patient. Il est clair que quand ELIZA déclarait « Je comprends », faute d'être capable d'une réponse plus adaptée au contexte du dialogue, elle bluffait totalement l'interlocuteur, alors qu'il s'agissait d'un système des plus bêtes.

Cette question des IA qui peuvent sembler « réalistes » à leurs utilisateurs, tout en s'appuyant sur des techniques très simples (voire simplistes) s'est longtemps retrouvée dans de nombreux jeux vidéos. Comme précisé dans [31, Section 1.2.2], dans les premiers temps, l'IA des jeux vidéos était généralement fondée sur des scripts pré-établis. Cela limite grandement les comportement des personnages non joueurs (PNJ, c'est-à-dire les personnages du jeu contrôlé par la machine). Le résultat est que, malgré les progrès importants réalisés par l'IA dans le domaine des jeux vidéos, il y a encore aujourd'hui de nombreux joueurs se plaignant de la bêtise des PNJ [3], y compris parmi les joueurs professionnels qui identifient les phases scriptées dans le déroulement du jeu [4]. Une conséquence à cela est qu'il est possible, pour un joueur suffisamment expérimenté, d'anticiper le comportement à venir de l'IA.

2. Ce qui n'empêche pas le bon sens de l'être également.

3. https://tinyurl.com/52mnzf85

4. https://tinyurl.com/yckpy784

3.3 Des IA qui disent des bêtises (ou sont utilisées bêtement)

Le cas Dong Mingzhu, ou l'IA bête et méchante. Dong Mingzhu est une femme d'affaires chinoise, à la tête d'une entreprise importante dans le domaine de la climatisation. Pour promouvoir son entreprise, elle a réalisé une campagne de publicité. Son portrait s'est ainsi retrouvé affiché dans les transports publics, sur des bus urbains. Or, en République Populaire de Chine, la vidéo-surveillance s'est généralisée. Une multitude de caméras et des algorithmes d'IA de reconnaissance de visages permettent à la puissance publique de détecter celles et ceux de leurs citoyens qui commettent des délits ou simplement des incivilités, comme traverser la chaussée quand c'est aux véhicules de passer. Un système de crédit social est en place et quand une incivilité est repérée, la personne l'ayant commise se voit retirer des points (et à un moment, perd par exemple la possibilité d'utiliser les transports publics). Dong Mingzhu a très vite perdu ses points car chaque fois qu'un bus affichant son portait passait devant une caméra de surveillance, le système d'IA en place concluait, à tort, que Dong Mingzhu avait traversé la route alors qu'elle n'en avait pas le droit. Ce scénario illustre bien à la fois la très grande qualité des algorithmes de reconnaissance faciale utilisés (ils ne sont pas intrinsèquement bêtes) mais aussi l'immense bêtise produite ici par le système d'IA utilisé, due à l'absence totale de contextualisation de la prise de décision.

Le traitement automatique de la langue (qui fourche). Pouvoir réaliser automatiquement (par algorithme) et de façon satisfaisante des tâches de traitement de la langue, comme les dialogues à base d'agents conversationnels animés (ou autres *bots*), le résumé ou la traduction d'une langue à une autre requiert une aptitude à comprendre, qui n'est pas acquise par les systèmes d'IA qui restent dénués de connaissances de bon sens sur le monde dans lequel on vit. Les outils de traduction automatique disponibles en ligne et que vous connaissez tous sont de formidables outils (les meilleurs disponibles) pour réaliser des traductions « superficielles ». Ils ne sont pas du tout bêtes mais pour autant, ce sont des générateurs de bêtises dont on peut s'amuser.

La résolution d'anaphores (en particulier, la satisfaction des références pronominales) est un problème difficile qui, pour pouvoir être résolu correctement, demande de disposer (et de savoir raisonner sur) des connaissances du monde. Ce problème est à la base des schémas de Winograd [30], qui peuvent être employés comme variantes au test de Turing.

Essayez par exemple d'utiliser un outil de traduction automatique disponible en ligne pour traduire de l'anglais vers le français la phrase suivante :

`"An AI algorithm cannot produce such a sillyness since` **`it`** `is too comic."`

La traduction résultante (via un outil bien connu et largement utilisé) est :

« `Un algorithme d'IA ne peut pas produire une telle bêtise car` **`il`** `est trop comique.` »

Ici, l'erreur est sur la référence du pronom « *it* ». Au départ, c'est la bêtise qui se veut trop comique, au final c'est l'algorithme d'IA qui l'est. Ce qui n'est pas tout à fait faux s'il s'agit de l'algorithme utilisé pour réaliser cette traduction.

Voici un autre exemple. Essayez de traduire en anglais la phrase suivante (qui présente un argument dont la justification est assez discutable) avec un outil de traduction automatique :

`La bêtise précède souvent l'intelligence. C'est normal, « bêtise » commence par un « b » et « intelligence » par un « i ».`

Une fois traduite en anglais, cela donne :

`Stupidity often precedes intelligence. This is normal, "stupidity" begins with a "b" and "intelligence" with an "i".`

La justification de l'argument traduit est encore plus discutable... L'algorithme d'IA ne sépare pas ici les lectures *de re* et *de dicto*, en dépit des guillemets utilisés.

Les outils de traduction automatique fournissent aussi des résultats amusants quand ils s'essaient à corriger nos erreurs... en en produisant à la place. Ainsi, la suite de mots

`Veille technologique.`

qui, une fois traduite automatiquement en anglais, devient :

`Old technology.`

Statistiquement, les mots « vieille » et « technologie » devraient apparaître plus souvent ensemble que les mots « veille » et « technologique », d'où l'erreur produite. Elle est cocasse car typiquement, quand on se lance dans une « veille technologique », c'est parce qu'on est en quête d'une technologie qui serait tellement récente qu'on ne la connaît pas encore, et donc pas d'une technologie qui serait désuète.

Bien entendu, la traduction est une tâche ardue ; il n'y a en règle générale, pour un texte un peu complexe, pas de traduction idéale, parfaite et ainsi les bêtises qui sont produites ne reflètent pas une distance trop importante par rapport à une norme qui serait universelle [5], mais plutôt un écart à ce qui est généralement attendu. Parmi les difficultés rencontrées figurent les ambiguïtés inhérentes au langage naturel qui peuvent se manifester dans la syntaxe même des phrases. Ainsi, la phrase (un classique du genre) « La petite brise la glace » est ambiguë puisque sa signification n'est pas du tout la même selon que l'on considère que le

5. Le problème de l'absence de norme universelle se pose, bien entendu, aussi pour les systèmes de reconnaissance faciale. Il n'existe pas une image de référence du visage de Dong Mingzhu (ou de n'importe qui d'autre).

verbe est « brise » ou que le verbe est « glace ». Prise en dehors d'un contexte qui permettrait de supprimer une des deux hypothèses possibles, cette phrase n'est pas traduisible puisque son propre sens dans sa langue d'origine n'est pas déterminé. Il est amusant d'observer comment les outils de traduction automatique s'en sortent avec ce type de phrases. Pour celle-ci, le résultat est surprenant :

`The little icebreaker`

qui signifie littéralement « Le petit brise-glace ». On se retrouve immédiatement transportés en hiver sur les bords de la Volga !

On pourrait multiplier les exemples. Les algorithmes d'IA ne comprenant pas les textes qu'ils traitent, le sens caché éventuel de ces textes reste inaccessible, même quand leur prise en compte est indispensable, comme dans les jeux de mots, calembours ou autres contrepèteries. Ainsi, « l'art de décaler les sons » une fois traduite en anglais devient "*the art of shifting sounds*", perdant au passage son statut de contrepèterie. La phrase devient clairement moins drôle ! Les chansons à double écoute, comme « Mon père et ses verres » de Boby Lapointe ou « La jeune fille du métro » de Colette Renard sont évidemment hors de portée d'une traduction automatique qui préserverait la double écoute possible.

4 Pourquoi construire un système de BA ?

La construction effective d'un système de BA peut être vue comme un exercice assez vain. Néanmoins, cette section réunit quelques raisons pour lesquelles on pourrait vouloir se lancer dans une telle aventure.

4.1 Dans l'optique de la réussite du test de Turing

Comme on l'a vu, le test de Turing vise à vérifier qu'un système automatique pourrait tromper des humains en leur faisant croire qu'il est humain lui-même, sous certaines conditions d'interaction [28]. Si un tel système répond trop intelligemment à une question, il risque d'échouer à ce test. Pour reprendre un exemple de la section 2.3, si un interlocuteur-machine répondant à la question « $12345 \times 67890 = ?$ » très rapidement et de façon correcte, il devrait échouer dans sa tromperie. Une réponse suffisamment éloignée du résultat sans être complètement absurde pourrait être une bêtise qui tromperait l'examinateur humain [6].

Ainsi, avec le but de la tromperie qu'induit le test de Turing, si l'on veut faire un système informatique réalisant ce test, celui-ci devrait relever de l'IA *et* de la BA. Et si on accepte le test de Turing comme une caractérisation de l'intelligence, cela conduit à accepter la bêtise comme une part de l'intelligence.

6. Une autre réponse de la machine pourrait être : « Tu ne serais pas un ordinateur qui essaie de passer son test de Turing, toi, des fois ? »

4.2 Bêtises délibérées

On pourrait imaginer un système d'IA qui produirait des bêtises de manière délibérée, peut-être dans l'esprit de [17]. On peut imaginer au moins deux raisons à cela :

— pour détendre l'atmosphère, pour faire rire l'interlocuteur, s'il s'agit d'un robot de compagnie.
— pour tromper, ou pour décontenancer un interlocuteur qui, par exemple, s'énerverait.

Il s'agit évidemment de défis très durs pour l'IA ! La compréhension et la génération automatique d'histoires drôles sont des problèmes très compliqués [23, 10].

4.3 La bêtise créative

Si séparer le vrai du faux est le Graal de la démarche scientifique et est indispensable « pour marcher avec assurance en cette vie » (pour paraphraser René Descartes), il ne faut pas avoir peur de se tromper, de faire des bêtises. La bêtise est féconde. C'est en en faisant que l'on apprend.

L'erreur est, en effet, source de création. L'Histoire regorge de scénarios où des maladresses ont conduit à des réussites, que l'on pense aux bêtises de Cambrai, à la tarte Tatin, ou pour quitter le domaine culinaire tout en restant dans la sérendipité, à la découverte du procédé de la vulcanisation par Charles Goodyear au XIXe siècle. Bien sûr, pour qu'une erreur soit fructueuse, il faut ensuite de l'intelligence pour reconnaître l'existence d'un potentiel à exploiter dans la situation créée par l'erreur. Ainsi, une erreur dans une preuve mathématique se révélera féconde si sa découverte peut conduire à une approche nouvelle du problème.

Chez l'humain, depuis la petite enfance, l'apprentissage passe ainsi par des phases d'essais et erreurs, par des généralisations abusives que l'éducation, l'enseignement, la raison permettent (parfois) de corriger. L'apprentissage par renforcement implémente cette idée en IA : elle passe par l'exploration de situations qui sont jugées insatisfaisantes ou non. Observons que pour être créatif, un apprentissage par renforcement devrait sans doute autoriser une certaine permissivité dans l'estimation du caractère insatisfaisant d'une situation.

Il en va de même chez la machine : la généralisation inductive, tout comme l'abduction ou l'analogie ou encore d'autres formes de raisonnement de sens commun, ne préserve pas la vérité. Un apprentissage, quand il ne se limite pas à un apprentissage par cœur (qui, lui, n'offre pas la capacité de prendre en compte des situations nouvelles) présuppose le risque de se tromper. Les algorithmes mis en place dans les machines pour apprendre visent à minimiser ce risque de façon empirique, en s'appuyant sur les situations disponibles. Ils ne peuvent pas l'éliminer totalement. Chez la machine, il faut noter que l'erreur commise peut être importante quand les données sont trop peu massives ou de mauvaise qualité. C'est la question des biais dans

les données utilisées pour apprendre, qui peuvent conduire des IA à reproduire des comportements répréhensibles [7] ou à générer des prédictions qui ne sont rien d'autres que des prophéties auto-réalisatrices, comme dans le cas du prédicteur *PredPol*, utilisé par le département de police de Los Angeles pour déterminer les zones à patrouiller ; évidemment, le LAPD pouvait observer plus de crimes, délits et infractions là où *PredPol* l'envoyait patrouiller que là où elle n'était pas (et n'était donc pas en situation de les observer) !

Pour que la bêtise soit créative, chez l'humain comme chez la machine, le risque de se tromper doit être accepté, mais il doit aussi être modulé et contrôlé par la mise en œuvre de mécanismes permettant de remettre en cause ses croyances et les décisions auxquelles elles conduisent. C'est indispensable pour éviter le *perseverare diabolicum* déjà mentionné et s'efforcer ainsi de ne pas rester bêtes. Ainsi, de nombreux travaux en IA se sont-ils tournés depuis une quarantaine d'années vers cette problématique à facettes multiples, incluant le raisonnement non monotone (où les conclusions changent selon les hypothèses faites, qu'elles soient implicites ou pas) ou encore la révision de croyances (grosso modo, comment remettre en cause ses croyances pour en changer suffisamment sans en changer trop ?).

5 Concevoir un système de BA

Supposons que l'on veuille construire un système de BA. Pour ce faire, nous considérons d'une part les systèmes d'intelligence/bêtise artificielle fondées sur des raisonnements et d'autre part des systèmes d'[I/B]A fondés sur l'apprentissage supervisé, étant bien entendu que cela ne couvre pas toute l'IA ni toute la BA, et que cela ne constitue pas des ensembles de systèmes disjoints : en particulier, le raisonnement à partir de cas (RàPC), peut être considéré à l'intersection des deux. Le RàPC et l'argumentation sont d'ailleurs les deux problématiques plus spécifiques étudiées en fin de section.

5.1 Raisonner bêtement

Commettre des erreurs de raisonnement, c'est sans doute raisonner bêtement (même si c'est à la portée de tout le monde). Le raisonnement déductif repose largement sur le schéma suivant $\frac{p \to q \quad p}{q}$ (si p est vrai, et si p implique q, alors q est vrai). Autoriser d'autres schémas, comme par exemple $\frac{p \to q \quad q}{p}$ ou $\frac{p \to q \quad p}{\neg q}$ est plus risqué ! Selon le premier, qui n'offre pas les garanties de la déduction, l'observation d'une conséquence de p suggère que p est vrai, ce qui n'est pas bête ! Tandis que le second schéma mène à coup sûr à l'incohérence. On pourra consulter [5] pour une étude des arguments fallacieux. Une mauvaise analogie, reposant sur un parallèle trop hasardeux, mène de même à l'absurdité : « Le hérisson est comme la brosse à dents, il a le poil raide. »

Utiliser improprement la déduction conduit aussi à des absurdités :

— *Les appartements bon marché sont rares.*
— *Tout ce qui est rare est cher.*
— *Donc les appartements bon marché sont chers.*

Le problème provient ici de ce que le second énoncé a des exceptions (comme l'indique d'ailleurs le premier), alors qu'il est traité comme s'il n'en avait pas. La vraie bêtise peut alors être de ne pas démordre d'une conclusion fausse ou absurde.

Si un système de BA veut paraître bête, produire des conclusions fausses ne suffit pas ! Il doit rendre perceptible la manière dont il les obtient. Le comble du raisonnement bête est peut-être d'arriver à une conclusion exacte par des inférences fausses... à moins que cela ne soit un signe de virtuosité intellectuelle !

De façon similaire, un processus de raisonnement parfaitement correct pourra être perçu comme bête simplement parce qu'il n'est pas astucieux, qu'il est trop complexe, qu'il passe par des calculs compliqués alors qu'il existe des approches beaucoup plus simples (à titre d'exemple, on pensera au fameux problème des cyclistes et de la mouche, cf. `https://membres-ljk.imag.fr/Bernard.Ycart/mel/sn/node16.html`).

Enfin, si la bêtise est d'abord, comme il a été dit, une incapacité à voir, à créer des liens entre des faits ou des énoncés de ces faits, alors, on peut utiliser des idées de systèmes à capacité de raisonnement limitée [11] pour concevoir un système de BA. Par exemple, le système peut être « conscient » que p et $p \to q$ sont vrais, sans pouvoir former leur conjonction et conclure q [22].

5.2 Apprendre des bêtises

Un système d'apprentissage supervisé en IA s'appuie sur des exemples sous la forme de couples $(\mathtt{x}, \mathtt{y}) \in \mathcal{P} \times \mathcal{S}$ où $\mathcal{P}$ est l'espace des problèmes (ou des entrées) et $\mathcal{S}$ est l'espace des solutions (ou des sorties), tel que $\mathtt{x}$ a pour solution $\mathtt{y}$. Pour simplifier la discussion qui suit, on supposera que la relation « a pour solution » est fonctionnelle et on notera $\mathtt{F}$ cette fonction : les exemples non bruités $(\mathtt{x}, \mathtt{y})$ d'un système d'IA d'apprentissage supervisé vérifient $\mathtt{y} = \mathtt{F}(\mathtt{x})$. L'objectif d'un tel système est donc d'approcher la fonction $\mathtt{F}$: si on note $\mathtt{EA}$ l'ensemble d'apprentissage et $\mathtt{F}_{\mathtt{EA}}$ une fonction apprise par un système d'apprentissage supervisé donné, l'erreur sera mesurée par une estimation de la différence entre les fonctions $\mathtt{F}$ et $\mathtt{F}_{\mathtt{EA}}$.

Pour construire un système d'apprentissage supervisé en BA, une idée naturelle est d'appliquer une technique d'ap-

7. Voir par exemple `https://tinyurl.com/yvaj4beh`.

prentissage supervisée d'IA qui s'appuie sur des exemples de bêtises. La question à laquelle il faut répondre alors est celle de la constitution de $\mathtt{EAB}$ (l'ensemble d'apprentissage bête). Partant de l'idée qu'une condition nécessaire pour que $(\mathtt{x}, \mathtt{y})$ soit un exemple de bêtise est que $(\mathtt{x}, \mathtt{y})$ soit une erreur, il faut donc que $\mathtt{y} \neq \mathtt{F}(\mathtt{x})$. Cette condition nécessaire est-elle suffisante pour faire un apprentissage bête ? Si non, quelle autre condition serait nécessaire ? Voilà deux questions auxquelles nous n'apporterons pas de réponse ici : nous nous contenterons de la condition nécessaire et allons examiner ce qu'implique le fait de la considérer comme suffisante dans quelques cadres.

Considérons dans un premier temps la classification binaire (i.e., $|\mathcal{S}| = 2$) et notons $\overline{\mathtt{y}}$ l'élément de $\mathcal{S}$ différent de $\mathtt{y} \in \mathcal{S}$. On peut donc constituer $\mathtt{EAB}$ en partant d'un ensemble d'apprentissage $\mathtt{EA}$ (destiné à une IA) : $\mathtt{EAB}$ serait alors l'ensemble des $(\mathtt{x}, \overline{\mathtt{y}})$ où $(\mathtt{x}, \mathtt{y}) \in \mathtt{EA}$. L'apprentissage mènerait alors à une fonction $\mathtt{F}_{\mathtt{EAB}}$ qui, si l'apprentissage est réussi, se tromperait la plupart du temps, ce qui signifie que la fonction $\overline{\mathtt{F}_{\mathtt{EAB}}} : \mathtt{x} \in \mathcal{P} \mapsto \overline{\mathtt{F}_{\mathtt{EAB}}(\mathtt{x})} \in \mathcal{S}$ donnerait une fonction d'apprentissage correct[8] : demander l'avis aux personnes qui sont bêtes dans ce sens-là est donc utile pour savoir ce qu'on ne devrait pas faire ! Par exemple, si on a un système de classification d'images permettant de distinguer les photos de rats-taupes glabres des photos de bicyclettes, un classifieur binaire efficace construit sur un ensemble d'apprentissage bête (au sens précédemment décrit), il prendra, la plupart du temps, une photo de rat-taupe glabre, pour celle d'un vélo et inversement.

Quand la taille de $\mathcal{S}$ augmente, les choses peuvent devenir différentes. Prenons le cas de la régression où $\mathcal{P} = \mathbb{R}^m$ et $\mathcal{S} = \mathbb{R}^n$, étant donné $(\mathtt{x}, \mathtt{y}) \in \mathtt{EA}$, comment choisir $(\mathtt{x}, \mathtt{y}') \in \mathtt{EAB}$ avec la seule condition $\mathtt{y}' \neq \mathtt{y}$? Si c'est par une bijection $\varphi : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, on se ramène à une situation similaire à la précédente ($\varphi^{-1} \circ \mathtt{F}_{\mathtt{EAB}}$ pourrait être une aussi bonne approximation de $\mathtt{F}$ que $\mathtt{F}_{\mathtt{EA}}$). Si le calcul de $\mathtt{y}'$ se fait de façon aléatoire (et sans tenir compte de $\mathtt{y}$), alors $\mathtt{F}_{\mathtt{EAB}}$ ne serait représentatif que de la distribution de probabilité du générateur aléatoire sur $\mathcal{S}$: on retombe sur l'hypothèse rollinesque d'un système de BA qui répond n'importe comment à la question posée (cf. introduction). Si le calcul de $\mathtt{y}'$ se fait de façon aléatoire mais proche de $\mathtt{y}$ (p. ex., pour $n = 1$, $\mathtt{y}'$ choisi aléatoirement dans $[\mathtt{y} - C, \mathtt{y} + C] \setminus \{\mathtt{y}\}$, où $C > 0$ est une constante relativement « petite »), on se retrouve dans le cadre de l'apprentissage avec des exemples bruités, avec un bruit additif.

Ces quelques exemples abstraits illustrent le fait que la condition nécessaire « être bête, c'est se tromper » n'est pas toujours suffisante pour faire un système d'apprentissage en BA qui ne soit ni un système d'apprentissage en IA « en creux » ni un système qui se contente de répondre n'importe comment.

8. Sous certaines hypothèses de symétrie entre l'appréhension des deux classes par le classifieur, il devrait être possible de montrer que $\overline{\mathtt{F}_{\mathtt{EAB}}}$ et $\mathtt{F}_{\mathtt{EA}}$ ont la même espérance d'erreur.

5.3 Le RàPB

Le RàPB serait le raisonnement à partir de cas (RàPC) produisant des bêtises. On rappelle que le RàPC consiste à (tenter de) résoudre un problème cible en s'appuyant sur une base de cas où un cas est une représentation d'un épisode de résolution de problèmes : un cas est souvent la donnée d'un couple $(\mathtt{x}, \mathtt{y})$ (i.e., ce qu'on appelle un exemple en apprentissage supervisé), parfois agrémenté par des explications sur des liens entre $\mathtt{x}$ et $\mathtt{y}$. Une session de RàPC consiste souvent en la sélection dans la base de cas d'un cas jugé similaire au problème cible (étape de remémoration) et en la modification de ce cas dans l'optique de la résolution du problème cible (étape d'adaptation).

Pour construire un système de RàPB, on pourrait s'inspirer du principe de l'apprentissage de bêtises évoqué ci-dessus, à savoir la constitution d'une base de cas-bêtises. Cela rencontre la problématique récente du RàPC utilisant des cas positifs et des cas négatifs (ces derniers peuvent être rapprochés — voire assimilés ? — à des cas-bêtises) et qui a donné de bons résultats pour l'apprentissage de connaissances d'adaptation [18], ce qui tend à dire que si on mémorise à la fois les expériences positives et les expériences de bêtises, en distinguant les deux, cela peut être utile pour mieux adapter en RàPC.

Une autre voie, éventuellement complémentaire, pour le RàPB, consisterait à changer les étapes du raisonnement (comme c'est le cas dans le cadre général de la section 5.1), par exemple en sélectionnant le cas le moins similaire au problème cible : si l'adaptation parvient à modifier le cas-recette du gratin dauphinois afin de résoudre le problème « J'aimerais une recette de sorbet au cassis. », le résultat pourrait être surprenant (et utile quand on a des invités qui s'attardent trop). On peut aussi imaginer une remémoration pertinente suivie d'une adaptation qui ne l'est pas : l'exemple de la section 3.1 illustre cela (pour un système de RàPC ou de RàPB résolvant des problèmes de calcul de limites).

On notera que ces idées rejoignent celle présentées par François Rollin d'un système de BA qui s'abstiendrait de trier (ici : de faire une remémoration bête) ou d'adapter (ici : de faire une adaptation bête).

5.4 Bêtise argumentée

En IA, l'argumentation formelle est le domaine qui s'intéresse à la représentation d'informations conflictuelles, et à la définition de méthodes de raisonnement non triviales à partir de ces informations, en se basant sur des notions de logique et de rhétorique : un argument est composé d'un ensemble d'informations considéré comme fiables (le support) et d'une information qui peut être déduite de ce support (la conclusion) ; si un argument a contredit un argu-

ment b, mais que rien ne vient contredire a, alors on considère que a est acceptable et que b ne l'est pas. Différents principes gouvernent les modèles de raisonnement argumentatif, il semble donc relativement aisé de définir un système de bêtise argumentée (BArg), via un système d'argumentation qui faillirait à satisfaire certains de ces principes de base.

Prenons d'abord le cas de l'argumentation abstraite [9]. Dans ce contexte, on s'intéresse aux relations entre les arguments pour déterminer lesquels sont acceptables, sans se soucier de leur nature précise (en particulier, de leur structure logique interne). Un système d'argumentation abstrait est donc un graphe $\mathtt{F} = \langle \mathtt{A}, \mathtt{R} \rangle$, dont les noeuds $\mathtt{A}$ sont les arguments, et les arcs $\mathtt{R} \subseteq \mathtt{A} \times \mathtt{A}$ sont les attaques, c'est-à-dire la représentation de la notion de « contre-argument ». Le raisonnement à partir d'un tel graphe se fait au moyen du concept d'extension, c'est-à-dire d'ensemble d'arguments conjointement acceptable, représentant une solution potentielle du problème représenté par le graphe argumentatif. Il existe différentes façon de définir une extension, selon les propriétés attendues de cet ensemble d'arguments. Parmi les approches classiques, nous parlerons uniquement (à titre illustratif) de la sémantique stable : un ensemble d'arguments $\mathtt{E} \subseteq \mathtt{A}$ est une extension stable si et seulement si 1) c'est un ensemble sans conflit ($\forall a, b \in \mathtt{E}$, $(a, b) \notin \mathtt{R}$), et 2) c'est un ensemble qui attaque son complémentaire ($\forall a \in \mathtt{A} \setminus \mathtt{E}$, $\exists b \in \mathtt{E}$ tel que $(b, a) \in \mathtt{R}$). Il est donc possible de définir un système de BArg comme un système qui retourne des ensembles d'arguments qui attaquent leur complémentaire, mais ne satisfont pas le principe d'absence de conflit. Prenons l'exemple d'un scénario très simple, où le dialogue suivant [9] :

– *C'est Charles Lytton qui a volé la pierre précieuse !* (a)
– *C'est impossible, il a une jambe dans le plâtre.* (b)
– *Il feint probablement d'être blessé.* (c)

est modélisé par un système d'argumentation $\mathtt{F} = \langle \mathtt{A}, \mathtt{R} \rangle$, où $\mathtt{A} = \{a, b, c\}$ et $\mathtt{R} = \{(c, b), (b, a)\}$. Alors que la sémantique stable requiert d'accepter $\{a, c\}$, une version bête peut mener à l'acceptabilité de $\{b, c\}$ (l'ensemble attaque bien son complémentaire, mais n'est pas sans conflit). Des versions bêtes des autres sémantiques de raisonnement argumentatif peuvent être définies de façon similaire, en choisissant des ensembles d'arguments qui ne satisfont qu'une partie des propriétés requises. Notons tout de même que dans certains contextes, comme l'existence d'informations au sujet de priorités à appliquer entre les arguments, certaines propriétés de base comme l'absence de conflit peuvent être violées [1, 2, 25]. Cela veut-il dire que ces cadres étendus sont bêtes ? Nous laissons le lecteur en juger.

Si on ne s'intéresse pas uniquement aux relations entre les arguments, mais également à la façon de les former à partir de connaissances structurées (généralement avec un formalisme logique [4]), il existe également plusieurs façons d'argumenter bêtement. En résumé, il y a deux étapes dans la conception d'un système d'argumentation structuré à partir de connaissances logiques : l'identification des arguments, et l'identification des attaques entre eux. Ces deux étapes peuvent être sources d'importantes bêtises. Supposons qu'on ait affaire à un individu énonçant la phrase suivante : « J'aime le chocolat, donc je suis la reine d'Angleterre. ». Cela revient à considérer $(\{a\}, b)$ (où a représente le fait d'aimer le chocolat, et b le fait d'être la reine d'Angleterre) comme un argument, alors qu'il n'y a aucune raison rationnelle de supposer que b soit une conséquence de a. S'il est déjà un peu bête de prétendre au trône britannique en raison de son amour du chocolat, ce genre de raisonnement peut naturellement poser d'autres problèmes si on applique l'argumentation à des sujets plus sensibles. Même dans un cas où les arguments sont bien construits, une piste pour concevoir un système de BArg se situe au niveau de l'étape d'identification des attaques. Ignorer certaines attaques, ou au contraire en ajouter certaines qui ne devraient pas l'être, va naturellement changer l'issue du raisonnement.

On peut raisonnablement [10] se demander si un système de BArg « sans limite » ne ressemblerait pas à un raisonnement chaotique, sans intérêt particulier. Comme on l'a mentionné en introduction, la bêtise d'un résultat peut se définir en fonction d'une distance par rapport au résultat correct attendu. Un « bon » système de BArg serait alors un système qui identifie incorrectement un ratio pré-défini des arguments et des attaques, et qui sélectionne les arguments acceptables en maintenant également une certaine proximité avec une solution correcte.

5.5 D'autres BA

Chaque domaine et problématique de l'IA pourrait être examiné sous le prisme de la BA. Voici un court inventaire non exhaustif de tels exercices.

Pour la prise de décision, une question serait : « Comment décider bêtement ? » Par exemple, en décision multi-critères, on peut décider finalement à partir d'un critère non pertinent : acheter une voiture parce qu'elle est rouge alors qu'on avait décidé de l'acheter sur des critères écologiques, de prix et de performance, peut apparaître comme bête.

Pour l'apprentissage non supervisé, une question serait : « Comment catégoriser bêtement un domaine ? » À titre d'exemple, on peut séparer les humains en deux catégories, en fonction de la parité du nombre de voyelles dans le prénom de leurs grands-mères paternelles.

Dans le cadre des systèmes multi-agents, une question serait : « Comment faire émerger de la bêtise d'un groupe d'agents individuellement pas bêtes ? » Dans un

9. Toute ressemblance avec une comédie policière des années 1960 est absolument non fortuite.

10. Est-ce vraiment si raisonnable ?

tel contexte, on notera au passage l'identification de cinq lois fondamentales de la stupidité humaine, proposée par Carlo M. Cipolla [6]. Cette théorie pose, en particulier (la troisième loi énoncée par l'auteur), qu'un agent (humain) stupide est un agent qui nuit à un autre (ou à un groupe d'autres agents) sans en tirer aucun bénéfice, voire en étant nuisible à lui-même. En quelque sorte, tirer sur quelqu'un d'autre en se mettant une balle dans le pied ! Ces lois ont donné lieu à des expérimentations visant à estimer la validité de la théorie de Cipolla dans des contextes de simulation à base d'agents artificiels [27], mais aussi à des travaux visant à fournir une interprétation biophysique à la théorie [21].

6 Conclusion provisoire

Cet article est né d'une réflexion autour de la notion de BA qu'a introduite François Rollin dans un ouvrage récent. L'objectif poursuivi ici est de réunir quelques réflexions sur la question de la bêtise vue du point de vue de l'IA. L'article ne prétend pas couvrir toutes les problématiques que cette notion de bêtise artificielle suscite, mais pose les questions suivantes sur les liens entre ces notions encore mal définies d'intelligence [artificielle] et de bêtise [artificielle] : la seconde peut-elle être vue comme le complémentaire de la première ? ou, à l'inverse, considère-t-on qu'il ne peut y avoir de bêtise [artificielle] sans (un peu d')intelligence [artificielle] ? Une autre question ou, peut-être, une autre façon de poser cette question est celle de ce que serait un système de BA : un système d'IA quand il se trompe ? ou un système destiné à proposer des bêtises ? La deuxième réponse pourrait poser problème si l'on comprend la bêtise comme étant involontaire, mais on peut contourner cette difficulté en affirmant que la volonté (que le système soit bête) est celle du concepteur de ce système, alors que le système n'aurait pas plus conscience de dire des bêtises qu'un système d'IA n'a (jusqu'à preuve du contraire) de conscience d'être (parfois) intelligent. Enfin, nous ne nous sentons pas exemptés d'être bêtes par le simple fait d'avoir écrit cet article (qui contient au moins 7 bêtises : saurez-vous les découvrir ?), mais nous nous consolons en appréciant le charme, sinon la beauté de la bêtise.

Ce travail offre plusieurs perspectives d'études, en partie évoquées dans l'article. Une d'entre elles est la tentative de modélisation formelle de la bêtise. La bêtise ne se réduit pas à l'erreur, elle doit être considérée comme un écart vis-à-vis d'une norme qu'il conviendra de caractériser. Cette norme peut être celle d'un observateur jugeant s'il y a bêtise. Elle peut également être une estimation des conséquences de la bêtise (par exemple par une fonction d'utilité).

Cet article pose plus de problèmes qu'il n'en résout (en résout-il un seul ?). En fait, on peut le voir comme une invitation à la communauté de l'IA de considérer ce champ de recherche sous un angle inhabituel, avec l'idée que ce pas de côté pourrait se révéler déambulatoire [11]. Et n'oublions pas Gustave Flaubert qui a écrit « Oui, la bêtise consiste à vouloir conclure » (Lettre à Louis Bouilhet, 4 septembre 1850). Sachons, en toutes choses, nous garder des conclusions définitives !

Toutes les sections de cet article, à l'exception de la dernière, ont été rédigées par ses quatre premiers auteurs (qui ont néanmoins cité le cinquième en introduction). La dernière section est rédigée par le cinquième auteur.

7 Le mot de François Rollin

Des réflexions et questionnements exposés ci-dessus, je crains de devoir déduire qu'il est encore plus difficile de produire de la Bêtise Artificielle que de produire de l'Intelligence Artificielle, dès lors que l'objectif est de générer de l'authentique bêtise, de la bonne vraie bêtise bien de chez nous, et non pas simplement de l'absurdité ou du non-sens.

Cette conclusion serait contre-intuitive si on voyait la bêtise comme un simple ratage. De fait, il est plus facile de rater que de réussir : en cuisine, en musique, en littérature, en médecine, au jeu d'échecs, en amour, et j'en oublie, il faut une vraie compétence pour bien faire, tandis qu'une simple incompétence suffit à mal faire. En matière d'intelligence, il semble que ce soit le contraire, du moins si on veut produire une bonne vraie bêtise.

Cette surprenante difficulté ne peut qu'appuyer la nécessité et l'urgence de travaux savants et studieux sur la BA. L'aventure a démarré, c'est une bonne chose, il faut maintenant la mener à son terme, et ce n'est pas à moi que la tâche incombe : je suis bien trop bête pour ça.

A Réponse à l'exercice de la section 3.1

La solution proposée par l'étudiant était $+\backsim$.

Références

[1] Amgoud, L. et C. Cayrol: *A Reasoning Model Based on the Production of Acceptable Arguments*. Ann. Math. Artif. Intell., 34(1-3) :197–215, 2002.

[2] Atkinson, K. et T. J. M. Bench-Capon: *Value-based Argumentation*. FLAP, 8(6) :1543–1588, 2021.

[3] Bechtel, G. et J.-C. Carrière: *Dictionnaire de la Bêtise et des erreurs de jugement, suivi du Livre des Bizarres*. Robert Laffont, Bouquins, 1965, 1991, 2014.

11. Une légende raconte que le choix de cet adjectif a été proposé par un système de BA. L'origine de cette légende se perd dans la nuit d'étang (noirâtre ?).

[4] Besnard, P. et A. Hunter: *Elements of Argumentation*. MIT Press, 2008, ISBN 978-0-262-02643-7.

[5] Bisquert, P., F. Dupin de Saint-Cyr et P. Besnard: *Assessing arguments with schemes and fallacies*. Dans Balduccini, M., Y. Lierler et S. Woltran (rédacteurs) : *Proc. 15th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'19), Philadelphia, June 3-7*, tome 11481 de *LNCS*, pages 61–74. Springer, 2019.

[6] Cipolla, C. M.: *The basic laws of human stupidity*. Bologna : il Mulino, 2011.

[7] Denis, P.: *Éloge de la Bêtise*. Presses Universitaires de France - PUF, 2001.

[8] Descartes, R.: *Discours de la méthode*. 1637.

[9] Dung, P. M.: *On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games*. Artif. Intell., 77(2) :321–358, 1995.

[10] Dupin de Saint-Cyr, F. et H. Prade: *La compréhension des histoires drôles : une affaire de révision de croyances*. Revue Ouverte d'Intelligence Artificielle, 2022.

[11] Fagin, R. et J. Y. Halpern: *Belief, Awareness, and Limited Reasoning.* Artif. Intell., 34(1) :39–76, 1987.

[12] Frankfurt, H.: *On Bullshit*. Princeton University Press, 2005. Traduction française : De l'Art de Dire des Conneries, 10/18, 2006.

[13] Godin, C.: *La bêtise existe-t-elle ?* Le Philosophoire, 42 :27–38, 2014. L'auteur développe son propos dans son Encyclopédie Conceptuelle et Thématique de la Philosophie (Editions Champ Vallon, 2018) au chapitre 96 sur l'intelligence qui comporte une section III sur la bêtise et une section IV sur l'intelligence artificielle.

[14] Jean-Paul: *Éloge de la Bêtise*. Corti, Domaine Romantique, 1782. Traduit de l'allemand par N. Briand, 1993.

[15] Kast, R.: *La Théorie de la Décision*. La Découverte, 1993.

[16] Latzarus, L.: *Éloge de la Bêtise* . Hachette, 1925.

[17] Lidén, L.: *Artificial stupidity : The art of intentional mistakes*. Dans Rabin, S. (rédacteur) : *AI Game Programming Wisdom*, tome 2, pages 41–48. Charles River Media Rockland, MA, 2003.

[18] Lieber, J. et E. Nauer: *Adaptation knowledge discovery using positive and negative cases*. Dans *Proc. of ICCBR 2021*, 2021.

[19] Musil, R.: *De la Bêtise*. Allia, 1937. Traduit de l'allemand par Ph. Jaccottet, 2000.

[20] Paulos, J. A.: *Innumeracy : Mathematical Illiteracy and Its Consequences*. Hill and Wang, 1988.

[21] Perissi, I. et U. Bardi: *The Sixth Law of Stupidity : A Biophysical Interpretation of Carlo Cipolla's Stupidity Laws*, mars 2021.

[22] Prade, H.: *Handling (un)awareness and related issues in possibilistic logic : A preliminary discussion*. Dans *Proc. of NMR 2006*, pages 219–225, 2006.

[23] Ritchie, G.: *The Comprehension of Jokes : A Cognitive Science Framework*. CRC Press, 2018.

[24] Rollin, F.: *Suis-je bête ! — L'héroïque Professeur Rollin foudroie la bêtise avec ruse et modestie*. Presses Universitaires de France, 2020. Illustrations de D. Gooseens, préface d'A. Astier.

[25] Rossit, J., J. G. Mailly, Y. Dimopoulos et P. Moraitis: *United we stand : Accruals in strength-based argumentation*. Argument Comput., 12(1) :87–113, 2021.

[26] Schneps, L. et C. Colmez: *Les Maths au tribunal. Quand les erreurs de calcul font les erreurs judiciaires*. Seuil, 1995.

[27] Tettamanzi, A. G. B. et C. da Costa Pereira: *Testing Carlo Cipolla's Laws of Human Stupidity with Agent-Based Modeling*. Dans *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, August 11-14, 2014 - Volume III*, pages 246–253. IEEE Computer Society, 2014. `https://doi.org/10.1109/WI-IAT.2014.174`.

[28] Turing, A.: *Computing Machinery and Intelligence*. Mind, LIX(236) :433–460, octobre 1950.

[29] Weizenbaum, J.: *ELIZA - A computer program for the study of natural language communication between man and machine*. Communications of the ACM, 9(1) :36–45, 1966.

[30] Winograd, T.: *Understanding natural language*. Cognitive psychology, 3(1) :1–191, 1972.

[31] Yannakakis, G. N. et J. Togelius: *Artificial Intelligence and Games*. Springer, 2018.

Session 4 : Argumentation

Actes JIAF 2022

Sémantiques à base d'extensions pour les systèmes d'argumentation incomplets

Jean-Guy Mailly

Université Paris Cité, LIPADE, F-75006 Paris, France
jean-guy.mailly@u-paris.fr

Résumé

Les systèmes d'argumentation incomplets (IAFs) introduisent de l'incertitude qualitative dans l'argumentation abstraite : on peut exprimer des informations comme « Je ne suis pas sûr que cet argument (ou attaque) existe ». Généralement, le raisonnement se fait via l'ensemble des complétions, c'est-à-dire des AFs qui représentent les mondes possibles encodés dans le IAF. Au contraire, nous suivons une approche « directe », en définissant de nouvelles formes d'*absence de conflit* et de *défense*, les propriétés à la base des sémantiques de Dung. Nous étudions les propriétés des nouvelles sémantiques que cela induit. En particulier, nous montrons que leur complexité est identique au cas classique. Nous proposons un encodage en logique qui permet d'utiliser des solvers SAT pour raisonner avec les IAFs. [1]

Abstract

Incomplete Argumentation Frameworks (IAFs) incorporate qualitative uncertainty in abstract argumentation: information such as "I am not sure whether this argument (or attack) exists" can be expressed. In general, reasoning relies on the set of completions, *i.e.* AFs that represent possible worlds encoded in the IAF. On the contrary, we follow a "direct" approach which consists in defining new forms of *conflict-freeness* and *defense*, the properties that underly the definition of Dung's semantics. We study the properties of the new semantics induced by this basic principles. In particular we show that their complexity is the same as in the case of Dung's AFs. Finally, we propose a logical encoding which allows to use SAT solvers for reasoning with IAFs.

1 Préliminaires formels

Un système d'argumentation [4] (AF, pour *argumentation framework*) est un graphe dirigé $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ dont les noeuds représentent des arguments et les arcs des attaques entre ces arguments. Des *sémantiques à base d'extensions* ont été définies pour déterminer l'acceptabilité des arguments. Elles sont en général basées sur deux principes : un ensemble acceptable $E \subseteq \mathcal{A}$ doit 1) être *sans conflit* ($\mathrm{cf}(\mathcal{F})$) : $\forall a, b \in E$, $(a, b) \notin \mathcal{R}$; 2) *défendre tous ses éléments* : $\forall a \in E$, $\forall b \in \mathcal{A}$, si $(b, a) \in \mathcal{R}$ alors $\exists c \in E$ tel que $(c, b) \in \mathcal{R}$. Un ensemble qui satisfait ces deux propriétés est appelé *admissible* ($\mathrm{ad}(\mathcal{F})$). Les extensions *complètes* ($\mathrm{co}(\mathcal{F})$) sont les ensembles admissibles qui contiennent tous les arguments qu'ils défendent ; les extensions *préférées* ($\mathrm{pr}(\mathcal{F})$) sont les extensions complètes maximales (pour l'inclusion) ; les extensions stables ($\mathrm{st}(\mathcal{F})$) sont les ensembles sans conflit qui attaquent tous les arguments extérieurs. Un *système d'argumentation incomplet* (IAF) est un AF dont certains éléments sont incertains. Formellement, $\mathcal{I} = \langle \mathcal{A}, \mathcal{A}^?, \mathcal{R}, \mathcal{R}^? \rangle$ où l'agent n'est pas sûr de l'existence des arguments dans $\mathcal{A}^?$ et des attaques dans $\mathcal{R}^?$. Les modes de raisonnement classiques dans ce cadre font appel à la notion de complétion, c'est-à-dire un AF standard qui représente un des mondes possibles encodés dans l'IAF. L'ensemble de complétions est généralement exponentiel par rapport au nombre d'arguments, ce qui peut avoir un impact sur la complexité du raisonnement [1]. Au contraire, l'approche *directe* proposée par [3] pour les AFs partiels (PAFs, c'est-à-dire des IAFs avec $\mathcal{A}^? = \emptyset$) adapte les notions d'absence de conflit et de défense aux différentes natures de l'information présente dans le PAF (certaine ou incertaine) pour proposer des sémantiques qui ne s'appuient pas sur l'ensemble de complétions. C'est cette approche que nous généralisons aux IAFs.

2 Sémantiques pour les IAFs

On considère deux façons de traiter l'acceptabilité des arguments dans un IAF. De façon optimiste, les attaques incertaines (ou provenant d'arguments incertains) ne sont pas des menaces sérieuses pour les arguments dont elles sont la cible. Au contraire, de façon pessimiste, toutes les

1. Cet article est un résumé de [6].

attaques sont des menaces sérieuses, et une défense n'est efficace que si elle est certaine. Formellement :

Définition 1 *Soit $\mathcal{I} = \langle \mathcal{A}, \mathcal{A}^?, \mathcal{R}, \mathcal{R}^? \rangle$. $E \subseteq \mathcal{A} \cup \mathcal{A}^?$ est 1)* faiblement sans conflit *ssi $\forall a, b \in E \cap \mathcal{A}$, $(a, b) \notin \mathcal{R}$; 2)* fortement sans conflit *ssi $\forall a, b \in E$, $(a, b) \notin \mathcal{R} \cup \mathcal{R}^?$.*
Soit $a \in \mathcal{A} \cup \mathcal{A}^?$, 1) E défend *a* faiblement *ssi $\forall b \in \mathcal{A}$ t.q. $(b, a) \in \mathcal{R}$, $\exists c \in E \cap \mathcal{A}$ t.q. $(c, b) \in \mathcal{R}$; 2) E* défend *a* fortement *ssi $\forall b \in \mathcal{A} \cup \mathcal{A}^?$ t.q $(b, a) \in \mathcal{R} \cup \mathcal{R}^?$, $\exists c \in E \cap \mathcal{A}$ t.q. $(c, b) \in \mathcal{R}$.*

En combinant ces notions, on peut envisager plusieurs formes d'admissibilité. Nous avons montré dans [6] que deux d'entre elles ont des propriétés intéressantes :

Définition 2 *Soit $\mathcal{I} = \langle \mathcal{A}, \mathcal{A}^?, \mathcal{R}, \mathcal{R}^? \rangle$. $E \subseteq \mathcal{A} \cup \mathcal{A}^?$ est* faiblement (resp. fortement) admissible *ssi E est faiblement (resp. fortement) sans conflit, et défend faiblement (resp. fortement) tous ses éléments.*

Pour $\sigma \in \{\text{cf}, \text{ad}, \text{co}, \text{pr}, \text{st}\}$, σ_x représente sa contrepartie faible (si $x = w$) ou forte (si $x = s$). On adapte alors les sémantiques $\sigma \in \{\text{co}, \text{pr}, \text{st}\}$ ainsi :

Définition 3 *Soient $\mathcal{I} = \langle \mathcal{A}, \mathcal{A}^?, \mathcal{R}, \mathcal{R}^? \rangle$, et $E \subseteq \mathcal{A} \cup \mathcal{A}^?$.*
1) $E \in \text{co}_x(\mathcal{I})$ ssi $E \in \text{ad}_x(\mathcal{I})$ et E contient tout ce qu'il x-défend ($x \in \{w, s\}$); 2) $E \in \text{pr}_x(\mathcal{I})$ ssi E est un élément $\subseteq$-maximal de $\text{ad}_x(\mathcal{I})$ ($x \in \{w, s\}$); 3) $E \in \text{st}_w(\mathcal{I})$ ssi $E \in \text{cf}_w(\mathcal{I})$ et $\forall a \in \mathcal{A} \setminus E$, $\exists b \in E \cap \mathcal{A}$ t.q. $(b, a) \in \mathcal{R}$; 4) $E \in \text{st}_s(\mathcal{I})$ ssi $E \in \text{cf}_s(\mathcal{I})$ et $\forall a \in (\mathcal{A} \cup \mathcal{A}^?) \setminus E$, $\exists b \in E \cap \mathcal{A}$ t.q. $(b, a) \in \mathcal{R}$.

Nous avons étudié les inclusions entre sémantiques, qui sont résumées en Table 1.[2] Pour une cellule de coordonnées (σ_1, σ_2), ✓ signifie : $\forall \mathcal{I}, \sigma_1(\mathcal{I}) \subseteq \sigma_2(\mathcal{I})$, et ✠ signifie : $\forall \mathcal{I}$, $\forall E \in \sigma_1(\mathcal{I})$ $\exists E' \in \sigma_2(\mathcal{I})$ t.q. $E \subseteq E'$.

σ_1 \ σ_2	cf_w	cf_s	ad_w	ad_s	co_w	co_s	pr_w	pr_s	st_w	st_s
cf_w	✓									
cf_s	✓	✓								
ad_w	✓		✓							
ad_s	✓	✓	✓	✓						
co_w	✓		✓		✓					
co_s	✓	✓	✓	✓	✠	✓	✠			
pr_w	✓		✓		✓		✓			
pr_s	✓	✓	✓	✓	✠	✓	✠	✓		
st_w	✓								✓	
st_s	✓	✓	✠	✓	✠	✓	✠	✓	✓	✓

Table 1 – Résumé des inclusions entre sémantiques

3 Complexité et algorithmes

La Table 2 résume la complexité du raisonnement avec nos sémantiques, où Ver correspond à la *vérification d'une extension*, Exist à l'*existence d'une extension*, Cred et Skep à l'*acceptation crédule* et l'*acceptation sceptique*, et NE à l'*existence d'une extension non vide*. On remarque que la complexité est équivalente à celle du raisonnement dans le cadre de Dung [5] (mis à part co-Skep, qui est polynomiale dans le cadre de Dung, et pour laquelle nous avons uniquement démontré une appartenance à coNP pour le moment). On remarque également que le choix d'une variante (faible ou forte) n'influence pas la complexité. Nous avons également proposé une traduction en logique propositionnelle des sémantiques, en se basant sur l'approche de [2] : on définit une formule $\phi^x_\sigma(\mathcal{I})$ telle que ses modèles correspondent à $\sigma_x(\mathcal{I})$, pour $\sigma \in \{\text{cf}, \text{ad}, \text{st}, \text{co}\}$. On peut alors utiliser un solver SAT pour résoudre tous les problèmes étudiés. La formule dédiée aux sémantiques ad_x ou co_x peut aussi être utilisée pour certains problèmes liés à pr_x, tandis que d'autres problèmes nécessitent des techniques propres liées à la maximisation. Des travaux récents (non inclus dans [6], mais actuellement soumis à un journal) montrent que cette approche passe bien à l'échelle, permettant par exemple de calculer une extension en (au plus) quelques secondes, pour les sémantiques σ_x avec $\sigma \in \{\text{co}, \text{st}\}$ et $x \in \{w, s\}$, avec des IAFs contenant jusque 200 arguments.

σ_x	σ_x-Ver	σ_x-Exist	σ_x-Cred	σ_x-Skep	σ_x-NE
cf_x	in L	trivial	in L	trivial	in L
ad_x	in L	trivial	NP-c	trivial	NP-c
st_x	in L	NP-c	NP-c	coNP-c	NP-c
co_x	in L	trivial	NP-c	in coNP	NP-c
pr_x	coNP-c	trivial	NP-c	Π^P_2-c	NP-c

Table 2 – Complexité de σ_x-Ver, σ_x-Exist, σ_x-Cred, σ_x-Skep et σ_x-NE for $\sigma \in \{\text{cf}, \text{ad}, \text{st}, \text{co}, \text{pr}\}$ et $x \in \{w, s\}$.

2. Ce tableau corrige une erreur incluse dans [6] au sujet de $\sigma = \text{st}_w$.

Références

[1] Baumeister, D., M. Järvisalo, D. Neugebauer, A. Niskanen et J. Rothe: *Acceptance in incomplete argumentation frameworks*. Artif. Intell., 295 :103470, 2021.

[2] Besnard, P. et S. Doutre: *Checking the acceptability of a set of arguments*. Dans *NMR'04*, pages 59–64, 2004.

[3] Cayrol, C., C. Devred et M. C. Lagasquie-Schiex: *Handling Ignorance in Argumentation : Semantics of Partial Argumentation Frameworks*. Dans *ECSQARU'07*, pages 259–270, 2007.

[4] Dung, P. M.: *On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games*. Artif. Intell., 77(2) :321–358, 1995.

[5] Dvorák, W. et P. Dunne: *Computational Problems in Formal Argumentation and their Complexity*. Dans *Handbook of Formal Argumentation*, pages 631–688. 2018.

[6] Mailly, J. G.: *Extension-Based Semantics for Incomplete Argumentation Frameworks*. Dans *CLAR'21*, pages 322–341, 2021.

Actes JIAF 2022

Admissibility in Strength-based Argumentation: Complexity and Algorithms

Yohann Bacquey[1] **Jean-Guy Mailly**[1]
Pavlos Moraitis[1,2] **Julien Rossit**[1]

[1]Université Paris Cité, LIPADE, F-75006 Paris, France
[2]Argument Theory, Paris, France

yohann.bacquey@etu.u-paris.fr jean-guy.mailly@u-paris.fr
pavlos.moraitis@u-paris.fr julien.rossit@u-paris.fr

Abstract

Strength-based Argumentation Frameworks (StrAFs) have been proposed to model situations where some quantitative strength is associated with arguments. In this setting, the notion of accrual corresponds to sets of arguments that collectively attack an argument. Some semantics have already been defined, which are sensitive to the existence of accruals that collectively defeat their target, while their individual elements cannot. However, until now, only the surface of this framework and semantics have been studied. Indeed, the existing literature focuses on the adaptation of the stable semantics to StrAFs. In this paper, we push forward the study and investigate the adaptation of admissibility-based semantics. We show that the strong admissibility defined in the literature does not satisfy a desirable property, namely Dung's fundamental lemma. We therefore propose an alternative definition that induces semantics behaving as expected. We then study computational issues for these new semantics, in particular we show that complexity of reasoning is similar to the complexity of the corresponding decision problems for standard argumentation frameworks in almost all cases. We propose a translation in pseudo-Boolean constraints for computing extensions. We conclude with an experimental evaluation of our approach which shows that it scales up well for solving the problem of providing one extension as well as enumerating them all.

1 Introduction

Among widespread knowledge representation and reasoning techniques proposed in the literature of Artificial Intelligence over the last decades, Abstract Argumentation Frameworks [7] are an intuitive but yet powerful tool for dealing with conflicting pieces of information. In this setting, directed graphs where nodes represent arguments, and edges represent attacks between them, are used to determine which sets of arguments can be (collectively) accepted. This convenient and elegant framework successfully captures many well-known problems like reasoning with exceptions [26], or the stable marriage problem [15].

Since then the initial work of Dung has been actively extended and enriched in many directions, *e.g.* considering other kinds of relations between arguments [6] or additional information associated with arguments or attacks [1, 5]. Among them, Strength-Based Argumentation Frameworks [27] allow to associate a quantitative information with each available argument. This quantitative information is a weight that intuitively represents the intrinsic strength of an argument, and is then naturally combined with attacks between arguments to induce a defeat relation that allows either to confirm an attack between two arguments or to cancel it, if the attacked argument is stronger than the attacking one (*i.e.* w.r.t. their respective weights). Strength Based Argumentation Frameworks extend further this notion of defeat among arguments by building a defeat that is based on a collective attack of a group of arguments (or accrual) and by offering associate semantics. Within these semantics, arguments can collectively defeat arguments that they cannot defeat individually. Intuitively speaking, these accrual-sensitive semantics allow some kind of compensation among arguments, where the accumulation of weak arguments can create a synergy and get rid of a stronger argument they collectively attack. This reasoning approach allows to produce extensions that are not considered when applying classical semantics.

Accrual of arguments has not received much attention in

the literature, although it is natural in various situations. Let us consider the following example, borrowed from [29]. Assume that John has robbed someone (without violence), thus he should be convicted. However, the judge considers that John should not be convicted, since he is a first offender. Similarly, assume that John has attacked someone. John may not be convicted since he is a first offender. But in the situation where John has attacked someone for robbing this person, then despite the fact that he is a first offender, the judge can choose to punish John.

In [27] the authors presented the basics of Strength-based Argumentation Frameworks (StrAFs) inspired by Dung's semantics for abstract argumentation along with some theoretical and computational results concerning classical issues related to abstract argumentation (*i.e.* acceptability semantics, semantics inclusion, extensions existence and verification, etc.). In this paper we propose a state of the art advancement in Strength-based Argumentation Frameworks (StrAFs) by presenting original theoretical and computational results related to different aspects. More particularly the contribution of this work lies into the following aspects. The semantics proposed in [27] exist in two versions (namely strong, and weak). Roughly speaking, a set is strongly conflict-free if and only if none of its elements attacks another one, whereas a set is weakly conflict-free if and only if it does not contain any (successful) accrual against one of its elements. After detecting that strong admissibility fails to satisfy a desirable property in Dung's abstract argumentation frameworks, namely his Fundamental Lemma [7], we propose an alternative definition for strong admissibility in order to remedy this issue and we define new admissibility-based semantics for $StrAFs$. We show that our new definition of strong admissible sets (as well as strong complete and preferred extensions) satisfies the expected properties. Besides, we also show that it is not required to adapt the definition of weak admissibility, since the one defined in [27] already behaves as expected. Furthermore, we study the complexity of reasoning with these semantics and in particular we show that, surprisingly, the complexity does not increase with respect to the complexity of reasoning with standard argumentation frameworks. For computing the extensions under these semantics, we propose algorithms based on pseudo-Boolean constraints.

The rest of the paper is organized as follows. We recall some background notions of abstract argumentation and strength-based argumentation in Section 2. Then, in Section 3, we study the property of admissibility-based semantics for StrAFs. We provide complexity results for several classical reasoning problems, as well as computational methods based on pseudo-Boolean encodings, in Section 4. This computational approach is empirically evaluated in Section 5. Finally, Section 6 describes some related work and concludes the paper with interesting topics for future research. Proofs are omitted for space reasons.

2 Background Notions

2.1 Abstract Argumentation Frameworks

Let us recall basic notions of abstract argumentation.

Definition 1 (Argumentation Framework [7])**.** *An* argumentation framework *(AF) is a pair* $\langle \mathcal{A}, \mathcal{R} \rangle$*, where* $\mathcal{A}$ *is a set of* arguments*,* $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ *is an* attack relation*.*

We focus on finite argumentation frameworks, *i.e.* AFs such that $\mathcal{A}$ is a finite set. For $a, b \in \mathcal{A}$, if $(a, b) \in \mathcal{R}$, then we say that *a attacks b*. Moreover, if $(b, c) \in \mathcal{R}$ also holds (for $c \in \mathcal{A}$), then *a defends c* against *b*. Given a set of arguments $S \subseteq \mathcal{A}$, we say that S attacks the argument a if there is an argument $b \in S$ that attacks a. Finally, $S \subseteq \mathcal{A}$ *defends* $a \in \mathcal{A}$ *against* $b \in \mathcal{A}$ if b attacks a and S attacks b. We say that S *defends* a if S attacks all the attackers of a.

In order to reason with AFs, Dung has defined semantics based on the notion of extensions, *i.e.* sets of arguments that can be jointly accepted. These semantics rely on basic principles of conflict-freeness and admissibility.

Definition 2 (Conflict-freeness and Admissibility [7])**.** *Let* $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework. For* $S \subseteq \mathcal{A}$*,*

- S *is conflict-free iff* $\nexists\, a, b \in S$ *s.t.* $(a, b) \in \mathcal{R}$*;*
- S *is an admissible set iff* S *is a conflict-free set that defends all its elements.*

The set of all conflict-free (resp. admissible) sets of AF is denoted $\mathrm{cf}(AF)$ *(resp.* $\mathrm{ad}(AF)$*).*

The intuition behind these concepts is that it does not seem "reasonable" to accept together arguments that are conflicting, or that cannot stand the attacks they receive.

We recall now the Fundamental Lemma of abstract argumentation [7, Lemma 10], which states that a "reasonable point of view" (*i.e.* an admissible set) can be extended with the arguments that it defends.

Lemma 1 (Fundamental Lemma)**.** *Let* $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework. Let* $S \subseteq \mathcal{A}$ *be an admissible set, and* a, a' *two arguments that are defended by* S*. 1.* $S' = S \cup \{a\}$ *is admissible, 2.* S' *defends* a'*.*

Now, classical extension semantics are defined by :

Definition 3 (Extension Semantics [7])**.** *Let* $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework. For* $S \subseteq \mathcal{A}$*,*

- S *is a complete extension iff it is an admissible set that contains all the arguments that it defends;*
- S *is a preferred extension iff it is a* $\subseteq$*-maximal admissible set;*
- S *is a stable extension iff it is a conflict-free set that attacks every argument in* $\mathcal{A} \setminus S$*.*

These sets of extensions are denoted (respectively) by $\mathrm{co}(AF)$*,* $\mathrm{pr}(AF)$ *and* $\mathrm{st}(AF)$*.*

The initial study [7] and some subsequent work define other semantics as well, that are out of the scope of this paper, so we choose to omit their definition. For more details on the semantics of AFs, we refer the interested reader to [4, 7]. Let us simply mention some basic properties of preferred and complete semantics. For any argumentation framework AF, there is at least one extension for the complete and preferred semantics, *i.e.* $\mathrm{co}(AF) \neq \emptyset$, and $\mathrm{pr}(AF) \neq \emptyset$. Moreover, for any AF, $\mathrm{st}(AF) \subseteq \mathrm{pr}(AF) \subseteq \mathrm{co}(AF)$.

2.2 Strength-based Argumentation Frameworks

Let us now recall the Strength-based Argumentation Frameworks introduced in [27] :

Definition 4. *(Strength-based Argumentation Framework) A* Strength-based Argumentation Framework *(StrAF) is a triple StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *where* $\mathcal{A}$ *and* $\mathcal{R}$ *are arguments and attacks, and* $\mathcal{S} : \mathcal{A} \to \mathbb{N}$ *is a* strength function.

Example 1. *Consider StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *depicted at Figure 1. Nodes represent arguments* $\mathcal{A}$*, edges represent attacks* $\mathcal{R}$*, and the numbers close to the nodes represent the arguments strength* $\mathcal{S}(a)$ *for each* $a \in \mathcal{A}$.

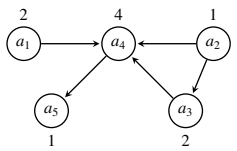


FIGURE 1 – A StrAF Example

These strengths represent the intrinsic robustness associated with an argument and allow to induce a *defeat* relation : an argument a defeat another argument b when a attacks b and the strength associated with b does not overcome that with a. Moreover, this framework offers the convenient notion of collective defeat, *i.e.* sets of arguments that can jointly defeat their target while they cannot do so separately. First, we call an *accrual* a set of arguments that collectively attack a same target :

Definition 5 (Accrual)**.** *Let StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a StrAF. A set of arguments* $\kappa \subseteq \mathcal{A}$ *is called* accrual *iff* $\exists c \in \mathcal{A}$ *s.t.* $\forall a \in \kappa$, $(a, c) \in \mathcal{R}$. *Moreover, we say that* κ *is* an accrual that attacks argument c. *Then, for* $\kappa' \subseteq \mathcal{A}$ *an accrual,* κ attacks κ' *iff* $\exists a \in \kappa'$ *s.t.* κ *attacks* a.

Example 2. *Consider again StrAF from Figure 1. We can observe several examples of accruals,* e.g. $\kappa_1 = \{a_1, a_2\}$ *and* $\kappa_2 = \{a_1, a_3\}$*, that both attack* a_4*. Notice that any attack* $(a_i, a_j) \in \mathcal{R}$ *induces an accrual* $\{a_i\}$ *attacking* a_j.

For evaluating whether an accrual (collectively) defeats its target, we first have to assess its collective strength.

Definition 6 (Collective Strength)**.** *Given StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *a StrAF and* $\kappa = \{a_1, ..., a_n\} \subseteq \mathcal{A}$ *an accrual, the collective strength associated with* κ *is* $\mathrm{coval}_{\oplus}(\kappa) = \oplus(\mathcal{S}(a_1), \ldots, \mathcal{S}(a_n))$ *where* $\oplus$ *is an aggregation operator,* i.e. *a mapping from a vector of integers to an integer satisfying :*

- **(non-decreasingness)** *if* $x_i \geq x'_i$*, then* $\oplus(x_1, \ldots, x_i, \ldots, x_n) \geq \oplus(x_1, \ldots, x'_i, \ldots, x_n)$*;*
- **(minimality)** $\oplus(x_1, \ldots, x_n) = 0$ *iff* $x_1 = \ldots = x_n = 0$*;*
- **(identity)** $\oplus(x) = x$*;*
- **(accrual)** $\oplus(x_1, \ldots, x_n) \leq \oplus(x_1, \ldots, x_n, y)$.

If $\oplus$ is clear from the context, we write coval for $\mathrm{coval}_{\oplus}$.

Definition 7 (Collective Defeat)**.** *Let StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a StrAF,* $a \in \mathcal{A}$*, and* $\oplus$ *an aggregation operator. Then, an accrual* κ defeats a *with respect to* $\mathrm{coval}_{\oplus}$*, denoted by* $\kappa \rhd_{\oplus} a$*, if and only if 1.* $\kappa \subseteq \mathcal{A}$ *is an accrual that attacks* a*; and 2.* $\mathrm{coval}_{\oplus}(\kappa) \geq \mathcal{S}(a)$*. If* $\oplus$ *is clear from the context, we use* $\kappa \rhd a$ *instead of* $\kappa \rhd_{\oplus} a$.

In the rest of the paper, we focus on $\oplus = \sum$ in examples and the pseudo-Boolean encoding defined in Section 4.2. But, unless explicitly stated otherwise, our results remain valid for any $\oplus$.

Definition 8. *Given StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *a StrAF,* coval *a collective strength evaluation and* $\kappa \subseteq \mathcal{A}$*,* $\kappa' \subseteq \mathcal{A}$ *two accruals,* κ defeats κ'*, denoted by* $\kappa \rhd \kappa'$*, iff* $\exists a \in \kappa'$ *s.t.* $\kappa \rhd a$.

Example 3. *Continuing Example 2, notice that* $\mathrm{coval}_{\sum}(\kappa_1) = \mathcal{S}(a_1) + \mathcal{S}(a_2) = 3 < \mathcal{S}(a_4)$*, (so* $\kappa_1 \not\rhd a_4$*), while* $\mathrm{coval}_{\sum}(\kappa_2) = \mathcal{S}(a_1) + \mathcal{S}(a_3) = 4 = \mathcal{S}(a_4)$*, (so* $\kappa_2 \rhd a_4$*).*

Extension-based semantics for StrAFs are based on a refinement of the notion of conflict-freeness, and can be defined in two ways :

Definition 9 (Conflict-freeness/Defense)**.** *Given StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *a StrAF,* $\oplus$ *an aggregation operator, and* $S \subseteq \mathcal{A}$.

- S *is* strongly conflict-free *iff* $\nexists a, b \in S$ *s.t.* $(a, b) \in \mathcal{R}$.
- S *is* weakly conflict-free *iff there are no accruals* $\kappa_1 \subseteq S$ *and* $\kappa_2 \subseteq S$ *s.t.* $\kappa_1 \rhd_{\oplus} \kappa_2$.
- S *defends an element* $a \in \mathcal{A}$ *iff for all accruals* $\kappa_1 \subseteq \mathcal{A}$*, if* $\kappa_1 \rhd_{\oplus} a$*, then there exists* $\kappa_2 \subseteq S$ *s.t.* $\kappa_2 \rhd_{\oplus} \kappa_1$.

Intuitively, strongly conflict-free sets are “classically" conflict-free, *i.e.* there is no attack between two arguments members of such a set. On the contrary, weakly conflict-free sets are “defeat-free" : attacks between arguments are permitted as long as they do not result in a defeat neither individual nor collective. We use (respectively) $\mathrm{cf}_S^{\oplus}$ and $\mathrm{cf}_W^{\oplus}$ to denote these sets (or simply cf_S and cf_W when $\oplus$ is clear from the context).

Example 4. *We illustrate the difference between strong and weak conflict-freeness. Given the StrAF depicted at Figure 1, we have* $\mathrm{cf}_S(StrAF) = \{\emptyset, \{a_1\}, \{a_2\},$

$\{a_3\}$,$\{a_4\}$, $\{a_5\}$, $\{a_1, a_2\}$, $\{a_1, a_3\}$, $\{a_1, a_5\}$, $\{a_2, a_5\}$, $\{a_3, a_5\}$, $\{a_1, a_2, a_5\}$, $\{a_1, a_3, a_5\}\}$, *while* $\mathrm{cf}_W(StrAF) = \mathrm{cf}_S(StrAF) \cup \{\{a_1, a_4\}, \{a_2, a_4\}, \{a_3, a_4\}, \{a_2, a_3\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_4\}, \{a_2, a_3, a_4\}, \{a_2, a_3, a_5\}, \{a_1, a_2, a_3, a_5\}\}$. *Notice that every strong conflict-free set of any StrAF is also weakly conflict-free. On the contrary, some sets that are not strongly conflict-free (*e.g. $\{a_2, a_3\}$*) are weakly conflict-free because the attack does not imply a defeat (*e.g. $\{a_2\} \not\rhd a_3$*).*

Then, admissibility and extension-based semantics can be defined either strong or weak. Namely :

Definition 10 (Semantics for StrAFs [27])**.** *Given $\langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ a StrAF, $\oplus$ an aggregation operator, and $S \subseteq \mathcal{A}$ a strong (resp. weak) conflict-free set,*

- *S is a* strong (resp. weak) admissible set *iff S defends all elements of S.*
- *S is a* strong (resp. weak) preferred extension *iff S is a $\subseteq$-maximal strong (resp. weak) admissible set.*
- *S is a* strong (resp. weak) stable extension *iff for each argument $a \in \mathcal{A} \backslash S$, there is $\kappa \subseteq S$ s.t. $\kappa \rhd_\oplus a$.*

For σ an extension-based semantics and $X \in \{S, W\}$ meaning respectively *strong* and *weak*, we use $\sigma_X^\oplus(StrAF)$ to denote the X-σ extensions of $StrAF$. We drop $\oplus$ from the notation where there is no possible ambiguity.

These are the only semantics defined in [27]. Here are a few results known about them. The first one states that the classical inclusion between semantics holds for strong (resp. weak) stable and preferred semantics.

Proposition 1 (Semantics Inclusion)**.** *Let $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ be a StrAF and $\oplus$ an aggregation operator. For $X \in \{S, W\}$, $\mathrm{st}_X^\oplus(StrAF) \subseteq \mathrm{pr}_X^\oplus(StrAF) \subseteq \mathrm{ad}_X^\oplus(StrAF)$.*

Then, it is proven that standard AFs are a specific subclass of StrAFs, where strong and weak semantics coincide, using the following transformation :

Definition 11. *Given an argumentation framework $AF = \langle \mathcal{A}, \mathcal{R} \rangle$, the StrAF associated with AF is $StrAF_{AF} = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ with $\mathcal{S}(a) = 1, \forall a \in \mathcal{A}$ and $\oplus = \sum$.*

Proposition 2 (Dung Compatibility)**.** *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an AF, and $StrAF_{AF} = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ its associated StrAF. For $\sigma \in \{\mathrm{cf}, \mathrm{ad}, \mathrm{pr}, \mathrm{st}\}$, $\sigma(AF) = \sigma_X^\sum(StrAF_{AF})$, for $X \in \{S, W\}$.*

This result is useful for proving complexity results. However, in [27] authors only focus on complexity issues for the (weak and strong) stable semantics.

3 Admissibility-based Semantics for StrAFs

In our study, we investigate computational issues for admissibility-based semantics. A formal definition of (weak or strong) complete semantics is missing in [27], but matching the definition of (weak or strong) admissibility with the classical definition of the complete semantics, a straightforward definition can be stated as follows :

Definition 12 (Straightforward Complete Semantics for StrAFs)**.** *Let $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ be a StrAF, and $\oplus$ an aggregation operator. A strong (resp. weak) admissible set $S \subseteq \mathcal{A}$ is a* strong (resp. weak) complete extension *of StrAF if S contains all the arguments that it defends.*

We study these semantics and in particular, we show that surprisingly this intuitive and natural definition of the complete semantics based on strong admissibility fails to satisfy a desirable property, namely the Fundamental Lemma. This leads to a redefinition of strong admissibility in Section 3.1. On the contrary, the definition proposed for weak admissibility is proved to be suitable in Section 3.2

3.1 Revisiting Strong Admissibility

First, we observe that the usual inclusion relation between the preferred and complete semantics is not satisfied for the strong semantics of StrAFs. Moreover, the universal existence of complete extensions does not hold either.

Proposition 3. *There exists StrAF such that $\mathrm{pr}_S(StrAF) \nsubseteq \mathrm{co}_S(StrAF)$, and $\mathrm{co}_S(StrAF) = \emptyset$.*

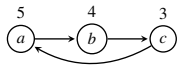


FIGURE 2 – Example proving that $\mathrm{pr}_S(StrAF) \nsubseteq \mathrm{co}_S(StrAF)$

Démonstration. For $StrAF$ from Figure 2, we have $\mathrm{cf}_S(StrAF) = \{\emptyset, \{a\}, \{b\}, \{c\}\}$. Trivially, $\emptyset$ is strongly admissible (since it is not defeated). Similarly, $\{a\}$ is strongly admissible (it is only attacked by c, but not defeated). $\{b\}$ is not strongly admissible ($\{a\} \rhd \{b\}$, and $\{b\}$ does not defend itself), neither $\{c\}$ ($\{b\} \rhd \{c\}$, and $\{c\}$ does not defend itself). With $\mathrm{ad}_S(StrAF) = \{\emptyset, \{a\}\}$, $\{a\}$ is the (only) $\subseteq$-maximal strongly admissible set, *i.e.* $\mathrm{pr}_S(StrAF) = \{\{a\}\}$. But $\{a\}$ is not complete : $\{a\}$ defends c against all its defeaters, but does not contain it. Hence $\mathrm{pr}_S(StrAF) \nsubseteq \mathrm{co}_S(StrAF)$. Moreover, $\emptyset$ is not complete either : $\emptyset$ defends a against all its defeaters. Thus $\mathrm{co}_S(StrAF) = \emptyset$. □

This example shows that the Fundamental Lemma does not hold for strong semantics : $\{a\}$ is strongly admissible, and defends c, but $\{a\} \cup \{c\}$ is not strongly admissible. The problem comes from the fact that a set of arguments can defend an argument, while the union is not (strongly) conflict-free. A solution to this issue is to redefine strong admissibility to require (strong) conflict-freeness of the union :

Definition 13 (Strong Admissibility Revisited)**.** *Let $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ be a StrAF and $\oplus$ an aggregation operator. A strong conflict-free set S* strongly defends *an argument a if S defends a against all the accruals that defeat it,* i.e. *$\forall \kappa \subseteq \mathcal{A}$ s.t. $\kappa \rhd a$, $\exists \kappa' \subseteq S$ s.t. $\kappa' \rhd \kappa$ and $S \cup \{a\}$ is strongly conflict-free. Then, a set $S \subseteq \mathcal{A}$ is* strongly admissible *if $S \in \mathrm{cf}_S(StrAF)$ and it strongly defends all its elements.*

With Definition 13, strong preferred extensions and strong complete extensions can be defined as follows :

Definition 14 (Strong Preferred and Complete Semantics Revisited)**.** *Let StrAF = $\langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ be a StrAF, $\oplus$ an aggregation operator, and S a strong admissible set.*

— *S is a* strong preferred extension *iff S is a $\subseteq$-maximal strong admissible set.*

— *S is a* strong complete extension *iff S contains all the arguments that it strongly defends.*

If we consider again the StrAF from Figure 2, observe that this time, the strongly admissible set $\{a\}$ does not strongly defends c, since $\{a, c\}$ is not strongly conflict-free. Thus $\{a\}$ is a strong complete extension of this StrAF, following Definition 14. Now, Dung's Fundamental Lemma can be adapted to strong admissibility.

Lemma 2 (Fundamental Lemma for Strong Admissibility)**.** *Let StrAF = $\langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ be a StrAF and $\oplus$ an aggregation operator. Given $S \subseteq \mathcal{A}$ be a strongly admissible set, and a an argument that is strongly defended by S against all its defeaters, $S' = S \cup \{a\}$ is strongly admissible.*

Notice that this version of the Fundamental Lemma has only one part of Dung's Lemma. However, this is enough to imply that strong preferred extensions are (as in Dung's framework) strong complete extensions.

Proposition 4. *For any StrAF and $\oplus$, $\mathrm{pr}_S^{\oplus}(StrAF) \subseteq \mathrm{co}_S^{\oplus}(StrAF)$.*

This guarantees the existence of at least one strong complete extension for any StrAF : since $\emptyset$ is a strong admissible set for any *StrAF*, then *StrAF* admits some $\subseteq$-maximal strong admissible sets, *i.e.* $\mathrm{pr}_S(StrAF) \neq \emptyset$, which implies $\mathrm{co}_S(StrAF) \neq \emptyset$.

Example 5. *Let us consider again the StrAF provided by Figure 1. From the strongly conflict-free sets (see Example 4), we deduce the strongly admissible sets $\mathrm{ad}_S(StrAF) = \{\emptyset, \{a_1\}, \{a_2\}, \{a_3\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_1, a_3, a_5\}\}$ by removing the sets that do not defend all their elements. Then, the strong preferred and complete extensions are $\mathrm{pr}_S(StrAF) = \mathrm{co}_S(StrAF) = \{\{a_1, a_2\}, \{a_1, a_3, a_5\}\}$.*

Finally, we prove that the new definition of strong admissibility does not change the fact that strong stable extensions are strongly preferred.

Proposition 5. *For any StrAF and $\oplus$, $\mathrm{st}_S^{\oplus}(StrAF) \subseteq \mathrm{pr}_S^{\oplus}(StrAF)$.*

3.2 Properties of the Weak Semantics

Regarding now weak semantics as defined in [27], the usual result still holds for StrAFs.

Lemma 3 (Fundamental Lemma for Weak Admissibility)**.** *Let StrAF = $\langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ be a StrAF and $\oplus$ an aggregation operator. Given $S \subseteq \mathcal{A}$ a weakly admissible set, and a, a' two arguments that are defended by S, 1. $S' = S \cup \{a\}$ is weakly admissible, and 2. S' defends a'.*

Proposition 6. *For any StrAF and $\oplus$, $\mathrm{pr}_W^{\oplus}(StrAF) \subseteq \mathrm{co}_W^{\oplus}(StrAF)$.*

Similarly to what we have noticed previously for strong admissibility, $\emptyset$ is weakly admissible for any StrAF. This implies the existence of at least one weak preferred extension, and then one weak complete extension for any StrAF.

Example 6. *Consider again the StrAF from Figure 1. Its weakly conflict-free sets were given at Example 4. From them, one identifies the weakly admissible sets $\mathrm{ad}_W(StrAF) = \mathrm{ad}_S(StrAF) \cup \{\{a_1, a_3, a_5\}, \{a_2, a_3\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_5\}\}$. Then, $\mathrm{pr}_W(StrAF) = \mathrm{co}_W(StrAF) = \{\{a_1, a_2, a_3, a_5\}\}$.*

Notice that, contrary to the case of stable semantics [27], we do not have $\mathrm{co}_S(StrAF) \subseteq \mathrm{co}_W(StrAF)$. This comes from the fact that our strong and weak complete semantics are not based on the same notion of defense. However, we observe that each strong complete extension is included in some weak preferred (and complete) extension. We prove that this is true for any StrAF.

Proposition 7 (Relation between Strong and Weak Semantics)**.** *Given StrAF = $\langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $\oplus$ an aggregation operator, $\forall E \in \mathrm{co}_S(StrAF)$, $\exists E' \in \mathrm{pr}_W(StrAF)$ s.t. $E \subseteq E'$.*

Notice finally that we do not need a counterpart to Proposition 5 : the definition of semantics based on weak admissibility is not modified (because the definition from the literature induces a notion of weak admissibility which satisfies the Fundamental Lemma), so the result from [27, Proposition 1] still holds in this case.

3.3 Dung Compatibility

Following the new definition of strong admissible sets, one might fear that Dung Compatibility does not hold for strong admissibility-based semantics. However, Proposition 8 shows that it still does, as well as for weak complete semantics.

Proposition 8 (Dung Compatibility)**.** *Let AF = $\langle \mathcal{A}, \mathcal{R} \rangle$ be an AF, and $StrAF_{AF} = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ its associated StrAF (see Definition 11). For $\sigma \in \{\mathrm{ad}, \mathrm{pr}, \mathrm{co}\}$, $\sigma(AF) = \sigma_X(StrAF_{AF})$, for $X \in \{S, W\}$.*

4 Complexity and Algorithms

Now we provide some insight on computational issues for admissibility-based semantics of StrAFs, *i.e.* we iden-

tify the computational complexity of several classical reasoning problems under these semantics, and we provide algorithms (based on pseudo-Boolean encoding) for solving them. While the complexity results are generic regarding the choice of $\oplus$, the algorithms focus on $\oplus = \sum$.

4.1 Complexity Analysis

Background on complexity Let us briefly recall some notions of computational complexity. P is the class of decision problems that can be solved in polynomial time (*i.e.* in $O(n^k)$ steps, where n is the size of the problem instance, and $k \in \mathbb{N}$) by a deterministic algorithm. NP is the class of problems that can be solved in polynomial time by a non-deterministic algorithm. It means that some execution of the algorithm must answer "YES" for all positive instances, and each possible execution of the algorithm must answer "NO" for all negative instances. coNP is the complement class of NP, *i.e.* $\mathcal{P} \in$ coNP iff $\overline{\mathcal{P}} \in$ NP, where $\mathcal{P}$ and $\overline{\mathcal{P}}$ have the same set of instances, and the answer to $\mathcal{P}$ on some instance i is "YES" iff the answer to $\overline{\mathcal{P}}$ is "NO" on i. A problem $\mathcal{P}$ is said to be *complete* for a complexity class C if it is one of the "hardest" problems in the class (*i.e.* $\mathcal{P} \in C$, and $\forall \mathcal{P}' \in C$, $\mathcal{P}'$ can be polynomially reduced to $\mathcal{P}$). Finally, $\Sigma_2^P = \mathsf{NP}^{\mathsf{NP}}$ is the class of problems that can be solved by a polynomial non-deterministic algorithm, with access to an oracle for some NP-complete problem, and Π_2^P is the complement class of Σ_2^P. We refer the interested reader to [8] for more details on complexity in formal argumentation, and [2] for a more general overview of computational complexity.

Reasoning with StrAFs We focus on three classical reasoning problems in abstract argumentation, namely *verification* ("Is a given set of arguments an extension ?"), *credulous acceptability* ("Is a given argument member of some extension ?") and *skeptical acceptability* ("Is a given argument member of each extension ?"). Formally, for $\sigma \in \{\text{ad}, \text{pr}, \text{co}\}$ and $X \in \{S, W\}$:

— σ-X-Ver : Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $S \subseteq \mathcal{A}$, is S a member of $\sigma_X(StrAF)$?

— σ-X-Cred : Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $a \in \mathcal{A}$, is a in some $S \in \sigma_X(StrAF)$?

— σ-X-Skep : Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $a \in \mathcal{A}$, is a in each $S \in \sigma_X(StrAF)$?

We recall that these reasoning problems are already considered only for the (weak and strong) stable semantics in [27]. In the following, we assume a fixed $\oplus$, that can be computed in polynomial time. This is not a very strong assumption, since it is the case with the classical aggregation operators (*e.g.* $\sum, \max, \dots$). Proposition 8 implies that the complexity of reasoning with standard AFs provides a lower bound complexity of reasoning with StrAFs. So we focus on identifying upper bounds.

Proposition 9 (Verification). *For $X \in \{S, W\}$, σ-X-Ver $\in P$, for $\sigma \in \{\text{ad}, \text{co}\}$, and* pr-$X$-Ver *is* coNP*-complete.*

Proposition 10 (Credulous Acceptability). *σ-X-Cred is* NP*-complete, for $\sigma \in \{\text{ad}, \text{co}, \text{pr}\}$ and $X \in \{S, W\}$.*

Proposition 11 (Skeptical Acceptability). *For $X \in \{S, W\}$,* ad-X-*Skep is trivial,* co-X-*Skep* $\in$ coNP*, and* pr-X-*Skep is Π_2^P-complete.*

Summary Table 1 summarizes the complexity results from Propositions 9, 10 and 11. We observe that the choice of the weak or strong variant of the semantics does not have an impact on the complexity of reasoning.

	σ-X-Ver	σ-X-Cred	σ-X-Skep
ad_X	P	NP-c	Trivial
co_X	P	NP-c	in coNP
pr_X	coNP-c	NP-c	Π_2^P-c

TABLE 1 – Complexity of reasoning for σ_X with $\sigma \in \{\text{ad}, \text{co}, \text{pr}\}$ and $X \in \{S, W\}$. Trivial means that all instances are trivially "NO" instances, and C-c means C-complete.

Discussion As it is the case for the strong (resp. weak) stable semantics [27], we prove here that the higher expressivity of StrAFs (compared to AFs) does not come at the price of a complexity blow-up. Only the case of skeptical acceptability under strong (resp. weak) complete semantics requires a deeper analysis, since we only provide the coNP upper bound. We have already explained that $\emptyset$ is a strong (resp. weak) admissible set of any StrAF. This implies that the existence problem ("Given a StrAF and a semantics, is there at least one extension ?") is trivial for the semantics studied in this paper. A related interesting problem is the non-empty existence ("Given a StrAF and a semantics, is there at least one non-empty extension ?"), which we keep for future work.

4.2 Algorithms

For computing the strong (resp. weak) admissible sets and complete extensions, we propose a translation of StrAF semantics in pseudo-Boolean (PB) constraints [28]. Such a constraint is an (in)equality $\sum_i w_i \times l_i \# k$ where w_i and k are positive integers, and $\# \in \{>, \geq, =, \neq, \leq, <\}$. l_i is a literal, *i.e.* $l_i = v_i$ or $l_i = \overline{v_i} = 1 - v_i$, where v_i is a Boolean variable (*i.e.* its value belongs to $\{0, 1\}$). Determining whether a set of PB constraints has a solution is a NP-complete problem, that generalizes the Boolean satisfiability (SAT) problem. Despite the high theoretical complexity of this problem, it can be efficiently solved in many cases, see *e.g.* [21, 12].

Strong and Weak Conflict-freeness Now we describe our PB encoding of StrAF semantics. For ensuring self-containment of the paper, we recall the encoding of strong and weak conflict-freeness [27]. Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and coval $= \sum$, we define a set of Boolean variables $\{x_i \mid a_i \in \mathcal{A}\}$ associated with each argument, where $x_i = 1$ means that a_i belongs to the set of arguments characterized by the solutions of the PB constraints. Then, strong conflict-freeness is encoded by :

(1) $\forall (a_i, a_j) \in \mathcal{R}$, add the constraint $x_i + x_j \leq 1$

and weak conflict-freeness is encoded by :

(1') $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, \ldots, a_n\}$,[1] add the constraint $\mathcal{S}(a_1) \times x_1 + \cdots + \mathcal{S}(a_n) \times x_n < x \times \mathcal{S}(a) + \overline{x} \times M$

where $\Gamma^-(a) = \{b \mid (b, a) \in \mathcal{R}\}$ is the set of attackers of $a \in \mathcal{A}$, and x is the Boolean variable associated with a. A solution to the set of constraints **(1)** (resp. **(1')**) yields a strong (resp. weak) conflict-free set $E = \{a_i \mid x_i = 1\}$. We formally prove this claim with Propositions 12 and 13. First, let us introduce some notations. Given a set of arguments $S \subseteq \mathcal{A}$, ω_S is a mapping from the set of x_i variables to $\{0, 1\}$ s.t. $\omega_S(x_i) = 1$ iff $a_i \in S$.

Proposition 12. *Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $S \subseteq \mathcal{A}$, $S \in \mathrm{cf}_S(StrAF)$ iff ω_S satisfies the set of constraints* **(1)**.

Proposition 13. *Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $S \subseteq \mathcal{A}$, $S \in \mathrm{cf}_W(StrAF)$ iff ω_S satisfies the set of constraints* **(1')**.

Strong and Weak Admissibility For encoding strong (resp. weak) admissibility, one must add to the set of constraints **(1)** (resp. **(1')**) some new constraints that represent the strong defense (resp. defense) property. To do so, one needs to introduce new Boolean variables $\{y_i \mid a_i \in \mathcal{A}\}$ s.t. $y_i = 1$ means that a_i is defeated by the set of arguments characterized by the solution of the PB constraints. Then, three constraints are added (the same ones for strong and weak admissibility) :

(2) $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, \ldots, a_n\}$, add the constraint $\mathcal{S}(a_1) \times x_1 + \cdots + \mathcal{S}(a_n) \times x_n \geq y \times \mathcal{S}(a)$

(3) $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, \ldots, a_n\}$, add the constraint $\mathcal{S}(a_1) \times x_1 + \cdots + \mathcal{S}(a_n) \times x_n \leq \overline{y} \times \mathcal{S}(a) + y \times M$

(4) $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, \ldots, a_n\}$, add the constraint $\mathcal{S}(a_1) \times \overline{y_1} + \cdots + \mathcal{S}(a_n) \times \overline{y_n} \leq x \times \mathcal{S}(a) + \overline{x} \times M$

with M an arbitrary large natural number that is greater than the sum of the strengths of the arguments (*i.e.* $M > \sum_{a \in \mathcal{A}} \mathcal{S}(a)$). The sets of constraints **(2)** and **(3)** ensure that $y = 1$ iff a is defeated by some $\kappa \subseteq E = \{a_i \mid x_i = 1\}$, and the constraints **(4)** ensure that E defends all its elements. The following propositions show the correctness of the encodings.

1. Notice that the constraints referring to $\Gamma^-(a)$ must be added even when $\Gamma^-(a) = \emptyset$.

Proposition 14. *Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $S \subseteq \mathcal{A}$, $S \in \mathrm{ad}_S(StrAF)$ iff ω_S satisfies the sets of constraints* **(1)**, **(2)**, **(3)** *and* **(4)**.

Proposition 15. *Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $S \subseteq \mathcal{A}$, $S \in \mathrm{ad}_W(StrAF)$ iff ω_S satisfies the sets of constraints* **(1')**, **(2)**, **(3)** *and* **(4)**.

Strong and Weak Complete Semantics Now, for computing the strong (resp. weak) extensions, one must consider the sets of constraints **(1)** (resp. **(1')**), **(2)**, **(3)** and **(4)**, and add a last set of constraints, respectively **(5)** for strong complete semantics, and **(5')** for weak complete semantics :

(5) $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, \ldots, a_n\}$ and $\Gamma^+(a) = \{a'_1, \ldots, a'_m\}$, add the constraint $\sum_{i=1}^{n}(\mathcal{S}(a_i) \times \overline{y_i}) + \sum_{i=1}^{n}(M \times x_i) + \sum_{i=1}^{m}(M \times x'_i) \geq \overline{x} \times \mathcal{S}(a)$

(5') $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, \ldots, a_n\}$, add the constraint $\mathcal{S}(a_1) \times \overline{y_1} + \cdots + \mathcal{S}(a_n) \times \overline{y_n} \geq \overline{x} \times \mathcal{S}(a)$

where $\Gamma^+(a) = \{b \in \mathcal{A} \mid (a, b) \in \mathcal{R}\}$ is the set of arguments attacked by a. These constraints ensure that an argument is not accepted only if it is not (strongly) defended. Again, we prove the correctness of the encodings :

Proposition 16. *Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $S \subseteq \mathcal{A}$, $S \in \mathrm{co}_S(StrAF)$ iff ω_S satisfies the sets of constraints* **(1)**, **(2)**, **(3)**, **(4)** *and* **(5)**.

Proposition 17. *Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and $S \subseteq \mathcal{A}$, $S \in \mathrm{co}_W(StrAF)$ iff ω_S satisfies the sets of constraints* **(1')**, **(2)**, **(3)**, **(4)** *and* **(5')**.

Acceptability and Verification Obtaining one (resp. each) solution for one of the sets of constraints defined previously corresponds to obtaining one (resp. each) extension of the StrAF under the corresponding semantics. For checking whether a given argument a_i is credulously accepted, one simply needs to add the constraint $x_i = 1$. If a solution exists, then it corresponds to an extension that contains a_i, proving that this argument is credulously accepted. Otherwise, a_i is not credulously accepted. For skeptical acceptability, one needs to add the constraint $x_i = 0$. In this case, a solution exhibits an extension that does not contain a_i, thus this argument is not skeptically accepted. In the case where no solution exists, then the argument is skeptically accepted. Finally, for checking whether a set of arguments $S \subseteq \mathcal{A}$ is an extension, one needs to add the constraints $x_i = 1$ for each $a_i \in S$, as well as $x_i = 0$ for each $a_i \in \mathcal{A} \setminus S$. A solution exists for the new set of constraints iff S is an extension under the considered semantics.

Strong and Weak Preferred Semantics Finally, let us mention an approach to handle reasoning with strong and

weak preferred semantics. Because of the higher complexity of skeptical reasoning under these semantics (recall Proposition 11), it is impossible (under the usual assumption that the polynomial hierarchy does not collapse) to find a (polynomial) encoding of these semantics in PB constraints. However, PB solvers can be used as oracles to find (with successive calls) preferred extensions. Algorithm 1 describes our method to do this for strong preferred semantics (replacing **(1)** by **(1')** provides an algorithm for weak preferred semantics). At start, we add the four constraints corresponding to a strong (resp. weak) admissible set and solve the instance, with the PB solver as a coNP oracle. Then we force the arguments within the extension to stay in the next one by adding the constraint on line 4. To avoid getting the same solution as in the previous step, we make sure that at least one argument outside the previous extension will be in the next one (line 5). This method iteratively extends an admissible set into a preferred extension, that is finally returned when the solver cannot find any (larger) solution. Such an approach is called a CounterExample Guided Abstraction Refinement (CEGAR). This approach has already demonstrated its power for handling problems that are hard for the second level of the polynomial hierarchy, especially reasoning with the preferred semantics of standard AFs, where the oracles are usually SAT solvers instead of PB solvers [11, 23].

Algorithm 1 Compute a strong preferred extension

P = PB problem with constraints **(1)**, **(2)**, **(3)** and **(4)**
while $P.solve() \neq null$ **do**
  $E \leftarrow P.solve()$
  $P.add_constr(x_1 + \cdots + x_n = n)$ for $E = \{a_1, \ldots, a_n\}$
  $P.add_constr(x'_1 + \cdots + x'_k \geq 1)$ for $\mathcal{A} \setminus E = \{a'_1, \ldots, a'_k\}$
end while
return E

5 Experimental Evaluation

For estimating the scalability of our method based on pseudo-Boolean constraints, we present now some results obtained from our experimental evaluation using two prominent PB solvers : Sat4j [18] and RoundingSat [12]. While Sat4j is based on saturation, RoundingSat uses the division rule (see [12] for a discussion on both approaches). We focus here on the most relevant results; additional results are presented in the supplementary material.

Benchmark Generation We generate benchmarks in a format adapted to StrAFs, inspired by ASPARTIX formalism [10]. We consider two classes of randomly generated graphs. First, with the Erdös–Rényi model (ER) [13], given a set of arguments $\mathcal{A}$, and $p \in [0, 1]$, we generate a graph such that for each $(a, b) \in \mathcal{A} \times \mathcal{A}$, a attacks b with a probability p. We consider two values for the probability, namely $p \in \{0.1, 0.5\}$. Then, with the Barabási–Albert (BA) model [3], a graph of n nodes is grown by attaching new nodes with m edges that are preferentially attached to existing nodes with a high degree. These types of graphs have been frequently used for studying computational aspects of formal argumentation, in particular during the ICCMA competitions [14]. The choice of a generation model provides the arguments $\mathcal{A}$ and attacks $\mathcal{R}$. We attach a random strength $\mathcal{S}(a) \in \{1, \ldots, 20\}$ to each $a \in \mathcal{A}$. For each generation model, we build 20 StrAFs for each $|\mathcal{A}| \in \{5, 10, 15, \ldots, 60\}$. Parameters ($p \in \{0.1, 0.5\}$ for ER, $m = 1$ for BA) are chosen to avoid graphs with a high density of attacks, that would prevent the existence of meaningful extensions (*e.g.* non-empty ones). Larger StrAFs (with $|\mathcal{A}| \in \{5, 10, \ldots, 250\}$) have been generated with the same parameters ($p \in \{0.1, 0.5\}$ for ER, $m = 1$ for BA) for studying the problem of providing one extension.

Experimental Setting The experiments were run on a Windows computer (using Windows Subsystem for Linux), with an Intel Core i5-6600K 3.50GHz CPU and 16GB of RAM. The timeout is set to 600 seconds, similarly to the runtime limit from ICCMA competitions [17].

Results We are interested in the semantics σ_X, with $\sigma \in \{\text{pr}, \text{st}, \text{co}\}$ and $X \in \{S, W\}$. The encodings for st_X ($X \in \{S, W\}$) are those proposed in [27], while the encoding for the other semantics are those described in Section 4.2. For each generated *StrAF*, and each of these semantics σ_X, the two tasks we are interested in consist in enumerating all extensions and finding one extension. We first focus on the runtime for enumerating σ_X extensions, which provides an upper bound of the runtime for solving other classical reasoning tasks, like deciding the credulous (or skeptical) acceptability of a given argument, or computing one extension [17]. To do so, we use a Python script that converts a StrAF into a set of PB constraints. The set of extensions is then obtained in a classical iterative way : once an extension is returned by the PB solver, we add a new constraint that forbids this extension, and we call again the solver on this updated set of PB constraints. This process is repeated until the set of constraints becomes unsatisfiable, which means that all the extensions have been obtained. Concerning the preferred extensions, this iterative approach is combined with Algorithm 1. In order to measure the performance of our approach, we also implemented a so-called *naive* algorithm that enumerates every subset of arguments and then verifies, for each of them, if it is a σ_X extension. The following figures present the average runtimes w.r.t. instance sizes (*i.e.* $|\mathcal{A}|$) for various semantics and StrAF families as described before. As a first result, we observe in Figure 3 that runtime for enumerating extensions (with the PB approach) is reasonable (*i.e.* less than a minute) for most of the cases considered in our study, when the PB approach is used, while the naive approach reaches the timeout for most of the large instances (in particular, all

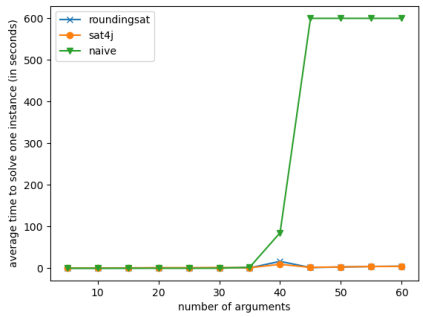


FIGURE 3 – Enumeration runtime for co_W on ER graphs ($p = 0.1$)

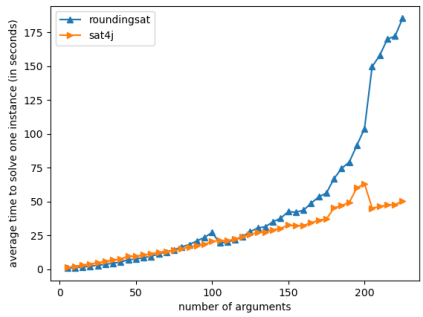


FIGURE 5 – Finding one extension runtime ($\sigma = pr_s$, BA graphs)

the instances with $|\mathcal{A}| \geq 45$). The average runtimes are higher in only two situations : the enumeration of strong preferred and strong complete extensions, with the BA graphs. However, even in such situations where the enumeration is harder (*e.g.* for the strong preferred semantics on BA graphs, as depicted on Figure 4), the PB solvers clearly outperform the naive algorithm, which reaches the timeout in every instance when $|\mathcal{A}| \geq 30$, while the PB approach can enumerate extensions for larger graphs. We also study the classical problem of providing one extension, for StrAFs of larger sizes (recall that here $|\mathcal{A}| \in \{5, 10, \ldots, 250\}$). Figure 5 shows that the PB solvers (in particular, Sat4j) provide one extension for these large graphs under two minutes, even for the preferred semantics (which is the hardest one, in our study, from the computational point of view). Concerning the respective performances of the two considered solvers, Figure 5 shows that Roundingsat processes faster for fast-to-compute instances (*i.e.* the smallest ones), while Sat4j outperforms it for instances of larger size. As a general conclusion on our experimental analysis,

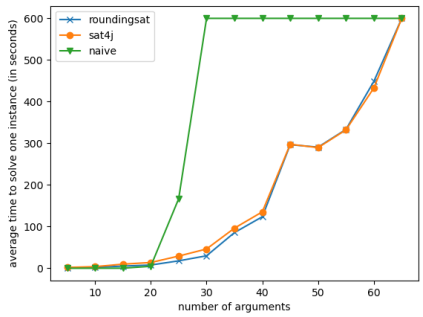


FIGURE 4 – Enumeration runtime ($\sigma = pr_S$, BA graphs)

we observe that the PB approach for reasoning with StrAFs generally scales up well, for both problems of enumerating extensions and providing one extension.

6 Related Work and Conclusion

Strength-based Argumentation Frameworks (StrAFs) have originally been proposed in [27]. Contrary to this work, in this paper we focused on admissibility-based semantics. We showed that the weak admissibility-based semantics defined in the original work satisfy some expected properties, namely Dung's Fundamental Lemma. However, the definition for strong admissibility proposed in [27] does not yield semantics that behave as expected. This has conducted us to revisit the definition of strong admissibility, and this allowed us to introduce strong complete and preferred semantics. We have also enhanced the StrAFs literature by studying the computational complexity of classical reasoning problems for these semantics, and we have shown that it is the same as for the corresponding tasks in Dung's framework, in spite of the increase of expressivity. Then we have proposed a method based on pseudo-Boolean constraints for computing the extensions of a StrAF under the various semantics defined in this paper, and we have empirically evaluated the scalability of this approach for the new semantics defined in this paper, as well as the (weak and strong) stable semantics from [27]. The accrual of arguments has also been studied in structured argumentation setting. For instance, in [24, 19, 20, 16, 25] authors propose to deal with accruals by adding a new argument that represents the accrual. In these works, the assumption is made that the arguments members of an accrual should not be taken into consideration on their own. On the other hand, in [29] conflicts are defined in terms of sets of arguments attacking other (sets of) arguments. This means that all accruals are explicitly given as input of the reasoning process. The notions introduced in these works strongly rely on the internal structure of the arguments, whereas in abstract argumentation, which is the topic of this paper, this structure is unknown. SETAFs [22] are also related to StrAFs, since they allow the representation of collective attacks. However, contrary to StrAFs,

SETAFs do not take into account the strength of individual arguments. Moreover, all the collective attacks (corresponding to the notion of defeating accrual in StrAFs) must be elicited in advance, while defeat in StrAFs are identified only when computing the extensions, which makes StrAFs more modular in dynamic contexts.

For our future work we have identified several promising research tracks, including the study of (weak and strong) grounded semantics, and tight complexity results for the skeptical reasoning under the (weak and strong) complete semantics. The question of whether a non-empty (weak or strong) extension exists is also appealing. We are also interested in a more detailed analysis of the relation between StrAFs and other frameworks, in particular the comparison of the signatures of StrAFs semantics and SETAFs semantics [9]. Finally, we want to study argument strength and accrual in a context of structured argumentation.

Références

[1] Amgoud, L. et C. Cayrol: *A Reasoning Model Based on the Production of Acceptable Arguments*. Annals of Mathematics and Artificial Intelligence, 34(1-3) :197–215, 2002.

[2] Arora, S. et B. Barak: *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[3] Barabasi, A. et R. Albert: *Emergence of Scaling in Random Networks*. Science, 286(5439) :509–512, 1999.

[4] Baroni, P., M. Caminada et M. Giacomin: *Abstract Argumentation Frameworks and Their Semantics*. Dans *Handbook of Formal Argumentation*, pages 159–236. 2018.

[5] Bench-Capon, T.: *Value-based Argumentation Frameworks*. Dans *NMR'02*, pages 443–454, 2002.

[6] Cayrol, C. et M. C. Lagasquie-Schiex: *Bipolarity in argumentation graphs : Towards a better understanding*. Int. J. Approx. Reason., 54(7) :876–899, 2013.

[7] Dung, P. M.: *On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games*. Artif. Intell., 77(2) :321–358, 1995.

[8] Dvorák, W. et P. Dunne: *Computational Problems in Formal Argumentation and their Complexity*. Dans *Handbook of Formal Argumentation*, pages 631–688. 2018.

[9] Dvorák, W., J. Fandinno et S. Woltran: *On the expressive power of collective attacks*. Argument Comput., 10(2) :191–230, 2019.

[10] Dvorák, W., S. A. Gaggl, A. Rapberger, J. P. Wallner et S. Woltran: *The ASPARTIX System Suite*. Dans *COMMA'20*, pages 461–462, 2020.

[11] Dvorák, W., M. Järvisalo, J. P. Wallner et S. Woltran: *Complexity-sensitive decision procedures for abstract argumentation*. Artif. Intell., 206 :53–78, 2014.

[12] Elffers, J. et J. Nordström: *Divide and Conquer : Towards Faster Pseudo-Boolean Solving*. Dans *IJCAI'18*, pages 1291–1299, 2018.

[13] Erdös, P. et A. Rényi: *On Random Graphs. I.* Publicationes Mathematicae, 6 :290–297, 1959.

[14] Gaggl, S. A., T. Linsbichler, M. Maratea et S. Woltran: *Design and results of the Second International Competition on Computational Models of Argumentation*. Artif. Intell., 279, 2020.

[15] Gale, D. et L. S. Shapley: *College Admissions and the Stability of Marriage*. The American Mathematical Monthly, 69(1) :9–15, 1962.

[16] Gordon, T.: *Defining argument weighing functions*. Journal of Applied Logics - IfCoLog Journal of Logics and their Application, 5 :747–773, 2018.

[17] Lagniez, J. M., E. Lonca, J G. Mailly et J. Rossit: *Introducing the Fourth International Competition on Computational Models of Argumentation*. Dans *Proc. of SAFA'20*, pages 80–85, 2020.

[18] Le Berre, D. et A. Parrain: *The Sat4j library, release 2.2*. J. Satisf. Boolean Model. Comput., 7(2-3) :59–6, 2010.

[19] Lucero, M. Gómez, C. Chesñevar et G. Simari: *On the Accrual of Arguments in Defeasible Logic Programming*. Dans *IJCAI'09*, pages 804–809, 2009.

[20] Lucero, M. Gómez, C. Chesñevar et G. Simari: *Modelling argument accrual with possibilistic uncertainty in a logic programming setting*. Information Sciences, 228 :1–25, 2013.

[21] Martins, R., V. Manquinho et I. Lynce: *Open-WBO : A Modular MaxSAT Solver,*. Dans *SAT'14*, pages 438–445, 2014.

[22] Nielsen, S. et S. Parsons: *A generalization of Dung's abstract framework for argumentation : Arguing with sets of attacking arguments*. Dans *ArgMAS'06*, pages 54–73. Springer, 2006.

[23] Niskanen, A. et M. Järvisalo: *μ-toksia : An Efficient Abstract Argumentation Reasoner*. Dans *KR'20*, pages 800–804, 2020.

[24] Prakken, H.: *A Study of Accrual of Arguments, with Applications to Evidential Reasoning*. Dans *ICAL'05*, pages 85–94, 2005.

[25] Prakken, H.: *Modelling Accrual of Arguments in ASPIC+*. Dans *ICAIL'19*, pages 285–297, 2019.

[26] Reiter, R.: *A Logic for Default Reasoning*. Artif. Intell., 13(1-2) :81–132, 1980.

[27] Rossit, J., J. G. Mailly, Y. Dimopoulos et P. Moraitis: *United We Stand : Accruals in Strength-based Argumentation*. Argument & Computation., 12(1) :87–113, 2021.

[28] Roussel, O. et V. Manquinho: *Pseudo-Boolean and Cardinality Constraints*. Dans *Handbook of Satisfiability*, pages 695–733. 2009.

[29] Verheij, B.: *Accrual of Arguments in Defeasible Argumentation*. Dans *Second Dutch/German Workshop on Nonmonotonic Reasoning*, pages 217–224, 1995.

Actes JIAF 2022

Restreindre l'Impact des Arguments Contradictoires dans les Sémantiques Graduées en Argumentation Abstraite

Vivien Beuselinck[1] **Jérôme Delobelle**[2] **Srdjan Vesic**[3]

[1] Aniti, Université Fédérale
[2] Université Paris Cité, LIPADE, F-75006 Paris, France
[3] CNRS, Univ. Artois, CRIL, Lens, France
vivien@beuselinck.fr jerome.delobelle@u-paris.fr vesic@cril.fr

Résumé

La question de savoir comment raisonner en présence d'arguments contradictoires a toujours été un sujet de débat parmi les spécialistes de l'argumentation formelle. Un consensus existe pour les sémantiques à base d'extensions car ces arguments sont toujours rejetés. Dans le cas des sémantiques graduées, la question est plus complexe, puisque d'autres critères sont pris en compte. Dans cet article, nous étudions l'impact des arguments contradictoires pour ces sémantiques via une approche axiomatique. Plus précisément, nous identifions deux ensembles maximaux de propriétés représentant chacun un traitement différent des arguments contradictoires. Nous définissons ensuite une sémantique graduée qui converge et satisfait l'un de ces ensembles de propriétés. Enfin, nous montrons expérimentalement l'efficacité de notre sémantique. Ce papier est un résumé du papier publié à la conférence CLAR 2021. [3]

Abstract

The issue of how a semantics should deal with self-attacking arguments was always a subject of debate amongst formal argumentation scholars. A consensus exists for classical extension-based semantics because those arguments are always rejected. In case of gradual semantics, the question is more complex, since other criteria are taken into account. In this paper we check the impact of those argument by using a principle-based approach. More precisely, we identify two maximal sets of principles each representing a different treatment of the self-attacking arguments. We then define a gradual semantics that converges and satisfies one of these sets of principles. Finally, we experimentally show that our semantics is computationally efficient.

1 État de l'art

Formellement, un **système d'argumentation** (AG) [4] est un couple $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ où $\mathcal{A}$ est un ensemble fini d'arguments et $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ est la relation d'attaque entre arguments. Bien que dans la plupart des cas une attaque peut exister entre deux arguments distincts, il arrive qu'un argument soit en conflit avec lui-même. Un tel argument est appelé un argument contradictoire (*self-attacking arguments* en anglais). Pour la partie raisonnement, il existe de nombreuses sémantiques permettant d'évaluer les arguments d'un AG. Dans ce papier, nous nous sommes focalisés sur les **sémantiques graduées** assignant à chaque argument un score (souvent compris entre 0 et 1) appelé *degré d'acceptabilité*.

De nombreuses propriétés ont été introduites [1] afin de mieux comprendre le comportement des sémantiques graduées, de choisir une sémantique pour une application particulière, de comparer les sémantiques entre elles, etc. Voici une liste partielle de propriétés existantes.

Equivalence (EQ) : deux arguments avec des attaquants de même force auront le même degré d'acceptabilité.

Counting (CT) : un attaquant de degré non nul doit avoir un impact sur l'acceptabilité des arguments qu'il attaque.

Weakening Soundness (WS) : si le degré d'acceptabilité d'un argument n'est pas maximal alors il est attaqué par au moins un attaquant ayant un degré d'acceptabilité non nul.

Reinforcement (RF) : l'acceptabilité d'un argument augmente si l'acceptabilité de ses attaquants diminue.

Resilience (RES) : aucun argument ne peut avoir le degré d'acceptabilité minimal.

En dehors des propriétés proposées dans [1], il existe deux autres propriétés qui concernent directement les arguments contradictoires :

Self-Contradiction (SC) : le degré d'acceptabilité d'un argument contradictoire doit être strictement inférieur au degré d'acceptabilité d'un argument qui ne l'est pas.

Strong Self-Contradiction (SSC) : les arguments contra-

dictoires sont les seuls à avoir le degré d'acceptabilité minimal.

2 Analyse des propriétés et de leurs liens

Notre analyse des implications et des incompatibilités entre les propriétés existantes a notamment mis en avant que EQ et SSC sont incompatibles. L'existence de cette incompatibilité nous a mené à étudier deux ensembles maximaux de propriétés compatibles entre elles : P_{CREW} incluant EQ et P_{2S2C} incluant SSC. L'intérêt d'étudier ces deux ensembles de propriétés est qu'ils proposent un traitement différent des arguments contradictoires. Concrètement, une sémantique satisfaisant P_{CREW} considère que l'auto-attaque d'un argument contradictoire est une attaque comme les autres, i.e. entre deux arguments distincts. Ainsi, un argument qui s'attaque lui-même (et qui n'est attaqué par aucun autre argument) peut être plus fort qu'un argument qui est attaqué par plusieurs arguments. Au contraire, une sémantique qui satisfait P_{2S2C} considère qu'un argument contradictoire est intrinsèquement erroné, sans même avoir besoin d'autres arguments pour le vaincre. C'est pourquoi son degré d'acceptabilité sera toujours minimal, limitant ainsi grandement l'impact de ces arguments sur l'acceptabilité des autres arguments.

3 Nouvelle sémantique graduée

Alors que la sémantique graduée h-categorizer [2] satisfait l'ensemble des propriétés de P_{CREW}, aucune sémantique graduée de la littérature ne satisfait toutes les propriétés de P_{2S2C} jusqu'à aujourd'hui. Afin d'y remédier, nous avons créé une nouvelle sémantique graduée, appelée no self-attack *h*-categorizer (`nsa`), qui est inspirée de *h*-categorizer. La principale différence est que nous forçons l'attribution du degré d'acceptabilité minimal (i.e. 0) aux arguments contradictoires tandis que le degré d'acceptabilité des autres arguments est calculé en utilisant la formule de la sémantique *h*-categorizer.

Définition 1 (nsa) *Pour tout système d'argumentation $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ et pour tout argument $x \in \mathcal{A}$,*

$$Deg^{\mathtt{nsa}}_{\mathcal{F}}(x) = \begin{cases} 0 & si\ (x,x) \in \mathcal{R} \\ \frac{1}{1+\sum_{y \in \mathtt{Att}_{\mathcal{F}}(x)} Deg^{\mathtt{nsa}}_{\mathcal{F}}(y)} & sinon \end{cases}$$

avec $\mathtt{Att}_{\mathcal{F}}(x)$ l'ensemble des attaquants directs de x.

Nous avons prouvé que `nsa` garantit la convergence et l'unicité du résultat. Enfin, nous avons fourni une caractérisation de `nsa`, c'est-à-dire une définition déclarative (non itérative), où chaque sémantique satisfaisant la définition déclarative coïncide forcément avec `nsa`.

	Propriétés	M&T	h-cat	nsa
⋆ ◊	Anonymity	✓	✓	✓
⋆ ◊	Independence	✓	✓	✓
⋆ ◊	Directionality	✓	✓	✓
⋆	Neutrality	×	✓	×
⋆	Equivalence	×	✓	×
⋆ ◊	Maximality	✓	✓	✓
⋆ ◊	Weakening	✓	✓	✓
⋆ ◊	Counting	×	✓	✓
⋆	Weakening Soundness	×	✓	×
⋆	Reinforcement	×	✓	×
⋆	Resilience	×	✓	×
	Cardinality Precedence	×	×	×
	Quality Precedence	×	×	×
⋆ ◊	Compensation	✓	✓	✓
◊	Self-Contradiction	✓	×	✓
◊	Strong Self-Contradiction	✓	×	✓

Table 1 – Propriétés satisfaites par les sémantiques M&T, h-categorizer et `nsa`. Le symbole ⋆ (resp. ◊) signifie que la propriété appartient à P_{CREW} (resp. P_{2S2C}).

4 Analyse axiomatique et empirique

L'étude axiomatique (résumée dans la Table 1) a permis de montrer que `nsa` est la première sémantique graduée de la littérature à satisfaire toutes les propriétés de P_{2S2C}.

L'étude empirique, basée sur un total de 2160 systèmes d'argumentation générés via différents modèles (Erdös-Rényi, Barabasi-Albert, Watts-Strogatz) sur des tailles allant de 5 à 500 arguments, a permis de mettre en avant que, sans surprise, `nsa` et h-categorizer ont des temps d'exécution très proches sur les instances testées et qui sont bien meilleurs que ceux de la sémantique M&T. De plus, `nsa` nous permet de calculer rapidement (avec une moyenne inférieure à une seconde) le degré d'acceptabilité de chaque argument, même pour les AG de grande taille.

Références

[1] Amgoud, L., J. Ben-Naim, D. Doder et S. Vesic: *Acceptability Semantics for Weighted Argumentation Frameworks*. Dans *IJCAI'17*, pages 56–62, 2017.

[2] Besnard, P. et A. Hunter: *A logic-based theory of deductive arguments*. Artificial Intelligence, 128(1-2) :203–235, 2001.

[3] Beuselinck, V., J. Delobelle et S. Vesic: *On Restricting the Impact of Self-attacking Arguments in Gradual Semantics*. Dans *CLAR'21*, tome 13040, pages 127–146, 2021.

[4] Dung, P.M.: *On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games*. Artificial Intelligence, 77(2) :321–358, 1995.

Actes JIAF 2022

Raisonner sur l'éthique avec un cadre d'argumentation fondé sur une logique modale normale

Christopher Leturc[1] **Grégory Bonnet**[2]

[1] Inria, Université Côte d'Azur, CNRS, I3S, France
[2] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, France
christopher.leturc@inria.fr gregory.bonnet@unicaen.fr

Résumé

Les agents autonomes deviennent récurrents dans de nombreux domaines tels que le transport, la santé ou le commerce à haute fréquence. Leurs comportements autonomes peuvent soulever des questions éthiques lorsque leurs actions ont un impact sur les humains, questions que les agents doivent prendre en compte dans leur raisonnement. Alors que l'éthique représente de manière relative (i.e. par rapport à un groupe, des normes ou un ensemble de valeurs morales) ce qui est juste ou injuste, la morale décrit de manière absolue ce qui est bien ou mal. Si certaines approches représentent le raisonnement moral avec des logiques déontiques, utilisant des opérateurs modaux, d'autres décrivent le raisonnement éthique avec un cadre d'argumentation valué comme celui proposé par Katie Atkinson et Trevor Bench-Capon. Cependant, il n'existe pas à notre connaissance de cadre qui combine à la fois logique modale et argumentation pour raisonner sur l'éthique. Dans cet article, nous proposons un nouveau cadre d'argumentation où les arguments sont construits à partir d'un cadre général de logique modale normale. Celui-ci permet d'exprimer et de raisonner sur un ensemble de valeurs morales via des opérateurs déontiques dyadiques (voire n-aires) ou encore de raisonner sur les états mentaux des agents. Toutefois dans ce cadre, les attaques classiques fondées sur les contradictions entre propositions (i.e. réfutation, contre-attaque, ou encore défaite) posent problème. En effet, il n'y a pas de contradiction nécessaire entre les opérateurs modaux et certaines attaques ne peuvent pas être déduites. Nous proposons alors un nouveau mécanisme permettant de définir des attaques fondées sur des logiques modales en décrivant les formes que doivent prendre les contradictions.

Abstract

Autonomous agents become recurrent in many domains such as transportation, health care, or high-frequency trading. When the actions impact humans, autonomous behaviors may rise ethical issues that agents must consider in their reasoning. While ethics represents in a relative way (with e.g. norms, or moral worth) what is fair or unfair, moral describes in absolute way what is right or wrong. Some approaches represent moral reasoning with deontic logics, which use modal operators, while reasoning about ethics may be represented with a value-based argumentation framework as it has been proposed in the literature. However, there is no work that combines both modal logic and argumentation to reason about ethics. In this article, we propose a new argumentation framework where arguments are built from a general normal modal logic framework. It allows to express different kinds of modal operators of interest in ethical reasoning, such as moral worth operators, dyadic deontic operators or the mental states of the agents. However in logic-based argumentation framework, since standard attacks (e.g. rebuttal, undercut, defeater) are defined as a contradiction between two propositions, it raises a problem: since there is no necessary contradiction between modal operators, they are no longer sufficient when we need to characterize attacks between modal operators. To solve this problem, we define attacks based on modal logic thank to a description of how the contradictions must arise.

1 Introduction

L'utilisation d'agents autonomes devient récurrente dans de nombreux domaines tels que le transport, les soins de santé ou le commerce à haute fréquence. Leurs comportements autonomes peuvent soulever des questions éthiques lorsque leurs actions ont un impact sur les humains, questions que les agents doivent représenter et intégrer dans leur raisonnement. Par exemple, raisonner sur les codes de conduite peut être important pour des agents autonomes médicaux qui doivent gérer le secret médical ou le respect de la dignité. À cette fin, de nombreux cadres ont été développés pour permettre la conception d'agents autonomes intégrant des concepts éthiques explicites [5, 6, 8, 15, 19,

25, 26, 27, 32, 34]. Si ces approches modélisent certains aspects spécifiques du raisonnement éthique, elles ne rendent pas entièrement compte de la manière dont les êtres humains effectuent un raisonnement éthique. Par exemple, lorsqu'un être humain prend une décision, il tient compte de ses désirs, de ses croyances sur une situation, de ses émotions ou de valeurs morales. Par *valeur morale*, nous entendons, ce qui décrit la conformité d'un comportement aux mœurs, valeurs et usages d'un groupe ou d'une seule personne en associant une valeur bonne ou mauvaise à des combinaisons d'actions et de contextes [17, 22, 33, 20]. Cependant, dans certaines situations, les êtres humains ne sont pas capables de donner une évaluation différente entre deux options, chacune soutenue par une valeur morale différente et chacune apportant un regret après leur execution. Nous parlons alors de *dilemme moral*. Pour résoudre les dilemmes moraux, les êtres humains se livrent à un *raisonnement éthique* pour rechercher des arguments qui soutiendront certaines valeurs morales, considérées comme importantes dans la situation [24].Dans cet article, nous avons pour but de modéliser un tel processus de raisonnement éthique en considérant différentes valeurs morales et en les confrontant les unes aux autres à travers un cadre d'argumentation (AF) [18]. Plus précisement, nous postulons comme [7] qu'un raisonnement éthique peut être représenté par un cadre d'argumentation comme un cadre d'argumentation fondé sur des valeurs (VAF). Dans un VAF, les arguments sont associés à une valeur qui peut correspondre à une valeur morale. Par exemple, un véhicule autonome dans une situation d'urgence peut se représenter deux arguments pour savoir si celui-ci doit dépasser la limitation vitesse pour arriver plus vite à l'hopital. Un premier argument pourrait indiquer que selon la loi, "il est interdit de dépasser la limite de vitesse". Cet argument promeut une valeur *respecter la loi*. Un second peut indiquer qu'"une passagère mourante doit arriver à l'hôpital le plus vite possible" et peut promouvoir une valeur *préserver la vie*. Cependant, dans un VAF, les associations entre arguments et valeurs sont données a priori. C'est pourquoi nous proposons d'aller plus loin en considérant un VAF fondé sur une logique modale. En effet, nous affirmons que le raisonnement selon les valeurs peut être représenté par une logique modale puisque les opérateurs modaux sont bien adaptés pour exprimer des notions de désirs (valeurs égoïstes ou hédoniques) ou de mondes idéaux déontiques. De plus, la logique modale est également adaptée pour exprimer et raisonner sur les connaissances et les croyances d'un agent. Par exemple, dans l'exemple précédent, nous pouvons considérer un autre argument qui exprimerait que "la passagère n'est pas mourante pas car elle souffre d'une luxation de l'épaule". Un tel argument ne promeut aucune valeur morale mais attaque clairement l'argument précédent. Ainsi, la principale contribution de cet article est de fournir un cadre d'argumentation où les arguments et les relations d'attaque sont sémantiquement définis du point de vue des logiques modales. Cet article est organisé de la manière suivante : la section 2 présente l'exemple qui permettra d'illustrer le cadre tout au long de cet article. La section 3 rappelle des notions préliminaires sur l'argumentation formelle. Enfin, nous définissons le cadre en section 4 et 5.

2 Application

L'exemple ci-après a pour objectif de modéliser le raisonnement éthique d'un agent devant décider d'un mode de transport. Pour prendre sa décision, nous faisons l'hypothèse que l'agent considère un ensemble de valeurs, a une préférence sur elles et connait les différents itinéraires associés au mode de transport choisi : prendre sa voiture, son vélo, les transports publics ou aller à pied. Nous supposons aussi que l'agent ne prend que des décisions rationnelles fondés ses préférences. L'exemple 1 introduit quelques éléments formels qui seront repris dans la suite de cet article.

Exemple 1. *Nous considérons un ensemble de valeurs $\mathcal{V}al$, constitué d'une valeur écologique associée au respect de l'environnement (v_{env}), une valeur économique associée à la maximisation de son intérêt personnel (v_{eco}), une valeur associée au respect de consignes données par une autorité gouvernementale (v_{aut}) ainsi qu'une valeur associée au respect de sa santé (v_{health}). Nous considérons donc que l'agent doit décider entre plusieurs actions représentées par les symboles suivants :*

- $\triangle_{car}$ *:= "prendre sa voiture"*
- $\triangle_{p.t.}$ *:= "prendre les transports publics"*
- $\triangle_{bike}$ *:= "prendre son vélo"*
- $\triangle_{walk}$ *:= "partir à pied"*

Nous représentons par $\mathcal{P}$ un ensemble de variables propositionnelles décrivant des informations sur la situation et l'agent. $\mathcal{P}$ contient les éléments suivants :

- has_{bike} *:= "l'utilisateur possède un vélo"*
- has_{car} *:= "l'utilisateur possède une voiture"*
- $crisis_{health}$ *:= "il y a une crise sanitaire"*
- *congestion := "il y a un embouteillage"*
- *emergency := "il s'agit d'une urgence médicale"*

3 Notions préliminaires

Dans un premier temps, nous rappelons quelques notions élémentaires sur l'argumentation valuée. Dans un second temps, nous présentons la littérature qui combine à la fois logiques modales et argumentation.

3.1 Argumentation valuée

Classiquement un VAF étend le cadre de l'argumentation abstraite introduit par Dung [18]. Ce dernier l'étend

en introduisant des concepts de valeurs, et de préférences sur les valeurs [7].

Définition 1 (VAF)**.** *Un* VAF *est un tuple* $\langle \mathcal{A}, \mathcal{R}, \mathcal{V}al, \mu, \succ \rangle$ *tel que :*

- $\mathcal{A}$ *est un ensemble non vide d'arguments,*
- $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ *est une relation binaire décrivant la notion d'attaque entre arguments de* $\mathcal{A}$*,*
- $\mathcal{V}al$ *est un ensemble non vide de valeurs,*
- $\mu : \mathcal{A} \rightarrow \mathcal{V}al$ *est une fonction associant à chaque argument une valeur,*
- $\succ \subseteq \mathcal{V}al \times \mathcal{V}al$ *est une relation transitive de préférence entre les valeurs.*

Un argument en écarte un autre s'il l'attaque et que l'argument attaqué n'est à l'argument attaquant au regard des valeurs qui leur sont respectivement associées.

Définition 2 (Écart)**.** *Soit* $\langle \mathcal{A}, \mathcal{R}, \mathcal{V}al, \mu, \succ \rangle$ *un VAF. Un argument* $A \in \mathcal{A}$ écarte *un argument* $B \in \mathcal{A}$*, noté* $A \mathcal{R}_{vaf} B$ *si, et seulement si,* $A\mathcal{R}B$ *i.e. A attaque B, et* $\mu(B) \not\succ \mu(A)$ *i.e. la valeur* $\mu(B)$ *n'est pas préférée à la valeur* $\mu(A)$*.*

Définition 3 (Sans conflit et acceptabilité)**.** *Soient* $\mathcal{R}_{vaf}$ *une attaque VAF sur* $\langle \mathcal{A}, \mathcal{R}, \mathcal{V}al, \mu, \succ \rangle$*, et* $S \subseteq \mathcal{A}$*.* S *est un ensemble d'argument* sans conflit *par rapport à* $\mathcal{R}_{vaf}$ *si, et seulement si,* $\nexists A, B \in S$ *t.q.* $A \mathcal{R}_{vaf} B$*. Pour tout* $A \in \mathcal{A}$*,* A *est* acceptable *pour* S *si, et seulement si,* $\forall B \in \mathcal{A}$ *si* $B \mathcal{R}_{vaf} A$*, alors* $\exists C \in S, C \mathcal{R}_{vaf} B$*.*

Pour caractériser des arguments dits "admissibles", nous utilisons la notion de sémantique d'admissibilité ou extensions [18]. Comme il en existe plusieurs dans la littérature, nous rappelons les principales :

Définition 4 (Extensions)**.** *Soit* $S \subseteq \mathcal{A}$*.*

- S *est une* extension admissible *ssi* S *est sans conflit et tout argument* $A \in S$ *est acceptable pour* S*.*
- S *est une* extension complète *ssi* S *est admissible et contient tous les arguments acceptables pour* S*.*
- S *est une* extension préférée *ssi* S *est une extension admissible maximale pour* $\subseteq$*.*
- S *est une* extension stable *ssi* S *est sans conflit, et* $\forall A \in \mathcal{A} \setminus S$*,* $S \mathcal{R}_{vaf} A$*.*

Dans la suite, nous ne faisons pas de supposition sur la sémantique utilisée.

3.2 L'argumentation fondée sur la logique

De nombreux travaux combinent logique et argumentation : d'une part les travaux qui utilisent la logique pour modéliser l'argumentation formelle [12, 14, 23, 31], et d'autre part les travaux qui considère l'argumentation comme une structure issue d'une logique [10, 21, 1, 4, 13, 29]. La première approche ne permet pas d'exprimer des arguments fondés sur des états mentaux ou des modalités déontiques car la relation d'accessibilité dans ces modèles représentent une relation d'attaque entre arguments. C'est pourquoi, nous nous intéressons uniquement à la seconde approche. Par exemple, [10] définit une argumentation fondée sur la logique propositionnelle : un argument est un couple (prémisse, conclusion) ; la relation d'attaque est définie sur les contradictions déduites dans la logique. Pour aller plus loin, [13] propose une argumentation fondée sur le calcul des hyperséquents et une logique modale S5. Ici, $A = (\Gamma, \phi)$ est un argument avec Γ un ensemble de formules de S5 supportant la conclusion ϕ si, et seulement si, $\Gamma \Rightarrow \phi$ est un séquent prouvé. Cependant, il n'y a qu'un type de modalité et [13] ne fournit pas de sémantique à leur opérateur. Enfin, plutôt que de considérer une sémantique particulière associée à une logique, [1] propose une généralisation de l'argumentation en considérant une fonction C représentant l'opérateur de conséquence syntaxique $\vdash$ (i.e. ce qui peut être déduit d'un système de preuve). Cependant, toutes ces approches ne fournissent pas de sémantique associé aux opérateurs de preuves syntaxiques. À notre connaissance, si certains travaux introduisent une sémantique de logique du premier ordre pour les arguments comme [9], il n'existe aucun travail proposant un cadre d'argumentation fondé sur la sémantique des logiques modales normales.

4 Logique multimodale normale et n-aire

Afin de pouvoir exprimer certaines logiques de la morale comme DL-MA [27] ou les logiques déontiques dyadiques [30], nous considérons un cadre de logique multimodal normale avec des opérateurs n-aires comme dans [11]. Pour ce faire, nous devons tout d'abord définir la notion de type de similarité.

Définition 5 (Type de similarité)**.** *Un* type de similarité *est un couple* $\tau = (O, \rho)$ *où* O *représente un ensemble non vide de symboles pour chaque opérateur modal et* $\rho : O \rightarrow \mathbb{N}$ *associe une arité à chaque opérateur modal.*

Un type de similarité représente un ensemble d'opérateurs modaux avec une arité qui lui est assignée. Cette arité peut être supérieure ou égale à 0. Les éléments de O sont usuellement représentés par un "triangle" comme $\triangle_0$, $\triangle_1$, ... pour les opérateurs modaux d'arité 0 (appelés opérateurs nullaires). Ces opérateurs peuvent être vus comme des variables propositionnelles au niveau du cadre de Kripke et pas au niveau d'un modèle. Par exemple, [27] représente le choix des agents par $choice(a_i)$ signifiant que dans le monde courant l'agent i choisit l'action a.

Exemple 2. *Considérons* $\triangle_{car}$*,* $\triangle_{p.t.}$*,* $\triangle_{bike}$*,* $\triangle_{walk}$ *les choix de l'agent de respectivement prendre sa voiture, prendre les transports publics, partir en vélo, ou partir à pieds. Il s'agit ici d'opérateurs nullaires qui représentent les choix.*

Concernant les arités supérieures, nous considérons par convention que l'ensemble O contient seulement des modalités de type "diamants". Ainsi, si un opérateur modal $\triangle$ est un opérateur monadique, i.e. $\rho(\triangle) = 1$, alors, il s'agit de l'opérateur standard $\diamond$ i.e. une modalité de type "existence" (noté $\exists$-type). Pour $\rho(\triangle) > 1$, un opérateur n-aire $\triangle$ signifie qu'il existe un N-uplet de mondes accessibles où chaque formule à l'interieur des paramètres est vraie. Nous représentons par ∇ le dual de $\triangle$. Quand ∇ est un opérateur monadique, il est équivalent à l'opérateur standard $\Box$. Pour un type de similarité $\tau = (O, \rho)$, nous représentons O_{dual} l'ensemble des opérateurs duaux dans O, i.e. tous les types "pour tout" (noté $\forall$-type). Pour les modalités d'arité supérieure, ∇ signifie que quelque soit les N-uplet de mondes accessibles, au moins une formule contenue dans les paramètres de la modalité est vérifiée. De manière intéressante, les valeurs peuvent être soit représentées par des opérateurs nullaires (pour exprimer des valeurs promues dans un monde) ou par des opérateurs d'arité supérieure pouvant être interprété comme des obligations déontiques au nom d'une valeur spécifique.

Exemple 3. *Nous pouvons représenter "prendre son vélo promeut la valeur environnementale" soit en considérant un opérateur nullaire $\triangle_{env}$ qui est vrai pour chaque monde où $\triangle_{bike}$ est vrai, ou avec un opérateur déontique monadique de SDL e.g. $\nabla_{env}(\triangle_{bike})$, ou de façon plus expressive, avec un opérateur n-aire. Dans la suite, nous considérons des opérateurs dyadiques [30] dont un paramètre représente le contexte d'application d'une règle déontique et le second paramètre représente ce qui devrait être dans un monde idéal associé à ce contexte. Intuitivement, ces opérateurs représentent une contrainte sur trois mondes (w, v, u) où w est un monde dans lequel une norme (v, u) peut s'appliquer, norme signifiant que u est un monde idéal pour un monde v où le contexte est vérifié.*

Contrairement au système logique introduit dans [11], nous considérons ici des sémantiques alternatives pour $\triangle$ et ∇ : un opérateur n-aire, noté $\Box$, est défini comme pour tout N-uplet de mondes accessibles, chaque formule à l'intérieur de $\Box$ est vérifiée. $\Box$ a aussi sa modalité duale, notée $\diamond$, qui est définie comme il existe un N-uplet tel que au moins une formule est vraie dans son monde respectif. Les ensembles des $\diamond$ et $\Box$ sont représentés respectivement par O^* et O^*_{dual}. Pour lier chaque opérateur modal avec son dual et ses sémantiques alternatives, nous notons $O^\Omega = O \cup O_{dual} \cup O^* \cup O^*_{dual}$ l'ensemble de tous les opérateurs modaux et une fonction $\varrho : O^\Omega \to O \times O_{dual} \times O^* \times O^*_{dual}$ qui associe à chaque modalité son dual et ses alternatives. Dans la suite, nous considérons que le domaine de ρ est étendu à O^Ω, i.e. $\rho : O^\Omega \to \mathbb{N}^*$ et ρ est telle que : $\forall \triangle \in O$, si $\varrho(\triangle) = (\triangle, \nabla, \diamond, \Box)$, alors $\rho(\triangle) = \rho(\nabla) = \rho(\diamond) = \rho(\Box)$.

Définition 6 (Langage des logiques modales)**.** *Soit $\mathcal{P}$ un ensemble d'atomes propositionnels et $\tau = (O, \rho)$ un type de similarité. Nous considérons le langage des formules bien formées (fbf) généré par la grammaire sous forme de Backus-Naur, pour tout $p \in \mathcal{P}$, $\triangle \in O$, $\star \in \varrho(\triangle)$:*

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \star(\phi_1, \ldots, \phi_{\rho(\triangle)})$$

De façon standard :
— $\top := \neg\bot$,
— $\phi \vee \psi := \neg(\neg\phi \wedge \neg\psi)$,
— $\phi \Rightarrow \psi := \neg\phi \vee \psi$

De plus si $\rho(\triangle) > 0$:
— $\nabla(\phi_1, \ldots, \phi_{\rho(\triangle)}) := \neg\triangle(\neg\phi_1, \ldots, \neg\phi_{\rho(\triangle)})$
— $\Box(\phi_1, \ldots, \phi_{\rho(\triangle)}) := \neg\diamond(\neg\phi_1, \ldots, \neg\phi_{\rho(\triangle)})$

Nous ne considérons ici qu'une sémantique de système K car elle fournit un cadre minimaliste, sans contrainte sur les relations d'accessibilité. Cependant, elle pourrait être étendue en contraignant les relations d'accessibilité.

Définition 7 (Sémantique des logiques multi-modales normales n-aires)**.** *Soit $\tau = (O, \rho)$ un type de similarité. Un τ-*modèle *est un N-uplet $\mathcal{M} = \langle \mathcal{W}, \{\mathcal{R}_\triangle\}_{\triangle \in O}, \mathcal{V} \rangle$ tel que :*
— *$\mathcal{W}$ est un ensemble non vide de mondes,*
— *$\forall \triangle \in O, \mathcal{R}_\triangle$ est une relation $(\rho(\triangle) + 1)$-aire* [1]*,*
— *$\mathcal{V} : \mathcal{P} \to 2^{\mathcal{W}}$ est une fonction de valuation.*

*$C = \langle \mathcal{W}, \{\mathcal{R}_\triangle\}_{\triangle \in O} \rangle$ est un τ-*cadre *et $\mathcal{M}$ est un modèle dans C ssi $\mathcal{M}$ appartient à la classe des modèles $C_{\mathbb{M}} = \{\mathcal{M} : \exists \mathcal{V}, \mathcal{M} = \langle C, \mathcal{V} \rangle\}$. Pour tout $w \in \mathcal{W}$, et $\phi, \psi \in \mathcal{L}$, $\triangle \in O$, $\varrho(\triangle) = (\triangle, \nabla, \diamond, \Box)$, et $p \in \mathcal{P}$:*

1. $\mathcal{M}, w \not\models \bot$
2. $\mathcal{M}, w \models p$ *ssi* $w \in V(p)$
3. $\mathcal{M}, w \models \neg\phi$ *ssi* $\mathcal{M}, w \not\models \phi$
4. $\mathcal{M}, w \models \phi \wedge \psi$ *ssi* $\mathcal{M}, w \models \phi$ *et* $\mathcal{M}, w \models \psi$

Pour les cas où $\rho(\triangle) \geq 1$:

5. $\mathcal{M}, w \models \triangle(\phi_1, \ldots, \phi_{\rho(\triangle)})$ *ssi* $\exists (v_1, \ldots, v_{\rho(\triangle)}) \in \mathcal{W}^{\rho(\triangle)}$ *:* $w\mathcal{R}_\triangle(v_1, \ldots, v_{\rho(\triangle)})$ *et* $\forall k \in \mathbb{N}^*, k \leq \rho(\triangle)$, $v_k \models \phi_k$
6. $\mathcal{M}, w \models \nabla(\phi_1, \ldots, \phi_{\rho(\triangle)})$ *ssi* $\forall (v_1, \ldots, v_{\rho(\triangle)}) \in \mathcal{W}^{\rho(\triangle)}$ *: si* $w\mathcal{R}_\triangle(v_1, \ldots, v_{\rho(\triangle)})$ *alors* $\exists k \in \mathbb{N}^*, k \leq \rho(\triangle)$, $v_k \models \phi_k$
7. $\mathcal{M}, w \models \Box(\phi_1, \ldots, \phi_{\rho(\triangle)})$ *ssi* $\forall (v_1, \ldots, v_{\rho(\triangle)}) \in \mathcal{W}^{\rho(\triangle)}$ *: si* $w\mathcal{R}_\triangle(v_1, \ldots, v_{\rho(\triangle)})$ *alors* $\forall k \in \mathbb{N}^*, k \leq \rho(\triangle)$, $v_k \models \phi_k$
8. $\mathcal{M}, w \models \diamond(\phi_1, \ldots, \phi_{\rho(\triangle)})$ *ssi* $\exists (v_1, \ldots, v_{\rho(\triangle)}) \in \mathcal{W}^{\rho(\triangle)}$ *:* $w\mathcal{R}_\triangle(v_1, \ldots, v_{\rho(\triangle)})$ *et* $\exists k \in \mathbb{N}^*, k \leq \rho(\triangle)$, $v_k \models \phi_k$

Pour les cas où $\rho(\triangle) = 0$:

9. $\mathcal{M}, w \models \triangle$ *ssi* $w \in \mathcal{R}_\triangle$
10. $\mathcal{M}, w \models \nabla$ *ssi* $\mathcal{R}_\triangle = \mathcal{W}$
11. $\mathcal{M}, w \models \Box$ *ssi* $\mathcal{R}_\triangle = \emptyset$

1. Remarquons que chaque opérateur $\star \in \varrho(\triangle)$ est défini sur la même relation $(\rho(\triangle) + 1)$-aire.

12. $\mathcal{M}, w \models \Diamond$ *ssi* $\mathcal{R}_\triangle \neq \emptyset$

Pour résumer, nous avons quatre types d'opérateurs modaux : $\triangle$ est une modalité $(\exists, \wedge)$-type appartenant à l'ensemble $\mathcal{O}$, ∇ est une modalité $(\forall, \vee)$-type, appartenant à l'ensemble $\mathcal{O}_{dual}$, $\Diamond$ est une modalité $(\exists, \vee)$-type, appartenant à l'ensemble $\mathcal{O}^*$ et $\Box$ est une modalité $(\forall, \wedge)$-type, appartenant à l'ensemble $\mathcal{O}^*_{dual}$. Par modalités $(\exists, \wedge)$-type, nous entendons les modalités ayant une sémantique où nous avons un quantificateur $\exists$ sur les N-uplets accessibles et une "conjonction" sur la vérité des formules par rapport à leurs mondes respectifs du N-uplet, tandis que les modalités $(\forall, \vee)$-type ont une sémantique fondée sur le quantificateur $\forall$ pour les mondes accessibles et une "disjonction" sur la vérité des formules par rapport à leur monde respectif du N-uplet. Pour le cas où $\rho(\triangle) = 1$, i.e. $\triangle$ est un opérateur monadique, il y a équivalence sémantique entre les opérateurs : $\triangle \equiv \Diamond$ et $\Box \equiv \nabla$.

Exemple 4. *Considérons la formule* $O(crisis_{health}, \neg\triangle_{p.t.})$ *où O est un opérateur déontique dyadique [30]. Cette formule peut signifier que dans le cas où il y a une crise sanitaire, il est fortement déconseillé de prendre les transports publics. Dans notre cadre, cet opérateur peut être exprimé sémantiquement avec un opérateur* ∇ *ayant donc une sémantique de* $(\forall, \vee)$*-type, i.e* $\nabla_O(\neg crisis_{health}, \neg\triangle_{p.t.})$*. En effet, le contexte doit être représenté avec une négation dans les modalités* $(Qt, \vee)$*-type où* $Qt \in \{\forall, \exists\}$*. Pour mieux comprendre, il suffit de remarquer que pour toutes les modalités* $(Qt, \vee)$*-type, celles-ci considèrent une "disjonction" sur les mondes accessibles du N-uplet, e.g.* $w \models \nabla(\neg\phi, \psi)$ *est vrai ssi* $\forall (u, v) \in \mathcal{R}_\triangle(w), u \models \neg\phi$ *ou* $v \models \psi$*. Ainsi, nous pouvons facilement traduire cette disjonction, de manière équivalente, en, si* $u \models \phi$*, alors* $v \models \psi$*.*

Rappelons les notions de validité et satisfiabilité d'une formule. ϕ est valide dans $\mathcal{M}$ (noté $\mathcal{M} \models \phi$) si, et seulement si, pour tous les mondes $w \in \mathcal{W}$, ϕ est satisfiable dans w i.e. $\mathcal{M}, w \models \phi$ est vraie. Une formule ϕ est valide dans un cadre $\mathcal{C}$ (noté $\models_{\mathcal{C}} \phi$ ou $\mathcal{C} \models \phi$) si, et seulement si, pour tous les modèles $\mathcal{M}$ construits sur $\mathcal{C}$, $\mathcal{M} \models \phi$. Dans ce cas, ϕ est une tautologie de $\mathcal{C}$. Soit $\Gamma \subseteq \mathcal{L}$ un ensemble de formules, $\Gamma \models_{\mathcal{C}} \phi$ signifie que ϕ est une *conséquence sémantique* de Γ si, et seulement si, pour tout modèle $\mathcal{M}$ dans $\mathcal{C}$, si $\mathcal{M} \models \Gamma$ i.e. $\mathcal{M} \models \bigwedge_{\psi \in \Gamma} \psi$ alors $\mathcal{M} \models \phi$.

Exemple 5. *Dans notre application, nous considérons le type de similarité* $\tau = (\mathcal{O}, \rho)$ *où* $\mathcal{O} = Dyadic_\triangle \cup Unary \cup Nullary_\triangle$*. Ici,* $Dyadic_\triangle = \{\triangle_{gov}, \triangle_{env}, \triangle_{sport}, \triangle_{time}, \}$ *représente tous les opérateurs dyadiques qui sont respectivement associés aux conseils des autorités, les recommandations environnementales, les recommandations pour la santé, et ce qui permet de favoriser le gain de temps.* $Unary = \{\Diamond\}$ *contient un opérateur unaire pour décrire ce qui est possible et* $Nullary = \{\triangle_{bike}, \triangle_{p.t.}, \triangle_{walk}, \triangle_{car}\}$ *sont les choix. Ainsi, la fonction* ρ *est telle que :* $\forall \star \in Dyadic_\triangle, \rho(\star) = 2, \rho(\Diamond) = 1, \forall \star \in Nullary, \rho(\star) = 0$*. Par conséquent, nous considérerons le* τ*-cadre* $\mathcal{C} = \langle \mathcal{W}, \{\mathcal{R}_\triangle\}_{\triangle \in \mathcal{O} \setminus \{\Diamond\}}, \mathcal{R}_\Diamond \rangle$ *où la sémantique de chaque opérateur est donnée par le type de symbole utilisé dans la formule et* $\mathcal{O}^\Omega$ *représente l'ensemble de tous les symboles pour les opérateurs modaux* $\triangle$*,* ∇ *et leur sémantique alternative* $\Box$ *et* $\Diamond$*.*

Système axiomatique GK En étendant les travaux de [11] aux modalités n-aires $\Box$, nous proposons un système axiomatique correct et complet, appelé GK, qui généralise les cadres de Kripke. Nous notons $\vdash \phi$ pour signifier que ϕ est un théorème. Nous considérons les axiomes classiques du calcul propositionnel (CP), le modus ponens (MP) et la substitution uniforme (SUB). Pour tout $\triangle \in \mathcal{O}$, t.q. $\rho(\triangle) \geq 1$, $\varrho(\triangle) = (\triangle, \nabla, \Diamond, \Box)$.

La règle de nécessitation (NEC) :

$$\text{Pour } \vdash \phi :\vdash \nabla(\bot, \ldots, \phi, \ldots, \bot) \wedge \Box(\top, \ldots, \phi, \ldots, \top)$$

Les axiomes $\forall k \in [|1, \rho(\triangle)|]$, (K^k_∇) :

$$\vdash \nabla(\phi_1, \ldots, \phi_k \Rightarrow \psi_k, \ldots, \phi_{\rho(\triangle)}) \\ \Rightarrow \nabla(\phi_1, \ldots, \phi_k, \ldots, \phi_{\rho(\triangle)}) \Rightarrow \nabla(\phi_1, \ldots, \psi_k, \ldots, \phi_{\rho(\triangle)})$$

Les axiomes $\forall k \in [|1, \rho(\triangle)|]$, $(K^k_\Box)$:

$$\vdash \Box(\phi_1, \ldots, \phi_k \Rightarrow \psi_k, \ldots, \phi_{\rho(\triangle)}) \\ \Rightarrow \Box(\phi_1, \ldots, \phi_k, \ldots, \phi_{\rho(\triangle)}) \Rightarrow \Box(\phi_1, \ldots, \psi_k, \ldots, \phi_{\rho(\triangle)})$$

Les axiomes d'interaction $(Int(\Box, \nabla))$:

$$\vdash \Box(\phi_1, \ldots, \phi_{\rho(\triangle)}) \Rightarrow \nabla(\phi_1, \ldots, \phi_{\rho(\triangle)})$$

Les axiomes de dualité pour $(Dual_\nabla)$ et $(Dual_\Box)$:

$$\vdash \nabla(\phi_1, \ldots, \phi_{\rho(\triangle)}) \Leftrightarrow \neg\triangle(\neg\phi_1, \ldots, \neg\phi_{\rho(\triangle)})$$

$$\vdash \Box(\phi_1, \ldots, \phi_{\rho(\triangle)}) \Leftrightarrow \neg\Diamond(\neg\phi_1, \ldots, \neg\phi_{\rho(\triangle)})$$

Les règles d'équivalence (RE), pour tout $\star \in \varrho(\triangle)$:

$$\text{Pour } \vdash \phi_k \Leftrightarrow \psi_k :\vdash \star(\ldots, \phi_k, \ldots)) \Leftrightarrow \star(\ldots, \psi_k, \ldots)$$

Nous avons aussi besoin de considérer deux cas spécifiques pour les opérateurs monadiques et nullaires. Si $\rho(\triangle) = 1$, nous considérons les axiomes précédents, et ajoutons : $\vdash \Box\phi \Leftrightarrow \nabla\phi$ et $\vdash \Diamond\phi \Leftrightarrow \triangle\phi$. Si $\rho(\triangle) = 0$, les axiomes précédents ne tiennent plus et nous avons seulement (PC), (MP), (RE), (SUB).

Définition 8 (Système de déduction)**.** *Soit* Σ *un ensemble de formules et* ϕ *une formule de* $\mathcal{L}$*. Nous disons que* ϕ *est* deductible *de* Σ*, noté* $\Sigma \vdash \phi$*, si, et seulement si :*

— *si* $\Sigma = \emptyset$*, alors* $\vdash \phi$*,*

— *sinon* $\exists n \in \mathbb{N}^*, \exists \psi_1, \ldots, \psi_n \in \Sigma, \vdash \psi_1 \wedge \ldots \wedge \psi_n \Rightarrow \phi$*.*

Nous conjecturons [2] que GK est fortement correct et fortement complet.

Conjecture 1. *Soit Γ un ensemble de fbf et ϕ une fbf.*

1. *GK vérifie les théorèmes de déduction (a) et (b) :*
   (a) $\Gamma \models \phi$ *ssi* $\models \Gamma \Rightarrow \phi$ *(forme sémantique)*
   (b) $\Gamma \vdash \phi$ *ssi* $\vdash \Gamma \Rightarrow \phi$ *(forme syntaxique)*
2. *GK est fortement correct et fortement complet w.r.t. le cadre des logiques multi-modales normales et généralisées aux opérateurs n-aires i.e. :*

$$\Gamma \vdash \phi \text{ si, et seulement si, } \Gamma \models \phi$$

Exemple 6. *Dans notre application, nous avons besoin d'exprimer une généralisation de l'axiome de vérité, notée T_τ pour les modalités de type $\Box$ et $\Diamond$. À cette fin, nous supposons que $\mathcal{R}_\Diamond$ du τ-cadre est telle que :*

$$\forall w \in \mathcal{W}, \mathcal{R}_\Diamond(w) = \{w\} \cup \bigcup_{\triangle' \in Dyadic} \{u, v : (u, v) \in \mathcal{R}_{\triangle'}(w)\}$$

Cette contrainte exprime la tautologie suivante :

$$\models \Box\phi \Rightarrow \Big(\phi \bigwedge_{\triangle' \in Dyadic : \varrho(\triangle') = (\triangle', \nabla', \Diamond', \Box')} (\Box'(\top, \phi) \wedge \Box'(\phi, \top) \wedge \Box'(\phi, \phi) \wedge \nabla'(\bot, \phi) \wedge \nabla'(\phi, \bot) \wedge \nabla'(\phi, \phi))\Big)$$

Elle signifie que si une formule est nécessairement vraie, alors elle est aussi vraie dans tous les mondes accessibles à partir des autres relations.

Exemple 7. *Nous considérons les hypothèses suivantes :*

$\Gamma_1 \stackrel{\text{def}}{=} \{\nabla_{aut}(\neg crisis_{health}, \neg\triangle_{p.t.})\}$

$\Gamma_2 \stackrel{\text{def}}{=} \{\triangle_{car} \rightarrow has_{car}, \triangle_{bike} \rightarrow has_{bike}\}$

$\Gamma_3 \stackrel{\text{def}}{=} \{\nabla_{health}(\bot, \triangle_{bike})\}$

$\Gamma_4 \stackrel{\text{def}}{=} \{\nabla_{time}(\neg emergency \vee congestion, \triangle_{car})\}$

$\Gamma_5 \stackrel{\text{def}}{=} \{\neg\triangle_{time}(emergency, \triangle_{walk})\}$

$\Gamma_6 \stackrel{\text{def}}{=} \{\nabla_{env}(\bot, \triangle_{bike} \vee \triangle_{walk} \vee \triangle_{p.t.})\}$

$\Gamma_7 \stackrel{\text{def}}{=} \{\nabla_{env}(\neg congestion, \neg\triangle_{car})\}$

$\Gamma_8 \stackrel{\text{def}}{=} \{\Box\texttt{XOR}(\triangle_{car}, \triangle_{p.t.}, \triangle_{bike}, \triangle_{walk})\}$

$\Gamma_9 \stackrel{\text{def}}{=} \{\Box(has_{car} \wedge \neg has_{bike} \wedge crisis_{health} \wedge \neg congestion \wedge \neg emergency)\}$

2. La preuve est laissée pour des travaux futurs. Intuitivement, la complétude découlerait de l'utilisation d'un modèle canonique $M = (W^c, \mathcal{R}*, V^c)$ où W^c est l'ensemble des ensembles maximaux cohérents de formules (comme défini par exemple dans [11]). Nous devrions alors montrer que le modèle canonique fonctionnerait pour GK avec des modalités de type $\Diamond$ et des modalités de type $\Box$. Cela signifierait que si $\Diamond(\phi_1, \ldots, \phi_n) \in w$, alors il existerait $i \in [|1, n|]$ tel que $\mathcal{R}_i * (w) \cup \phi_i$ serait cohérent et où $\mathcal{R}_i * (w) := \{\phi'_i : \Box(\phi'_1, ..., \phi'_i, ..., \phi'_n) \in w\}$. Cela signifierait que notre modèle canonique fonctionnerait pour l'existence d'un monde accessible dans W^c, c'est-à-dire un ensemble maximal cohérent par application du lemme de Lindenbaum. La preuve du lemme de vérité découlerait ensuite d'une preuve par récurrence sur le degré des formules comme il est standard de le faire. Nous conclurions alors que pour tout w dans W^c, $M^c, w \models \phi$ i.e. ϕ dans w. Enfin, la preuve de la complétude découlerait de l'utilisation des propriétés sur les ensembles maximaux cohérents et du lemme de vérité.

Γ_1 *signifie que les autorités recommendent de ne pas prendre les transports publics quand il y a une crise sanitaire.* Γ_2 *représente la condition, faire du vélo ou conduire implique d'avoir un vélo ou une voiture.* Γ_3 *signifie que faire du vélo est obligatoire dans tous les contextes du point de vue de la santé.* Γ_4 *signifie que s'il y a une urgence et qu'il n'y a pas d'embouteillage, alors il est obligatoire de prendre sa voiture pour économiser du temps.* Γ_5 *signifie qu'il n'est pas possible qu'en cas d'urgence, marcher vous fasse gagner du temps.* Γ_6 *signifie que peu importe le contexte, il est obligatoire de prendre son vélo, marcher ou prendre les transports publics pour le bien de l'environnement.* Γ_7 *signifie qu'il est obligatoire de ne pas prendre sa voiture quand il y a des embouteillages.* Γ_8 *signifie qu'il est nécessaire que l'agent choisisse un seul choix au même moment (e.g. l'agent ne peut pas à la fois prendre sa voiture et son vélo).* Γ_9 *décrit la situation initiale de l'agent.*

Exemple 8. *Nous montrons à partir de l'exemple 7 :*

$$\Gamma_5 \vdash \nabla_{time}(\neg emergency, \neg\triangle_{walk})$$

$$\Gamma_2 \vdash \neg has_{bike} \Rightarrow \neg\triangle_{bike}$$

$$\Gamma_2 \cup \Gamma_9 \vdash \neg\triangle_{bike}$$

$$\Gamma_6 \cup \Gamma_8 \vdash \nabla_{env}(\bot, \neg\triangle_{car})$$

$$\Gamma_8 \vdash \texttt{XOR}(\triangle_{car}, \triangle_{p.t.}, \triangle_{bike}, \triangle_{walk})$$

$$\Gamma_9 \vdash \neg emergency \vee congestion$$

Remarquons que nous déduisons de Γ_6 *et* Γ_8 *que, peu importe le contexte, il n'est pas bon pour l'environnement de prendre sa voiture, ce qui est plus fort que l'hypothèse* Γ_7.

5 VAF fondé sur les logiques modales

Nous définissons les arguments puis les relations d'attaque pour les VAF fondés sur les logiques modales.

5.1 Arguments

Dans la littérature, il est usuel de considérer un argument comme un couple (Γ, ϕ) où Γ est un ensemble de formules, nommées *prémisses* ou *support*, et implique une formule ϕ, appelée *conclusion* [10, 29, 1, 4, 13]. Usuellement, de tels arguments sont définis sur la base des *implications syntaxiques* en supposant que la logique est connue et bien construite. Ici, nous donnons ensuite une définition pour les arguments déduits à partir de notre système de preuve. Nous appelons un *argument valide* un couple (Γ, ϕ) où la conclusion ϕ est une *conséquence sémantique* des prémisses de Γ.

Définition 9 (Argument valide)**.** *Soit $\tau = (O, \rho)$ un type de similarité, et C un τ-cadre. Un* argument valide *dans C est un couple $A = (\Phi, \alpha)$ t.q. :*

1. $\Phi \subseteq \Delta$ *où* $\Delta \subseteq \mathcal{L}$ *est fini* [3],
2. $\Phi \not\models_C \bot$ *i.e.* Φ *n'est pas inconsistant :* $\exists \mathcal{M}, w \models \Phi$,
3. $\Phi \models_C \alpha$,
4. Φ *est minimal pour* $\subseteq$ *t.q. 1, 2 et 3 sont respectés.*

Φ *est appelée* ***support****, noté* $P(A) = \Phi$*, et* α *sa* ***conclusion****, notée* $C(A) = \alpha$.

Un argument est *déductible* quand il est obtenu à partir d'un système de preuve syntaxique, i.e. avec l'opérateur de déduction $\vdash$ plutôt que $\models$. Il y a généralement équivalence entre argument déductible et argument valide dans un système logique correct et complet. Par abus de langage, nous écrivons $A = (_, \phi)$ comme un raccourci signifiant que l'ensemble des prémisses Φ n'est pas spécifié. Cela correspond à $\exists\, \Phi \subseteq \mathcal{L}$ t.q. $A = (\Phi, \phi)$.

Exemple 9. *Quelques exemples d'arguments déductibles :*

$$A_1 = (\Gamma_1, \neg congestion)$$
$$A_2 = (\Gamma_7, \nabla_{env}(\neg congestion, \neg\triangle_{car}))$$
$$A_3 = (\Gamma_9, \Box\neg congestion)$$
$$A_4 = (\Gamma_8, \Box\texttt{XOR}(\triangle_{car}, \triangle_{p.t.}, \triangle_{bike}, \triangle_{walk}))$$
$$A_5 = (\Gamma_3, \nabla_{sport}(\bot, \triangle_{bike}))$$
$$A_6 = (\Gamma_2 \cup \Gamma_9, \Box\neg\triangle_{bike})$$
$$A_7 = (\Gamma_4, \nabla_{time}(\neg emergency \vee congestion, \triangle_{car}))$$
$$A_8 = (\Gamma_8, \neg emergency \vee congestion)$$
$$A_9 = (\Gamma_5, \nabla_{time}(\neg emergency, \neg\triangle_{walk}))$$
$$A_{10} = (\Gamma_9, \neg emergency)$$
$$A_{11} = (\Gamma_6 \cup \Gamma_8, \nabla_{env}(\bot, \neg\triangle_{car}))$$
$$A_{12} = (\Gamma_1, \nabla_{aut}(\neg crisis_{health}, \neg\triangle_{p.t.}))$$

D'autres arguments peuvent être déduits (ex. $(\Gamma_9, crisis_{health})$*) mais ne sont pas considérés car ils ne supportent aucune décision ou n'attaquent aucun argument supportant une décision.*

5.2 Attaques modales

Différents types de relations d'attaque sont définies dans l'argumentation fondée sur la logique [28, 10, 2, 21]. Nous rappelons la sémantique des relations d'attaque les plus utilisées : le **rebuttal**, l'**undercut** et le **defeater**.

Définition 10 (Rebuttal)**.** $A = (\Phi, \alpha)$ *est un* rebuttal *pour* $B = (\Psi, \beta)$ *(noté A Reb B) ssi* $\models \alpha \Leftrightarrow \neg\beta$

Définition 11 (Undercut)**.** $A = (\Phi, \alpha)$ *est un* undercut *pour* $B = (\Psi, \beta)$ *(noté A Und B) ssi il existe :*

$$\Psi' = \{\psi_1, \ldots, \psi_n\} \subseteq \Psi \text{ t.q. } \models \alpha \Leftrightarrow \neg \bigwedge_{\psi_i \in \Psi'} \psi_i$$

Définition 12 (Defeater)**.** $A = (\Phi, \alpha)$ *est un* defeater *pour* $B = (\Psi, \beta)$ *(noté A Def B) ssi il existe :*

$$\Psi' = \{\psi_1, \ldots, \psi_n\} \subseteq \Psi \text{ t.q. } \alpha \models \neg \bigwedge_{\psi_i \in \Psi'} \psi_i$$

3. Puisque il y a un nombre infini de fbf sur $\mathcal{L}$, il est usuel de considérer Δ un ensemble de fbf sur lesquelles les agents peuvent raisonner [10].

Cependant ces attaques ne permettent pas d'exprimer les attaques entre formules utilisant des opérateurs modaux différents car ces formules n'aboutissent pas nécessairement à des incohérences.

Exemple 10. *Considérons les arguments* A_7 *et* A_{11} *qui concluent que (resp.) "s'il y a une urgence et pas d'embouteillage, alors prendre sa voiture fait gagner du temps" et "peu importe le contexte, il est bon pour l'environnement de ne pas prendre sa voiture". Puisque* ∇_{time} *et* ∇_{env} *sont deux opérateurs modaux différents qui n'impliquent pas que les formules idéales soient vraies dans le même monde, il n'y a aucune contradiction. Ainsi, les attaques définies précédemment ne permettent pas de capturer cette relation d'attaque intuitive entre* A_7 *et* A_{11}.

Ainsi, nous proposons un nouveau type d'attaque pour toute logique modale, que nous appelons *attaque modale*. Comme pour les attaques standards, nous avons la notion de *rebuttal modal*, *undercut modal* et *defeater modal*. Cependant dans un cadre de logique modale, les opérateurs modaux peuvent être associés à différentes arités avec des sémantiques différentes. Considérons une formule déontique dyadique $\nabla_O(\neg\phi, \psi)$ signifiant que quand ϕ est vraie, il est obligatoire d'avoir ψ à vrai, et une formule monadique déontique $\Box_K\neg\psi$ signifiant qu'il est obligatoire pour une certaine doctrine que ψ soit au contraire fausse. Supposons que ψ représente "dire un mensonge" et ϕ représente "une vie est menacée". Selon $\Box_K$ il n'est pas permis de mentir, et ce, peu importe le contexte, alors que pour ∇_O il devrait être nécessaire de mentir dans ce contexte ϕ. Intuitivement une attaque entre deux arguments $C = (_, \nabla_O(\neg\phi, \psi))$ et $K = (_, \Box_K\neg\psi)$ serait fondée sur une contradiction entre le second paramètre de l'opérateur dyadique ∇_O et la formule contenue dans l'opérateur monadique $\Box_K$. Cependant, C devrait être attaqué si ϕ est faux, i.e. le contexte n'est pas vérifié. Ainsi, un argument $F = (_, \neg\phi)$ qui conclue "aucune vie n'est menacée dans cette situation" serait un *defeater* de C. De plus, puisque l'opérateur $\Box_K$ est supposé être un opérateur monadique et ∇_O est un opérateur dyadique, l'attaque entre K et C est fondée sur une contradiction entre le premier paramètre de $\Box_K$ et le second paramètre de ∇_O. Nous disons ici que les paramètres sont *non alignés*. D'autre part, l'attaque entre F et C est fondée sur une **équivalence** entre $\neg\phi$ et le premier paramètre de ∇_O. Nous disons que ces paramètres sont *alignés*. Ainsi, nous avons besoin de spécifier quels paramètres se fondent l'attaque et quelle est la nature de cette attaque : *alignée* ou *non alignée*. Tout d'abord, nous définissons $d^{\star_2}_{\star_1}$*-mapping* une fonction qui associe les paramètres de deux modalités distinctes $\star_1$ et $\star_2$, qui doivent être incohérentes pour caractériser une attaque. Dans l'exemple précédent, $K = (_, \Box_K\psi_1)$ avec $\psi_1 := \neg\psi$, et $C = (_, \nabla_O(\phi_1, \phi_2))$ avec $\phi_1 := \neg\phi$ et $\phi_2 := \psi$, s'attaque mutuellement car $\models \phi_2 \equiv \neg(\psi_1)$. Ainsi, le d-mapping est défini comme $d^{\nabla_C}_{\Box_K} = \{(1, 2)\}$, i.e. le premier paramètre de $\Box_K$ est associé avec le second paramètre

de ∇_O et il doit y avoir une contradiction pour définir une relation d'attaque. Dans un second temps, nous définissons un $a^{\star_2}_{\star_1}$*-mapping* qui est une fonction qui associe les paramètres de deux modalités distinctes qui doivent être équivalentes pour pouvoir définir une attaque. Considérons la modalité standard $\Box$ et la formule $\Box\theta_1$ où $\theta_1 := \neg\phi$ signifie "$\neg\phi$ est nécessairement vraie". Considérons $\nabla_O(\phi_1, \phi_2)$, θ_1 est nécessairement vraie, i.e. $\neg\phi$, il ne peut pas être possible que le contexte de $\nabla_O(\phi_1, \phi_2)$ soit vérifié, i.e. ϕ est vraie. Ainsi, le a-mapping est défini comme $a^{\nabla_C}_{\Box} = \{(1, 1)\}$ et est défini lorsque $\models \phi_1 \equiv \theta_1$ est une tautologie. Alors qu'il pourrait sembler contre intuitif de considérer une équivalence pour caractériser une attaque, cela fait sens pour les modalités de type $(\forall, \vee)$ et $(\exists, \vee)$, i.e. ∇ et $\Diamond$, comme elles représentent le contexte avec une négation. Troisièmement, nous définissons un $d^{\star_2}_{\emptyset}$*-mapping* (resp. $a^{\star_2}_{\emptyset}$*-mapping*) correspondant aux paramètres de $\star_2$ qui doivent être incohérents (reps. équivalent) par rapport aux vérités factuelles, i.e. un argument dont la conclusion n'est pas précédée par un opérateur modal. Par exemple, $d^{\nabla_O}_{\emptyset} = \{1\}$ signifie que le paramètre contextuel de l'opérateur dyadique déontique ∇_O peut être attaqué par une vérité factuelle. Au lieu de représenter quatre mapping différents, nous proposons une notation plus compacte où d- et a-mappings sont représentés de manière jointe comme un ensemble d'ensembles de règles, où les règles décrivent quels paramètres attaquent quels paramètres pour n'importe quel type de mapping. Par exemple, nous pouvons représenter un $\langle d, a\rangle^{\star_2}_{\star_1}$-mapping comme $\{\{(1, 2, d), (2, 3, a)\}, \{(3, 4, d)\}\}$. Ici, $\langle d, a\rangle^{\star_2}_{\star_1}$ peut être vue comme une formule normale conjonctive (FNC) définissant les règles qui doivent être satisfaites pour avoir une attaque, i.e. il y a une attaque si le paramètre 1 dans $\star_1$ et le paramètre 2 dans $\star_2$ se contredisent, ou le paramètre 2 dans $\star_1$ et le paramètre 3 dans $\star_2$ sont équivalents, et le paramètre 3 dans $\star_1$ et le paramètre 4 dans $\star_2$ se contredisent.

Définition 13 ($\langle d, a\rangle$-mappings modaux)**.** *Soient $\tau = (O, \rho)$ un type de similarité, O^Ω un ensemble de symboles fondés sur τ, $\{d, a\}$ un ensemble de symboles, et $(\star_1, \star_2) \in O^\Omega \times O^\Omega$ deux modalités.*

$\langle d, a\rangle^{\star_2}_{\star_1}$ *est un* mapping modal *pour* $(\star_1, \star_2)$ *ssi :*

$$\langle d, a\rangle^{\star_2}_{\star_1} \subseteq 2^{[|1,\rho(\star_1)|]\times[|1,\rho(\star_2)|]\times\{d,a\}}$$

$\langle d, a\rangle^{\star_2}_{\emptyset}$ *est un* mapping modal *pour* $\star_2$ *ssi :*

$$\langle d, a\rangle^{\star_2}_{\emptyset} \subseteq 2^{[|1,\rho(\star_2)|]\times\{d,a\}}$$

$\langle d, a\rangle_\tau$ *est un* mapping modal pour le τ-cadre*, une fonction qui associe pour chaque paire de modalités $(\star_1, \star_2) \in O^\Omega \times O^\Omega$ un mapping modal pour $(\star_1, \star_2)$ et pour chaque modalité $\star_2 \in O^\Omega$ assigne un mapping modal pour $\star_2$, i.e. $\langle d, a\rangle_\tau$ est tel que :*

— $\langle d, a\rangle_\tau : (O^\Omega \cup \{\emptyset\}) \times O^\Omega \to 2^{2^{((\mathbb{N}^*\times\mathbb{N}^*)\cup\mathbb{N}^*)\times\{d,a\}}}$

— *Pour tout $(\star_1, \star_2) \in O^\Omega \times O^\Omega$, $\langle d, a\rangle_\tau(\star_1, \star_2) \subseteq 2^{[|1,\rho(\star_1)|]\times[|1,\rho(\star_2)|]\times\{d,a\}}$*

— *Pour tout $\star_2 \in O^\Omega$, $\langle d, a\rangle_\tau(\emptyset, \star_2) \subseteq 2^{[|1,\rho(\star_2)|]\times\{d,a\}}$*

Dans la suite, nous considérons une notion de mapping modal *consistant*, i.e. est satisfiable, puisqu'un mapping modal entre deux modalités peut être vu comme une FNC. En effet, si pour deux modalités nous avons $\langle d, a\rangle^{\star_2}_{\star_1} = \{\{(i, j, d)\}, \{(i, j, a)\}\}$, alors cela signifierait que les paramètres devraient être à la fois incohérents et équivalents, ce qui n'a pas de sens.

Définition 14 (Mapping modal consistant)**.** *Soit $\tau = (O, \rho)$ un type de similarité, O^Ω l'ensemble des symboles de τ, et $\langle d, a\rangle_\tau$ un mapping modal défini sur le τ-cadre. Nous appelons $\langle d, a\rangle_\tau$ un* mapping modal consistant *ssi (1) et (2) sont vérifiés :*

(1) Pour tout $\star_1, \star_2 \in O^\Omega$, il existse une fonction I : $[|1, \rho(\star_1)|] \times [|1, \rho(\star_2)|] \to \{\top, \bot\}$ t.q. :

$$\forall S \in \langle d, a\rangle_\tau(\star_1, \star_2), \exists(i, j, t) \in S, I(i, j) = \begin{cases} \top, & \text{if } t = a \\ \bot, & \text{if } t = d \end{cases}$$

(2) Pour tout $\star_2 \in O^\Omega$, il existe une fonction I : $[|1, \rho(\star_2)|] \to \{\top, \bot\}$ t.q. :

$$\forall S \in \langle d, a\rangle_\tau(\emptyset, \star_2), \exists(j, t) \in S, I(j) = \begin{cases} \top, & \text{if } t = a \\ \bot, & \text{if } t = d \end{cases}$$

Vérifier si un mapping modal est consistant est un problème NP-complet car cela revient à résoudre un problème de satisfiabilité pour la logique propositionnelle. La preuve de NP-complétude découle du théorème suivant :

Théorème 1. *Soit $\langle d, a\rangle_\tau$ un mapping modal consistant ssi la formule propositionnelle suivante ϕ est satisfiable, avec $\mathcal{P} = \{p_{i,j} : (i, j) \in [|1, \rho(\star_1)|] \times [|1, \rho(\star_2)|], (\star_1, \star_2) \in (O^\Omega)^2\} \cup \{p_j : j \in [|1, \rho(\star_2)|], \star_2 \in O^\Omega\}$ un ensemble d'atomes propositionnels :*

$$\phi = \bigwedge_{(\star_1,\star_2)\in O^\Omega\times O^\Omega} \Big(\bigwedge_{S\in\langle d,a\rangle^{\star_2}_{\star_1}} \Big(\bigvee_{\substack{(i,j,t)\in S:\\ t=a}} p_{i,j} \bigvee_{\substack{(i,j,t)\in S:\\ t=d}} \neg p_{i,j}\Big)\Big)$$
$$\bigwedge_{\star_2\in O^\Omega} \Big(\bigwedge_{S\in\langle d,a\rangle^{\star_2}_{\emptyset}} \Big(\bigvee_{\substack{(j,t)\in S:\\ t=a}} p_j \bigvee_{\substack{(j,t)\in S:\\ t=d}} \neg p_j\Big)\Big)$$

La preuve est évidente en remarquant qu'elle est une traduction directe d'un $(i, j)^{\star_2}_{\star_1}$ variable propositionnelle $p_{i,j}$ quand il s'agit d'un mapping de type aligné et la négation $\neg p_{i,j}$ quand il est de type non aligné. Nous faisons de même pour les mappings de la forme $(j)^{\star_2}_{\emptyset}$. Prouver la réciproque correspond à construire un bon modal mapping correspondant à une formule ϕ en FNC.

Exemple 11. *Supposons un mapping modal non aligné entre chaque second paramètre des opérateurs dyadiques*

et considérons un mapping modal aligné pour chaque premier paramètre des opérateurs dyadiques et, entre les formules factuelles et les formules nécessairement vraies. Ainsi, le mapping modal $\langle d, a\rangle_\tau$ est tel que :

— $\forall \star_1, \star_2 \in Dyadic_\nabla, \langle d, a\rangle_\tau(\star_1, \star_2) = \{\{(2, 2, d)\}\}$

— $\forall \star_2 \in Dyadic_\nabla, \langle d, a\rangle_\tau(\emptyset, \star_2) = \{\{(1, a)\}\}$

— $\forall \star_2 \in Dyadic_\nabla, \langle d, a\rangle_\tau(\Box, \star_2) = \{\{(1, 1, a)\}\}$

Nous ne spécifions pas le mapping modal des opérateurs alternatifs puisqu'ils ne sont pas utilisés ici.

Ces mappings modaux permettent de définir différents types d'attaques modales comme un rebuttal modal, un undercut modal ou un defeater modal. Cependant, en raison des interactions entre modalités et mappings le nombre d'attaques à caractériser devient trop important pour les énumérer. Afin de proposer une notation compacte, nous introduisons une fonction Υ exprimant les règles pour définir ces nouvelles attaques. L'idée consiste à diviser les définitions en deux cas : les règles qui représentent les attaques entre deux modalités, et les règles qui représentent les faits et les modalités. De plus, nous introduisons un paramètre dans Υ, pour décrire le *type*, distinguant les attaques rebuttals et undercuts ($\Leftrightarrow$), et defeaters ($\Rightarrow$).

Définition 15 (Règle Υ-attaque pour $\langle d, a\rangle$-mapping)**.** *Soient $\tau = (O, \rho)$ un type de similarité, O^Ω un ensemble de symboles fondés sur τ, $\langle d, a\rangle$ un mapping modal sur le τ-cadre, et $\Upsilon = \{\Upsilon^{\star_2}_\star\}_{(\star,\star_2)\in M}$ un ensemble de fonctions où $M = (O^\Omega \cup \{\emptyset\}) \times O^\Omega$:*

$$\forall(\star, \star_2) \in M, \Upsilon^{\star_2}_\star : \mathcal{L} \times \mathcal{L} \times \{\Rightarrow, \Leftrightarrow\} \to \{\bot, \top\}$$

Pour tout $(\phi, \psi) \in \mathcal{L}^2$, pour tout $(\star_1, \star_2) \in O^\Omega \times O^\Omega$.

Nous appelons Υ l'ensemble des règles d'attaque pour le $\langle d, a\rangle$-mapping si, et seulement si, Υ est t.q. (1) et (2) sont respectées :

1. *si $\phi = \star_1(\phi_1, \ldots, \phi_{\rho(\star_1)})$, et pour tout $i \in [|1, \rho(\star_1)|]$, $\phi_i \equiv \phi_i^1 \wedge \ldots \wedge \phi_i^{n_i}$ sa Forme Canonique Normale Conjonctive (FCNC), $\Upsilon^{\star_2}_{\star_1}(\phi, \psi, type) = \top$ si, et seulement si, $\forall S \in \langle d, a\rangle^{\star_2}_{\star_1}$, $\exists(i, j, t) \in S$, $\exists K \subseteq [|1, n_i|]$:*
   — *si $t = d$, alors :*
   $$\{\bigwedge_{k\in K} \phi_i^k\} \models \neg\psi_j \text{ et } (\text{si type} = \Leftrightarrow, \{\neg\psi_j\} \models \bigwedge_{k\in K} \phi_i^k)$$
   — *si $t = a$, alors :*
   $$\{\bigwedge_{k\in K} \phi_i^k\} \models \psi_j \text{ et } (\text{si type} = \Leftrightarrow, \{\neg\psi_j\} \models \bigwedge_{k\in K} \phi_i^k)$$
2. *si $\phi \equiv \Phi$ où $\Phi = \phi^1 \wedge \ldots \wedge \phi^n$ sa FCNC de ϕ, et $\psi = \star_2(\psi_1, \ldots, \psi_{\rho(\star_2)})$, alors $\Upsilon^{\star_2}_\emptyset(\phi, \psi, type) = \top$ si, et seulement si, $\forall S \in \langle d, a\rangle^{\star_2}_\emptyset$, $\exists(j, t) \in S, \exists K \subseteq [|1, n|]$:*
   — *si $t = d$, alors :*
   $$\{\bigwedge_{k\in K} \phi^k\} \models \neg\psi_j \text{ et } (\text{if type} = \Leftrightarrow, \{\neg\psi_j\} \models \bigwedge_{k\in K} \phi^k)$$
   — *si $t = a$, alors :*
   $$\{\bigwedge_{k\in K} \phi^k\} \models \psi_j \text{ et } (\text{if type} = \Leftrightarrow, \{\psi_j\} \models \bigwedge_{k\in K} \phi^k)$$

Nous pouvons désormais définir les attaques modales standards de l'argumentation fondée sur les logiques modales.

Définition 16 (Types d'attaques modales)**.** *Soit $\tau = (O, \rho)$ un type de similarité, et $\langle d, a\rangle$ un mapping modal consistant sur le τ-cadre, et $(\star, \star_2) \in (O^\Omega \cup \{\emptyset\}) \times O^\Omega$,*

— *Un argument A est un $\langle d, a\rangle^{\star_2}_\star$*rebuttal modal direct *pour B, noté $(A, B) \in$ Reb $\langle d, a\rangle^{\star_2}_\star$ si, et seulement si, $A = (_, \phi)$, $B = (_, \star_2(\psi_1, \ldots, \psi_{\rho(\star_2)}))$ et*
$$\Upsilon^{\star_2}_\star(\phi, \star_2(\psi_1, \ldots, \psi_{\rho(\star_2)}), \Leftrightarrow) = \top$$

— *Un argument A est $\langle d, a\rangle^{\star_2}_\star$*undercut modal direct *pour B, noté $(A, B) \in$ Und $\langle d, a\rangle^{\star_2}_\star$ si, et seulement si, $A = (_, \phi)$, $B = (\Psi, _)$, il existe $\psi \in \Psi$, $\psi = \star_2(\psi_1, \ldots, \psi_{\rho(\star_2)})$,*
$$\Upsilon^{\star_2}_\star(\phi, \star_2(\psi_1, \ldots, \psi_{\rho(\star_2)}), \Leftrightarrow) = \top$$

— *Un argument A est un* defeater modal direct *$\langle d, a\rangle^{\star_2}_\star$ pour B, noté $(A, B) \in$ Def $\langle d, a\rangle^{\star_2}_\star$ si, et seulement si, $A = (_, \phi)$, $B = (\Psi, _)$, il existe $\psi \in \Psi$, $\psi = \star_2(\psi_1, \ldots, \psi_{\rho(\star_2)})$,*
$$\Upsilon^{\star_2}_\star(\phi, \star_2(\psi_1, \ldots, \psi_{\rho(\star_2)}), \Rightarrow) = \top$$

— *Nous définissons $\langle d, a\rangle^{\star_2}_\star$* attaque modale *comme :*
$$Att\langle d, a\rangle^{\star_2}_\star = Reb\ \langle d, a\rangle^{\star_2}_\star \cup Und\ \langle d, a\rangle^{\star_2}_\star \cup Def\ \langle d, a\rangle^{\star_2}_\star$$

Remarquons que les attaques sur les opérateurs nullaires sont considérées comme pour les formules factuelles. En effet puisque les opérateurs nullaires sont vrais dans des mondes spécifiques, quelque soit les modèles du cadre, nous pouvons les considérer comme des formules propositionnelles et donc des formulles factuelles.

5.3 VAF fondés sur les logiques modales

Nous pouvons définir un AF fondé sur le système GK. Nous appelons alors ce cadre d'argumentation un τ-VAF.

Définition 17 (AF fondé sur la logique modale)**.** *Soit $\tau = (O, \rho)$ un type de similarité. Nous appelons τ-*AF *un N-uplet $\langle C, \langle d, a\rangle, \mathcal{A}, \mathcal{R}\rangle$ où C est un τ-cadre, et $\langle \mathcal{A}, \mathcal{R}\rangle$ est un cadre d'argumentation abstrait t.q. :*

— *$\mathcal{A}$ contient seulement des arguments valides sur C i.e. valides wrt $\models_C$,*

— *$\langle d, a\rangle$ est un mapping modal consistant,*

— *$\mathcal{R} = Reb \cup Und \cup Def \bigcup_{(\star,\star_2)\in(O^\Omega\cup\{\emptyset\})\times O^\Omega} Att\langle d, a\rangle^{\star_2}_\star$*

Nous appelons un τ-VAF un N-uplet $\langle C, \langle d, a\rangle, \mathcal{A}, \mathcal{R}, \mathcal{V}al, \mu, \succ\rangle$ qui est un τ-AF et $\langle \mathcal{A}, \mathcal{R}, \mathcal{V}al, \mu, \succ\rangle$ est un VAF.

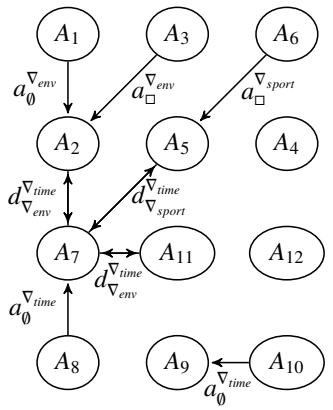


FIGURE 1 – Graphe d'argumentation

Exemple 12. *En se basant sur les exemples précédents, nous déduisons le τ-VAF comme décrit par la figure 1. Les attaques modales sont données avec leur type de modal mapping appliqué. Nous supposons μ est t.q. pour tout $\star \in \{\Box, \Diamond, \nabla, \triangle\}$:*

1. $\forall \phi, \psi \in \mathcal{L}, \mu(_, \star_{sport}(\phi, \psi)) = \nu_{health}$
2. $\forall \phi, \psi \in \mathcal{L}, \mu(_, \star_{env}(\phi, \psi) = \nu_{env}$
3. $\forall \phi, \psi \in \mathcal{L}, \mu(_, \star_{time}(\phi, \psi)) = \nu_{eco}$
4. $\forall \phi, \psi \in \mathcal{L}, \mu(_, \star_{gov}(\phi, \psi)) = \nu_{aut}$
5. $\forall \phi \in \mathcal{L}, \mu(_, \Box\phi) = \emptyset$
6. $\forall A \in \mathcal{A}$ *t.q. A n'est pas de la forme* $(_, \star(\ldots))$, $\mu(A) = \emptyset$

Ainsi, $\mu = \{(A_1, \emptyset), (A_2, \nu_{env}), (A_3, \emptyset), (A_4, \emptyset), (A_5, \nu_{health}), (A_6, \emptyset), (A_7, \nu_{eco}), (A_8, \emptyset), (A_9, \nu_{env}), (A_{10}, \emptyset), (A_{11}, \nu_{env}), (A_{12}, \nu_{aut})\}$. *Considérons ensuite les préférences sur les valeurs suivantes :* $\succ$ *:*

$$\emptyset \succ \nu_{health} \succ \nu_{aut} \succ \nu_{env} \succ \nu_{eco}$$

Ainsi, certains arguments comme A_7 sont defaits par A_2, A_5, A_{11} et sont associés à une valeur préférée. La VAF-attaque $= \{(A_1, A_2), (A_3, A_2), (A_6, A_5), (A_5, A_7), (A_2, A_7), (A_{11}, A_7), (A_8, A_7), (A_{10}, A_9)\}$. *Enfin, considérons l'extension stable $\mathcal{E}$:*

$$\mathcal{E} = \{\{A_1, A_3, A_4, A_6, A_8, A_{10}, A_{11}, A_{12}\}\}$$

Le résultat conclut que l'utilisateur ne peut pas prendre son vélo puisqu'il n'a pas de vélo. De plus, puisqu'il n'y a pas d'embouteillage et pas d'urgence, l'utilisateur n'a pas non plus besoin de prendre sa voiture pour réduire son temps de trajet. De plus, pour l'environnement, il est meilleur de prendre les transports publics ou marcher. Cependant, selon A_{12}, puisqu'il y a une crise sanitaire, il est recommandé de ne pas prendre les transports publics. Par conséquent, la décision éthique dans notre exemple est donné par la décision de marcher.

Même si nous n'explicitons pas le modèle de décision dans notre VAF, nous pourrions étendre notre VAF avec le modèle de [3] où une fonction $\mathcal{F}_f$ (resp. $\mathcal{F}_c$) assigne une décision à un ensemble d'arguments qui la supporte (resp. ne la supporte pas). Dans l'exemple 12, une décision peut être considérée acceptable éthiquement si elle est supportée par un argument crédule i.e. au moins un argument supporte la décision et appartient à au moins une extension.

6 Conclusion

Nous avons proposé de modéliser le raisonnement éthique avec un cadre d'argumentation basé sur les valeurs, appelé τ-VAF. Celui-ci est fondé sur une logique modale normale généralisée aux opérateurs n-aires. Une telle logique permet entre autres d'exprimer de raisonner avec des opérateurs déontiques dyadiques. Le fait de fonder un cadre d'argumentation sur la logique modale nous a conduit à proposer de nouveaux types d'attaques, appelées les attaques modales, qui généralisent les attaques logiques standard. Puisque le système logique utilisé dans les τ-VAF est une généralisation des logiques multimodales normales généralisée aux opérateurs n-aires, les travaux futurs consisteront à instancier un τ-VAF avec une logique modale comme DL-MA [27] ou avec une logique de description déontique comme celle proposée dans [16].

Références

[1] Amgoud, Leila: *Postulates for logic-based argumentation systems*. International Journal of Approximate Reasoning, 55(9) :2028–2048, 2014.

[2] Amgoud, Leila et Philippe Besnard: *A formal analysis of logic-based argumentation systems*. Dans *4th International Conference on Scalable Uncertainty Management*, pages 42–55, 2010.

[3] Amgoud, Leila et Henri Prade: *Using arguments for making and explaining decisions*. Artificial Intelligence, 173(3) :413–436, 2009.

[4] Arieli, Ofer et Christian Straßer: *Sequent-based logical argumentation*. Argument & Computation, 6(1) :73–99, 2015.

[5] Arkoudas, Konstantine, Selmer Bringsjord et Paul Bello: *Toward ethical robots via mechanized deontic logic*. Dans *AAAI Fall Symposium*, 2005.

[6] Atkinson, Katie et Trevor J. M. Bench-Capon: *Addressing moral problems through practical reasoning*. Journal of Applied Logic, 6(2) :135–151, 2008.

[7] Bench-Capon, Trevor JM: *Persuasion in practical argument using value-based argumentation frameworks*. Journal of Logic and Computation, 13(3) :429–448, 2003.

[8] Berreby, Fiona, Gauvain Bourgne et Jean Gabriel Ganascia: *Modelling moral reasoning and ethical responsibility with logic programming*. Dans *20th International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 532–548, 2015.

[9] Besnard, Philippe et Anthony Hunter: *Practical first-order argumentation*. Dans *20th National Conference on Artificial Intelligence*, pages 590–595, 2005.

[10] Besnard, Philippe et Anthony Hunter: *Argumentation based on classical logic*. Dans Simari, Guillermo et Iyad Rahwan (rédacteurs) : *Argumentation in artificial intelligence*, pages 133–152. Springer, 2009.

[11] Blackburn, Patrick, Maarten De Rijke et Yde Venema: *Modal logic : graph. Darst*, tome 53. Cambridge University Press, 2002.

[12] Boella, Guido, Joris Hulstijn et Leendert Van Der Torre: *A logic of abstract argumentation*. Dans *2nd International Workshop on Argumentation in Multi-Agent Systems*, pages 29–41, 2005.

[13] Borg, AnneMarie et Ofer Arieli: *Hypersequential argumentation frameworks : An instantiation in the modal logic S5*. Dans *17th International Conference on Autonomous Agents and MultiAgent System*, pages 1097–1104, 2018.

[14] Caminada, Martin WA et Dov M Gabbay: *A logical account of formal argumentation*. Studia Logica, 93(2-3) :109, 2009.

[15] Cointe, Nicolas, Grégory Bonnet et Olivier Boissier: *Ethical judgment of agents' behaviors in multi-agent systems*. Dans *15th International Conference on Autonomous Agents and Multiagent Systems*, pages 1106–1114, 2016.

[16] Dalmonte, Tiziano, Andrea Mazzullo et Ana Ozaki: *On non-normal modal description logics*. Dans Šimkus, Mantas et Grant Weddell (rédacteurs) : *32nd International Workshop on Description Logics*, tome 2373 de *CEUR-WS.org*, page 14, 2019.

[17] Damasio, Antonio: *Descartes's error : Emotion, reason and the human brain*. Avon : New York, 1994.

[18] Dung, Phan Minh: *On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games*. Artificial Intelligence, 77(2) :321–357, 1995.

[19] Ganascia, Jean Gabriel: *Modelling ethical rules of lying with answer set programming*. Ethics and Information Technology, 9(1) :39–47, 2007.

[20] Gert, Bernard et Joshua Gert: *The definition of morality*. Dans Zalta, Edward N. (rédacteur) : *The Stanford Encyclopedia of Philosophy*. Stanford University, fall 2020 édition, 2020.

[21] Gorogiannis, Nikos et Anthony Hunter: *Instantiating abstract argumentation with classical logic arguments : Postulates and properties*. Artificial Intelligence, 175(9–10) :1479–1497, 2011.

[22] Greene, Joshua et Jonathan Haidt: *How (and where) does moral judgment work ?* Trends in Cognitive Sciences, 6(12) :517–523, 2002.

[23] Grossi, Davide: *On the logic of argumentation theory*. Dans *9th International Conference on Autonomous Agents and MultiAgent Systems*, pages 409–416, 2010.

[24] Haidt, Jonathan: *The emotional dog and its rational tail : a social intuitionist approach to moral judgment*. Psychological Review, 108(4) :814–834, 2001.

[25] Horty, John F.: *Moral dilemmas and nonmonotonic logic*. Journal of Philosophical Logic, 23(1) :35–65, 1994.

[26] Kuipers, Benjamin: *Perspectives on ethics of AI : Computer science*. Dans Dubber, Markus, Frank Pasquale et Sunit Das (rédacteurs) : *The Oxford Handbook of Ethics of AI*. Oxford University Press, 2019.

[27] Lorini, Emiliano: *On the logical foundations of moral agency*. Dans *11th International Conference on Deontic Logic in Computer Science*, pages 108–122, 2012.

[28] Pollock, John L: *Defeasible reasoning*. Cognitive science, 11(4) :481–518, 1987.

[29] Prakken, Henry: *An abstract framework for argumentation with structured arguments*. Argument and Computation, 1(2) :93–124, 2010.

[30] Prakken, Henry et Marek Sergot: *Dyadic deontic logic and contrary-to-duty obligations*. Dans *Defeasible deontic logic*, pages 223–262. Springer, 1997.

[31] Proietti, Carlo, Davide Grossi, Sonja Smets et Fernando R Velázquez-Quesada: *Bipolar argumentation frameworks, modal logic and semantic paradoxes*. Dans *7th International Workshop on Logic, Rationality and Interaction*, pages 214–229, 2019.

[32] Saptawijaya, Ari et Luis Moniz Pereira: *Towards modeling morality computationally with logic programming*. Dans *International Symposium on Practical Aspects of Declarative Languages*, pages 104–119, 2014.

[33] Timmons, Mark: *Moral theory : an introduction*. Rowman & Littlefield Publishers, 2012.

[34] Wiegel, Vincent et Jan van den Berg: *Combining moral theory, modal logic and MAS to create well-behaving artificial agents*. International Journal of Social Robotics, 1(3) :233–242, 2009.

Session 5 : Optimisation et planification

Actes JIAF 2022

Generalisation of alpha-beta search for AND-OR graphs with partially ordered values*

Junkang Li[1,2] **Bruno Zanuttini**[2]
Tristan Cazenave[1,3] **Véronique Ventos**[1]

[1]NukkAI, Paris, France
[2]Normandie Univ.; UNICAEN, ENSICAEN, CNRS, GREYC, 14 000 Caen, France
[3]LAMSADE, Université Paris-Dauphine, PSL, CNRS, France
junkang.li@nukk.ai bruno.zanuttini@unicaen.fr
tristan.cazenave@lamsade.dauphine.fr vventos@nukk.ai

Résumé

Nous proposons un cadre pour l'évaluation de graphes ET-OU (orientés, acycliques) portant des valeurs partiellement ordonnées. De tels graphes apparaissent naturellement dans la résolution de jeux à information incomplète (par exemple, la plupart des jeux de cartes, comme le bridge) ou multicritères. En particulier, notre cadre généralise l'évaluation standard de graphes ET-OU et le calcul de stratégies optimales pour les jeux à information complète.

Dans ce cadre, nous proposons un nouvel algorithme, qui utilise l'élagage alpha-beta et un cache des valeurs déjà calculées. Cet article présente l'algorithme, démontre sa correction, et fournit des résultats expérimentaux sur des jeux aléatoires et sur un jeu de cartes à information incomplète.

Abstract

We define a new setting related to the evaluation of AND-OR directed acyclic graphs with partially ordered values. Such graphs arise naturally when solving games with incomplete information (e.g. most card games such as Bridge) or games with multiple criteria. In particular, this setting generalises standard AND-OR graph evaluation and computation of optimal strategies in games with complete information.

Under this setting, we propose a new algorithm which uses both alpha-beta pruning and cached values. In this paper, we present our algorithm, prove its correctness, and give experimental results on random games and on a card game with incomplete information.

*This article is a long version with full proofs of the article published in the proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022).

1 Introduction

Search in graphs containing AND- and OR-nodes is used as a basis of many algorithmic solutions to Artificial Intelligence problems. In such graphs, OR-nodes typically model choice nodes where an agent can choose a successor, while AND-nodes model an opponent. For instance, in robust planning with nondeterministic actions, an AND-node models the outcome of an action: a strategy must be valid whatever the outcome [11]. Similarly, when solving a zero-sum two-player (sequential) game for a player, OR-nodes are those at which it is her turn to play, while AND-nodes correspond to her opponent [22]: the value of an OR-node is 1 if and only if at least one move leads to a node with value 1; dually, at AND-nodes, the values of the children are conjoined. More generally, in games that can have more than two outcomes, such as chess or checkers [20], AND-nodes (respectively OR-nodes) correspond to a minimum (respectively maximum) operator on the values of their children.

A fundamental question is that of evaluating rooted AND-OR directed acyclic graphs (DAGs), which means computing the value of their root given a value for each of their leaves. For instance, in games with complete information, selecting the best move for the current turn amounts to evaluate each of the children of the root. This problem has been thoroughly studied in the literature [16] under the setting of totally ordered values (Boolean or real) and the standard AND/OR or min/max operators. Following [8], we investigate here a more general setting, where the values are taken from a distributive lattice $(V, \wedge, \vee)$ (i.e. a partially ordered set with least upper bound and greatest lower

bound for any two elements), and operators for AND- and OR nodes are taken to be the meet $\wedge$ and join $\vee$, respectively. This setting arises naturally in many applications, in particular in games with incomplete information. Example of such games are Skat [13, 19, 5], Bridge [14, 9, 2], Hearts and Spades [21].

A well-known technique for evaluating AND-OR graphs is alpha-beta pruning, which maintains a lower bound α (respectively upper bound β) on the value of each OR-nodes (respectively AND-nodes), and uses them to prune some of their successors. This technique is currently used in strong chess programs [10] combined with sophisticated evaluation functions such as NNUE neural networks that were first used in Shogi [17]. However, the generalisation of alpha-beta pruning to AND-OR DAGs with partially ordered values is nontrivial since two values from a lattice are not always comparable. We build on the seminal work by [8] and generalise it by proving the correctness of lattice-valued alpha-beta pruning with the consideration for heuristic functions.

Orthogonally, we investigate caching techniques for alpha-beta pruning in lattice-valued DAGs. The question is again nontrivial because nodes are in general revisited with different α and β than during previous visits. For this, we propose a new algorithm called 'alpha-beta duo'. We state its correctness and experimentally evaluate its efficiency.

The paper is organised as follows. Preliminaries are given in Sections 2 and 3. We extend the work by [8] in Section 4, and present alpha-beta duo in Section 5. We then report experimental results and conclude.

2 Preliminaries

The following definitions on posets and lattices are based on [4].

Definition 1. *Let V be a set and $\preceq$ be a binary relation on V. Then $(V, \preceq)$ is called a* partially ordered set *(poset) if $\preceq$ is a partial order (i.e. reflexive, transitive, and antisymmetric).*

For a poset $(V, \preceq)$ and $S \subseteq V$, an element $x \in V$ is called an *upper bound* (UB) of S if $s \preceq x$ holds for all $s \in S$, and x is called a *least upper bound* (LUB) if in addition $x \preceq y$ holds for any UB y of S. If S has an LUB, then it is unique. The greatest lower bound (GLB) of S is defined dually. For $x, y \in V$, we write $x \vee y$ ('*x join y*') and $x \wedge y$ ('*x meet y*') respectively for the LUB and GLB of $\{x, y\}$, when they exist.

Definition 2. *A poset $(V, \preceq)$ is called a* distributive lattice *if*

- *for all $x, y \in V$, $x \vee y$ and $x \wedge y$ exist,*
- *for all $x, y, z \in V$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$,*
- *for all $x, y, z \in V$, $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.*

It is moreover said to be bounded *if there are elements $\bot, \top \in V$ satisfying $\bot \preceq x$ and $x \preceq \top$ for any $x \in V$.*

In the remainder of this paper, we denote by $(V, \preceq, \wedge, \vee)$ an arbitrary bounded distributive lattice.

Example 1. *Let S be a set and let 2^S denote its powerset. Then $(2^S, \subseteq, \cap, \cup)$ is a bounded distributive lattice with set inclusion $\subseteq$ as partial order, set intersection $\cap$ and set union $\cup$ respectively as meet and join, $\emptyset$ as $\bot$, and S as $\top$.*

We denote any directed acyclic graph (DAG) by $G = (N, C)$, where N is the set of nodes, and $C : N \to 2^N$ is a function that yields the set of *children* of each node. A *root* r is a node without predecessor (i.e. for any $n \in N$, $r \notin C(n)$), and a *leaf* is a node without child. We denote the set of leaves of a DAG G by L_G. We only consider *rooted* DAGs, which contain a (necessarily unique) root r such that there exists a directed path to every vertex from r.

An *AND-OR DAG* is a rooted DAG (N, C, r) equipped with a labelling function $\ell : N \to \{\mathrm{A}, \mathrm{O}\}$. Nodes labelled by A and O are respectively called *AND-nodes* and *OR-nodes*. Note that we do not impose nodes to be alternating.

3 Problem setting

We are interested in the problem of evaluating the root value of an AND-OR DAG, given values for all its leaves. Formally, given an AND-OR DAG $G = (N, C, r, \ell)$, a bounded distributive lattice $(V, \preceq, \wedge, \vee)$, and an evaluation function $e : L_G \to V$ assigning a value in V to each leaf of G, the goal is to compute $v(r)$, where the value $v(n)$ of $n \in N$ is defined recursively by:

- for a leaf node n, $v(n) := e(n)$;
- for an internal AND-node n, $v(n) := \bigwedge_{c \in C(n)} v(c)$;
- for an internal OR-node n, $v(n) := \bigvee_{c \in C(n)} v(c)$.

Since G is a DAG, the function $v : N \to V$ is well-defined.

Example 2. *Consider the DAGs in Figure 1, where circle and square nodes represent AND-nodes and OR-nodes, respectively. On the left, the lattice is the set of Boolean vectors of length 4 (denoted as words), with bitwise AND and bitwise OR as meet and join, respectively. One can easily verify that $v(r) = 1100$. On the right, the lattice is the set of Boolean vectors of length 3, and $v(R) = 001$.*

Example applications

Many important problems are in fact AND-OR DAG evaluation in disguise. The simplest one is Boolean circuit evaluation. Here AND- and OR-nodes model AND and OR gates, the lattice is the Boolean algebra (i.e. $V = \{0, 1\}$ with $0 \preceq 1$ and logical conjunction and disjunction as meet

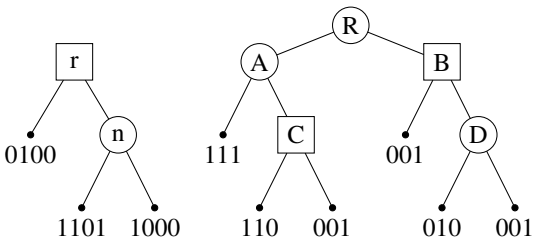


Figure 1: Two AND-OR DAGs with partially ordered values.

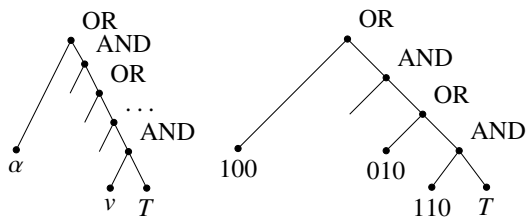


Figure 2: Deep pruning *vs* expected pruning.

and join), and the evaluation function encodes the inputs of the circuit.

Solving a game with complete information typically involves computing the minimax value of a game tree, which can be regarded as evaluating an AND-OR DAG: AND- and OR-nodes are respectively choice nodes of player MIN and of player MAX, the lattice is $(V, \leq, \min, \max)$ with V a totally ordered set such as $\mathbb{Z}$ or $\mathbb{R}$, and the evaluation function gives the value of terminal nodes, or a heuristic value if the search is cut at some depth. Then the root value is the minimax value of the game.

In games with incomplete information, nontrivial lattices come into play. For example, [9] shows that computing the maxmin value of a player amounts to evaluate the game DAG with the lattice $(2^{2^S}, \preceq, \sqcap, \cup)$[1], where S is a finite set and $f \sqcap g = \{\alpha \cap \beta \mid \alpha \in f, \beta \in g\}$ for any $f, g \in 2^{2^S}$ (i.e. f and g are sets of subsets of S). We will discuss more about this in Section 6.

4 Alpha-beta pruning under partial order

Most of the literature on alpha-beta pruning concerns only totally ordered values, such as real numbers. Since AND-OR DAGs with partially ordered values are useful to model richer problems, [3] proposed alpha-beta pruning in this new setting for multi-criteria game. [8] gave the first thorough study on this subject, and proved in particular that deep pruning is sound for rational players if and only if the set of values is a distributive lattice. [15] showed that deep pruning is sound for tropical algebras if rationality is relaxed.

The form of deep α pruning considered by [8] is given in Figure 2 (left). If $v \preceq \alpha$, then the subtree T can be deeply pruned. To show why this definition of deep pruning does not capture every cut an alpha-beta search should perform when values are partially ordered, consider Figure 2 (right). The lattice is again the set of Boolean vectors of length 3, with bitwise AND and bitwise OR as meet and join, respectively. When an alpha-beta search algorithm descends to the bottommost AND-node, the value of α would be 110, which is the join of the value of an already explored child of two ancestor OR-nodes. We would like the algorithm to prune subtree T since the value of its parent node cannot be better than the current value of α (due to the sibling of T). However, deep pruning, as it is defined in the literature such as [8], does not apply since the value of α does not come from a child of one single ancestor node. Note that this phenomenon is specific to lattices that are *not* totally ordered, since otherwise the meet or join (i.e. min or max) of two values is always one of them, hence deep pruning captures any pruning in a standard alpha-beta search.

Another question not formally addressed in the literature is the initialisation of node values. In standard alpha-beta search, one typically initialises the value of an OR-node (respectively AND-node) to be α (respectively β) [12] or $-\infty$ (respectively $+\infty$) [16] (note that $-\infty$ and $+\infty$ translate to $\bot$ and $\top$ in our context). However, one may have access to a heuristic evaluation of nodes, typically by evaluating a relaxed version of the problem which is easier to solve. For instance, a player can do no better in a game with incomplete information than in the same game but with complete information. The latter being much easier to solve, the value obtained can be used as a heuristic in the original game with incomplete information. Ideally, initialising values with an accurate heuristic should accelerate the search by finding cuts earlier.

In order to fill these two gaps in the literature, we first formalise alpha-beta search under partial order with initialisation function in Algorithm 1. We denote by h the initialisation function. In general, its value depends on the current α and β, so we define it to yield a value $h(n, \alpha, \beta)$ for any node n and bounds α and β. Note that since non-trivial initial value can be used for a node n (Line 3), a cut may happen even before the first child of n is explored, hence we update α and β (Lines 7 and 9) and determine whether there is a cut (Line 10) at the beginning of the main loop. This is otherwise the same algorithm as in the literature [16, for instance].

It can be seen that Algorithm 1 will perform the wished pruning in the example in Figure 2 (right). By mimicking the proof by [12], we prove that such pruning is indeed sound, thereby extending the result by [8] to its full form,

[1]We abuse the notation to denote by 2^{2^S} the set of subsets of S closed under subsets, i.e. the set of down-sets of 2^S.

Algorithm 1: Alpha-beta search

```
def AlphaBeta(node n, α, β):
    I = h(n, α, β)
    v ← I
    determine the successor nodes n_1, ..., n_b of n
    for i in {1, ..., b}:
        if n is an OR-node:
            α ← α ∨ v
        else:
            β ← β ∧ v
        if α ≥ β:
            break
        v_child ← AlphaBeta(n_i, α, β)
        if n is an OR-node:
            v ← v ∨ v_child
        else:
            v ← v ∧ v_child
    return v
```

provided that the initialisation function h satisfies a certain admissibility condition:

Definition 3. *A heuristic function h is said to be* admissible *for G and V if for any node n in G and any $\alpha, \beta \in V$,*

$$h(n,\alpha,\beta)\begin{cases} = v(n) & \text{if } n \text{ is a leaf node;} \\ \geq v(n) \wedge \beta & \text{if } n \text{ is an AND-node;} \\ \leq v(n) \vee \alpha & \text{if } n \text{ is an OR-node.} \end{cases}$$

Note that this condition is satisfied for the initialisation values usually used in the literature, such as α or $-\infty$ for OR-nodes (and β or $+\infty$ for AND-nodes). It is also satisfied when $h(n,\alpha,\beta)$ overestimates $v(n)$ for internal AND-nodes and underestimates it for internal OR-nodes.

We denote the value returned by Algorithm 1 with input n, α, β by $f(n,\alpha,\beta)$. The correctness of Algorithm 1 is a consequence of the following central result.

Proposition 1. *If h is an admissible heuristic function for G and V, then for any node n of G and any $\alpha, \beta \in V$, we have*

$$\beta \wedge \big(\alpha \vee f(n,\alpha,\beta)\big) = \beta \wedge \big(\alpha \vee v(n)\big). \tag{1}$$

Proof. The proof is based on structural induction. We will only focus on OR nodes, since the case for AND nodes is completely symmetric. So let n be an OR node and let us consider the execution of the function call `AlphaBeta`(n,α,β).

If n is a leaf, then $f(n,\alpha,\beta) = h(n,\alpha,\beta) = v(n)$ since h is admissible, hence equality (1) holds trivially.

Now consider an internal OR node n. Let $b \geq 1$ be the number of children of n and let $n_1, \ldots, n_b$ be the children of n, listed in the same order as in Algorithm 1. By definition of the value function,

$$v(n) = \bigvee_{j=1}^{b} v(n_j).$$

We assume by induction that all function calls on the children of n satisfy equality (1).

Let k be the index of loop during which a break happens (i.e. a cut is found). If no break happens, then k is taken to be $b+1$. For $0 \leq i < k$, let v_i and α_i denote the value of v and α after the ith loop[2]. Then we have

$$v_i = I \vee \bigvee_{j=1}^{i} f(n_j, \alpha_j, \beta).$$

In particular, $v_0 = I$. In addition, $\alpha_0 = \alpha$, and $\alpha_i = \alpha \vee v_{i-1}$ for $1 < i < k$.

To prove equality (1) for node n, we need the following lemma:

Lemma 1. *For any $0 \leq i < k$, we have*

$$\beta \wedge (\alpha \vee v_i) = \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{i} v(n_j)\right). \tag{2}$$

Proof. We prove equality (2) by an induction on i. When $i = 0$, both sides of equality (2) equals $\beta \wedge (\alpha \vee I)$, hence the equality holds.

For $i \geq 1$, $f(n_i, \alpha_i, \beta)$ satisfies equality (1) by induction from the main proposition, which means

$$\beta \wedge \big(\alpha_i \vee f(n_i,\alpha_i,\beta)\big) = \beta \wedge \big(\alpha_i \vee v(n_i)\big). \tag{3}$$

Notice that $\alpha_i = \alpha \vee v_{i-1}$, we have

$$\begin{aligned} \beta \wedge (\alpha \vee v_i) &= \beta \wedge \big(\alpha \vee v_{i-1} \vee f(n_i,\alpha_i,\beta)\big) \\ &= \beta \wedge \big(\alpha \vee v_{i-1} \vee v(n_i)\big) \\ &= \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{i-1} v(n_j) \vee v(n_i)\right) \\ &= \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{i} v(n_j)\right), \end{aligned}$$

where the second line is due to equality (3), the third line is due to distributivity and equality (2) applied to v_{i-1}. Therefore, equality (2) holds for all $0 \leq i < k$. □

Now we can complete the proof of Proposition 1. For a non-leaf OR-node n with $b \geq 1$ children, two cases are possible:

[2] By after the 0th loop, we mean before the beginning of the first loop.

- No break has taken place, which means $k = b + 1$ and the algorithm has looped through all children of n. Then we have $f(n, \alpha, \beta) = v_b$. Plugging $i = b$ into equality (2), we get

$$\begin{aligned}\beta \wedge (\alpha \vee v_b) &= \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{b} v(n_j)\right)\\ &= \beta \wedge \big(\alpha \vee I \vee v(n)\big)\\ &= \beta \wedge \big(\alpha \vee v(n)\big)\end{aligned}$$

  where the last equality is due to $I = h(n, \alpha, \beta) \preceq v(n) \vee \alpha$ for an OR node since h is admissible. Hence, equality (1) holds for node n.

- A break happens during the kth loop where $1 \leq k \leq b$, which means $\alpha_k = \alpha \vee v_{k-1} \succeq \beta$ (Line 10 in Algorithm 1) and $f(n, \alpha, \beta) = v_{k-1}$. On the right hand side of equality (1), we have

$$\begin{aligned}\beta \wedge \big(\alpha \vee v(n)\big) &= \beta \wedge \big(\alpha \vee I \vee v(n)\big)\\ &= \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{b} v(n_j)\right)\\ &\succeq \beta \wedge \left(\alpha \vee I \vee \bigvee_{j=1}^{k-1} v(n_j)\right)\\ &= \beta \wedge (\alpha \vee v_{k-1}),\end{aligned}$$

  where the first line is due to $I \preceq \alpha \vee v(n)$, the second one is by definition of $v(n)$, and the last line is due to equality (2) applied to $i = k - 1$. Hence,

$$\beta \succeq \beta \wedge \big(\alpha \vee v(n)\big) \succeq \beta \wedge (\alpha \vee v_{k-1}) = \beta,$$

  which means all inequalities are equalities. Hence,

$$\beta \wedge \big(\alpha \vee f(n, \alpha, \beta)\big) = \beta \wedge \big(\alpha \vee v_{k-1}\big) = \beta \wedge \big(\alpha \vee v(n)\big).$$

  which means, equality (1) holds for node n.

□

Intuitively, Proposition 1 states that the value returned by Algorithm 1 is the exact value of n up to a factor of α and β. So even for partially ordered values, alpha-beta search can be interpreted as search with a pruning window.

Now it follows that Algorithm 1 is correct in the sense that if we use a lower and an upper bound as the initial search window at a node, we can recover its exact value using the returned value of the algorithm.

Corollary 1. *If h is admissible for G and V, then for any node n of G and any $\underline{v}, \overline{v} \in V$ satisfying $\underline{v} \preceq v(n) \preceq \overline{v}$, we have $v(n) = \overline{v} \wedge \big(\underline{v} \vee f(n, \underline{v}, \overline{v})\big)$. In particular, $v(n) = f(n, \bot, \top)$.*

Proof. Plug $\alpha = \underline{v}$ and $\beta = \overline{v}$ into the equality in Proposition 1, and use the fact that $\underline{v} \vee v(n) = v(n) = \overline{v} \wedge v(n)$. □

Importantly, contrary to the case of totally ordered values, it is *not* true in general that the stronger equality $v(n) = f(n, \underline{v}, \overline{v})$ holds, as the example in Figure 1 (left) shows. Let $h(r, \cdot, \cdot) = \bot = 0000$ and $h(n, \cdot, \cdot) = \top = 1111$, so that h is admissible. Recall that $v(r) = 1100$. For $\underline{v} = 1000$ and $\overline{v} = 1110$, indeed $\underline{v} \preceq v(r) \preceq \overline{v}$. However, $f(r, \underline{v}, \overline{v}) = 1101 \neq v(r)$ since an α-cut happens in the call on n (Line 10) after examining its first child, since at this point $\alpha = \beta = 1100$. However, since $\alpha = \underline{v}$ and $\beta = \overline{v}$ yield no constraint on the fourth component of the values, we still have $v(r) = \overline{v} \wedge \big(\underline{v} \vee f(r, \underline{v}, \overline{v})\big)$, as stated in Corollary 1.

5 Alpha-beta duo algorithm

We now come to the main contribution of our work, namely a caching scheme for reusing previously computed values in an alpha-beta search for partially ordered values. The trouble of standard alpha-beta search is that the returned value is equal to the exact value only up to a factor of α and β. It is therefore a nontrivial question how to reuse a previously computed value, since subsequent revisits of a node may come with different search windows.

In alpha-beta search with cache for totally ordered values [16, for instance], one can exploit the fact that with usual value initialisation, the value $f(n, \alpha, \beta)$ satisfies $f(n, \alpha, \beta) < \beta \Rightarrow v(n) \leq f(n, \alpha, \beta)$ and $f(n, \alpha, \beta) > \alpha \Rightarrow v(n) \geq f(n, \alpha, \beta)$. In particular, if $\alpha < f(n, \alpha, \beta) < \beta$, then $f(n, \alpha, \beta) = v(n)$. Hence by comparing $f(n, \alpha, \beta)$ to α and β, one can determine whether it is exact, a lower, or an upper bound, and store it in the cache with an appropriate flag.

However, this does not hold in general for partially ordered values, as shown on Figure 1 (right). For $\alpha = 010$ and $\beta = 110$, a β-cut happens after evaluating the first child of C, and an α-cut after evaluating the first child of D. Hence, the algorithm returns 010 for R, which is neither a lower nor an upper bound of the exact value 001. In fact, these two values are incomparable in the lattice. If R is an internal node in a DAG, then caching this returned value 010 may cause an evaluation error when the algorithm revisits R.

To tackle this difficulty, we propose a new algorithm named 'alpha-beta duo', which computes a pair of values for all nodes instead of one single value. The algorithm is presented in Algorithm 2, where Cache refers to a transposition table the entries of which are pairs of values indexed by nodes of G, and $(\underline{h}, \overline{h})$ refers to a pair of initialisation functions for which we assume the following property.

Definition 4. *A pair $(\underline{h}, \overline{h})$ of initialisation functions is said to be* admissible *for G and V if for any node n in G, $\underline{h}(n) \preceq v(n) \preceq \overline{h}(n)$ holds, and in addition, if n is a leaf node, $\underline{h}(n) = \overline{h}(n) = v(n)$ holds.*

In other words, admissible $\underline{h}$ and $\overline{h}$ respectively underestimates and overestimates the value of a node. Note that $\underline{h}$ and $\overline{h}$ that assign respectively $\bot$ and $\top$ to all internal nodes form an admissible pair, which can always be used if one does not have better heuristic functions.

Algorithm 2: Alpha-beta duo search

```
1  def AlphaBetaDuo(node n, α, β):
2      if there is an entry for n in the cache:
3          (c̲, c̄) ← Cache(n)
4      else:
5          (c̲, c̄) ← (h̲(n), h̄(n))
6      if c̲ = c̄:
7          return (c̲, c̄)
8      α ← α ∨ c̲
9      β ← β ∧ c̄
10     if n is an OR-node:
11         (v̲, v̄) ← (⊥, ⊥)
12     else:
13         (v̲, v̄) ← (⊤, ⊤)
14     determine the children n_1, ..., n_b of n
15     for i in {1, ..., b}:
16         if α ≥ β:
17             if n is an OR-node:
18                 v̄ = c̄
19             else:
20                 v̲ = c̲
21             break
22         v̲′, v̄′ ← AlphaBetaDuo(n_i, α, β)
23         if n is an OR-node:
24             v̲ ← v̲ ∨ v̲′
25             v̄ ← v̄ ∨ v̄′
26             α ← α ∨ v̲′
27         else:
28             v̲ ← v̲ ∧ v̲′
29             v̄ ← v̄ ∧ v̄′
30             β ← β ∧ v̄′
31     v̲ ← v̲ ∨ c̲
32     v̄ ← v̄ ∧ c̄
33     store (v̲, v̄) in the cache under an entry for n
34     return (v̲, v̄)
```

Alpha-beta duo search works in the following manner:

- First, variables $\underline{c}$ and $\overline{c}$ denote respectively the best lower and upper bound of $v(n)$ available to the algorithm before this call. If n has already been visited, then $\underline{c}$ and $\overline{c}$ are retrieved from the cache. Otherwise, they are given by the initialisation functions $\underline{h}$ and $\overline{h}$. If $\underline{c} = \overline{c}$, $(\underline{c}, \overline{c})$ is returned immediately.
- Otherwise, by symmetry consider the case when n is an OR-node. During the main loop, $\underline{v}$ and $\overline{v}$ are respectively the cumulative lower and upper bound of $v(n)$ (notice that they are both initialised to $\bot$ for an OR-node). If a cut ever happens, it means not all children of n have been evaluated, hence $\overline{v}$ is not a valid upper bound of $v(n)$. Then we take $\overline{v}$ to be $\overline{c}$, the best upper bound previously known. On the other hand, $\underline{v}$, which is the join of lower bounds of children of n that have been evaluated, is a valid lower bound so we keep it.
- Finally, after the main loop, $\underline{v}$ and $\overline{v}$ are the lower and upper bounds of $v(n)$ computed by the current call. Hence they are combined with the previously known bounds $\underline{c}$ and $\overline{c}$ to yield to best currently known bounds on $v(n)$ and they are cached.

We now prove that alpha-beta duo is correct. For this, we first need the following notion.

Definition 5. *A cache* Cache *is said to be* coherent *for G and V if for any node n in G, if there is an entry for n in the cache, then* $\mathsf{Cache}(n) = (\underline{c}, \overline{c})$ *where $\underline{c} \preceq v(n) \preceq \overline{c}$, and in addition, if n is a leaf node, then $\underline{c}(n) = \overline{c}(n) = v(n)$.*

Obviously, an empty cache is coherent for any G and V.

In the following, we denote the pair of values returned by Algorithm 2 with input n, α, β by $(\underline{f}(n, \alpha, \beta), \overline{f}(n, \alpha, \beta))$. We first show that if the cache is initially coherent, then it remains coherent after the execution, and that any interval stored in it cannot become looser.

Proposition 2. *If $(\underline{h}, \overline{h})$ is admissible and* Cache *is initially coherent for G and V, then for any node n in G and any $\alpha, \beta \in V$, we have*

$$\underline{f}(n, \alpha, \beta) \preceq v(n) \preceq \overline{f}(n, \alpha, \beta). \tag{4}$$

Moreover, if there is an entry $(\underline{c}, \overline{c})$ in the cache for n before the call, then $\underline{c} \preceq \underline{f}(n, \alpha, \beta)$ and $\overline{f}(n, \alpha, \beta) \preceq \overline{c}$.

Proof. $\underline{c} \preceq \underline{f}(n, \alpha, \beta)$ and $\overline{f}(n, \alpha, \beta) \preceq \overline{c}$ are direct consequence of Line 31 and 32.

The proof of inequality (4) is based on structural induction. We will only focus on OR nodes since the case for AND nodes is completely symmetric. So let n be an OR node and let us consider the execution of the function call $\texttt{AlphaBetaDuo}(n, \alpha, \beta)$.

If n is a leaf node, then whether or not there is an entry for n in the cache, on Line 6 we have $\underline{c} = \overline{c} = v(n)$ since $(\underline{h}, \overline{h})$ is admissible and Cache is coherent. Hence the function returns immediately, so inequality (4) holds and the cache remains coherent.

Otherwise, n is an internal OR-node. Again, whether or not there is an entry for n in the cache, on Line 6 and forward we have $\underline{c} \preceq v(n) \preceq \overline{c}$, since $(\underline{h}, \overline{h})$ is admissible and Cache is coherent.

Let $b \geq 1$ be the number of children of n. We assume by induction that all function calls on the children of n satisfy

inequality (4) and maintain the coherence of the cache. Let k be the index of loop during which a break happens (if no break happens, then k is taken to be $b+1$). For $1 \le i \le k-1$, let $\underline{v}^i$ and $\overline{v}^i$ denote the values returned by the function call on the child n_i during the ith loop. Then by induction assumption, inequality (4) yields $\underline{v}^i \preceq v(n_i) \preceq \overline{v}^i$ for $1 \le i \le k-1$.

Hence after the loop (i.e. just before Line 31),

$$\underline{v} = \bigvee_{i=1}^{k-1} \underline{v}^i \preceq \bigvee_{i=1}^{k-1} v(n_i) \preceq \bigvee_{i=1}^{b} v(n_i) = v(n).$$

As for $\overline{v}$, we have two cases.

- No break happens, i.e. $k = b+1$. Then

$$\overline{v} = \bigvee_{i=1}^{b} \overline{v}^i \succeq \bigvee_{i=1}^{b} v(n_i) = v(n).$$

- Otherwise, a break happens, and $\overline{v} = \overline{c} \succeq v(n)$.

So in both cases, $\overline{v} \succeq v(n)$.

Therefore, after the final updates on Line 31 and 32, we have $\underline{v} \preceq v(n) \preceq \overline{v}$. As a result, the returned values of the function call $\texttt{AlphaBetaDuo}(n, \alpha, \beta)$ satisfy inequality (4). And the cache remains coherent after the function call. □

We can now prove results parallel to those in Section 4.

Proposition 3. *If $(\underline{h}, \overline{h})$ is admissible and* Cache *is initially coherent for G and V, then for any node n in G and any $\alpha, \beta \in V$ we have*

$$\beta \wedge \big(\alpha \vee \underline{f}(n, \alpha, \beta)\big) = \beta \wedge \big(\alpha \vee \overline{f}(n, \alpha, \beta)\big). \quad (5)$$

Proof. Again, we will only focus on OR nodes since the case for AND nodes is symmetric.

If n is a leaf node, then $\underline{f}(n, \alpha, \beta) = \overline{f}(n, \alpha, \beta) = v(n)$ since $(\underline{h}, \overline{h})$ is admissible and Cache is coherent. Hence equality (5) holds.

Otherwise, let $b \ge 1$ be the number of children of n. We assume by induction that all function calls on the children of n satisfy inequality (5). Let k be the index of loop during which a break happens (if no break happens, then k is taken to be $b+1$). For $1 \le j < k$, let $\underline{v}^j$ and $\overline{v}^j$ denote the values returned by the function call on n_j during the jth loop. For $0 \le i < k$, let $\underline{v}_i$, $\overline{v}_i$, and α_i denote the value of $\underline{v}$, $\overline{v}$, and α after the ith loop. Then for $0 \le i < k$, we have $\underline{v}_i = \bigvee_{j=1}^{i} \underline{v}^j$, $\overline{v}_i = \bigvee_{j=1}^{i} \overline{v}^j$, and $\alpha_i = \alpha \vee \underline{c} \vee \underline{v}_i$. In particular, $\underline{v}_0 = \overline{v}_0 = \bot$ and $\alpha_0 = \alpha \vee \underline{c}$. For $1 \le j < k$, since function call on the child n_j has the form $\texttt{AlphaBetaDuo}(n_j, \alpha_j, \beta \wedge \overline{c})$, by equality (5), we have

$$\beta \wedge \overline{c} \wedge \big(\alpha_j \vee \underline{v}^j\big) = \beta \wedge \overline{c} \wedge \big(\alpha_j \vee \overline{v}^j\big).$$

Hence by distributivity, we have

$$\beta \wedge \overline{c} \wedge \bigvee_{j=1}^{k-1} \big(\alpha_j \vee \underline{v}^j\big) = \beta \wedge \overline{c} \wedge \bigvee_{j=1}^{k-1} \big(\alpha_j \vee \overline{v}^j\big). \quad (6)$$

We distinguish two cases.

- No break happens (i.e. $k = b+1$). Then

$$\underline{f}(n, \alpha, \beta) = \underline{c} \vee \underline{v}_b = \underline{c} \vee \bigvee_{j=1}^{b} \underline{v}^j,$$

$$\overline{f}(n, \alpha, \beta) = \overline{c} \wedge \overline{v}_b = \overline{c} \wedge \bigvee_{j=1}^{b} \overline{v}^j.$$

First notice that the distributivity of the lattice implies that for any $x, y, z \in V$,

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) = x \vee \big(y \wedge (x \vee z)\big), \quad (7)$$

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) = x \wedge \big(y \vee (x \wedge z)\big). \quad (8)$$

Applying equalities (7) and (8), one has

$$\begin{aligned}
\beta \wedge \big(\alpha \vee \overline{f}(n, \alpha, \beta)\big) &= \beta \wedge \left(\alpha \vee \left(\overline{c} \wedge \bigvee_{j=1}^{b} \overline{v}^j\right)\right) \\
&= \beta \wedge \left(\alpha \vee \left(\overline{c} \wedge \left(\alpha \vee \bigvee_{j=1}^{b} \overline{v}^j\right)\right)\right) \\
&= \beta \wedge \left(\alpha \vee \left(\beta \wedge \overline{c} \wedge \left(\alpha \vee \bigvee_{j=1}^{b} \overline{v}^j\right)\right)\right).
\end{aligned}$$

We will first focus on the term $\beta \wedge \overline{c} \wedge (\alpha \vee \bigvee_{j=1}^{b} \overline{v}^j)$. Our goal is to massage it into a form to which the induction assumption (6) can apply.

Recall that for $j \le b$, $\alpha_j = \alpha_0 \vee \underline{v}_j = \alpha_0 \vee \bigvee_{l=1}^{j} \underline{v}^l$. Hence,

$$\begin{aligned}
\bigvee_{j=1}^{b} (\alpha_j \vee \overline{v}^j) &= \bigvee_{j=1}^{b} \left(\alpha_0 \vee \bigvee_{l=1}^{j} \underline{v}^l \vee \overline{v}^j\right) \\
&= \alpha_0 \vee \bigvee_{j=1}^{b} \bigvee_{l=1}^{j} \underline{v}^l \vee \bigvee_{j=1}^{b} \overline{v}^j \\
&= \alpha_0 \vee \bigvee_{j=1}^{b} \underline{v}^j \vee \bigvee_{j=1}^{b} \overline{v}^j \\
&= \alpha_0 \vee \bigvee_{j=1}^{b} \overline{v}^j,
\end{aligned}$$

where in the last equality we use the fact that by Proposition 2, we have $\underline{v}^j \preceq \overline{v}^j$ for any $j \leq b$. Similarly, we have

$$\bigvee_{j=1}^{b}(\alpha_j \vee \underline{v}^j) = \bigvee_{j=1}^{b}\left(\alpha_0 \vee \bigvee_{l=1}^{j} \underline{v}^l \vee \underline{v}^j\right) = \alpha_0 \vee \bigvee_{j=1}^{b} \underline{v}^j.$$

Hence, by $\alpha_0 = \alpha \vee \underline{c} \succeq \alpha$, the two previous equalities, distributivity, and the induction assumption (6),

$$\begin{aligned}
\beta \wedge \overline{c} \wedge \left(\alpha \vee \bigvee_{j=1}^{b} \overline{v}^j\right) &\preceq \beta \wedge \overline{c} \wedge \left(\alpha_0 \vee \bigvee_{j=1}^{b} \overline{v}^j\right) \\
&= \beta \wedge \overline{c} \wedge \bigvee_{j=1}^{b}(\alpha_j \vee \overline{v}^j) \\
&= \beta \wedge \overline{c} \wedge \bigvee_{j=1}^{b}(\alpha_j \vee \underline{v}^j) \\
&= \beta \wedge \overline{c} \wedge \left(\alpha_0 \vee \bigvee_{j=1}^{b} \underline{v}^j\right) \\
&\preceq \beta \wedge \left(\alpha_0 \vee \bigvee_{j=1}^{b} \underline{v}^j\right).
\end{aligned}$$

Therefore, applying again equalities (7) and (8) yields

$$\begin{aligned}
&\beta \wedge \left(\alpha \vee \overline{f}(n, \alpha, \beta)\right) \\
&= \beta \wedge \left(\alpha \vee \left(\beta \wedge \overline{c} \wedge \left(\alpha \vee \bigvee_{j=1}^{b} \overline{v}^j\right)\right)\right) \\
&\preceq \beta \wedge \left(\alpha \vee \left(\beta \wedge \left(\alpha_0 \vee \bigvee_{j=1}^{b} \underline{v}^j\right)\right)\right) \\
&= \beta \wedge \left(\alpha \vee \alpha_0 \vee \bigvee_{j=1}^{b} \underline{v}^j\right) \\
&= \beta \wedge \left(\alpha \vee \underline{c} \vee \bigvee_{j=1}^{b} \underline{v}^j\right) \\
&= \beta \wedge \left(\alpha \vee \underline{f}(n, \alpha, \beta)\right).
\end{aligned}$$

- A break happens (i.e. $1 \leq k \leq b$). Then $\overline{f}(n, \alpha, \beta) = \overline{c}$ and $\underline{f}(n, \alpha, \beta) = \underline{c} \vee \underline{v}_{k-1}$. Since a break happens during the kth loop, according to Line 16 in Algorithm 2 we have $\beta \wedge \overline{c} \preceq \alpha_{k-1} = \alpha \vee \underline{c} \vee \underline{v}_{k-1}$. Hence by distributivity,

$$\begin{aligned}
\beta \wedge \left(\alpha \vee \overline{f}(n, \alpha, \beta)\right) &= \beta \wedge (\alpha \vee \overline{c}) \\
&= \beta \wedge \left(\alpha \vee (\beta \wedge \overline{c})\right) \\
&\preceq \beta \wedge \left(\alpha \vee (\alpha \vee \underline{c} \vee \underline{v}_{k-1})\right) \\
&= \beta \wedge \left(\alpha \vee \underline{f}(n, \alpha, \beta)\right).
\end{aligned}$$

Therefore, no matter a break happens or not, we have

$$\beta \wedge \left(\alpha \vee \overline{f}(n, \alpha, \beta)\right) \preceq \beta \wedge \left(\alpha \vee \underline{f}(n, \alpha, \beta)\right).$$

On the other hand, by Proposition 2, $\overline{f}(n, \alpha, \beta) \succeq \underline{f}(n, \alpha, \beta)$, which means

$$\beta \wedge \left(\alpha \vee \overline{f}(n, \alpha, \beta)\right) \succeq \beta \wedge \left(\alpha \vee \underline{f}(n, \alpha, \beta)\right).$$

As a consequence, equality (5) holds. □

Corollary 2. *If $(\underline{h}, \overline{h})$ is admissible and* Cache *is initially coherent for G and V, then for any node n in G and any $\underline{v}, \overline{v} \in V$ satisfying $\underline{v} \preceq v(n) \preceq \overline{v}$, we have $v(n) = \overline{v} \wedge \left(\underline{v} \vee \underline{f}(n, \underline{v}, \overline{v})\right) = \overline{v} \wedge \left(\underline{v} \vee \overline{f}(n, \underline{v}, \overline{v})\right)$. In particular, $v(n) = \underline{f}(n, \bot, \top) = \overline{f}(n, \bot, \top)$.*

Proof. By Proposition 2, $\underline{f}(n, \underline{v}, \overline{v}) \preceq v(n) \preceq \overline{f}(n, \underline{v}, \overline{v})$. Using $\overline{v} \wedge \left(\underline{v} \vee v(n)\right) = v(n)$, we get

$$\overline{v} \wedge \left(\underline{v} \vee \underline{f}(n, \underline{v}, \overline{v})\right) \preceq v(n) \preceq \overline{v} \wedge \left(\underline{v} \vee \overline{f}(n, \underline{v}, \overline{v})\right).$$

These inequalities are in fact equalities since from Proposition 3 we have $\overline{v} \wedge \left(\underline{v} \vee \underline{f}(n, \underline{v}, \overline{v})\right) = \overline{v} \wedge \left(\underline{v} \vee \overline{f}(n, \underline{v}, \overline{v})\right)$. □

6 Experiments

To assess the efficiency of alpha-beta duo (hereafter 'ABD'), we ran experiments comparing it to three other algorithms:

- alpha-beta without cache (Algorithm 1, 'AB' for short);
- an alpha-beta search which only caches the value computed for a node if no cut is found during the search in the subtree rooted at this node (hereafter 'ABC');
- a minimax search algorithm without alpha-beta pruning, but with a cache (hereafter 'MMC').

The code of ABD was slightly optimized by refining the values computed on Lines 31 and 32 with the corresponding bounds of all fully explored children. It is easy to show that this preserves the correctness of the algorithm.

For all experiments, we measured the number of nodes of the DAG visited at least once, the total number of node visits (equivalently, the total number of recursive calls), and the time taken for solving the problem. Intuitively, we expect ABD to be better than ABC, ABC to be better than MMC (because MMC does not use alpha-beta pruning), and MMC to be better than AB (because the latter does not cache its results and hence, recomputes several times for the same node).

We used two synthetic sets of benchmarks. The first (hereafter 'random') consists of random DAGs of the same kind as the one in Figure 1. Random DAGs with parameters d (depth), b (branching factor), and v (number of variables) are generated in the following manner:

- d layers $0, 1, \ldots, d-1$ are built: layer i consists of $3^{\min(i, \frac{3d}{2}-i)}$ nodes (which yields diamond-shapes DAGs);
- from each node n, a set of b nodes is randomly chosen from nodes in the next layer to be $C(n)$;
- each internal node is labelled AND or OR at random;
- the value of each terminal node is a uniformly drawn subset of $\{1, \ldots, v\}$, or equivalently a random Boolean vector of length v.

We also consider strictly alternating DAGs, in which all nodes in layer i are OR-nodes (respectively AND-nodes) if i is even (respectively odd). In particular, the root is an OR-node.

The second set of benchmarks consists of a simplified version of the card game Bridge that we call 'racing'. There are two players, MIN and MAX. Each has a hand of h cards drawn uniformly from the deck $\{1, \ldots, d\}$ where $d \geq 2h$. Players only see their own hand. MIN begins the game. During each trick, the player who begins plays a card from her hand, the other sees it and plays a card in turn. The player who played the highest card wins this trick and starts the next one. No new card is ever drawn from the deck. The game ends when the players have no more cards or when one has won in total g tricks. MAX wins if she is the first one to reach g tricks. For the benchmark, each instance with parameter h, d, and g consists of a randomly drawn hand with h cards for MAX and a randomly drawn card that is supposed to be played by MIN during the first trick. Notice that when $d > 2h$, each player has incomplete information. We use evaluation of AND-OR DAGs to compute optimal strategies for MAX against the best defence adversary model defined in [6].

In games with incomplete information where S is the set of all possible hidden configurations, [6, 7] define the maxmin value of player MAX to be the set of all subsets S' of S such that there is a uniform strategy winning in any configuration of S'. [9] shows that computing this value amounts to evaluate the game DAG with the lattice $(2^{2^S}, \preceq, \sqcap, \cup)$. Intuitively, using $\cup$ at OR-nodes models the fact that player MAX can choose any child as her strategy, and $\sqcap$ at AND-nodes models the fact that a strategy must be robust to all adversarial strategies, hence a strategy wins in $s \in S$ if and only if it wins in s whatever action her opponent chooses.

In the same setting of games with incomplete information, one can also be interested in non-uniform strategies which allow player MAX's actions to depend on the hidden information. This can be useful for computing heuristic values of for the game with incomplete information. It can be seen that the set of all configurations for which there is a non-uniform winning strategy can be computed as the value of the game DAG with the lattice $(2^S, \subseteq, \cap, \cup)$.

Hence, for both benchmarks, we consider the two lattices $(2^{2^S}, \sqcap, \cup)$ and $(2^S, \cap, \cup)$. In 'random', $S = \{1, \ldots, v\}$, while in 'racing' S is the set of all possible hands of player MIN.

For space reasons, we only give the most representative results, in terms of computation time. For each parameter setting and each algorithm, we averaged over 10 runs. Figure 3 shows two examples where, as can be expected, it is more efficient to cache bounds, even more to perform cuts, and still more to compute and store two bounds per node. On the top plot, the gain of using ABD is exponential: with the branching factor increasing, ABD gets a better advantage of computing and caching two bounds. On the bottom plot, ABC and AB (not represented) are exponentially worse, and ABD is better than MMC when the branching factor is high.

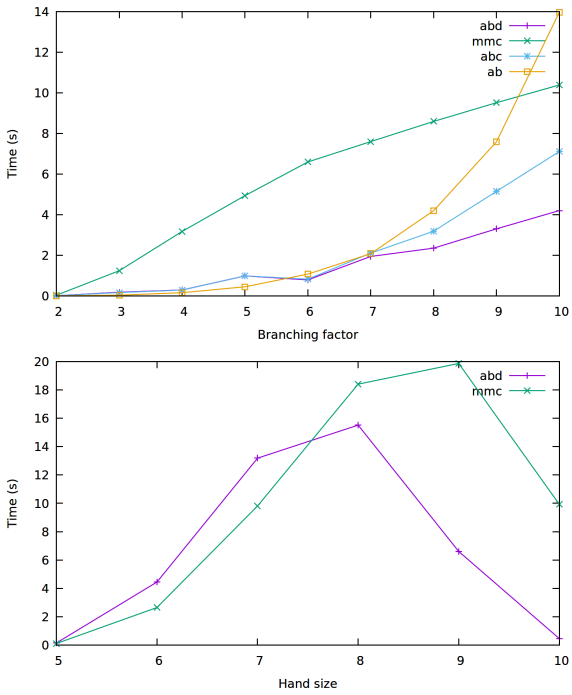


Figure 3: Experimental results on random (top) and racing (bottom). Top: $d = 15$, $v = 10$ (varying b), alternating DAGs. Bottom: $d = 20$, $g = 5$ (varying h). The lattice is 2^S in both cases.

Now Figure 4 shows two examples where it turns out that it is not always better to use alpha-beta pruning with cache.

On the top plot, not caching results at all turns out to be better: the overhead due to the additional operations from the lattice 2^{2^S} (which are necessary to maintain the cache) seems to compensate the advantage of ABC or ABD in terms of number of visited nodes and recursive calls (the curves are reversed for this metrics, not shown here).

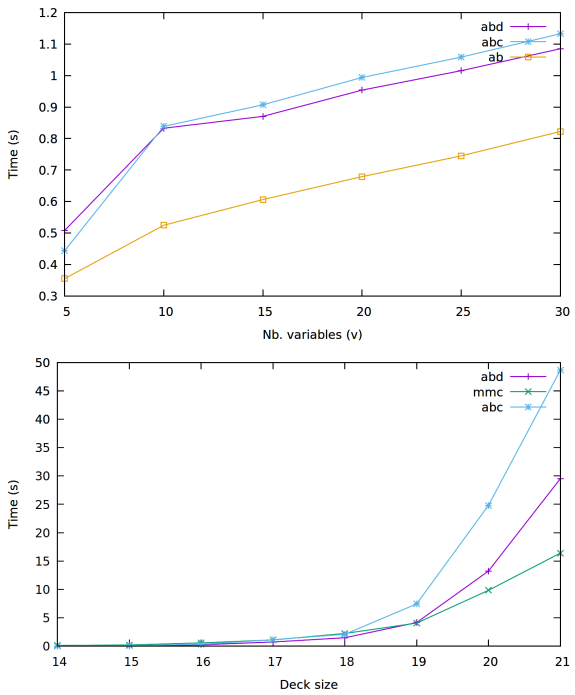


Figure 4: Experimental results on random (top) and racing (bottom). Top: $d = 15$, $b = 4$ (varying v), alternating DAGs, lattice 2^{2^S}. Bottom: $h = 7$, $g = 5$ (varying d), lattice 2^S.

On the bottom plot, it turns out that sometimes alpha-beta pruning even degrades performance. Again, this is due to the overhead of manipulating values from a large lattice (for a fixed hand size, the lattice grows exponentially with the deck size).

To complete these results, let us mention that in most experiments, the numbers of nodes explored and visited by each method are ordered as expected, with ratios varying from linear to exponential. In particular, for these metrics, ABD is most of the time better, and often much better, than the other three algorithms.

Summarising, ABD seems to provide a real gain in (brute) performance for DAGs with high branching factors. Contrastingly, when $\wedge$ and $\vee$ from the lattice are too expensive to compute (as is the case in some large lattices), it may sometimes be better not to use cache and alpha-beta pruning together, due to the overhead to maintain the coherence of the cache.

7 Conclusion

We investigated alpha-beta search for AND-OR DAGs with values from a lattice, which has direct applications such as solving games with incomplete information. We have extended previous formal analyses, in particular to the use of heuristic as initialisation functions. Then we have proposed a new algorithm named 'alpha-beta duo', which caches both a lower and an upper bound of the value of each visited node, and we have formally proved its correctness. Experiments show that it is more efficient than other algorithms in terms of number of visited nodes and recursive calls. As for time efficiency, alpha-beta duo turns out to be more efficient than other algorithms for DAGs with large branching factors and reasonably-sized lattices. As an interesting conclusion, our experiments also put forth that in other cases, it may be better not to use a cache with alpha-beta pruning.

Our future work includes algorithmic optimisations for alpha-beta search with cache applied to games with incomplete information. We will also investigate the use of efficient knowledge representations [1, 18] to accelerate lattice operations in such context. Another perspective is to apply our work to games with multiple criteria instead of scalar outcomes.

References

[1] Bienvenu, Meghyn, Hélène Fargier, and Pierre Marquis: *Knowledge Compilation in the Modal Logic S5*. In *Proc. 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, pages 261–266, 2010. `https://hal.archives-ouvertes.fr/hal-00624482`.

[2] Cazenave, Tristan and Véronique Ventos: *The $\alpha\mu$ search algorithm for the game of bridge*. In *Proc. Monte Carlo Search International Workshop*, pages 1–16. Springer, 2020.

[3] Dasgupta, Pallab, P. P. Chakrabarti, and S. C. De Sarkar: *Searching game trees under a partial order*. Artif. Intell., 82(1-2):237–257, 1996. `https://doi.org/10.1016/0004-3702(94)00085-9`.

[4] Davey, Brian A. and Hilary A. Priestley: *Introduction to Lattices and Order*. Cambridge University Press, 2nd edition, 2002, ISBN 978-0-521-78451-1. `https://doi.org/10.1017/CBO9780511809088`.

[5] Edelkamp, Stefan: *Representing and reducing uncertainty for enumerating the belief space to improve endgame play in skat*. In *Proc. 24th European Conference on Artificial Intelligence (ECAI 2020)*, pages 395–402. IOS Press, 2020.

[6] Frank, Ian and David A. Basin: *Search in games with incomplete information: A case study using bridge card play*. Artif. Intell., 100(1-2):87–123, 1998. `https://doi.org/10.1016/S0004-3702(97)00082-9`.

[7] Frank, Ian and David A. Basin: *A theoretical and empirical investigation of search in imperfect in-*

formation games. Theor. Comput. Sci., 252(1-2):217–256, 2001. https://doi.org/10.1016/S0304-3975(00)00083-9.

[8] Ginsberg, M and Alan Jaffray: *Alpha-beta pruning under partial orders*. In Nowakowski, Richard (editor): *More Games of No Chance*, number 42 in *Mathematical Sciences Research Institute Publications*, pages 37–48. Cambridge University Press, 2002.

[9] Ginsberg, Matthew L.: *GIB: imperfect information in a computationally challenging game*. J. Artif. Intell. Res., 14:303–358, 2001. https://doi.org/10.1613/jair.820.

[10] Haworth, Guy and Nelson Hernandez: *The 20th top chess engine championship, TCEC20*. ICGA Journal, 43(1):62–73, 2021.

[11] Kissmann, Peter and Stefan Edelkamp: *Solving fully-observable non-deterministic planning problems via translation into a general game*. In Mertsching, Bärbel, Marcus Hund, and Muhammad Zaheer Aziz (editors): *Proc. 32nd Annual German Conference on Artificial Intelligence (KI 2009)*, pages 1–8. Springer, 2009.

[12] Knuth, Donald E. and Ronald W. Moore: *An analysis of alpha-beta pruning*. Artif. Intell., 6(4):293–326, 1975. https://doi.org/10.1016/0004-3702(75)90019-3.

[13] Kupferschmid, Sebastian and Malte Helmert: *A skat player based on monte-carlo simulation*. In *Proc. 5th International Conference on Computers and Games (CG 2006)*, pages 135–147. Springer, 2006.

[14] Levy, David NL: *The million pound bridge program*. Heuristic Programming in Artificial Intelligence: The First Computer Olympiad, pages 95–103, 1989.

[15] Loddo, Jean-Vincent and Luca Saiu: *How to correctly prune tropical trees*. In Autexier, Serge, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton (editors): *Proc. 10th International Conference on Intelligent Computer Mathematics*, volume 6167 of *Lecture Notes in Computer Science*, pages 101–115. Springer, 2010. https://doi.org/10.1007/978-3-642-14128-7_10.

[16] Marsland, T. A.: *A review of game-tree pruning*. J. Int. Comput. Games Assoc., 9(1):3–19, 1986. https://doi.org/10.3233/ICG-1986-9102.

[17] Nasu, Yu: *Efficiently updatable neural-network-based evaluation functions for computer shogi*. The 28th World Computer Shogi Championship Appeal Document, 2018.

[18] Niveau, Alexandre and Bruno Zanuttini: *Efficient Representations for the Modal Logic S5*. In *Proc. 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 2016. https://hal.archives-ouvertes.fr/hal-01311642.

[19] Rebstock, Douglas, Christopher Solinas, Michael Buro, and Nathan R Sturtevant: *Policy based inference in trick-taking card games*. In *Proc. 2019 IEEE Conference on Games (CoG 2019)*, pages 1–8. IEEE, 2019.

[20] Schaeffer, Jonathan, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen: *Checkers is solved*. Science, 317(5844):1518–1522, 2007.

[21] Sturtevant, Nathan R and Adam M White: *Feature construction for reinforcement learning in hearts*. In *Proc. 5th International Conference on Computers and Games (CG 2006)*, pages 122–134. Springer, 2006.

[22] Van Den Herik, H Jaap, Jos WHM Uiterwijk, and Jack Van Rijswijck: *Games solved: Now and in the future*. Artif. Intell., 134(1-2):277–311, 2002.

Actes JIAF 2022

A KC Map for Variants of Nondeterministic PDDL

Sergej Scheck Alexandre Niveau Bruno Zanuttini
Normandie Univ.; UNICAEN, ENSICAEN, CNRS, GREYC, 14 000 Caen, France
{sergej.scheck,alexandre.niveau,bruno.zanuttini}@unicaen.fr

Résumé

Nous étudions différents langages permettant de représenter des actions non-déterministes pour la planification automatique, du point de vue de la compilation de connaissances. Précisément, nous considérons la question de la concision des langages (quelle est la taille de la description d'une action dans chaque langage ?) et des questions de complexité (traitabilité ou dureté de plusieurs requêtes et transformations qui surviennent naturellement dans la planification et le suivi des croyances). Nous étudions une version abstraite et nondéterministe de PDDL, STRIPS conditionnel nondéterministe, et les langages $\text{NPDDL}_{\text{seq}}$ et $\text{NPDDL}_{\text{not}}$ obtenus en ajoutant séquence et négation à PDDL nondéterministe. Nous montrons que ces langages ont une concision et complexité différente pour les requêtes les plus naturelles.

Abstract

We study different languages for representing nondeterministic actions in planning from the point of view of knowledge compilation. Precisely, we consider succintness issues (how succinct is the description of an action in each language?), and complexity issues (tractability or hardness of several queries and transformations which arise naturally in planning and belief tracking). We study an abstract, nondeterministic version of PDDL, nondeterministic conditional STRIPS, and the languages $\text{NPDDL}_{\text{seq}}$ and $\text{NPDDL}_{\text{not}}$ obtained by adding sequence and negation to nondeterministic PDDL. We show that these languages have different succinctness and different complexity for the most natural queries.

1 Introduction

In automated planning, a central aspect of the description of problems is the formal representation of actions. Such representations are indeed needed for specifying the actions available to the agent (PDDL [13] is a standard language for this), and also for planners to operate on them while searching for a plan.

In this paper, we consider different representation languages within the formal framework of the knowledge compilation map [5]. This framework deals with the study of formal languages under the point of view of queries (how efficient is it to answer various queries, depending on the language ?), transformations (how efficient is it to transform or combine different representations in a given language ?), and succinctness (how concise is it to represent knowledge in each language ?).

The knowledge compilation map has been introduced for representations of Boolean functions [5]. As far as we know there has been no systematic study of languages for representing actions per themselves. This is however an important problem, as planners need to query action representations again and again while searching for a plan (e.g., to find out which actions are applicable at the current node of the search tree), and many of them start by transforming the action specifications into some representation suited for this [11, 24, 3, 22]. Hence having a clear picture of the properties of languages is of interest for developing such planners.

This paper is an attempt at a systematic study of various action languages from the point of view of knowledge compilation. Works with related objectives do exist, but the focus has been on other aspects of planning, like the representation of plans [2], axioms [23], or action costs [21]. The only studies about action languages which we are aware of are those pioneered by Bäckström, but they essentially consider the representation of actions up to preservation of plan length [1, 16, 18], while we are interested in a stricter notion whereby the semantics is precisely preserved, hence preserving more aspects of the planning tasks (like the number of plans, relationships between variables, etc.).

We are interested here in (purely) nondeterministic actions, which lie at the core of fully observable nondeterministic planning and of conformant planning [19, 17, 7, 14, 24, 8]. We also focus on propositional domains, in which states are assignments to a given set of propositions.

Our mid-term goal is to give a systematic picture of languages arising from combinations of allowed constructs among the ones introduced in the literature (like nondeterministic choice, iteration, persistency by default, etc.),

and in this work we consider abstract languages which resemble variants and extensions of the well-known STRIPS and PDDL. Orthogonally, we also study two concrete representations of expressions, as syntactic *trees* or as more compact *circuits*, where identical subexpressions are not repeated. The former representation gives a natural measure of the size of action specifications, while the latter is more compact and is the one typically used in the knowledge compilation literature [5].

The paper is structured as follows. We first give background about actions and logic (Section 2), then formally define the action languages which we consider (Section 3). Then we give our results : results about the complexity of queries (Section 4); separation results, which allow us to determine the relative succinctness of the languages (Section 5), and results about the complexity of transformations (Section 6). Finally we conclude, discussing some open problems and perspectives of this work (Section 7).

2 Preliminaries

We consider a countable set of propositional *state* variables $\mathbb{P} := \{p_i \mid i \in \mathbb{N}\}$. Let $P \subset \mathbb{P}$ be a nonempty finite set of state variables; a subset of P is called a *P-state*, or simply a *state*. The intended interpretation of a state $s \in 2^P$ is the assignment to P in which all variables in s are true, and all variables in $P \setminus s$ are false. For instance, for $P := \{p_1, p_2, p_3\}$, $s := \{p_1, p_3\}$ denotes the state in which p_1, p_3 are true and p_2 is false. We write $\mathrm{V}(\varphi)$ for the set of variables occuring in an expression φ.

Actions We consider (purely) nondeterministic actions, which map states to sets of states. Hence a single state may have several successors through the same action, in contrast with deterministic actions (which map states to states), and no relative likelihood is encoded between the successors of a state, in contrast with stochastic actions (which map states to probability distributions over states).

Definition 1 (action)**.** Let $P \subset \mathbb{P}$ be a finite set of variables. A *P-action* is a mapping a from 2^P to $2^{(2^P)}$. The states in $a(s)$ are called *a-successors* of s and P is called the *scope* of a.

Note that $a(s)$ is defined for all states s. For our results explicit preconditions are not important; we will consider a to be *applicable* in s if and only if $a(s) \neq \emptyset$. Every action a can be identified with a binary *transition relation* $\|a\|$ on states which is defined via $\|a\| := \{(s, s') \mid s' \in a(s)\}$. Elements of $\|a\|$ are called *state transitions*.

In this article, we are interested in the properties of *representations* of actions in various languages.

Definition 2 (action language)**.** An *action language* is an ordered pair $\langle L, I\rangle$, where L is a set of expressions and I is a partial function from $L \times 2^{\mathbb{P}}$ to the set of all actions such that, when defined on $\alpha \in L$ and $P \subset \mathbb{P}$, $I(\alpha, P)$ is a P-action. If $\langle L, I\rangle$ is an action language and $L' \subseteq L$ then we call $\langle L', I|_{L' \times 2^{\mathbb{P}}}\rangle$ a *sublanguage* of $\langle L, I\rangle$.

We call the expressions in L *action descriptions*, and call I the *interpretation function* of the language. In this article $I(\alpha, P)$ will be defined if and only if $\mathrm{V}(\alpha) \subseteq P$.

If the language $\langle L, I\rangle$ and the set P are fixed or clear from the context, then we write $\alpha(s)$ instead of $I(\alpha, P)(s)$ for the set of all α-successors of s. We might also say "action α", meaning the action described by the action description α, and write $\|\alpha\|$ for $\|I(\alpha, P)\|$.

Translations In this article, we are interested in the existence of translations between languages which preserve the scope of action descriptions, i.e., which do not add new state variables to the scope in the destination language.

Definition 3 (translation)**.** A *translation* from an action language $\langle L_1, I_1\rangle$ to another language $\langle L_2, I_2\rangle$ is a function $f\colon L_1 \times 2^{\mathbb{P}} \to L_2$ such that $I_1(\alpha, P) = I_2(f(\alpha, P), P)$ holds for all $\alpha \in L_1$ and $P \subset \mathbb{P}$ such that $I_1(\alpha, P)$ is defined.

In words, this means that the L_1-expression α and the L_2-expression $f(\alpha, P)$ describe the same P-action.

Here, a translation is *not* allowed to introduce auxiliary variables. This is in contrast with many studies of compilation for planning. The reason why we focus on this setting is that we are interested in translating the *actions*, and not only the solutions to a planning problem. Our strict notion guarantees that translating the actions of a domain will preserve all properties : the existence of solution plans of course, but also their length, their number, and measures of complexity like many notions of width [17, 12, for instance].

The translation f is said to be *polynomial-time* if it can be computed in time polynomial in the size of α and P, and *polynomial-size* if the size of $f(\alpha, P)$ is bounded by a fixed polynomial in the size of α and P. Clearly, a polynomial-time translation is necessarily also a polynomial-size one, but the converse is not true in general.

Negation Normal Form A Boolean formula φ over a set Q of variables is said to be in *negation normal form* (NNF) if it is built up from literals using conjunctions and disjunctions, i.e., if it is generated by the following grammar (where q ranges over Q) :

$$\varphi ::= q \mid \neg q \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

3 Action Languages

We first define the semantics for operators which then will be combined to obtain the languages we study. Applying a nondeterministic action a in state s can be seen as choosing

nondeterministically which groups of variables to assign from a set of possible combinations. This motivates the following definition.

Definition 4 (effect)**.** An *effect* over a set of variables $P \subset \mathbb{P}$ is an ordered pair $\langle Q^+, Q^- \rangle$ with $Q^+, Q^- \subseteq P$ and $Q^+ \cap Q^- = \emptyset$. The set Q^+ (resp. Q^-) is called a *positive* (resp. *negative*) effect.

In the following we are going to define the set $\mathrm{E}(\alpha, P, s)$ of *explicit effects of* α *in* s for action descriptions α, but we already use it to define the interpretation function, which simply formalizes the fact that variables not explicitly set by the action retain their value. For all action descriptions α and sets of variables P with $\mathrm{V}(\alpha) \subseteq P \subset \mathbb{P}$, for all $s \subseteq P$:

$$I(\alpha, P)(s) := \{(s \cup Q^+) \setminus Q^- \mid \langle Q^+, Q^- \rangle \in \mathrm{E}(\alpha, P, s)\}$$

An effect $\langle Q^+, Q^- \rangle$ is said to *witness* a transition (s, s') if $(s \cup Q^+) \setminus Q^- = s'$. We emphasize that for a transition (s, s') there may exist several effects witnessing it. For example, given $P := \{p_1, p_2\}$, both effects $\langle \{p_2\}, \emptyset \rangle$ and $\langle \{p_1, p_2\}, \emptyset \rangle\}$ cause the same transition from $s := \{p_1\}$ to $s' := \{p_1, p_2\}$.

We say that two effects $e_1 := \langle Q_1^+, Q_1^- \rangle$, $e_2 := \langle Q_2^+, Q_2^- \rangle$ *are consistent* if $Q_1^+ \cap Q_2^- = Q_2^+ \cap Q_1^- = \emptyset$; in other words, if one effect assigns $\top$ to a variable, the other does not assign $\bot$ to it. Then the *combination* of e_1 and e_2 is defined to be the effect $\langle Q_1^+ \cup Q_2^+, Q_1^- \cup Q_2^- \rangle$. It can be interpreted as simultaneously executing the assignments of e_1 and e_2.

Definition 5. Our action languages will be defined as various combinations of constructs from the grammar

$$\alpha ::= \varepsilon \mid +p \mid -p \mid \varphi \rhd \alpha \mid (\alpha \cup \alpha) \mid \alpha \sqcap \alpha \mid \alpha \,;\, \alpha \mid \neg_{\min} \alpha$$

Intuitively, ε describes the action with no effect ($\forall s\colon \varepsilon(s) = \{s\}$), $+p$ (resp. $-p$) is the action which sets p true (resp. false), $\rhd$ denotes conditional execution, $\cup$ denotes (exclusive) nondeterministic choice, $\sqcap$ denotes simultaneous execution, ; denotes sequential execution, and $\neg_{\min}$ describes the negation, and, importantly, variables not explicitly set by the action are assumed to keep their value. Also observe that auxiliary variables are not allowed — only variables in $\mathbb{P}$ can occur.

Definition 6. Now we can define the effects $\mathrm{E}(\alpha, P, s)$ of action descriptions, and thus the semantics of all action languages in this article :

1. $\mathrm{E}(\varepsilon, P, s) := \{\langle \emptyset, \emptyset \rangle\}$,
2. $\mathrm{E}(+p, P, s) := \{\langle \{p\}, \emptyset \rangle\}$, and $\mathrm{E}(-p, P, s) := \{\langle \emptyset, \{p\} \rangle\}$,
3. $\mathrm{E}(\varphi \rhd \alpha, P, s) := \begin{cases} \mathrm{E}(\alpha, P, s) & \text{if } s \models \varphi, \\ \{\langle \emptyset, \emptyset \rangle\} & \text{otherwise,} \end{cases}$
4. $\mathrm{E}(\alpha \cup \beta, P, s) := \mathrm{E}(\alpha, P, s) \cup \mathrm{E}(\beta, P, s)$,
5. $\mathrm{E}(\alpha \sqcap \beta, P, s) := \{\langle Q_\alpha^+ \cup Q_\beta^+, Q_\alpha^- \cup Q_\beta^- \rangle \mid \langle Q_\alpha^+, Q_\alpha^- \rangle \in \mathrm{E}(\alpha, P, s), \langle Q_\beta^+, Q_\beta^- \rangle \in \mathrm{E}(\beta, P, s), Q_\alpha^+ \cap Q_\beta^- = Q_\alpha^- \cap Q_\beta^+ = \emptyset\}$,
6. $\mathrm{E}(\alpha \,;\, \beta, P, s) := \{\langle Q_\beta^+ \cup (Q_\alpha^+ \setminus Q_\beta^-), Q_\beta^- \cup (Q_\alpha^- \setminus Q_\beta^+) \rangle \mid \langle Q_\alpha^+, Q_\alpha^- \rangle \in \mathrm{E}(\alpha, P, s), t := (s \cup Q_\alpha^+) \setminus Q_\alpha^-, \langle Q_\beta^+, Q_\beta^- \rangle \in \mathrm{E}(\beta, P, t)\}$,
7. $\mathrm{E}(\neg_{\min} \alpha, P, s) := \{\langle s' \setminus s, s \setminus s' \rangle \mid s' \notin \alpha(s)\}$

Item 5 says that the effects of $\alpha \sqcap \beta$ in s are all *possible* combinations of effects of α and β in s. As an example, for $\alpha := (+p_1 \cup (-p_2 \sqcap +p_3)) \sqcap (-p_2 \cup +p_2)$, $P := \{p_1, p_2, p_3\}$, and any s, we have $\mathrm{E}(\alpha, P, s) = \{\langle \{p_1\}, \{p_2\} \rangle, \langle \{p_1, p_2\}, \emptyset \rangle, \langle \{p_3\}, \{p_2\} \rangle\}$, since in the last combination, $-p_2 \sqcap +p_3$ and $+p_2$ are not consistent.

Item 6 says that if a variable is assigned by both α and β, then it appears in the effects of $\alpha \,;\, \beta$ with the same polarity as in the effects of β. For example, for $\alpha := +p_1 \sqcap +p_2$ and $\beta := -p_1$, the only effect of $\alpha \,;\, \beta$ in the state $s := \emptyset$ is $\langle \{p_2\}, \{p_1\} \rangle$, because the $-p_1$ in β "beats" the $+p_1$ in α.

And item 7 says that $\neg_{\min} \alpha$ defines the transition from s to exactly all the states which were not accessible via α, with the witnessing effect being the one that assigns only the variables which change their value during this transition. For example, if $P := \{p_1, p_2, p_3\}$, $s := \{p_1\}$ and $\alpha := +p_2 \cup (-p_1 \sqcap +p_3)$, then $(\neg_{\min} \alpha)(s) = \{\emptyset, \{p_1\}, \{p_2\}, \{p_1, p_3\}, \{p_2, p_3\}, \{p_1, p_2, p_3\}\}$, and the effect of $\neg_{\min} \alpha$ witnessing the transition from $\{p_1\}$ to $\{p_1, p_3\}$ is $\langle \{p_3\}, \emptyset \rangle$.

We emphasize that explicit effects (of subactions) matter only for $\sqcap$, in the sense that for other constructs, the transition relation $\|\alpha\|$ of α only depends on the transition relation of its subactions.

Note that the expression $+p \sqcap -p$ (for an arbitrary $p \in \mathbb{P}$) defines an action with no successor (which can be interpreted as an execution failing). We use $\bot$ as a shorthand for it (hence $I(\bot, P)(s) = \emptyset$ for all $P \subset \mathbb{P}, s \subseteq P$). We also highlight that $\rhd$ is not necessarily expressing a precondition. If φ is not satisfied in a state s then $\varphi \rhd \alpha$ simply modifies no variable in s. If we want to express that φ is a precondition for α being applicable we need to write $(\varphi \rhd \alpha) \sqcap (\neg \varphi \rhd \bot)$.

Being given the grammar from Definition 5 and the interpretation from Definition 6 we can now define our action languages. In the following definitions p ranges over $\mathbb{P}$ and φ over formulas in NNF over $\mathbb{P}$.

Definition 7 (**NPDDL**)**.** An **NPDDL** *action description* is an expression α generated by the grammar

$$\alpha ::= \varepsilon \mid +p \mid -p \mid \varphi \rhd \alpha \mid (\alpha \cup \alpha) \mid \alpha \sqcap \alpha$$

NPDDL can be seen as an abstract version of grounded PDDL extended to nondeterminism (in the sense of [4]).

1. Note that this is in contrast to the usual semantics of STRIPS, where addition would override deletion, thus "forgetting" $-p_2$.

The next language can be seen as a nondeterministic extension of STRIPS [6] with arbitrary boolean NNF conditions.

Definition 8 (conditional STRIPS)**.** An **NSTRIPS** *action description* is an **NPDDL** expression of the form

$$\prod_{i=0}^{n}\Big(\varphi_i \rhd \big((\ell_i^{1,1} \sqcap \ldots \sqcap \ell_i^{1,j_1}) \cup \ldots \cup (\ell_i^{k_i,1} \sqcap \ldots \sqcap \ell_i^{k_i,j_{k_i}})\big)\Big)$$

where each $\ell_i^{k,j}$ is either ε, $+p$ or $-p$ for some $p \in \mathbb{P}$. In words, an **NSTRIPS** action description specifies a set of conditions so that, when the action is applied in a state s, for each condition satisfied by s exactly one of the corresponding effects occurs.

The last two languages can be seen as extensions of **NPDDL** via the sequence or the negation operator.

Definition 9 (**NPDDL**$_{\text{seq}}$, **NPDDL**$_{\text{not}}$)**.** An **NPDDL**$_{\text{seq}}$ action description α is generated by the grammar

$$\alpha \ ::= \ \varepsilon \mid +p \mid -p \mid \varphi \rhd \alpha \mid (\alpha \cup \alpha) \mid \alpha \sqcap \alpha \mid \alpha\,;\alpha$$

An **NPDDL**$_{\text{not}}$ action description α is generated by the grammar

$$\alpha \ ::= \ \varepsilon \mid +p \mid -p \mid \varphi \rhd \alpha \mid (\alpha \cup \alpha) \mid \alpha \sqcap \alpha \mid \neg_{\min}\alpha$$

Obviously **NSTRIPS** is a sublanguage of **NPDDL** in the sense of Definition 2, and **NPDDL** is a sublanguage of **NPDDL**$_{\text{not}}$ and **NPDDL**$_{\text{seq}}$. **NSTRIPS** is complete, since any action a can at least be represented by one condition for each state s (satisfied only by s), associated to either (1) a choice ($\cup$) between some "conjunctions" of atoms, one conjunction per a-successor s' of s (setting all variables as in s'), or (2) to the degenerate choice of conjunctions $\bot$, when $a(s)$ is empty. Therefore all languages in this article are complete.

While the choice $\cup$ is a natural construct to allow for expressing nondeterminism, and $\pm p$, $\sqcap$ and $\rhd$ are abstractions of the natural properties of PDDL and STRIPS, the sequence and the negation operators require an illustration of their possible use.

Example 10. *The sequence operator is particularly useful for describing actions featuring an intermediate nondeterministic process whose outcome influences the final result. For example, the* **NPDDL**$_{\text{seq}}$ *expression*

$$\begin{aligned}
&+p_{\text{even}}\,;\\
&\big(+p_1 \sqcap (p_{\text{even}} \rhd -p_{\text{even}}) \sqcap (\neg p_{\text{even}} \rhd +p_{\text{even}})\big) \ \cup \ -p_1\,;\\
&\ldots\\
&\big(+p_n \sqcap (p_{\text{even}} \rhd -p_{\text{even}}) \sqcap (\neg p_{\text{even}} \rhd +p_{\text{even}})\big) \ \cup \ -p_n\,;\\
&\neg p_{\text{even}} \rhd \bot
\end{aligned}$$

describes an action which maps any state to the set of all states with an even number of p_i's, and p_{even}, set to true. An action description without the sequence connective would need to enumerate all possible combinations directly, resulting in an exponential tree.

Example 11. *Imagine a self-driving car on a road where there might be an obstacle. Then a possible scenario could be a collision, with the following* **NPDDL** *description over the state variables* $P = \{\text{high_speed}, \text{obstacle}, \text{tire_flat}, \text{tank_leaking}, \text{bumper_dented}, \text{door_blocked}, \text{traffic_barrier}, \text{upside_down}\}$ *:*

$$\beta := \text{obstacle} \rhd \left(\begin{array}{l}
\big((+\text{tire_flat} \cup \varepsilon) \sqcap (-\text{high_speed} \cup \varepsilon)\big)\\
\sqcap\Big(\text{high_speed} \rhd \left(\begin{array}{l}(+\text{tank_leaking} \cup \varepsilon)\\ \sqcap \text{bumper_dented}\\ \sqcap \text{door_blocked}\end{array}\right)\Big)\\
\sqcap\big(\neg\text{traffic_barrier} \rhd +\text{upside_down} \cup \varepsilon\big)
\end{array}\right)$$

This "action" describes the effect of the collision, hence $\neg_{\min}\beta$ describes that the effects of the collision do not *occur. So, if α otherwise describes the effects of one of the actions of the car, action $\alpha \wedge \neg_{\min}\beta$ (shorthand for $\neg_{\min}(\neg_{\min}\alpha \vee \beta)$) describes the same action but taking into account the fact that the car has a built-in collision avoidance system.*

Representations Since we are interested in the succinctness of languages, it is crucial to define the *size* of action descriptions. For this we consider two variants of each language. The first variant corresponds to a representation of the expressions α in the language by their syntactic *tree* (including the Boolean formulas occuring in the expression, if any). The second variant corresponds to the representation of these expressions α by the directed acyclic graph, or *circuit*, obtained from the syntactic tree of α by iteratively identifying the roots of two isomorphic subexpressions to each other, until no more reduction is possible. Clearly, for all expressions α, the circuit associated to α in this manner is unique, and it can be computed in polynomial time from the tree or from a nonreduced circuit.

We write **T-NPDDL**, **T-NPDDL**$_{\text{seq}}$ and **T-NPDDL**$_{\text{not}}$ for those languages with expressions represented as trees, and **C-NPDDL**, **C-NPDDL**$_{\text{seq}}$ and **C-NPDDL**$_{\text{not}}$ for those languages with expressions represented as reduced circuits. Since **NSTRIPS** is flat (the depth of the underlying graph is bounded), there is no difference between the circuit and the tree versions up to polynomial-time transformations, except for the representation of conditions φ; since, as it turns out, the representation of conditions does not affect the complexity results in this paper, we only write **NSTRIPS** without specifying the representation.

4 Complexity of Queries

We now turn to studying the complexity of *queries* on action descriptions. We concentrate on three natural queries

for planning, corresponding to checking the existence of a transition, deciding applicability of an action, and deciding whether a (sequential) plan reaches a goal.

Definition 12 (queries)**.** Let $\langle L, I\rangle$ be a fixed action language, α, $\alpha_1, \ldots, \alpha_k$ be action descriptions in L, $P \subset \mathbb{P}$ be a set of variables such that $I(\alpha, P)$ and $I(\alpha_1, P), \ldots, I(\alpha_k, P)$ are defined, and s, s' be P-states. We consider the following decision problems.

— Is-Succ : given α, P, s, s', decide whether s' is an α-successor of s.

— Is-Applic : given α, P, s, decide whether $\alpha(s)$ is non-empty.

— Entails : given $\alpha_1, \ldots, \alpha_k$, P, s, and an **NNF** formula φ over P, decide whether $s' \models \varphi$ for all $s' \in (\alpha_1\,; \ldots; \alpha_k)(s)$.

Note that if $\alpha_1\,; \ldots; \alpha_k$ has no successors for s, then it automatically entails any formula φ, especially any unsatisfiable formula, say $\varphi := p \wedge \neg p$.

Is-Succ is the most basic query which corresponds to model checking for Boolean formulas. We will see later that it is usually enough to know the complexity of Is-Succ to conclude about the other queries.

The first result is a technical one and will be used later in succinctness proofs. It can be easily deduced from the fact that for a bounded number of variables the number of possible effects is bounded, too.

Lemma 13. *Let $k \in \mathbb{N}$ be fixed, and assume $|P| \leq k$ (i.e. the size of the scope of the described actions is fixed). Then Is-Succ can be solved in polynomial time for all languages which we consider.*

The next two statements were proven in [20].

Proposition 14. *Is-Succ is* NP*-complete for* **NSTRIPS**, **T-NPDDL**, **C-NPDDL** *and* **T-NPDDL**$_{\text{seq}}$.

Proposition 15. *Is-Succ is* PSPACE*-complete for* **C-NPDDL**$_{\text{seq}}$.

In order to prove the theorem about Is-Succ in **NPDDL**$_{\text{not}}$, we require a notation and a lemma.

Notation 16. Let $n \in \mathbb{N}$, and let $X_n := \{x_1, \ldots, x_n\}$ be a set of variables. Observe that there are a cubic number N_n of clauses of length 3 over X_n (any choice of 3 variables with a polarity for each). We fix an arbitrary enumeration $\gamma_1, \gamma_2, \ldots, \gamma_{N_n}$ of all these clauses, and we define $P_n \subset \mathbb{P}$ to be the set of state variables $\{p_1, p_2, \ldots, p_{N_n}\}$. Then to any 3-CNF formula φ we associate the P_n-state $s(\varphi) = \{p_i \mid i \in \{1, \ldots, N_n\}, \gamma_i \in \varphi\}$, and dually, to any P_n-state s, we associate the 3-CNF formula over X_n, written $\varphi(s)$, which contains exactly those clauses γ_i for which $p_i \in s$ holds.

2. By $s' \in (\alpha_1\,; \ldots; \alpha_k)(s)$ we mean that there are $s_0, s_1, \ldots, s_k$ with $s = s_0$, $s_i \in \alpha_i(s_{i-1})$ for $i = 1, \ldots, k$, and $s_k = s'$.

Example 17. *Let $n := 2$, and consider an enumeration of all 3-clauses over $X_2 := \{x_1, x_2\}$ which starts with $\gamma_1 := (x_1 \vee x_1 \vee x_2), \gamma_2 := (x_1 \vee x_1 \vee \neg x_2), \gamma_3 := (x_1 \vee \neg x_1 \vee x_2), \gamma_4 := (x_1 \vee \neg x_1 \vee \neg x_2), \gamma_5 := (\neg x_1 \vee \neg x_1 \vee x_2), \ldots$.*

Then the 3-CNF *$\varphi := (x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_1 \vee x_2)$ is encoded by the state $s(\varphi) = \{p_1, p_5\}$, and $\psi = (x_1 \vee \neg x_1 \vee \neg x_2)$ is encoded by by $s(\psi) = \{p_4\}$.*

The statement of the lemma can be shown by an easy induction.

Lemma 18. *Let φ, ψ be* 3-CNF *formulas, let $s(\varphi)$ and $s(\psi)$ be as in Notation 16, and define the QBF $\Psi^n_\varphi := \exists x_1 \colon \neg(\exists x_2 \colon \neg(\ldots \neg(\exists x_n \colon \varphi) \ldots))$. Then the following hold :*

1. *If n is odd : if Ψ^n_φ is true and $s(\psi) \subseteq s(\varphi)$ then Ψ^n_ψ is true*
   dually : if Ψ^n_φ is false and $s(\psi) \supseteq s(\varphi)$ then Ψ^n_ψ is false
2. *If n is even : if Ψ^n_φ is true and $s(\psi) \supseteq s(\varphi)$ then Ψ^n_ψ is true*
   dually : if Ψ^n_φ is false and $s(\psi) \subseteq s(\varphi)$ then Ψ^n_ψ is false

Notation 19. Using Notation 16, for all $n \in \mathbb{N}$ we define the **T-NPDDL** action description α^{sat}_n :

$$\alpha^{\text{sat}}_n := \bigsqcap_{x \in X_n} \left(\Big(\bigsqcap_{\gamma_i \colon x \in \gamma_i} (+p_i \cup \varepsilon) \Big) \bigcup \Big(\bigsqcap_{\gamma_i \colon \neg x \in \gamma_i} (+p_i \cup \varepsilon) \Big) \right)$$

Intuitively, α^{sat}_n chooses an assignment (true or false) to each variable in X_n (outermost $\cup$). Whenever it chooses an assignment for x, for each possible clause which is satisfied by this assignment (innermost $\sqcap$), it chooses whether to include this clause into the result, or not (innermost $\cup$). Hence it builds a formula which is satisfied at least by its choices, and clearly, any satisfiable 3-CNF formula can be built in this manner.

Lemma 20. *Let $n \in \mathbb{N}$, and let φ be a* 3-CNF *formula over X_n. Then φ is satisfiable if and only if $s(\varphi)$ is an α^{sat}_n-successor of the state $\emptyset$.*

Proposition 21. *Is-Succ is* PSPACE*-complete for* **T-NPDDL**$_{\text{not}}$ *and* **C-NPDDL**$_{\text{not}}$.

Proof. For membership observe that every execution of the circuit can be simulated in polynomial space and we can do it until we find a witness for the transition (s, s'), if any.

Not let the sets P_n, X_n and clauses γ_i be as in Notation 16. For hardness, we show that a quantified Boolean formula $\Phi := \exists x_1 \colon \neg(\exists x_2 \colon \neg(\ldots(\exists x_n \colon \varphi)))$, with φ a 3-CNF, is true if and only if $s(\varphi) \in \alpha^n_1(\emptyset)$ with

1. $\alpha^n_n := \chi^n_n$
2. $\alpha^n_k := \chi^n_k \sqcap \Big(\neg_{\min} \neg_{\min} (\rho^n_k \sqcap (\neg_{\min} \alpha^n_{k+1}))\Big)$ with
   — $\chi^n_k := \Big(\bigsqcap_{\gamma_i \colon x_k \in \gamma_i} (+p_i \cup \varepsilon) \Big) \cup \Big(\bigsqcap_{\gamma_i \colon \neg x_k \in \gamma_i} (+p_i \cup \varepsilon) \Big)$

— $\rho_k^n := (\bigsqcap_{\gamma_i\colon x_k \in \gamma_i} -p_i) \cup (\bigsqcap_{\gamma_i\colon \neg x_k \in \gamma_i} -p_i)$

In the following we give an intuition for what these (sub)actions do in $s := \emptyset$. ρ_k^n chooses an assignment to x_k and explicitly prohibits to add any of the clauses that contain a literal which is true under this assignment (i.e. is satisfied by this assignment). χ_k^n chooses an assignment to x_k and then produces nondeterministically any possible set of clauses which are satisfied by this assignment. Recall that for all actions δ we have $\|\delta\| = \|\neg_{\min}\neg_{\min}\delta\|$; the double negation here serves to make sure that transitions are witnessed by minimal effects.

Throughout the proof we use the following notation : $\nu_{=k}$ refers to an assignment to x_k and $\nu_{\leq \ell}$ to an assignment to $x_1, \ldots, x_\ell$. If $\nu_{\leq k-1}$ and $\nu_{=k}$ are clear from the context, then $\nu_{\leq k}$ refers to the induced joint assignment to $x_1, \ldots, x_k$. Finally, for an assignment ν we denote by $t(\nu)$ the encodings (via Notation 16) of 3-clauses which are satisfied by ν.

We denote by $\Phi_{\nu_{\leq k}}$ the formula $\exists x_{k+1}\colon \neg(\exists x_{k+2}\colon \neg(\ldots \neg(\exists x_n\colon \varphi_{|\nu_{\leq k}})))$. Now we prove the claim that for every $0 \leq k \leq n-1$ we have :

$$\text{for all } \nu_{\leq k}\colon [\Phi_{\nu_{\leq k}} \text{ true } \iff s(\varphi) \setminus t(\nu_{\leq k}) \in \alpha_{k+1}^n(\emptyset)]$$

After having shown the claim for $k = 0$ we are done with showing the main claim because $\Phi_{\nu_{\leq 0}} = \Phi$and $t(\nu_{\leq 0}) = \emptyset$ (since $\nu_{\leq 0}$ is the empty assignement). .

We start with $k := n-1$. In this case, $\Phi_{\nu_{\leq k}}$ is just an existentially quantified 3-CNF which is true (being already conditioned by an assignment $\nu_{\leq k}$ to $x_1, \ldots, x_{n-1}$) if and only if the clauses which have not yet been satisfied by $\nu_{\leq k}$ (i.e. $s(\varphi) \setminus t(\nu_{\leq k})$) can be satisfied by an assignment to x_n, which is equivalent to $s(\varphi) \setminus t(\nu_{\leq k})$ being an α_n^n-successor of $\emptyset$ (the statement is analogous to the statement of Lemma 20). Now we assume that the claim has already been shown for k, and we want to show it for $k-1$: we consider an assignment $\nu_{\leq k-1}$ to $x_1, \ldots, x_{k-1}$, and we prove that $\Phi_{\nu_{\leq k-1}}$ is true if and only if $s(\varphi) \setminus t(\nu_{\leq k-1}) \in \alpha_k^n(\emptyset) = \left(\chi_k^n \sqcap \left(\neg_{\min}\neg_{\min}(\rho_k^n \sqcap (\neg_{\min}\alpha_{k+1}^n))\right)\right)(\emptyset)$.

" $\Longrightarrow$:" Suppose that $\Phi_{\nu_{\leq k-1}}$ is true. Then there exists an assignment $\nu_{=k}$ to x_k such that $\Phi_{\nu_{\leq k}}$ is false. By the induction hypothesis this means that $T := s(\varphi) \setminus t(\nu_{\leq k}) \notin \alpha_{k+1}^n(\emptyset)$ and thus $T \in \neg_{\min}\alpha_{k+1}^n(\emptyset)$. Since $t(\nu_{=k}) \subseteq t(\nu_{\leq k})$, we have $t(\nu_{=k}) \cap T = \emptyset$, therefore the execution of ρ_k^n (re)assigning variables in $t(\nu_{=k})$ to $\bot$ is consistent with the effect of $\neg_{\min}\alpha_{k+1}^n$ leading to T. Hence $T \in (\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$, and adding the double negation we have $T \in \neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$ with the guarantee that the effect e_T witnessing this transition is minimal – and thus, since the transition starts from $\emptyset$, that e_T has no negative effects.

Now observe that, by construction of χ_k^n, any subset U of $t(\nu_{=k})$ is a successor of $\emptyset$ via χ_k^n and the transition is witnessed by a purely positive effect e_U. This applies to the subset $U := (s(\varphi) \setminus t(\nu_{\leq k-1})) \cap t(\nu_{=k})$. All in all, combining e_T and e_U (which are necessarily consistent since both contain only positive effects), we get that $U \cup T \in \chi_k^n \sqcap \left(\neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)\right)(\emptyset) = \alpha_k^n(\emptyset)$. Given that $t(\nu_{\leq k})) = t(\nu_{\leq k-1}) \cup t(\nu_{=k})$, it is not hard to see that $U \cup T = s(\varphi) \setminus t(\nu_{\leq k-1})$, which proves the claim.

" $\Longleftarrow$:" Now assume that $s(\varphi) \setminus t(\nu_{\leq k-1}) \in \alpha_k(\emptyset) = \chi_k^n \sqcap \left(\neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)\right)(\emptyset)$.

We first consider the case when $n-k$ is odd. From the assumption we get that there exist U, T such that $s(\varphi) \setminus t(\nu_{\leq k-1}) = U \cup T$ with $U \in \chi_k^n(\emptyset)$ and $T \in \neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$. So $T \subseteq s(\varphi) \setminus t(\nu_{\leq k-1})$ so there exists a V such that $(s(\varphi) \setminus t(\nu_{\leq k-1})) \setminus V \in \neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$, and therefore $(s(\varphi) \setminus t(\nu_{\leq k-1})) \setminus V \in (\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$. Let $\langle Q^+, Q^- \rangle$ be the effect of $\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n$ witnessing this transition. By definition of ρ_k^n (which is responsible for Q^- since $\neg_{\min}\alpha_{k+1}^n$ has only positive effects in $\emptyset$) there exists an assignment $\nu_{=k}$ to x_k such that $t(\nu_{=k}) = Q^-$. Since $Q^+ \cap Q^- = \emptyset$, it holds that $Q^+ \subseteq (s(\varphi) \setminus t(\nu_{\leq k-1})) \setminus t(\nu_{=k}) = s(\varphi) \setminus t(\nu_{\leq k})$. It also holds that $Q^+ \in \neg_{\min}\alpha_{k+1}^n(\emptyset)$ because ρ_k^n has only negative effects. Thus Q^+ is a set of clauses which encodes a 3-CNF formula ψ with $s(\psi) \subseteq s(\varphi) \setminus t(\nu_{\leq k}) \subseteq s(\varphi)$ and by the inductive assumption $\exists x_{k+1}\colon \neg(\exists x_{k+2}\colon \neg \ldots \neg(\exists x_n\colon \psi_{|\nu_{\leq k}}) \ldots)$ is false. With the first statement of Lemma 18 (recall that $n-k$ and hence the number of quantifiers was odd) it follows that $\exists x_{k+1}\colon \neg(\exists x_{k+2}\colon \neg \ldots \neg(\exists x_n\colon \varphi_{|\nu_{\leq k}}) \ldots)$ is false, too, so $\Phi_{\nu_{\leq k-1}}$ is true.

We now consider the case that $n-k$ is even. For the effect $\langle Q^+, Q^- \rangle$ witnessing the transition from $\emptyset$ to $s(\varphi) \setminus t(\nu_{\leq k-1})$ there must be an effect $\langle U^+, \emptyset \rangle$ of χ_k^n with $U^+ \subseteq Q^+$ and by construction of χ_k^n there exists an assignment $\nu_{=k}$ with $U^+ \subseteq t(\nu_{=k})$. Therefore there must exist an effect $\langle T, \emptyset \rangle$ of $\neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)$ in $\emptyset$ with $T \supseteq Q^+ \setminus U^+ \supseteq (s(\varphi) \setminus t(\nu_{\leq k-1})) \setminus t(\nu_{=k}) = s(\varphi) \setminus t(\nu_{\leq k})$. It follows that $T \in \neg_{\min}\neg_{\min}(\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$ and therefore $T \in (\rho_k^n \sqcap \neg_{\min}\alpha_{k+1}^n)(\emptyset)$ and thus $T \in \neg_{\min}\alpha_{k+1}^n(\emptyset)$ since ρ_k^n does not have positive effects. T encodes a 3-CNF formula ψ such that $s(\psi) \setminus t(\nu_{\leq k}) \supseteq s(\varphi) \setminus t(\nu_{\leq k})$ and by the inductive assumption $\exists x_{k+1}\colon \neg(\exists x_{k+2}\colon \neg \ldots \neg(\exists x_n\colon \psi_{|\nu_{\leq k}}) \ldots)$ is false. Since $n-k$ is even we apply the second statement of Lemma 18 and conclude that $\exists x_{k+1}\colon \neg(\exists x_{k+2}\colon \neg \ldots \neg(\exists x_n\colon \varphi_{|\nu_{\leq k}}) \ldots)$ is false, too. Therefore $\Phi_{\nu_{\leq k-1}}$ is true. □

Lemma 22. *Is-Succ is polynomial-time reducible to Is-Applic for* **T-NPDDL**$_{\text{seq}}$, **C-NPDDL**$_{\text{seq}}$, **T-NPDDL**$_{\text{not}}$ *and* **C-NPDDL**$_{\text{not}}$.

Proof. For a state s let ψ_s denote the formula $(\bigwedge_{p \in s} p) \wedge (\bigwedge_{p \notin s} \neg p)$. Let α be a (tree- or circuit) **NPDDL**$_{\text{seq}}$ action. For all states s, s' it is easy to see that we have $s' \in \alpha(s)$ if and only if the action $(\alpha\,; (\neg\psi_{s'} \rhd \bot))$ is applicable in s.

Now let β be a (tree- or circuit) **NPDDL**$_{\text{not}}$ action description and s, s' be P-states. We define $\rho_{s'} := \left((\bigsqcap_{p \in s'} +p) \sqcap (\bigsqcap_{p \notin s'} -p)\right)$, the deterministic action leading from all states

to s'. Then we have $s' \in \beta(s)$ if and only if the action $\gamma := \neg_{\min}(\neg_{\min}\beta \cup \neg_{\min}\rho_{s'})$ is applicable in s because it is easy to see that γ satisfies $\gamma(s) = \beta(s) \cap \rho_{s'}(s)$. □

Corollary 23. *Is-Applic is* NP*-complete for* **NSTRIPS**, **T-NPDDL**, **C-NPDDL**, *and* **T-NPDDL**$_{\text{seq}}$, *and it is* PSPACE*-complete for* **C-NPDDL**$_{\text{seq}}$, **T-NPDDL**$_{\text{not}}$ *and* **C-NPDDL**$_{\text{not}}$.

Proof. For **NPDDL**$_{\text{seq}}$ and **NPDDL**$_{\text{not}}$ hardness follows from Lemma 22 together with the previous hardness results about Is-Succ (Propositions 15 and 21). For **NSTRIPS** the result has been proven in [20]. Membership is clear because applicability can be witnessed by giving a successor and verifying successorship. □

Lemma 24. *The complement of Is-Applic is polynomial-time reducible to Entails for all languages.*

Proof. An action described by α is non-applicable in s (i.e. it has no successors in s) if and only if all α-successors s' of s satisfy $p \wedge \neg p$ for an arbitrary p, i.e. if and only if α entails $\varphi := p \wedge \neg p$ in s. □

Corollary 25. *Entails is* coNP*-complete for* **NSTRIPS**, **T-NPDDL**, **C-NPDDL**, *and* **T-NPDDL**$_{\text{seq}}$, *and it is* PSPACE*-complete for* **C-NPDDL**$_{\text{seq}}$, **T-NPDDL**$_{\text{not}}$, *and* **C-NPDDL**$_{\text{not}}$.

Proof. We first recall that PSPACE = coPSPACE. Then hardness follows for all the languages from the reduction of non-applicability to entailment together with the hardness results about Is-Applic (Lemma 22). Membership has been shown in [20]. □

5 Succinctness

We now give negative results about the existence of polynomial-*size* translations, hence about the relative *succinctness* of languages [5]. Recall that all the languages which we study are complete, thus our definition is a bit simpler than the usual one.

Definition 26 (succinctness)**.** A complete action language L_1 is at least as succinct as a complete action language L_2 if there exists a polynomial-size translation from L_2 into L_1.

Clearly, if L_2 is a sublanguage of L_1 then L_1 is at least as succinct as L_2.

Our succinctness *separation* results rely on assumptions about the *nonuniform* complexity classes P/poly and NP/poly. Recall that P/poly is the class of all decision problems such that for all $n \in \mathbb{N}$, there is a polynomial-time algorithm which decides the problem for all inputs of size n, NP/poly is defined analogously. The assumptions NP $\not\subseteq$ P/poly, coNP $\not\subseteq$ NP/poly and PSPACE $\not\subseteq$ NP/poly which we use are standard ones.

Given two disjoint sets of variables P, Q a $(P \cup Q)$-action a, and an assignment $t \subseteq Q$ to the variables in Q, we define the *t-conditioning* of a to be the P-action $a_{|t}$ satisfying $\forall s \subseteq P\colon a_{|t}(s) = \{s' \mid (s' \cup t) \in \alpha(s \cup t)\}$. The next lemma will be used to partially separate **NPDDL**$_{\text{seq}}$ from **NPDDL**$_{\text{not}}$.

Lemma 27. *Let P and Q be disjoint sets of variables, α be a* **T-NPDDL**$_{\text{not}}$ *(resp.* **C-NPDDL**$_{\text{not}}$*) expression for a $(P \cup Q)$-action, and $t \subseteq Q$ be an assignment to the variables in Q. Then we can compute a* **T-NPDDL**$_{\text{not}}$ *(resp.* **C-NPDDL**$_{\text{not}}$*) expression $f(\alpha)$ for the t-conditioning of α in time polynomial in $|\alpha|$.*

Proof. Simply replace all leaves of the form $+q$ with $q \in Q \cap t$ by ε, all leaves of the form $+q$ with $q \in Q \setminus t$ by $\perp$ (failure), and dually for $-q$, and for all subexpressions $\varphi \rhd \beta$, simplify φ by the assignment t to Q. □

Proposition 28. *If* NP $\not\subseteq$ P/poly *then there is no polynomial-size translation from* **T-NPDDL**$_{\text{seq}}$ *to* **T-NPDDL**$_{\text{not}}$, *nor from* **C-NPDDL**$_{\text{seq}}$ *to* **C-NPDDL**$_{\text{not}}$.

Proof. Let $n \in \mathbb{N}$, and let φ be a 3-CNF formula over a set of variables $X_n := \{x_1, \ldots, x_n\}$. Recall from Notation 16 that we can encode φ over a set $P_n \subseteq \mathbb{P}$. Finally, let $p_{\text{sat}} \in \mathbb{P}$ be a fresh variable. We define γ_n^{sat} to be the following action :

$$\prod_{i=1}^{n}(+x_i \cup -x_i); (\psi_n \rhd +p_{\text{sat}}) \sqcap (\neg\psi_n \rhd -p_{\text{sat}}); \prod_{i=1}^{n} -x_i$$

where ψ_n is the NNF $\bigwedge_{i=1}^{N_n}(\neg p_i \vee \bigvee_{\ell \in \gamma_i} \ell)$, which is satisfied if and only if each clause γ_i which is in φ (as witnessed by p_i being true) is also satisfied. In words, γ_n^{sat} guesses an assignment to $\mathrm{V}(\varphi)$, then sets p_{sat} according to whether φ is satisfied by this assignment, and finally resets all guessed variables to false. Note that γ_n^{sat} depends on n but not on φ, and that γ_n^{sat} is polynomial in n.

Clearly, $s(\varphi) \cup \{p_{\text{sat}}\}$ is a γ_n^{sat}-successor of $s(\varphi)$ if and only if φ is satisfiable. Hence the following decision problem is NP-hard :

— *Input :* a 3-CNF formula φ

— *Question :* is $s(\varphi) \cup \{p_{\text{sat}}\}$ a γ_n^{sat}-successor of $s(\varphi)$?

Now assume that there is a polynomial-size translation f from **T-NPDDL**$_{\text{seq}}$ to **T-NPDDL**$_{\text{not}}$, and for all $n \in \mathbb{N}$ let $\delta_n^{\text{sat}} := f(\gamma_n^{\text{sat}})$. Let φ be a 3-CNF formula over n variables. Since δ_n^{sat} is in **NPDDL**$_{\text{not}}$, we can apply Lemma 27 with $Q := P_n \cup \{x_1 \ldots, x_n\}$ and $t := s(\varphi)$, to get an expression in which the only occuring variable is p_{sat}, and $\{p_{\text{sat}}\}$ is a successor of $\emptyset$ if and only if $s(\varphi) \cup \{p_{\text{sat}}\}$ is a δ_n^{sat}-sucessor of $s(\varphi)$, that is, if and only if φ is satisfiable. Since this expression has only one variable, with Lemma 13 we see that successorship can be decided in polynomial time, implying NP $\subseteq$ P/poly. The proof is exactly the same for circuits. □

Proposition 29. *If* PSPACE $\not\subseteq$ NP/poly *then there is no polynomial-size translation neither from* **C-NPDDL**$_{\text{not}}$ *into* **C-NPDDL** *nor from* **T-NPDDL**$_{\text{not}}$ *into* **T-NPDDL**.

Proof. If **NPDDL**$_{\text{not}}$ was translatable into **NPDDL** with only a polynomial increase in size, we could translate α_1^n from Proposition 21 into a polynomial-sized equivalent action $f(\alpha_1^n)$ in **NPDDL**. Since α_1^n does not depend on a formula but only on the number of variables, it would follow that deciding the validity of a quantified Boolean formula of the form $\exists x_1 : \neg(\exists x_2 : \neg(\ldots \neg(\exists x_n : \varphi)))$ with a 3-CNF φ amounts to checking $s(\varphi) \in \alpha_1^n(\emptyset)$, and we would obtain a nonuniform NP-algorithm for a PSPACE-complete problem,implying PSPACE $\subseteq$ NP/poly. □

Proposition 30. *There exists no polynomial-size translation of* **C-NPDDL** *into* **NSTRIPS**.

Proof. The majority function (which returns true if and only if at least half of its arguments are true) can be computed by a boolean circuit ψ of linear size and logarithmic depth [15]. Consider an action a over $P_n = \{p_1, \ldots, p_n\}$ which is applicable only in $s = \emptyset$ and in this state it produces nondeterministically all s' with $|s'| \geq \frac{n}{2}$. Thus it is obviously representable by a polynomial-size circuit NNF action theory ψ'_n, and this can be translated into **C-NPDDL** in polynomial time [20]. Now every **NSTRIPS** representation of a can without loss of generality be assumed to be of the form $(\varphi_n \rhd \alpha_n) \sqcap (\neg\varphi_n \rhd \bot)$ with $\varphi_n = \neg p_1 \wedge \ldots \wedge \neg p_n$ and α_n being an unconditioned **NSTRIPS** expression. By replacing $\sqcap$ by $\wedge$, $\cup$ by $\vee$, $+p$ by p and $-p$ by $\neg p$ we would then obtain a formula over P_n whose models are α_n-successors of $\emptyset$, thus obtaining a boolean circuit of bounded depth of size polynomial in n for the majority function, which contradicts a result from [9]. □

6 Transformations

One of the key aspects of knowledge compilation is the study of transformations that a language supports [5]. In general, we are interested in finding out whether a given transformation is possible in polynomial time, and otherwise, whether it is at least possible without a superpolynomial explosion in size.

Let $\langle L, I\rangle$ be a fixed action language, let $\alpha, \alpha_1, \alpha_2$ be action descriptions in L, $P \subset \mathbb{P}$ be a set of variables such that $I(\alpha, P)$, $I(\alpha_1, P)$, and $I(\alpha_2, P)$ are defined, and let s, s' be P-states. We study the following computational problems.

— Sequence : given $\alpha_1, \alpha_2 \in L$, compute an action description $\beta \in L$ such that for all $s : \beta(s) = \{s' \mid \exists t$ such that $t \in \alpha_1(s)$ and $s' \in \alpha_2(t)\}$.

— Choice : given $\alpha_1, \alpha_2 \in L$, compute an action description $\beta \in L$ such that for all $s : \beta(s) = \alpha_1(s) \cup \alpha_2(s)$.

— Negation : given $\alpha \in L$, compute an action description $\beta \in L$ such that for all $s : \beta(s) = \{s' \mid s' \notin \alpha(s)\}$.

We insist that for each one it is required that the output be in the same language as the input description(s). We also consider the problem of extracting the precondition.

— Ex-Prec : given $\alpha \in L$ and $P \subset \mathbb{P}$, compute an NNF formula φ such that for all $s : \alpha(s) \neq \emptyset \iff s \models \varphi$.

We see it as a transformation because it amounts to computing an action description of the form $\neg\varphi \rhd \bot$ expressing when the action does not satisfy its precondition.

Of course, if a language allows for arbitrary nesting of an operator then it trivially supports a transformation. For example, **NPDDL** allows to express Choice of α_1 and α_2 via $\alpha_1 \cup \alpha_2$. In other cases, determining the complexity of a transformation seems to be not much easier than determining the succinctness of the language enriched with the corresponding operator.

Proposition 31. *The following transformations are polynomial-time, in all cases for both the tree and circuit representations :*

— *Choice for* **NPDDL**, **NPDDL**$_{\text{not}}$ *and* **NPDDL**$_{\text{seq}}$ *;*

— *Sequence for* **NPDDL**$_{\text{seq}}$ *;*

— *Negation for and* **NPDDL**$_{\text{not}}$.

Proposition 32. *If* NP $\not\subseteq$ P/poly *then* **NSTRIPS**, **T-NPDDL**, **C-NPDDL**, **T-NPDDL**$_{\text{not}}$ *and* **C-NPDDL**$_{\text{not}}$ *do not support polynomial-size Sequence.*

Proof. Consider the action γ_n^{sat} from the proof of Proposition 28. Each of its subactions which are connected via the sequence operator (these are $\bigsqcap_{i=1}^n (+x_i \cup -x_i)$, $(\psi_n \rhd +p_{\text{sat}}) \sqcap (\neg\psi_n \rhd -p_{\text{sat}})$ and $\bigsqcap_{i=1}^n -x_i$) is a **NSTRIPS** (and therefore **NPDDL**, **NPDDL**$_{\text{not}}$) action. Thus the proof of Proposition 28 can be reused to show that if NP $\not\subseteq$ P/poly then Sequence cannot be polynomial-size for **NSTRIPS**, **NPDDL**, nor **NPDDL**$_{\text{not}}$. □

Proposition 33. *If* coNP $\not\subseteq$ NP/poly *then Negation is not polynomial-size for* **T-NPDDL**, **C-NPDDL**, *nor* **T-NPDDL**$_{\text{seq}}$.

Proof. Recall for α_n^{sat} from Lemma 20 that $s(\varphi) \in \alpha_n^{\text{sat}}(\emptyset)$ holds if and only if φ is satisfiable. Now suppose that Negation is polynomial-size in **T-NPDDL**, **C-NPDDL**, or **T-NPDDL**$_{\text{seq}}$. Then there exists a polynomial-sized equivalent $f(\alpha_n^{\text{sat}})$ of the negation of α_n^{sat}. Thus $s(\varphi) \in f(\alpha_n^{\text{sat}})$ holds if and only if φ is unsatisfiable, and so there is a nonuniform NP-algorithm for a coNP-complete problem.□

The following results show that it is in general more efficient to represent the precondition of actions implicitly in the description rather than as a separate formula.

Proposition 34. *If* NP $\not\subseteq$ P/poly *then Ex-Prec is not polynomial-size for* **NSTRIPS**, *nor for* **NPDDL**, **NPDDL**$_{\text{not}}$, **NPDDL**$_{\text{seq}}$ *under neither the tree nor the circuit representation.*

Proof. Consider the **NSTRIPS** action over $X_n \cup P_n$ (as in Notation 16) : $\xi_n := \bigsqcap_{p_i} \left(p_i \rhd \left((\bigcup_{x \in \gamma_i} +x) \cup (\bigcup_{\neg x \in \gamma_i} -x)\right)\right)$

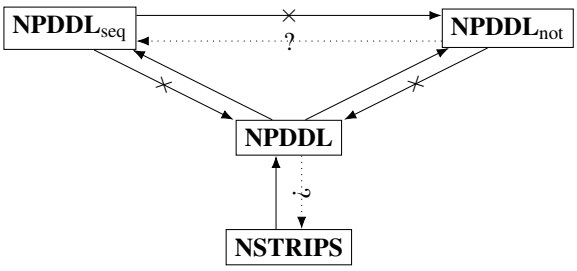


FIGURE 1 – Succinctness results. An arc from L_1 to L_2 denotes that L_1 is a sublanguage of L_2. A crossed out arc from L_1 to L_2 means that there is no polynomial-size translation from L_1 into L_2. These relations hold for both tree and circuit representations. Dotted arcs denote open questions (see text).

and observe that that ξ_n is applicable in $s(\varphi)$ if and only if φ is satisfiable. If there was a polynomial-size NNF representation of the precondition ψ_n of ξ_n, we could check φ for satisfiability by checking whether $s(\varphi) \models \psi_n$, and thus we would obtain a non-uniform polynomial-time algorithm for 3-SAT. To see the claim for the other languages, observe that if EX-PREC was polynomial-size for **NPDDL**, **NPDDL**$_{not}$ or **NPDDL**$_{seq}$, it would be polynomial-size for their sublanguage **NSTRIPS** (since the representation of the output does not depend on the language). □

7 Conclusion

We have studied several languages for describing nondeterministic actions along two criteria : succinctness and complexity of different decision problems and transformations natural to automated planning. We have also considered two representations, by trees and by circuits. Our results are summarized in Table 1 and on Figure 1.

An interesting result is the different complexity of queries for **NPDDL**$_{seq}$ with trees or circuits. While it is intuitively clear that there must be some languages which are strictly less succinct with trees than with circuits, and languages with less tractable queries with circuits than with trees, this gives a concrete example of this phenomenon. Finally, it is interesting that the complexity of the three queries is the same for tree-represented **NPDDL** with or without sequence. Since the language is strictly more succinct with sequence, **T-NPDDL**$_{seq}$ seems to be a strictly more interesting language than **T-NPDDL**.

We leave some problems open for transformations (see Table 1) and succinctness. For succinctness, a dotted arc from L_1 to L_2 on Figure 1 means that the existence of a polynomial-size translation from L_1 to L_2 is open; for all these arcs we conjecture that there is actually none.

It would also be interesting to determine the complexity of other queries and transformations. Queries obviously useful to planning are to count and enumerate successors, generate a successor of a given state uniformly at random (as needed in Monte-Carlo approaches), to determine whether an action is deterministic, and whether all executions of a given action sequence are free of dead-ends. As concerns transformations, the complexity of combining two actions via logical conjunction $\wedge$ does not seem so easy to determine. Another interesting transformation, in particular for regression approaches to planning, is the computation of the "reverse" action $\bar{\alpha}$ of α with $s' \in \bar{\alpha}(s) \Leftrightarrow s \in \alpha(s')$.

Our main perspective is a more systematic study, for languages constructed using combinations of features like the sequence operator, modalities, Kleene star, etc. A very expressive language to consider is **DL-PPA** [10]. Another perspective is to consider languages for stochastic actions, and for actions with observations (and hence queries on *belief states* rather than states). The ultimate goal is to draw clear pictures of what language to choose depending on the queries which are used by, e.g., a planning algorithm or a simulator.

Acknowledgements

This work has been supported by the French National Research Agency (ANR) through project PING/ACK (ANR-18-CE40-0011).

Références

[1] Bäckström, Christer: *Expressive Equivalence of Planning Formalisms*. Artificial Intelligence, 76(1-2) :17–34, 1995.

[2] Bäckström, Christer et Peter Jonsson: *Algorithms and Limits for Compact Plan Representations*. Journal of Artificial Intelligence Research, 44 :141–177, 2012.

[3] Benthem, Johan van, Jan van Eijck, Malvin Gattinger et Kaile Su: *Symbolic Model Checking for Dynamic Epistemic Logic — S5 and Beyond*. Journal of Logic and Computation, 28(2) :367—402, 2018.

[4] Bertoli, Piergiorgio, Alessandro Cimatti, Ugo Dal Lago et Marco Pistore: *Extending PDDL to nondeterminism, limited sensing and iterative conditional plans*. Dans *Proceedings of ICAPS 2003 Workshop on PDDL*, 2003.

[5] Darwiche, Adnan et Pierre Marquis: *A knowledge compilation map*. Journal of Artificial Intelligence Research, 17 :229–264, 2002.

[6] Fikes, Richard E et Nils J Nilsson: *STRIPS : A new approach to the application of theorem proving to problem solving*. Artificial intelligence, 2(3-4) :189–208, 1971.

[7] Geffner, Hector et Blai Bonet: *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

Language	Is-Succ	Is-Applic	Entails	Choice	Sequence	Negation	Ex-Prec
NSTRIPS	NP-c	NP-c	coNP-c	?	○	?	○
T-NPDDL, **C-NPDDL**	NP-c	NP-c	coNP-c	✓	○	○	○
T-NPDDL$_{\text{seq}}$	NP-c	NP-c	coNP-c	✓	✓	○	○
C-NPDDL$_{\text{seq}}$	PSPACE-c	PSPACE-c	PSPACE-c	✓	✓	?	○
T-NPDDL$_{\text{not}}$, **C-NPDDL**$_{\text{not}}$	PSPACE-c	PSPACE-c	PSPACE-c	✓	○	✓	○

Table 1 – Complexity results for queries and transformations. "✓" means that the transformation can be done in time polynomial in the size of the input. "?" means that the question is open. ○ means that under some complexity-theoretic assumption (see formal statements) the size of the result of the transformation is in general not polynomial in the size of the input.

[8] Geffner, Tomas et Hector Geffner: *Compact Policies for Fully Observable Non-Deterministic Planning as SAT*. Dans *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, pages 88–96, 2018.

[9] Hastad, John: *Almost optimal lower bounds for small depth circuits*. Dans *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20, 1986.

[10] Herzig, Andreas, Frédéric Maris et Julien Vianey: *Dynamic logic of parallel propositional assignments and its applications to planning*. Dans *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 5576–5582, 2019.

[11] Hoey, Jesse, Robert St Aubin et Craig Boutilier: *SPUDD : stochastic planning using decision diagrams*. Dans *Proceedings of Uncertainty in Artificial Intelligence (UAI). Stockholm, Sweden. Page (s)*, 1999.

[12] Lipovetzky, Nir et Hector Geffner: *Width and Serialization of Classical Planning Problems*. Dans *ECAI*, tome 242 de *Frontiers in Artificial Intelligence and Applications*, pages 540–545. IOS Press, 2012.

[13] McDermott, Drew: *PDDL–the planning domain definition language*. Rapport technique CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[14] Muise, Christian J., Sheila A. McIlraith et Vaishak Belle: *Non-Deterministic Planning With Conditional Effects*. Dans *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, pages 370—374. AAAI Press, 2014.

[15] Muller, David E. et Franco P. Preparata: *Bounds to complexities of networks for sorting and for switching*. Journal of the ACM (JACM), 22(2) :195–201, 1975.

[16] Nebel, Bernhard: *On the compilability and expressive power of propositional planning formalisms*. Journal of Artificial Intelligence Research, 12 :271–315, 2000.

[17] Palacios, Héctor et Hector Geffner: *Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width*. Journal of Artificiall Intelligence Research, 35 :623–675, 2009.

[18] Rintanen, Jussi: *Expressive Equivalence of Formalisms for Planning with Sensing*. Dans *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003)*, pages 185–194. AAAI Press, 2003.

[19] Rintanen, Jussi: *Complexity of Planning with Partial Observability*. Dans *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 345–354. AAAI Press, 2004.

[20] Scheck, Sergej, Alexandre Niveau et Bruno Zanuttini: *Knowledge Compilation for Nondeterministic Action Languages*. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, tome 31, pages 308–316, 2021.

[21] Speck, David, David Borukhson, Robert Mattmüller et Bernhard Nebel: *On the Compilability and Expressive Power of State-Dependent Action Costs*. Dans *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, pages 358–366. AAAI Press, 2021.

[22] Speck, David, Florian Geißer et Robert Mattmüller: *Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams*. Dans *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, pages 250–258. AAAI Press, 2018.

[23] Thiébaux, Sylvie, Joerg Hoffmann et Bernhard Nebel: *In defense of PDDL axioms*. Artificial Intelligence, 168 :38–69, 2005.

[24] To, Son Thanh, Tran Cao Son et Enrico Pontelli: *A generic approach to planning in the presence of incomplete information : Theory and implementation*. Artificial Intelligence, 227 :1–51, 2015.

Actes JIAF 2022

Une sémantique opérationnelle pour les QCHR

Igor Stéphan[1] **Vincent Barichard**[1]

[1] LERIA, Université d'Angers, France

igor.stephan@univ-angers.fr vincent.barichard@univ-angers.fr

Abstract

Nous passons du formalisme QCSP (Quantified Constraint Satisfaction Problems) à QCHR (Quantified Constraint Handling Rules) en permettant la construction d'un lieur dynamique et l'utilisation de contraintes définies par l'utilisateur. QCSP offre un cadre naturel pour exprimer les problèmes PSPACE comme des jeux finis à deux joueurs. Pour définir un modèle QCSP, le lieur doit être connu au préalable et ne peut pas être construit dynamiquement même si le pire cas ne se produira pas. Pour surmonter ce problème, nous définissons le nouveau formalisme QCHR qui permet de construire le lieur dynamiquement pendant la résolution. Nos modèles QCHR présentent d'excellentes performances lorsque le lieur est statique et surpassent les approches QCSP lorsque le lieur est dynamique. Cet article présente une sémantique opérationnelle sous la forme d'un ensemble de règles de transition, très proche de l'implantation effective, par extension de la sémantique opérationnelle de CHR (Constraint Handling Rules).

1 Introduction

Le passage du formalisme des QCSP (pour *Quantified Constraint Satisfaction Problems*) à celui de QCHR (pour *Quantified Constraint Handling Rule*) est motivé par les difficultés de développement d'applications réelles en QCSP. Une sémantique formelle de QCHR ainsi qu'une preuve de concept utilisant un système de calcul des séquents est présenté dans [4]. Dans ce présent article, nous définissons une sémantique opérationnelle décrivant précisément le fonctionnement de la résolution d'un QCHR.

QCSP [8, 25, 17, 5, 23, 21, 3] est une généralisation des *problèmes de satisfaction de contraintes* (CSP) dans lesquels les variables peuvent être quantifiées existentiellement (comme dans CSP) mais également universellement. Un QCSP est une alternance de variables quantifiées existentiellement et universellement sur des domaines finis, appelé le lieur, suivi d'un CSP. Les variables universellement quantifiées représentent des paramètres incontrôlables tels que les événements météorologiques. Un QCSP peut être vu comme un jeu à deux joueurs dans lequel les variables existentiellement quantifiées représentent un joueur A et les variables universellement quantifiées représentent un joueur B.

QCSP+ [5], une extension de QCSP, a été proposée pour faciliter la modélisation des jeux à deux joueurs. QCSP+ utilise des séquences de quantifications restreintes au lieu des séquences de quantifications standards. Un QCSP/QCSP+ est valide si le joueur A a une stratégie pour gagner, c'est-à-dire une stratégie pour fixer les variables existentiellement quantifiées telle que, quel que soit le paramètre choisi par le joueur B, le CSP sous-jacent est satisfait. QCSP/QCSP+ est un cadre de modélisation riche qui conduit à une modélisation succincte. Mais cette extension augmente également la complexité de la résolution de NP-complet à PSPACE-complet. Pour intégrer un problème réel dans un QCSP/QCSP+, il faut modéliser chaque partie du problème et tout doit être énoncé a priori. Certains problèmes (tic-tac-toe, reversi, puissance-4) répondent à cette exigence, mais d'autres (dames, échecs) ne le font pas. QCSP/QCSP+ ne peut pas être utilisé pour modéliser des jeux et des problèmes dont le nombre de coups n'est pas connu à priori. De plus, même si tout peut être énoncé statiquement, le modèle QCSP implique tous les coups possibles et surestime le nombre de coups dans le pire des cas. Par exemple, il est difficile d'encoder des jeux tels que certaines règles contraignent les coups futurs en fonction des coups passés [6] : la résolution d'un QCSP cherchera des solutions pour tous les coups possibles du joueur B, alors que certains d'entre eux ont été rendus impossibles par les coups précédents. Lorsqu'un QCSP/QCSP+ est défini, la résolution repose sur un solveur QCSP [17, 3, 25]. Ce solveur est un programme boîte noire qui résout une instance d'un modèle donné. Comme la plupart des solveurs QCSP/QCSP+ sont basés sur des solveurs CSP, il n'y a pas de moyen facile d'aider et guider le processus de résolution en prenant en compte les propriétés spécifiques d'un problème quantifié. Dans ce travail, nous utilisons un nouveau cadre pour modéliser les problèmes quantifiés d'une manière dynamique.

CHR (pour *Constraint Handling Rules*) [11, 12, 16, 13, 14, 15] est un langage à base de règles d'inférence à chaînage avant qui remplacent les contraintes du problème par d'autres plus simples jusqu'à la résolution complète. Chaque règle est composée d'une tête (un ensemble de contraintes devant exister dans un store d'éléments pour déclencher la règle), d'une garde (conditions à remplir, comme la valeur d'un paramètre d'une des contraintes capturées par la tête, pour exécuter la règle) et d'un corps (liste de contraintes à ajouter lorsque la règle est déclenchée). CHR est un langage dont l'objectif est de manipuler des contraintes déclaratives au sens de la programmation logique par contraintes [18, 19, 20]. CHR est une extension qui permet d'introduire des contraintes définies par l'utilisateur, c'est-à-dire des prédicats du premier ordre, dans un langage hôte donné comme Prolog, Lisp, Java, ou C/C++. CHR définit la simplification des contraintes définies par l'utilisateur, qui remplace les contraintes par des contraintes plus simples tout en préservant l'équivalence logique. CHR définit également la propagation des contraintes définies par l'utilisateur qui ajoute de nouvelles contraintes ; ces contraintes sont logiquement redondantes mais peuvent entraîner des simplifications supplémentaires. CHR permet d'utiliser des gardes qui sont des séquences d'instructions du langage hôte définissant des conditions à satisfaire pour déclencher la règle. CHR définit enfin la simpagation sur les contraintes définies par l'utilisateur qui mélange et subsume la simplification et la propagation. Les règles de CHR (simpagation) sont appliquées sur des multi-ensembles de contraintes. L'application répétée de ces règles sur un multi-ensemble de contraintes initiales résout incrémentalement ces contraintes. Le principe du chaînage avant exprime un non-déterminisme *don't care*, permettant des implémentations efficaces.

$\text{CHR}^{\vee}$ [1] est une extension de CHR introduisant le *don't know* non-déterminisme [7]. Ce non-déterminisme est offert gratuitement lorsque le langage hôte est Prolog. Ce non-déterminisme permet de spécifier facilement des problèmes de la classe de complexité NP mais ce n'est pas le cas pour le reste de la Hiérarchie Polynomiale et, en général, tout problème exprimé avec des quantifications alternées (bien que le formalisme soit Turing-complet).

Cet article reprend le formalisme *QCHR* (pour *Quantified Constraint Handling Rules*) dans lequel la règle de simpagation est étendue à une règle de simpagation *existentielle* mais aussi à une règle de simpagation *universelle*. La simpagation existentielle (resp. universelle) a les mêmes conditions pour être appliquée qu'une règle de simpagation mais le corps est existentiellement (resp. universellement) quantifié sur une variable et un domaine (fini). Nous obtenons un formalisme pour lequel il n'est pas nécessaire de déclarer *à priori* l'alternance des quantificateurs mais où les quantificateurs sont générés quand ils sont nécessaires. Cette propriété offre une facilité de programmation par rapport à QCSP/QCSP+ où il faut toujours déclarer un lieur *fini*. Avec QCHR, nous pouvons spécifier certains jeux potentiellement infinis par récursion.

Le nouveau cadre des QCHR a été défini dans [4] par une sémantique exprimée sous la forme d'un calcul des séquents directement issu d'une sémantique, exprimée elle aussi dans ce formalisme venant de la théorie de la preuve, mais pour CHR. Dans ce présent article, nous donnons une autre sémantique, que nous prouvons comme étant équivalente à la première, sous une forme opérationnelle basée sur la notion d'état et de transition, comme une extension de la sémantique opérationnelle ω_r de CHR [10] qui décrit de manière fine le fonctionnement du langage au plus près de son implantation dans son langage hôte, le C++.

La section 2 présente intuitivement la syntaxe de notre formalisme QCHR, illustre sur deux exemples emblématiques pourquoi il n'est pas toujours approprié de modéliser et résoudre avec QCSP/QCSP+, et montre la facilité de modélisation et l'efficacité de résolution de notre nouveau formalisme. La section 3 présente notre proposition, le langage QCHR : en premier lieu, sa syntaxe et en second lieu sa sémantique opérationnelle. La section 4 présente une discussion sur le lien entre QCHR et d'autres travaux connexes ainsi que des liens vers l'implémentation et des expérimentations déjà réalisées. La section 5 conclut et trace quelques perspectives.

2 Exemples motivant notre approche

Notre objectif principal est de pouvoir modéliser des problèmes quantifiés lorsque le lieur peut être construit dynamiquement pendant la résolution. Deux problèmes bien connus sont utilisés comme fil conducteur : le jeu Nim et le Puissance 4. De manière informelle, le formalisme CHR est étendu avec deux nouvelles règles : la règle existentielle (simpagation QCHR)

$$K_1, \ldots, K_m \backslash D_1, \ldots, D_n <\exists> \{it, inf, sup\}\ garde \mid \Omega^B$$

et la règle universelle (simpagation QCHR)

$$K_1, \ldots, K_m \backslash D_1, \ldots, D_n <\forall> \{it, inf, sup\}\ garde \mid \Omega^B$$

La tête $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n)$, le corps Ω^B et la $garde$ de ces règles sont interprétés de la même manière que dans le formalisme CHR : Les contraintes $K_1, \ldots, K_n$ sont conservées comme dans la propagation et les contraintes $D_1, \ldots, D_m$ sont supprimées comme dans la simplification ; les contraintes du corps $\Omega^B = B_1, \ldots, B_p$ sont les contraintes ajoutées ; si $\Omega^B = true$, rien n'est ajouté ; si $\Omega^B = false$, le calcul échoue (*true* et *false* sont deux symboles réservés). Les deux symboles *inf* et *sup* désignent, respectivement, la borne inférieure et la borne supérieure d'un intervalle entier. La variable *it* est censée apparaître dans le corps Ω^B. La sémantique informelle de ces règles est la

suivante : Le corps Ω^B de la règle existentielle conduit à un succès (resp. un échec) si au moins une valeur (resp. toutes les valeurs) v prise dans l'intervalle $[inf..sup]$ conduit le corps $[it \leftarrow v](\Omega^B)$ (où les occurrences de it dans Ω^B sont remplacées par v) à un succès (resp. un échec). De la même manière, le corps Ω^B de la règle universelle conduit à un succès (resp. un échec) si toutes les valeurs (resp. au moins une valeur) v prises dans l'intervalle $[inf..sup]$ conduisent le corps $[it \leftarrow v](\Omega^B)$ à un succès (resp. un échec). Si une règle QCHR existentielle (resp. universelle) conserve toutes les contraintes de sa tête multiple, c'est une règle de propagation existentielle (resp. universelle) et elle est notée $(K_1, \ldots, K_m \exists > \{it, inf, sup\}\ garde \mid \Omega^B)$ (resp. $(K_1, \ldots, K_m \forall > \{it, inf, sup\}\ garde \mid \Omega^B)$); si une règle QCHR existentielle (resp. universelle) ne conserve aucune des contraintes de sa tête multiple, il s'agit d'une règle de simplification existentielle (resp. universelle) et elle est notée $(D_1, \ldots, D_n < \exists > \{it, inf, sup\}\ garde \mid \Omega^B)$ (resp. $(D_1, \ldots, D_n < \forall > \{it, inf, sup\}\ garde \mid \Omega^B)$).

Le jeu de Nim. Le jeu de Nim est un jeu à deux joueurs qui se joue avec un tas de pièces ou d'allumettes. Le but du jeu est de prendre la dernière allumette. Chaque joueur peut prendre de une à trois allumettes. Dans la variante Fibonacci, le nombre minimum est d'une allumette et le maximum au premier tour de jeu est d'une allumette de moins que le nombre initial d'allumettes. Ensuite, chaque joueur peut prendre de un à deux fois plus d'allumettes que l'adversaire au tour de jeu précédent. Le joueur A commence. Par exemple, avec 4 allumettes, le joueur A prend 1 allumette. Ensuite, le joueur B peut prendre 3 allumettes mais s'il le fait, il perd immédiatement puisque le coup n'est pas autorisé. Le joueur B peut donc prendre 1 ou 2 allumettes. Mais quoi qu'il prenne, il perd puisque le joueur A prendra les allumettes restantes. La partie la plus longue possible a 4 tours : chaque joueur prend une allumette à chaque tour de jeu. Ainsi, le QCSP a 4 quantificateurs puisque le lieur est défini statiquement. Avec un nombre pair p d'allumettes, la spécification QCSP est la suivante (x_i, resp. y_i, correspond au nombre d'allumettes choisies par le joueur A, resp. B, au tour i) : $R_A(1)$ est vrai (le joueur A choisit entre 1 et $p-1$ allumettes) et pour tous les i, $1 < i \leq \frac{p}{2}$, $R_A(i) = (1 \leq x_i \leq 2*y_{i-1}) \wedge (x_i + \Sigma_{1 \leq j < i}(x_j + y_j) \leq p)$ et pour tout i, $1 \leq i \leq \frac{p}{2}$, $R_B(i) = (1 \leq y_i \leq 2*x_i) \wedge (\Sigma_{1 \leq j \leq i}(x_j + y_j) \leq p)$ (chaque joueur prend de un à deux fois plus d'allumettes que l'adversaire au tour précédent) [3] :

$$\begin{array}{l}
\exists x_1 \forall y_1 \ldots \exists x_{\frac{p}{2}} \forall y_{\frac{p}{2}} \exists o_1 \ldots \exists o_{\frac{p}{2}} \\
\quad R_A(1) \wedge (R_B(1) \rightarrow o_1) \wedge (o_1 \leftrightarrow (R_A(2) \wedge\ o_2)) \wedge \\
\quad (o_2 \leftrightarrow (R_B(2) \rightarrow o_3)) \wedge \ldots \wedge (o_{\frac{p}{2}} \leftrightarrow (R_B(\frac{p}{2}) \rightarrow \bot))
\end{array}$$

avec $x_1, y_1, \ldots, x_{\frac{p}{2}}, y_{\frac{p}{2}} \in [1..p-1]$ et $o_1, \ldots, o_{\frac{p}{2}}$ des variables booléennes. Pour chaque nombre initial d'allumettes, le modèle général doit être instancié. Par exemple, avec $p = 4$, on obtient le QCSP suivant :

$$\begin{array}{l}
\exists x_1 \forall y_1 \exists x_2 \forall y_2 \exists o_1 \exists o_2 \\
(((y_1 \leq 2 * x_1) \wedge\ (x_1 + y_1 \leq 4)) \rightarrow o_1) \wedge \\
(o_1 \leftrightarrow (((x_2 \leq 2 * y_1) \wedge\ (x_1 + y_1 + x_2 \leq 4)) \wedge\ o_2)) \wedge \\
(o_2 \leftrightarrow (((y_2 \leq 2 * x_2) \wedge\ (x_1 + y_1 + x_2 + y_2 \leq 4)) \rightarrow \bot))
\end{array}$$

avec $x_1, y_1, x_2, y_2 \in \{1, 2, 3\}$ et o_1, o_2 des variables intermédiaires booléennes (permettant de décomposer le problème en plusieurs contraintes réifiées). Ce QCSP est valide si le premier joueur a une stratégie pour choisir les valeurs des variables existentiellement quantifiées telle que, quelle que soit le choix de l'adversaire pour ses variables universellement quantifiées, ce premier joueur gagne (c'est-à-dire que le CSP est vrai).

Le programme QCHR suivant résout le jeu Nim avec la variante Fibonacci d'un nombre pair ou impair d'allumettes (N représente deux fois le nombre d'allumettes choisies par un joueur et R représente le nombre restant d'allumettes dans le tas) :

$$\begin{array}{l}
u@nfB(N, R) < \forall > \{it, 1, min(N, R)\}\ nfA(2 * it, R - it) \\
e@nfA(N, R) < \exists > \{it, 1, min(N, R)\}\ nfB(2 * it, R - it)
\end{array}$$

Ces règles sont construites de la même manière : ce sont des règles de simplification existentielles/universelles avec une garde vide (omise). Le nom (optionnel) de la règle est séparé du reste de la règle par le symbole @. Deux contraintes $nfA(,)$ et $nfB(,)$ sont utilisées pour mémoriser les informations d'un tour du jeu. À chaque exécution d'une des deux règles, la contrainte identifiée par la tête est supprimée, puis une nouvelle est postée dans le corps. Les paramètres du quantificateur ($\forall$ ou $\exists$) donnent la définition des bornes inférieure et supérieure, et le corps qui exprime que si un joueur a choisi de prendre it allumettes, son adversaire ne peut choisir qu'entre 1 et $min(2 * it, R)$ allumettes. Le premier joueur peut choisir librement entre 1 et le nombre initial d'allumettes moins 1 :

$l@nim_fibo(R) \Rightarrow nfA(R - 1, R)$

Le lieur n'est pas défini statiquement comme pour un QCSP mais dynamiquement. Notons que le cas d'arrêt de la récursion est caché dans la sémantique de la règle universelle lorsque la borne inférieure devient plus grande que la borne supérieure.

Le jeu de Puissance 4. Le jeu de Puissance 4 est un jeu à deux joueurs dans lequel les joueurs choisissent d'abord une couleur. Il se joue sur une grille suspendue verticalement. À chaque tour, un joueur choisit une colonne de la grille et laisse tomber une pièce de sa couleur à partir du haut de celle-ci. Les pièces tombent directement vers le bas, occupant l'emplacement le plus bas disponible dans la colonne. Le gagnant est le premier joueur à former une ligne horizontale, verticale ou diagonale de quatre de ses propres pièces.

Le jeu de Puissance 4 peut également être modélisé par un QCSP. Afin de modéliser tous les déroulements possibles, une grille est construite pour chaque tour de jeu. Il y a autant de tours de jeu qu'il y a d'emplacements dans la grille. Par conséquent, le modèle QCSP implique *nombre de lignes* × *nombre de colonnes* grilles liées entre elles par des contraintes (voir [22] pour plus de détails sur le modèle QCSP). De plus, des contraintes doivent être ajoutées pour détecter les colonnes pleines, les coups invalides et les grilles gagnantes. Mais, même si un QCSP peut être utilisé, le lieur est dynamique (un joueur peut gagner la partie sans remplir complètement le plateau) et le modèle statique est volumineux et difficilement compréhensible. En comparaison, le modèle QCHR est plus adapté, plus léger et plus lisible. Soit NC le nombre de colonnes de la grille. Le modèle QCHR suivant calcule une stratégie gagnante si une telle stratégie existe :

$$
\begin{aligned}
&if_{\bot} \ @\ \mathit{ifRule}(\bot, N) \Leftrightarrow \mathrm{coin}(N), \mathit{cfe}(\mathrm{isWon}(N))\\
&u_{\top} \ @\ \mathit{cfu}(\top) \Leftrightarrow \mathit{true}\\
&u_{\bot} \ @\ \mathit{cfu}(\bot)<\forall>\{it, 1, NC\}\ \mathit{ifRule}(\mathrm{isFull}(it), it)\\
&e_{\top} \ @\ \mathit{cfe}(\top) \Leftrightarrow \mathit{false}\\
&e_{\bot} \ @\ \mathit{cfe}(\bot)<\exists>\{it, 1, NC\}\ \mathrm{coin}(it), \mathit{cfu}(\mathrm{isWon}(it))
\end{aligned}
$$

où coin(), isFull() et isWon() sont des prédicats intégrés (fonctions implémentées dans le langage hôte) déjà définies : coin() place une pièce dans la colonne donnée et génère un échec si la colonne est pleine ; isFull() renvoie $\top$ si la colonne est pleine et $\bot$ sinon ; isWon() renvoie $\top$ si un alignement de quatre pièces est trouvé et $\bot$ sinon. Ces fonctions analysent une grille qui est remplie en fonction des déplacements des joueurs.

Le lieur n'est pas défini statiquement comme pour QCSP mais dynamiquement. Par conséquent, la taille du lieur est égale au nombre de coups effectués pendant la partie et non au pire cas possible.

3 Le langage QCHR

Une contrainte est considérée comme un prédicat du premier ordre. Seuls trois types de contraintes prédéfinies sont nécessaires : *true*, *false* et la contrainte d'égalité syntaxique désignée par $\doteq$, avec une théorie de l'égalité désignée ET pour les variables et les constantes [1]. Pour deux séquences de contraintes $K_1, \ldots, K_m \doteq k_1, \ldots, k_m$ signifie $K_1 \doteq k_1, \ldots, K_m \doteq k_m$ et pour deux contraintes $c(t_1, \ldots, t_n) \doteq c(s_1, \ldots, s_n)$ signifie $t_1 \doteq s_1, \ldots, t_n \doteq s_n$.

1. Nous n'utilisons pas de symboles fonctionnels mais le langage est plus riche que les simples variables et constantes puisqu'on peut utiliser les instructions du langage hôte qui sont *évaluées* avant que l'égalité ne soit appliquée sur une expression complètement instanciée. C'est le cas pour les opérations arithmétiques dans la modélisation du jeu Nim ou pour des fonctions plus *ad hoc* comme $isFull$ et $isWon$ dans la modélisation du jeu du Puissance 4.

3.1 Syntaxe de QCHR

Le formalisme QCHR est défini comme suit : une règle QCHR est une règle de la forme $(k_1, \ldots, k_m, d_1, \ldots, d_n, b_1, \ldots, b_p$ des contraintes utilisateur ; *inf* et *sup* des entiers ; *it* une variable entière ; *garde* une séquence d'instructions dans le langage hôte ou des égalités) :

[Règle de simpagation]
$(k_1, \ldots, k_m \backslash d_1, \ldots, d_n \Leftrightarrow garde \mid \Omega^B)$
avec $n \geq 0$ et $m \geq 0$,
[Règle de simpagation existentielle]
$(k_1, \ldots, k_m \backslash d_1, \ldots, d_n <\exists> \{it, inf, sup\}\ garde \mid \Omega^B)$
avec $n \geq 0$ et $m \geq 0$,
[Règle de simpagation universelle]
$(k_1, \ldots, k_m \backslash d_1, \ldots, d_n <\forall> \{it, inf, sup\}\ garde \mid \Omega^B)$
avec $n \geq 0$ et $m \geq 0$.
et $\Omega^B = b_1, \ldots, b_p$ avec $p > 0$ ou *true* ou *false*.

Une règle de *propagation* (resp. *simplification*) est un cas particulier de la règle de simpagation où l'ensemble $\{d_1, \ldots, d_n\}$ (resp. $\{k_1, \ldots, k_m\}$) est vide. Ceci est vrai également pour les règles de simpagation *existentielles* et *universelles*.

3.2 La sémantique $\omega_r^{\exists\forall}$

Notre système de transitions est basé sur les *états d'exécution* de [10] pour CHR.

Definition 3.1 (État d'exécution) *Une* contrainte identifiée *A#i (ou token) est une contrainte A définie par l'utilisateur avec un certain entier unique i, son identité. Un token* $<\exists\Omega^B>$ *et un token* $<\forall\Omega^B>$ *sont composés (dans cet ordre) d'une variable entière, de deux entiers (les bornes inférieure et supérieure) et d'une séquence de contraintes définies par l'utilisateur. Une* contrainte numérotée *A#i:o est une contrainte identifiée A#i avec un entier unique o fonction de i. Un* état d'exécution *est soit un des symboles réservés : Failure et Success, soit un tuple* $\langle \Omega, S, H \rangle_c$ *où* Ω *(le* but *actuel) représente une séquence de contraintes (numérotées) ou de contraintes définies par l'utilisateur ou false, S (le store actuel) représente un multi-ensemble de contraintes identifiées, H (l'*historique *courant) représente un ensemble de mots, chacun enregistrant le nom d'une règle et les identités des contraintes identifiées, c représente un compteur qui identifie le prochain nombre entier libre pouvant être utilisé pour numéroter une contrainte identifiée.*

Definition 3.2 (Continuation d'exécution) *Un lieur est une séquence de* $\exists$ *et de* $\forall$. *Une* continuation de décision *est une séquence d'états d'exécution. Une* continuation d'exécution *est une paire* $\mathcal{B}\Sigma$ *où* $\mathcal{B}$ *est un lieur et* Σ *est une continuation de décision.*

Pour un but initial Ω, la continuation d'exécution initiale est $[\langle \Omega, \emptyset, \emptyset \rangle_1]$.

La sémantique opérationnelle $\omega_r^{\exists\forall}$ est basée sur un ensemble de règles de transition qui font correspondre une continuation d'exécution à une autre. Les six premières règles [2] sont des extensions directes des règles de transition de la sémantique ω_r [10] pour CHR : le lieur est ignoré et le premier état d'exécution de la continuation d'exécution est considéré comme un état d'exécution de ω_r.

3.3 Extension de la sémantique ω_r

Les règles de transition suivantes [*Activate*], [*Reactivate*], [*Drop*], [*Simplify*], [*Propagate*], et [*Default*] sont des extensions directes des règles de transition ω_r de [10]. Dans ce qui suit, certaines contraintes (A, $k_1, \ldots, k_m$, $d_1, \ldots, d_n$), des ensembles de token (S, S^K, S^D, ...), des séquences de contraintes (Ω, Ω^B) et certains ensembles d'historique (H, H') sont utilisés, un lieur $\mathcal{B}$ peut être vide. Le but est composée d'une séquence de contraintes donnée sous forme de liste en intention ([tête | queue]) ou en extension. La majorité des règles suivantes sont illustrées dans l'exemple 3.1.

La règle de transition [*Activate*] **:**

$$\begin{array}{l}\mathcal{B}[\langle [A \mid \Omega], S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}[\langle [A\#c : 1 \mid \Omega], \{A\#c\} \uplus S, H\rangle_{c+1}|\Sigma]\end{array}$$

La règle de transition [*Activate*] active la première contrainte A du but $[A \mid \Omega]$. A est identifié par c (le prochain identifiant libre), ajouté au *store* de contraintes S et numéroté par 1 avant d'être remplacé dans le but. Le prochain identifiant libre est incrémenté. La règle de transition [*Activate*] prépare la contrainte A qui est prête à être testée avec sa première occurrence dans la tête des règles du programme.

La règle de transition [*Reactivate*] **:**

$$\mathcal{B}[\langle [A\#i \mid \Omega], S, H\rangle_c|\Sigma] \rightsquigarrow_r^{\exists\forall} \mathcal{B}[\langle [A\#i : 1 \mid \Omega], S, H\rangle_c|\Sigma]$$

Lorsqu'une contrainte A n'est pas supprimée par une règle du programme, elle reste dans le *store* de contraintes. Si un de ses paramètres est mis à jour (par exemple lorsque deux variables logiques sont placées dans la même classe d'équivalence), la contrainte A est réactivée et placée au début du but. La règle de transition [*Reactivate*] prépare la contrainte identifiée $A\#i$ et la numérote avec 1 afin d'être prête à être testée avec sa première occurrence dans la tête des règles du programme.

2. Puisque nous ne considérons pas la contrainte construite, nous n'avons pas la transition $[Solve]$ de la sémantique ω_r

La règle de transition [*Drop*] **:**

$$\mathcal{B}[\langle [A\#i : o \mid \Omega], S, H\rangle_c|\Sigma] \rightsquigarrow_r^{\exists\forall} \mathcal{B}[\langle \Omega, S, H\rangle_c|\Sigma]$$

La règle de transition [*Drop*] supprime la contrainte identifiée $A\#i$ du but si elle n'a pas d'o-ième occurrence dans le programme. A reste toujours dans le *store* de contraintes S et peut être réactivé si nécessaire.

La règle de transition [*Simplify*] **:** Si la o-ième occurrence de la contrainte A correspond à d_j dans une règle de simpagation

$$r@(k_1, \ldots, k_m \backslash d_1, \ldots, d_{j-1}, d_j, d_{j+1}, \ldots, d_n \Leftrightarrow \Omega^B)$$

du programme, alors

$$\begin{array}{l}\mathcal{B}[\langle [A\#i : o \mid \Omega], \{A\#i\} \uplus S^K \uplus S^{D^1} \uplus S^{D^2} \uplus S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}[\langle \Omega^B \oplus \Omega, S^K \uplus S, H'\rangle_c|\Sigma]\end{array}$$

avec
- r le nom (optionnel) de la règle ;
- $S^K = \{K_1\#i_1, \ldots, K_m\#i_m\}$,
  $S^{D^1} = \{D_1\#i_{m+1}, \ldots, D_{j-1}\#i_{m+j-1}\}$,
  $S^{D^2} = \{D_{j+1}\#i_{m+j+1}, \ldots, D_n\#i_{m+n}\}$ tels que :
  - $K_1, \ldots, K_m \doteq k_1, \ldots, k_m$
  - $D_1, \ldots, D_{j-1} \doteq d_1, \ldots, d_{j-1}$
  - $D_{j+1}, \ldots, D_n \doteq d_{j+1}, \ldots, d_n$
  - $A \doteq d_j$
- $H' = H$

La règle de transition [*Simplify*] applique une règle de simpagation si sa tête correspond à certaines contraintes du *store* de contraintes. Les (ensembles de) *token* S^{D^1}, S^{D^2} et $A\#i$ sont supprimés du *store* de contraintes, $A\#i$ est supprimé du but tandis que Ω^B, le corps de la règle, est ajouté au but. L'historique reste le même puisque la règle ne sera plus jamais déclenchée avec le même ensemble de *token* (les (ensembles de) *token* S^{D^1}, S^{D^2} et $A\#i$ ayant été supprimés).

La règle de [*Propagate*] **:** Si la o-ième occurrence de la contrainte A correspond à k_j dans une règle de simpagation

$$r@(k_1, \ldots, k_{j-1}, k_j, k_{j+1}, \ldots, k_m \backslash d_1, \ldots, d_n \Leftrightarrow \Omega^B)$$

du programme alors (avec $S^K = S^{K^1} \uplus S^{K^2}$)

$$\begin{array}{l}\mathcal{B}[\langle [A\#i : o \mid \Omega], \{A\#i\} \uplus S^K \uplus S^D \uplus S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}[\langle \Omega^B \oplus [A\#i : o \mid \Omega], \{A\#i\} \uplus S^K \uplus S, H'\rangle_c|\Sigma]\end{array}$$

avec
- r le nom (optionnel) de la règle ;
- $S^{K^1} = \{K_1\#i_1, \ldots, K_{j-1}\#i_{j-1}\}$,
  $S^{K^2} = \{K_{j+1}\#i_{j+1}, \ldots, K_m\#i_m\}$,
  $S^D = \{D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}\}$ tels que :
  - $K_1, \ldots, K_{j-1} \doteq k_1, \ldots, k_{j-1}$
  - $K_{j+1}, \ldots, K_m \doteq k_{j+1}, \ldots, k_m$

— $D_1, \ldots, D_n \doteq d_1, \ldots, d_n$
— $A \doteq k_j$
— $H' = H$ if $(n \neq 0)$
— $r.i.i_1 \ldots i_m \notin H$ and $H' = H \cup \{r.i.i_1 \ldots i_m\}$ si $(n = 0)$

La règle de transition [*Propagate*] applique une règle de simpagation si sa tête correspond à certaines contraintes du *store* de contraintes. De plus, si la règle est une règle de propagation (c'est-à-dire $(n = 0)$), nous devons être sûrs que la règle n'a pas été déclenchée auparavant en vérifiant $r.i.i_1 \ldots i_m \notin H$. Les contraintes à supprimer S^D sont supprimées du *store* de contraintes et Ω^B, le corps de la règle, est ajouté au but. Pour une règle de propagation, l'historique est mis à jour avec $r.i.i_1 \ldots i_m$, sinon, il reste identique.

La règle de transition [*Default*] **:**

$$\mathcal{B}[\langle [A\#i : o \mid \Omega], S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}[\langle [A\#i : o+1 \mid \Omega], S, H\rangle_c|\Sigma]$$

La règle de transition [*Default*] met à jour la contrainte active du but. Elle incrémente son nombre d'occurrences afin d'être prête pour la règle de transition suivante.

3.4 Règles de transition pour la quantification existentielle

La règle de transition [$\exists -$ *Simplify*] **:** Si la o-ième occurrence de la contrainte A correspond à la contrainte d_j dans une règle de simpagation existentielle

$$r@k_1, \ldots, k_m \backslash d_1, \ldots, d_{j-1}, d_j, d_{j+1}, \ldots, d_n \\ <\exists>\{it, inf, sup\}\ \Omega^B$$

du programme et $inf \leq sup$ alors

$$\mathcal{B}[\langle [A\#i : o \mid \Omega], \{A\#i\} \uplus S^K \uplus S^{D^1} \uplus S^{D^2} \uplus S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \mathcal{B}\exists[\langle [<\exists>(it, [inf, sup], \Omega^B)|\Omega], S^K \uplus S, H'\rangle_c|\Sigma]$$

avec les mêmes conditions que la règle de transition [*Simplify*].

La règle de transition [$\exists -$ *Simplify*] applique une règle de simpagation existentielle si les contraintes de sa tête correspondent à des *token* du *store*. Les (ensembles de) *token* S^{D^1}, S^{D^2} et $A\#i$ sont éliminés du *store*; de plus, le *token* $A\#i$ est effacé du but. Un nouveau niveau existentiel est ajouté au lieur $\mathcal{B}$ et un *token* $<\exists\Omega^B>$ est ajouté au but. Ce *token* $<\exists\Omega^B>$ est prêt à être déplié par la règle de transition [$\exists -$ *Unfolding*]. L'historique demeure le même puisque la règle ne pourra jamais être déclenchée avec le même ensemble de *token* (les (ensembles de) *token* S^{D^1}, S^{D^2} et $A\#i$ ayant été supprimés).

La règle de transition [$\exists -$ *Propagate*] **:** Si la o-ième occurrence de la contrainte A correspond à la contrainte k_j d'une règle de simpagation existentielle

$$r@k_1, \ldots, k_{j-1}, k_j, k_{j+1}, \ldots, k_m \backslash d_1, \ldots, d_n \\ <\exists>\{it, inf, sup\}\ \Omega^B$$

du programme et $inf \leq sup$ ($S^K = S^{K^1} \uplus S^{K^2}$) alors

$$\mathcal{B}[\langle [A\#i : o \mid \Omega], \{A\#i\} \uplus S^K \uplus S^D \uplus S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}\exists[\langle [<\exists>(it, [inf, sup], \Omega^B)|\Omega], S^K \uplus S, H'\rangle_c|\Sigma]$$

avec les mêmes conditions que la règle de transition [*Propagate*].

La règle de transition [$\exists -$ *Propagate*] applique une règle de simpagation existentielle si les contraintes de sa tête correspondent à des *token* du *store*. De plus, si la règle est une règle de propagation (c'est à dire $(n = 0)$), nous devons être sûrs que la règle n'a pas déjà été déclenchée auparavant en vérifiant que $r.i.i_1 \ldots i_m \notin H$. Les *token* de l'ensemble S^D sont éliminées du *store*. Un nouveau niveau existentiel est ajouté au lieur $\mathcal{B}$ et un *token* $<\exists\Omega^B>$ est ajouté au but. Ce *token* $<\exists\Omega^B>$ est prêt à être déplié par la règle de transition [$\exists -$ *Unfolding*]. Pour une règle de propagation, l'historique est mis-à-jour avec $r.i.i_1 \ldots i_m$, sinon, il reste identique.

La règle de transition [$\exists -$ *Unfolding*] **:** Si $inf \leq sup$ alors

$$\mathcal{B}\exists[\langle [<\exists>(it, [inf, sup], \Omega^B)|\Omega], S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}\exists[\ \langle [it \leftarrow inf](\Omega^B), S, H\rangle_c, \\ \langle [<\exists>(it, [inf+1, sup], \Omega^B)|\Omega], S, H\rangle_c|\Sigma]$$

La règle de transition [$\exists -$ *Unfolding*] déplie le *token* $<\exists\Omega^B>$ d'une étape. Une telle étape consiste en l'extraction et la vérification du corps Ω^B dans lequel la variable locale it est substituée par la borne inférieure inf du domaine du *token* $<\exists\Omega^B>$; elle ajoute aussi au but Ω un nouveau *token* $<\exists\Omega^B>$ avec une borne inférieure incrémentée. Ce nouveau *token* $<\exists\Omega^B>$ sera déplié si $[it \leftarrow inf](\Omega^B)$ échoue.

La règle de transition [$\exists -$ *Failure*] **:** Si $inf > sup$ alors

$$\mathcal{B}\exists[\langle [<\exists>(it, [inf, sup], \Omega^B)|\Omega], S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}[Failure|\Sigma]$$

La règle de transition [$\exists -$ *Failure*] s'applique si $inf > sup$ et si les *token* $<\exists\Omega^B>$ dépliés échouent. Cela clôt et élimine le niveau courant du lieur. En éliminant le niveau courant du lieur, le *token* $<\exists\Omega^B>$ et le reste du but Ω sont aussi éliminés et l'état d'exécution *Failure* prend la place de l'état d'exécution courant.

Le règle de transition [$\exists -$ *SuccessPropagate*] **:**

$$\mathcal{B}\exists[Success, \langle [<\exists>(it, [inf, sup], \Omega^B)|\Omega], S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}[\langle \Omega, S, H\rangle_c|\Sigma]$$

La règle de transition [$\exists -$ *SuccessPropagate*] s'applique lorsque qu'un corps Ω^B déplié est un succès (c-à-d l'état d'exécution *Success* est rencontré). La règle de transition [$\exists -$ *SuccessPropagate*] clôt et élimine le niveau courant du lieur. Comme le *token* $<\exists\Omega^B>$ réussit, le reste du but Ω doit être poursuivi au niveau supérieur du lieur.

La règle de transition $[\exists - \mathit{FailurePropagate}]$ **:**

$$\mathcal{B}\exists[\mathit{Failure}|\Sigma] \rightsquigarrow_r^{\exists\forall} \mathcal{B}\exists\Sigma$$

La règle de transition $[\exists - \mathit{FailurePropagate}]$, lorsqu'un corps Ω^B déplié est un échec (c'est-à-dire l'état d'exécution $\langle[it \leftarrow \mathit{inf}](\Omega^B), S, H\rangle_c$ s'est réécrit en l'état d'exécution *Failure*), poursuit le parcours des éléments de la quantification existentielle (car alors $\Sigma = [\langle[<\exists>(it, [\mathit{inf}+1, \mathit{sup}], \Omega^B)|\Omega], S, H\rangle_c \mid _])$.

3.5 Règles de transition pour la quantification universelle

La règle de transition $[\forall - \mathit{Simplify}]$ **:** Si la o-ième occurrence de la contrainte A correspond à la contrainte d_j de la règle de simpagation universelle

$$\begin{array}{l} r@k_1, \ldots, k_m \backslash d_1, \ldots, d_{j-1}, d_j, d_{j+1}, \ldots, d_n \\ \quad <\forall>\{it, \mathit{inf}, \mathit{sup}\}\ \Omega^B \end{array}$$

du programme et $\mathit{inf} \leq \mathit{sup}$ alors

$$\begin{array}{l} \mathcal{B}[\langle[A\#i : o \mid \Omega], \{A\#i\} \uplus S^K \uplus S^{D^1} \uplus S^{D^2} \uplus S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \mathcal{B}\forall[\langle[<\forall>(it, [\mathit{inf}, \mathit{sup}], \Omega^B)|\Omega], S^K \uplus S, H'\rangle_c|\Sigma] \end{array}$$

avec les mêmes conditions que la règle de transition $[\mathit{Simplify}]$.

La règle de transition $[\forall - \mathit{Simplify}]$ applique une règle de simpagation universelle si les contraintes de sa tête correspondent à des *token* du *store*. Les (ensembles de) tokens S^{D^1}, S^{D^2} et $A\#i$ sont éliminés du *store* ; de plus, le *token* $A\#i$ est éliminé du but. Un nouveau niveau universel est ajouté au lieur et un *token* $<\forall\Omega^B>$ est ajouté au but. Ce *token* $<\forall\Omega^B>$ est prêt à être déplié par la règle de transition $[\forall - \mathit{Unfolding}]$. L'historique demeure le même puisque la règle ne pourra jamais être déclenchée avec le même ensemble de *token* (les (ensembles de) *tokens* S^{D^1}, S^{D^2} et $A\#i$ ayant été supprimés).

La règle de transition $[\forall - \mathit{Propagate}]$ **:** Si la o-ième occurrence de la contrainte A correspond à la contrainte k_j d'une règle de simpagation universelle

$$\begin{array}{l} r@k_1, \ldots, k_{j-1}, k_j, k_{j+1}, \ldots, k_m \backslash d_1, \ldots, d_n \\ \quad <\forall>\{it, \mathit{inf}, \mathit{sup}\}\ \Omega^B \end{array}$$

du programme et $\mathit{inf} \leq \mathit{sup}$ alors

$$\begin{array}{l} \mathcal{B}[\langle[A\#i : o \mid \Omega], \{A\#i\} \uplus S^{K^1} \uplus S^{K^2} \uplus S^D \uplus S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}\forall[\langle[<\forall>(it, [\mathit{inf}, \mathit{sup}], \Omega^B)|\Omega], S^K \uplus S, H'\rangle_c|\Sigma] \end{array}$$

avec les mêmes conditions que la règle de transition $[\mathit{Propagate}]$.

La règle de transition $[\forall - \mathit{Propagate}]$ applique une règle de simpagation universelle si les contraintes de sa tête correspondent avec des *token* du *store*. De plus, si la règle est une règle de propagation (c'est-à-dire $(n = 0)$) nous devons être sûrs que la règle n'a pas été déclenchée auparavant en vérifiant $r.i.i_1 \ldots i_m \notin H$. Un nouveau niveau universel est ajouté au lieur $\mathcal{B}$ et un *token* $<\forall\Omega^B>$ est ajouté au but. Ce *token* $<\forall\Omega^B>$ est prêt à être déplié par la règle de transition $[\forall - \mathit{Unfolding}]$. Pour une règle de propagation, l'historique est mis-à-jour avec $r.i.i_1 \ldots i_m$, sinon, il reste identique.

Nous remarquons que les règles de transition $[\forall - \mathit{Simplify}]$ et $[\forall - \mathit{Propagate}]$ ajoutent toutes les deux un niveau au lieur $\mathcal{B}$. Ce faisant, elles ouvrent un sous-espace pour la résolution du but actuel. Le but sera résolu et un succès ou un échec sera traité en retour. Cela correspond au concept d'arbre de recherche et d'ouverture de nœuds dans le domaine de la résolution de problèmes combinatoires.

La règle de transition $[\forall - \mathit{Unfolding}]$ **:**

$$\begin{array}{l} \mathcal{B}\forall[\langle[<\forall>(it, [\mathit{inf}, \mathit{sup}], \Omega^B)|\Omega], S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}\forall[\ \langle[it \leftarrow \mathit{inf}](\Omega^B), S, H\rangle_c, \\ \qquad\qquad \langle[<\forall>(it, [\mathit{inf}+1, \mathit{sup}], \Omega^B)|\Omega], S, H\rangle_c|\Sigma] \end{array}$$

avec $\mathit{inf} \leq \mathit{sup}$.

La règle de transition $[\forall - \mathit{Unfolding}]$ traite et fait avancer le *token* $<\forall\Omega^B>$ d'une étape. Une étape consiste à extraire et vérifier le corps Ω^B où la variable locale *it* est substituée par la borne inférieure *inf* du domaine du *token* $<\forall\Omega^B>$. Il ajoute également au but Ω un nouveau *token* $<\forall\Omega^B>$ avec une limite inférieure incrémentée. Le nouveau *token* $<\forall\Omega^B>$ sera développé si le Ω^B actuel réussit.

La règle de transition $[\forall - \mathit{Success}]$ **:** Avec $\mathit{inf} > \mathit{sup}$

$$\begin{array}{l} \mathcal{B}\forall[\langle[<\forall>(it, [\mathit{inf}, \mathit{sup}], \Omega^B)|\Omega], S, H\rangle_c|\Sigma] \\ \rightsquigarrow_r^{\exists\forall} \quad \mathcal{B}[\langle\Omega, S, H\rangle_c|\Sigma] \end{array}$$

La règle de transition $[\forall - \mathit{Success}]$ s'applique si $\mathit{inf} > \mathit{sup}$ ce qui signifie que tous les *token* $<\forall\Omega^B>$ développés ont réussi. La règle de transition $[\forall - \mathit{Success}]$ ferme le niveau actuel du lieur. Comme le *token* $<\forall\Omega^B>$ a réussi, le Ω restant doit être poursuivi au niveau précédent de $\mathcal{B}$.

La règle de transition $[\forall - \mathit{FailurePropagate}]$ **:**

$$\mathcal{B}\forall[\mathit{Failure}, \sigma|\Sigma] \rightsquigarrow_r^{\exists\forall} \mathcal{B}[\mathit{Failure}|\Sigma]$$

La règle de transition $[\forall - \mathit{FailurePropagate}]$ s'applique lorsqu'un Ω^B développé a échoué. La règle de transition $[\forall - \mathit{FailurePropagate}]$ ferme et rejette le niveau actuel du lieur. En rejetant le niveau actuel, le *token* $<\forall\Omega^B>$ et le Ω restant sont également supprimés, puis un *Failure* est placé à la place de l'appel de la contrainte au niveau supérieure précédent de $\mathcal{B}$.

La règle de transition $[\forall - \mathit{SuccessPropagate}]$ **:**

$$\mathcal{B}\forall[\mathit{Success}|\Sigma] \rightsquigarrow_r^{\exists\forall} \mathcal{B}\forall\Sigma$$

La règle de transition $[\forall - \mathit{SuccessPropagate}]$, lorsqu'un corps Ω^B déplié est un succès (c'est-à-dire l'état d'exécution $\langle[it \leftarrow \mathit{inf}](\Omega^B), S, H\rangle_c$ s'est réécrit en l'état d'exécution *Success*), poursuit le parcours des éléments de la quantification universelle (car alors $\Sigma = [\langle[<\forall>(it, [\mathit{inf} + 1, \mathit{sup}], \Omega^B)|\Omega], S, H\rangle_c \mid _])$.

3.6 Le but est une conjonction

La règle de transition $[\mathit{Success}]$ **:**

$$\mathcal{B}[\langle[], S, H\rangle_c|\Sigma] \rightsquigarrow_r^{\exists\forall} \mathcal{B}[\mathit{Success}|\Sigma]$$

La règle de transition $[\mathit{Success}]$ introduit l'état d'exécution *Success* lorsqu'une suite d'exécutions a réussi. Un tel cas se produit lorsque le but de l'état d'exécution devient vide (c'est-à-dire que tous les éléments ont été traités avec succès).

La règle de transition $[\mathit{Failure}]$ **:**

$$\mathcal{B}[\langle[\mathit{false}|\Omega], S, H\rangle_c|\Sigma] \rightsquigarrow_r^{\exists\forall} \mathcal{B}[\mathit{Failure}|\Sigma]$$

La règle de transition $[\mathit{Failure}]$ remplace une continuation d'exécution qui a échouée par l'état d'exécution *Failure*. Un état d'exécution échoue uniquement si une contrainte $false$ est rencontrée lors du traitement du but.

La règle de transition $[\mathit{true}]$ **:**

$$\mathcal{B}[\langle[\mathit{true}|\Omega], S, H\rangle_c|\Sigma] \rightsquigarrow_r^{\exists\forall} \mathcal{B}[\langle\Omega, S, H\rangle_c|\Sigma]$$

La plupart des règles de transition sont illustrées dans l'exemple suivant, qui applique la sémantique $\omega_r^{\exists\forall}$ au jeu du NimFibo décrit en section 2.

Exemple 3.1

$$\begin{array}{rcl} u & @ & _\backslash \mathit{nfB}(N,R) <\forall> \{it, 1, \mathit{min}(N,R)\} \\ & & \mathit{nfA}(2*it, R-it) \\ e & @ & _\backslash \mathit{nfA}(N,R) <\exists> \{it, 1, \mathit{min}(N,R)\} \\ & & \mathit{nfB}(2*it, R-it) \\ \mathit{init} & @ & _\backslash \mathit{nim_fibo}(R) \Leftrightarrow \mathit{nfA}(R-1, R) \end{array}$$

$$[\ \langle[\mathit{nim_fibo}(4)], \emptyset, \emptyset\rangle_1] \quad (0)$$
$$\rightsquigarrow_r^{\exists\forall} [\mathit{Activate}]$$
$$[\ \langle[\mathit{nim_fibo}(4)\#1 : 1], \{\mathit{nim_fibo}(4)\#1\}, \emptyset\rangle_2] \quad (1)$$
$$\rightsquigarrow_r^{\exists\forall} [\mathit{Simplify}]$$
$$[\ \langle[\mathit{nfA}(3,4)], \emptyset, \emptyset\rangle_2] \quad (2)$$
$$\rightsquigarrow_r^{\exists\forall} [\mathit{Activate}] \rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Simplify}]$$
$$\exists\ [\ \langle[<\exists>(i, [1,3], \mathit{nfB}(2*i, 4-i))], \emptyset, \emptyset\rangle_3] \quad (4)$$
$$\rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Unfolding}]$$
$$\exists\ [\ \langle[\mathit{nfB}(2,3)], \emptyset, \emptyset\rangle_3, \langle[<\exists>(i, [2,3], \mathit{nfB}(2*i, 4-i))], \emptyset, \emptyset\rangle_3] \quad (5)$$

$$\mathit{avec}\ \sigma_1^{\exists} = \langle[<\exists>(i, [2,3], \mathit{nfB}(2*i, 4-i))], \emptyset, \emptyset\rangle_3$$

$$\rightsquigarrow_r^{\exists\forall} [\mathit{Activate}] \rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Simplify}]$$
$$\exists\forall\ [\ \langle[<\forall>(i, [1,2], \mathit{nfA}(2*i, 3-i))], \emptyset, \emptyset\rangle_4, \sigma_1^{\exists}] \quad (7)$$
$$\rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Unfolding}]$$
$$\exists\forall\ [\ \langle[\mathit{nfA}(2,2)], \emptyset, \emptyset\rangle_4, \langle[<\forall>(i, [2,2], \mathit{nfA}(2*i, 3-i))], \emptyset, \emptyset\rangle_4, \sigma_1^{\exists}] \quad (8)$$

$$\mathit{avec} \begin{cases} \sigma_{1.1}^{\forall} = \langle[<\forall>(i, [2,2], \mathit{nfA}(2*i, 3-i))], \emptyset, \emptyset\rangle_4 \\ \sigma_{1.1.1}^{\exists} = \langle[<\exists>(i, [2,2], \mathit{nfB}(2*i, 2-i))], \emptyset, \emptyset\rangle_5 \\ \sigma_{1.1.1.1}^{\forall} = \langle[<\forall>(i, [2,1], \mathit{nfA}(2*i, 1-i))], \emptyset, \emptyset\rangle_6 \end{cases}$$

$$\rightsquigarrow_r^{\exists\forall} [\mathit{Activate}] \rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Simplify}] \rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Unfolding}]$$
$$\rightsquigarrow_r^{\exists\forall} [\mathit{Activate}] \rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Simplify}] \rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Unfolding}]$$
$$\rightsquigarrow_r^{\exists\forall} [\mathit{Activate}] \rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Simplify}]$$
$$\rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Failure}]$$

$$\exists\forall\exists\forall\ [\ \mathit{Failure}, \sigma_{1.1.1.1}^{\forall}, \sigma_{1.1.1}^{\exists}, \sigma_{1.1}^{\forall}, \sigma_1^{\exists}] \quad (17)$$
$$\rightsquigarrow_r^{\exists\forall} [\forall - \mathit{FailurePropagate}]$$
$$\exists\forall\exists\ [\ \mathit{Failure}, \langle[<\exists>(i, [2,2], \mathit{nfB}(2*i, 2-i))], \emptyset, \emptyset\rangle_5, \sigma_{1.1}^{\forall}, \sigma_1^{\exists}] \quad (18)$$
$$\rightsquigarrow_r^{\exists\forall} [\exists - \mathit{FailurePropagate}]$$
$$\exists\forall\exists\ [\ \langle[<\exists>(i, [2,2], \mathit{nfB}(2*i, 2-i))], \emptyset, \emptyset\rangle_5, \sigma_{1.1}^{\forall}, \sigma_1^{\exists}] \quad (19)$$
$$\rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Unfolding}]$$
$$\exists\forall\exists\ [\ \langle[\mathit{nfB}(4,0)], \emptyset, \emptyset\rangle_5, \langle[<\exists>(i, [3,2], \mathit{nfB}(2*i, 2-i))], \emptyset, \emptyset\rangle_5, \sigma_{1.1}^{\forall}, \sigma_1^{\exists}] \quad (20)$$

$$\mathit{avec} \begin{cases} \sigma_{1.1.2}^{\exists} = \langle[<\exists>(i, [3,2], \mathit{nfB}(2*i, 2-i))], \emptyset, \emptyset\rangle_5 \\ \sigma_{1.2}^{\forall} = \langle[<\forall>(i, [3,2], \mathit{nfA}(2*i, 3-i))], \emptyset, \emptyset\rangle_4 \\ \sigma_{1.2.1}^{\exists} = \langle[<\exists>(i, [2,1], \mathit{nfB}(2*i, 1-i))], \emptyset, \emptyset\rangle_5 \end{cases}$$

$$\rightsquigarrow_r^{\exists\forall} [\mathit{Activate}] \rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Simplify}] \rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Success}]$$
$$\exists\forall\exists\ [\ \langle[], \emptyset, \emptyset\rangle_6, \sigma_{1.1.2}^{\exists}, \sigma_{1.1}^{\forall}, \sigma_1^{\exists}] \quad (23)$$
$$\rightsquigarrow_r^{\exists\forall} [\mathit{Success}]$$
$$\exists\forall\exists\ [\ \mathit{Success}, \langle[<\exists>(i, [3,2], \mathit{nfB}(2*i, 2-i))], \emptyset, \emptyset\rangle_5, \sigma_{1.1}^{\forall}, \sigma_1^{\exists}] \quad (24)$$
$$\rightsquigarrow_r^{\exists\forall} [\exists - \mathit{SuccessPropagate}]$$
$$\exists\forall\ [\ \langle[], \emptyset, \emptyset\rangle_5, \sigma_{1.1}^{\forall}, \sigma_1^{\exists}] \quad (25)$$
$$\rightsquigarrow_r^{\exists\forall} [\mathit{Success}] \rightsquigarrow_r^{\exists\forall} [\forall - \mathit{SuccessPropagate}]$$
$$\exists\forall\ [\ \langle[<\forall>(i, [2,2], \mathit{nfA}(2*i, 3-i))], \emptyset, \emptyset\rangle_4, \sigma_1^{\exists}] \quad (27)$$

$$\begin{array}{l}
\rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Unfolding}] \rightsquigarrow_r^{\exists\forall} [\mathit{Activate}] \rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Simplify}] \\
\rightsquigarrow_r^{\exists\forall} [\exists - \mathit{Unfolding}] \rightsquigarrow_r^{\exists\forall} [\mathit{Activate}] \rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Simplify}] \\
\rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Success}] \rightsquigarrow_r^{\exists\forall} [\mathit{Success}] \\
\rightsquigarrow_r^{\exists\forall} [\exists - \mathit{SuccessPropagate}] \rightsquigarrow_r^{\exists\forall} [\mathit{Success}] \\
\rightsquigarrow_r^{\exists\forall} [\forall - \mathit{SuccessPropagate}] \rightsquigarrow_r^{\exists\forall} [\forall - \mathit{Success}] \\
\rightsquigarrow_r^{\exists\forall} [\mathit{Success}] \rightsquigarrow_r^{\exists\forall} [\exists - \mathit{SuccessPropagate}] \\
\rightsquigarrow_r^{\exists\forall} [\mathit{Success}]
\end{array}$$

$$[\quad \mathit{Success}] \quad (41)$$

Le théorème suivant établit la correction et la complétude de la sémantique $\omega_r^{\exists\forall}$ par rapport à la sémantique $\omega^{\exists\forall}$ ($\overset{*}{\rightsquigarrow}_r^{\exists\forall}$ représente la fermeture transitive de la relation $\rightsquigarrow_r^{\exists\forall}$).

La sémantique $\omega^{\exists\forall}$ de QCHR est définie par un calcul des séquents qui opère sur des séquents ($P \blacktriangleright \Omega \blacktriangleleft S \vdash S'$) où P est un programme QCHR, Ω est un but, S est le *store* avant la résolution du but et S' est le but à l'issue de la résolution (voir [4] pour une description complète du système de calcul des séquents).

Théorème 3.1 (Correction et complétude de $\omega_r^{\exists\forall}$)
Correction et complétude de la sémantique $\omega_r^{\exists\forall}$ par rapport à la sémantique $\omega^{\exists\forall}$.

Soit P un programme QCHR. Il existe une preuve dans $\omega^{\exists\forall}$ pour le séquent ($P \blacktriangleright \Omega \blacktriangleleft S \vdash S'$) si et seulement si il existe une exécution dans $\omega_r^{\exists\forall}$ telle que $[\langle \Omega, S, \emptyset \rangle_1] \overset{*}{\rightsquigarrow}^{\exists\forall} [\langle \emptyset, S', H \rangle_c]$.

4 Discussion

Tous les solveurs QCSP de l'état de l'art ont le même inconvénient : ils explorent des espaces combinatoires beaucoup plus grands que l'espace de recherche naturel du problème original. Dans [3], la signification de la notion du "talon d'Achille", initialement introduite pour les formules booléennes quantifiées (QBF) [9, 24, 27, 28] dans [2] correspond à la difficulté de détecter que les contraintes booléennes sont nécessairement vraies sous une certaine affectation partielle. Cette notion a été étendue aux QCSP, mais sa portée a été étendue. Il s'agit désormais de déterminer comment éviter l'exploration d'espaces combinatoires dont on sait qu'ils sont inutiles par construction. Cette définition inclut la capture des actions illégales du joueur B mais aussi la fin du jeu avant le dernier tour, évitant ainsi d'explorer un espace de recherche sur-dimensionné. Déjà cité, [5, 26] propose le nouveau langage *QCSP+* qui utilise des séquences de quantification restreintes au lieu des séquences de quantification standard. [3] introduit un nouvel outil, inspiré de la règle de coupe de Prolog comme outil sous la responsabilité du concepteur du QCSP, pour élaguer les parties de l'espace de recherche qui sont par construction connues pour être inutiles. D'autres approches proposent de modifier plus ou moins le langage QCSP/QCSP+ pour surmonter les différents inconvénients : [6] propose le nouveau langage *Strategic CSP* où les variables universelles adaptent leur domaine pour être compatibles avec les choix précédents ; [23] propose également un nouveau langage mais restreint aux CSP de jeux markoviens, basé explicitement sur la notion d'état, afin de modéliser et de résoudre efficacement les problèmes de contrôle de systèmes dynamiques complètement observables et markoviens. Ce langage est facilement et efficacement exprimé en QCHR. La sémantique opérationnelle $\omega_r^{\exists\forall}$ pour les QCHR présentée dans cet article a été implémentée dans ^{Q}CHR++, un solveur QCHR basé sur le solveur CHR++ [3]. Ce dernier ayant été étendu par l'ajout de deux nouveaux éléments à sa grammaire : `exists` et `forall`. Les résultats obtenus sur trois jeux connus (le jeu de la Matrice, le jeu de Nim (version Fibonacci) et le jeu du Puissance 4) sont présentés et commentés dans [4]. Il en ressort qu'un problème modélisé en QCHR est plus léger et plus compréhensible qu'un modèle QCSP. De plus, les performances du solveur ^{Q}CHR++ sont équivalentes à celles du solveur QCSP voire supérieures de plusieurs ordres de grandeur lorsque le lieur peut être construit dynamiquement.

5 Conclusion

Cet article reprend le formalisme QCHR, une extension de CHR avec quantification, et propose une sémantique opérationnelle décrivant de manière fine l'exécution d'un programme QCHR. QCHR permet de modéliser des lieurs dynamiques qui se développent au fur et à mesure de l'exploration de l'arbre de recherche. Ceci permet de surmonter l'un des principaux inconvénients du cadre QCSP. Nous avons d'abord donné quelques modèles QCHR très intuitifs démontrant l'intérêt d'une modélisation dynamique où le lieur se développe au fur et à mesure de l'exploration de l'arbre de recherche. Nous avons ensuite présenté une sémantique opérationnelle basée sur des états d'exécution ainsi que les règles de transition permettant sa mise en œuvre. Nous avons enfin montré la correction et la complétude de notre sémantique opérationnelle par rapport à la sémantique formelle présentée dans [4].

Nous pensons que ce nouveau formalisme permet de modéliser plus facilement les problèmes contraints avec quantification et offre une nouvelle approche lorsque le lieur ne peut être défini statiquement. Dans le futur, nous prévoyons de nous attaquer à des problèmes qui ne peuvent pas être traités avec QCSP tels que les problèmes où un lieur statique ne peut pas être trouvé.

Références

[1] Abdennadher, S. et H. Schütz: *CHR : A Flexible Query Language*. Dans *Proceedings of the 3rd Internatio-*

3. CHR++ est un solveur CHR construit au dessus du langage C++, il peut être téléchargé à l'adresse `https://gitlab.com/vynce/chrpp`

nal Conference on Flexible Query Answering Systems, pages 1–14, 1998.

[2] Ansotegui, C., C. Gomes et B. Selman: *Achilles' Heel of QBF*. Dans *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*, pages 275–281, 2005.

[3] Barichard, V. et I. Stéphan: *The cut tool for QCSP*. Dans *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'14)*, pages 883–890, 2014.

[4] Barichard, V. et I. Stéphan: *Quantified Constraint Handling Rules*. Dans *Proceedings 35th International Conference on Logic Programming (ICLP'19)*, tome 306 de *EPTCS*, pages 210–223, 2019.

[5] Benedetti, M., A. Lallouet et J. Vautard: *QCSP made Practical by virtue of Restricted Quantification*. Dans *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 38–43, 2007.

[6] Bessiere, C. et G. Verger: *Strategic constraint satisfaction problems*. Dans *Proceedings of the Workshop on Modelling and Reformulation*, 2006.

[7] Betz, H. et T.W. Frühwirth: *Linear-Logic Based Analysis of Constraint Handling Rules with Disjunction*. ACM Transactions on Computational Logic, 14(1) :37, 2013.

[8] Bordeaux, L. et E. Monfroy: *Beyond NP : Arc-Consistency for Quantified Constraints*. Dans *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP'02)*, pages 371–386, 2002.

[9] Cadoli, M., A. Giovanardi et M. Schaerf: *An Algorithm to Evaluate Quantified Boolean Formulae*. Dans *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, pages 262–267, 1998.

[10] Duck, G.J., P.J. Stuckey, M.G. de la Banda et C. Holzbaur: *The Refined Operational Semantics of Constraint Handling Rules*. Dans *Proceedings of the 20th International Conference on Logic Programming (ICLP'04)*, pages 90–104, 2004.

[11] Frühwirth, T.W.: *Constraint Handling Rules*. rapport technique, ECRC, 1992.

[12] Frühwirth, T.W.: *Constraint Handling Rules*. Dans *Constraint Programming : Basics and Trends*, pages 90–107, 1994.

[13] Frühwirth, T.W.: *Constraint Handling Rules*. Cambridge University Press, 2009.

[14] Frühwirth, T.W. et S. Abdennadher: *Essentials of Constraint Programming*. Springer-Verlag, 2003.

[15] Frühwirth, T.W. et F. Raiser (rédacteurs): *Constraint Handling Rules : Compilation, Execution, and Analysis*. March 2011.

[16] Frühwirth, T.W.: *Theory and Practice of Constraint Handling Rules*. Journal of Logic Programming, 37(1-3) :95–138, 1998.

[17] Gent, I.P, P. Nightingale, A. Rowley et K. Stergiou: *Solving quantified constraint satisfaction problems*. Artificial Intelligence, 172(6-7) :738–771, 2008.

[18] Hentenryck, P. Van: *Constraint logic programming*. Knowledge Engineering Review, 6(3) :151–194, 1991.

[19] Jaffar, J. et J. L. Lassez: *Constraint Logic Programming*. Dans *Proceedings of the 14th Annual ACM Symposium on Principles of Programming Languages*, pages 111–119, 1987.

[20] Jaffar, J. et M.J. Maher: *Constraint Logic Programming : A Survey*. Journal of Logic Programming, 19/20 :503–581, 1994.

[21] Mamoulis, N. et K. Stergiou: *Algorithms for Quantified Constraint Satisfaction Problems*. Dans *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP'04)*, pages 752–756, 2004.

[22] Nightingale, P.: *Consistency and the Quantified Constraint Satisfaction Problem, PhD thesis, , University of St Andrews*, 2007.

[23] Pralet, C. et G. Verfaillie: *Beyond QCSP for Solving Control Problems*. Dans *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11)*, pages 744–758, 2011.

[24] Rabe, M. et L. Tentrup: *CAQE : A Certifying QBF Solver*. Dans *Formal Methods in Computer-Aided Design (FMCAD'15)*, pages 136–143, 2015.

[25] Verger, G. et C. Bessiere: *BlockSolve : a Bottom-Up Approach for Solving Quantified CSPs*. Dans *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP'06)*, pages 635–649, 2006.

[26] Verger, G. et C. Bessiere: *Guiding Search in QCSP$^+$ with Back-Propagation*. Dans *Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming (CP'08)*, pages 175–189, 2008.

[27] Zhang, L.: *Solving QBF with Combined Conjunctive and Disjunctive Normal Form*. Dans *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI'06)*, 2006.

[28] Zhang, L. et S. Malik: *Conflict Driven Learning in a Quantified Boolean Satisfiability Solver*. Dans *Proceedings of the International Conference on Computer Aided Design (ICCAD'02)*, pages 442–449, 2002.

Session 6 : Représentation logique

Modelling a faillibilistic and perspectivistic reasoning

Arnaud Kohler
arnaud.kohler@pacariane.com

Abstract

It is commonly accepted that propositional logic is insufficient to capture human reasoning. To extend its capabilities, we propose to automatically integrate silent propositions into the set of atomic propositions. We call them thoughts. They are used to define a semantic interpretation function on models. Our contribution to the family of non-monotonic formalisms is to formalise a faillibilistic and perspectivistic reasoning. We illustrate the interest of these properties by developing an example of application.

1 Introduction

Formalisms based on propositional logic L_p classically focus on modelling knowledge that is deemed to be true or false. It is insufficient to model the full diversity and complexity of human reasoning, which obviously also exploits inconsistent or uncertain information. Another difficulty is to move from a monotonic semantic interpretation (what is supposed to be true given a state of knowledge remains true even if a new piece of knowledge appears) to a non-monotonic semantic interpretation (what is supposed to be true given a state of knowledge can become false if a new piece of knowledge appears).

Many propositions have been presented to address these needs: epistemic modal logics, paraconsistent logics, default logic, intuitionistic logic, adaptive logics, or multivalued logics for example. They each manage to capture different properties. But they have not succeeded in modelling the many modes of reasoning empirically observed in humans (D. Andler [2]).

These formalisms most often approach human reasoning through the principle of proof inherited from mathematics. In this article, we propose the contextual logic L_c, which uses the principle of non-refutability. To achieve this, L_c models a faillibilistic reasoning:

> *Each piece of knowledge is uncertain, and our belief is constructed by identifying those that are justifiable.*

We will find that the exercise leads, mechanically, to a perspectivist reasoning:

> *Belief is constructed by aggregating some different justifiable points of view, accepting the possibility that they are incomplete, incorrect, or inconsistent with each other.*

L_c remains within the monotonic syntax of propositional logic L_p and enrich the set of atomic propositions by *silent* propositions. We call them thoughts. Integrated and consumed automatically, they identify the formulae belonging to the set of knowledge. We use them to define a non-monotonic semantic interpretation function based on the analysis of their behaviour in models of the theory.

We first present the propositional logic to share the vocabulary we use and the associated definitions. The supposed limits of the syntax of L_p are recalled. The principles of L_c and its main properties are described. The assumed limitations of the propositional logic are then revisited. To illustrate our point, we conclude our presentation by developing an example of application of L_c. It calls upon a sufficiently broad knowledge base to demonstrate, through a practical case, the non-monotonic expressiveness of the language and to give meaning to the various technical examples used throughout the text.

2 The propositional logic

In this article, we refer to several formal languages. We do not detail them in general so as not to make the presentation unnecessarily heavy, inviting the reader to refer to the many documents available on these formalisms. However, we think it is useful to pause for a moment on the propositional logic L_p.

This paragraph does not contain anything new. There are many equivalent ways of modeling a formal language. Here we share the definitions of the vocabulary and symbols that we use out of habit.

<u>The syntax of L_p</u>

The language of the propositional logic L_p is composed of the set P_{Lp} of atomic propositions, the negation connector $\neg$, the implication connector $\rightarrow$, and the parenthesis symbols, which are used according to classical mathematical rules. The rules for forming a **well-formed formula** are:

- any atomic proposition is a well-formed formula,
- if f and g are well-formed formulae, then the expressions (f), $\neg f$ and $f \rightarrow g$ are well-formed formulae,
- a well-formed formula is obtained only by applying the two precedent rules a finite number of times.

Let f, g and h be some well-formed formulae. The following formulae are some **axioms**:

- $f \rightarrow (g \rightarrow f)$
- $(f \rightarrow (g \rightarrow h)) \rightarrow ((f \rightarrow g) \rightarrow (f \rightarrow h))$
- $(\neg f \rightarrow \neg g) \rightarrow ((\neg f \rightarrow g) \rightarrow f)$

This axiom scheme is sufficient to cover all the axioms of L_p. For example, $a \rightarrow (a \rightarrow a)$ is an axiom obtained by applying the first formula for f is a and g is a. $f \rightarrow f$ is another axiom, which can be demonstrated with this scheme and the **theorem formation rules**:

- any axiom is a theorem,
- let f and g be two well-formed formulae. If f and $f \rightarrow g$ are theorems, then g is a theorem (this rule is called the *modus ponens*),
- a theorem can only be obtained by applying the two previous rules a finite number of times.

The statement f is a theorem is denoted $\vdash_{Lp} f$. A **theory** E_{Lp} is a set of well-formed formulae. The formulae $f \in E_{Lp}$ represent the **hypotheses** of E_{Lp}. A formula f is said **to be provable** in E_{Lp} if, and only if, it can be produced from E_{Lp} by applying the theorem formation rules, for all hypotheses of E_{Lp} behaving as theorems. In this case, f is said a theorem of E_{Lp}, and this is denoted $E_{Lp} \vdash_{Lp} f$:

A theory is said to be **inconsistent** if it produces the negation of a theorem. Otherwise, it is said to be **consistent**.

The semantic of L_p

Classically, the logician's attitude is to consider in formal language only mathematical symbols:

"*A formal language is, by definition, a language with only syntax and no semantics*" – (translation) J. Hebenstreit, Enclycopedia Universalis

What is relevant is the study of the mechanisms and laws of reasoning, modelled by syntactic rules. Any reference to semantic content is discarded. However, it is possible to attribute a meaning to connectors if it is strictly symbolic and univocal.

Let E_{Lp} be a theory of L_p, and f and g two well-formed formulae. The **syntactic interpretation function** of L_p is defined by a function I_{Lp} such that:

- $I_{Lp}(E_{Lp}, f) = true$ or $I_{Lp}(E_{Lp}, f) = false$,
- If f is a hypothesis of E_{Lp}, then $I_{Lp}(E_{Lp}, f) = true$.

The meaning of the connectors is then defined by:

- $I_{Lp}(E_{Lp}, \neg f) = true$ if, and only if, $I_{Lp}(E_{Lp}, f) = false$,
- $I_{Lp}(E_{Lp}, f \rightarrow g) = true$ if, and only if, $I_{Lp}(E_{Lp}, f) = false$ or $I_{Lp}(E_{Lp}, g) = true$.

The symbol $\vDash_{Lp}$ is defined by:

$$E_{Lp} \vDash_{Lp} f \text{ if, and only if, } I_{Lp}(E_{Lp}, f) = true$$

The syntactic interpretation of an axiom is always *true*, and L_p is correct and complete: everything that is produced (using $\vdash_{Lp}$) is *true* (according to $\vDash_{Lp}$), and everything that is *true* is produced: $E_{Lp} \vDash_{Lp} f$ if, and only if, $E_{Lp} \vdash_{Lp} f$.

If the constraints of a theory do not allow the *true* or *false* truth value of a formula to be calculated, it is said to have an *unknown* value for this theory – for example, the value of $I_{Lp}(\{a \rightarrow b\}, a)$: a can be *true* (in this case, b is *true*) or *false* (in this case, b can be *true* or *false*).

This remark introduces the definition of **model**. A model of a theory E_{Lp} is obtained by associating to each atomic proposition only one truth value (*true* or exclusively *false*) such that the result verifies I_{Lp}. E_{Lp} is consistent if it has at least one model. It is inconsistent otherwise.

For example, the theory $\{a \rightarrow b, c\}$ is verified by three models:

$$\{(a, true), (b, true), (c, true)\}$$
$$\{(a, false), (b, true), (c, true)\}$$
$$\{(a, false), (b, false), (c, true)\}$$

So, it is consistent. As a counter example, $\{a, \neg a\}$ does not accept a model: if a is assumed to be *true*, $\neg a$ is not verified - and *vice versa*. a is true and false. It is inconsistent.

We end this paragraph by presenting three often used connectors. To simplify the expression of formulae, the language is extended to disjunction (denoted $\vee$), conjunction (denoted $\wedge$) and equivalence (denoted $\leftrightarrow$) connectors. For f and g two well-formed formulae, they are defined by:

- $f \vee g$ is equivalent to $(\neg f \rightarrow g)$,
- $f \wedge g$ is equivalent to $\neg (f \rightarrow \neg g)$,
- $f \leftrightarrow g$ is equivalent to $(f \rightarrow g) \wedge (g \rightarrow f)$.

A **literal** is an atomic proposition or the negation of an atomic proposition. A **clause** is a disjunction of literals. A formula is said to be in **conjunctive normal form** if it is a conjunction of clauses. Any well-formed formula admits a logically equivalent rewriting in conjunctive normal form (A. Thayse, [17]). For example, a conjunctive normal form of the formula $((\neg f \rightarrow g) \rightarrow h)$ is $((\neg f \vee h) \wedge (\neg g \vee h))$. P. Siegel [16] proposes a linear complexity process that rewrites any well-formed formula into a conjunctive normal form.

3 The limits of L_p

Modelling human reasoning with L_p faces several difficulties. This article focuses on three of them.

1) A difficulty is to move from a monotonic semantic interpretation to a **non-monotonic** semantic interpretation. This topic is covered in paragraph 5.

2) Human reasoning sometimes seems incoherent. But whatever f and g two well-formed formulae of L_p, $\{f, \neg f\} \vdash_{Lp} g$. This is the **explosion principle**. It forbids the appearance of a syntactic inconsistency in a theory. This topic is covered in paragraph 6.

3) Human reasoning uses **semantic links between propositions**. But the symmetrical behaviour of connectors prohibits this type of modelling in L_p. Put more explicitly with an example, $f \rightarrow (g \rightarrow h)$ is syntactically equivalent to $g \rightarrow (f \rightarrow h)$: f and g have the same behaviour in the formula, and it is not possible to model a privileged relationship between one of them and h. This topic is covered in paragraph 7.

Two other difficulties are the modelling of induction and abduction, and the computational complexity of the algorithms. We do not address them in this article.

The successive failures of logic researchers to solve these problems have led many to conclude that the modelling of human reasoning probably escapes L_p, and logic formalisms more generally (D. Andler [2]).

4 The contextual logic

Let us consider a thought. We perceive it in the sense defined by R. Descartes [4]:

"*By the name of thought, I understand all that is so much in us that we are immediately aware of it*" (translation)

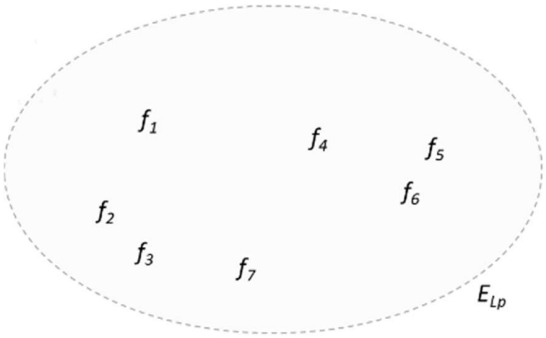
and we describe it with a set of sentences. For example, we can have the thought of a painted picture. Consider the Mona Lisa, by Leonardo da Vinci. We can describe it with some sentences. But even if this description would be ideally complete and perfect, we are immediately aware that it is not the thought that we have of the Mona Lisa.

We model this observation by distinguishing two notions in the syntax of the language: a unit sign c, which symbolises a thought, and a combination of signs f, which reproduces the sentences that describe it. This leads to the need to define a relationship between c and f. To this end, we consider the following postulate [8]:

Contextual postulate *Let L be a formal language with the functions of syntactic production* $\vdash_L$ *and of syntactic interpretation* $\vDash_L$. *A well-formed formula f of L is a set of signs that has no meaning. Its meaning is carried by a thought, which is an atomic proposition of L "which is not pronounced". For c symbolising this thought, the relation between c and f is* $c \vDash_L f$.

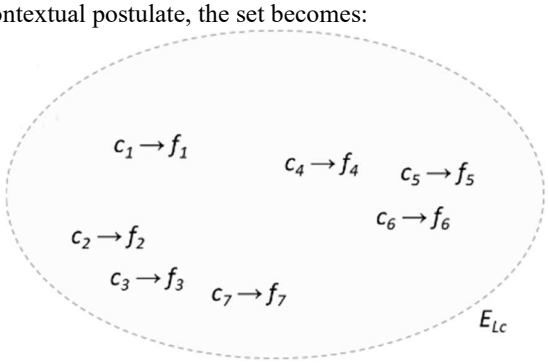
The expression $c \vDash_L f$ asserts neither the thought c nor the sentence f. It models that the sentence f expresses the thought c. c is an atomic proposition which respects the syntactic properties of L.

For example, consider seven sentences $f_1, f_2, \ldots, f_7$ of L_p such that:



Pic. 1 - A theory of L_p

E_{Lp} admits a syntactic behaviour, but it has no semantic meaning according to contextual postulate. To overcome this, we need to consider the thoughts $c_1, c_2, \ldots, c_7$. $c \vDash_L f$ is equivalent to $\vDash_L c \rightarrow f$ if L is L_p. So, after applying contextual postulate, the set becomes:



Pic. 2 - A theory of L_c

E_{Lc} is a theory of L_p. It models: "*each thought* c_i *is expressed by a sentence* f_i". We thus agree with L. Wittgenstein when he states [18]:

> "*We should not say: The complex sign aRb says that a is in the relation R with b, but: That a is in a certain relation R with b says that aRb*" (translation)

The application of contextual postulate to a formalism L produces the contextualised logic L. By language convention, we call contextual logic, denoted L_c, the contextualised propositional logic.

Each contextual formula takes a form $c \rightarrow f$, for c a thought and f a well-formed formula in the sense of L_p. Expressions in L_c accept a pre-order:

- an atomic proposition that is not a thought is of rank 0,
- a thought is of rank 1 or higher. We will see later that we propose to automatically handle the assignment of a rank to a thought,
- the rank of a well-formed formula in the sense of L_p is equal to the maximum rank of the atomic propositions (including thoughts) that compose it.

We define a well-formed formula in the sense of L_c to be a formula $c \rightarrow f$, for c a thought of rank i and f a well-formed formula in the sense of L_p of rank j, such that $i > j$.

This defines a syntactical restriction. Without these notions of rank and well-formed formula in the sense of L_c, thoughts would have a reflexive property that is not easy to understand: it could be expressed by a sentence containing it: c could participate in f in $c \rightarrow f$.

Knowledge reflexivity is a concept defended by J. Pitrat [12], but we don't master it. The restricted language we propose remains sufficient to cover the level of expressiveness that we want to achieve in this article. However, we do not know if this restriction is necessary.

Given the syntax $c \rightarrow f$ of the contextual formulae, the set $\{(c_i, false), c_i$ are the thoughts$\}$ characterises some models that verify any contextual theory. For example, $\{(c_1, false), (c_2, false), \ldots, (c_7, false)\}$ characterises some models that verify $\{c_1 \rightarrow f_1, c_2 \rightarrow f_2, \ldots, c_7 \rightarrow f_7\}$. The first consequence is that each contextual theory admits at least one model. So, it is always consistent.

The second consequence is that any thought is possibly false. In L_c, uncertainty is intrinsically embedded in the syntax. To remedy this problem, we adopt the following principles:

- because any formula can be false, we cannot interrogate a contextual theory with a question such as "Is f true (or false)?". But we can say: "What can I conclude if I suppose that f is true (or false)?".

Notation *f is an event (a question, a fact, a new thought, etc.) which generates a need for semantic interpretation. We call this a* ***stimulus****.*

- because every thought is possibly false, we propose to relativize the semantic interpretation to the subsets of thoughts identified as *the most relevant*. We cannot conclude that f is true or false, but we can say "f is true (or false) with respect to the most relevant sets of thoughts".

For example, let a and b be two atomic propositions of L_p, and c_1, c_2 and c_3 be three thoughts. Consider the following set:

$$E_{Lc} = \{c_1 \rightarrow a, c_2 \rightarrow \neg a, c_3 \rightarrow b\}$$

We cannot prove that a or b is true or false. But we can say that a is true considering $\{c_1\}$, or that a is false and b is true considering $\{c_2, c_3\}$, etc. There are many possible combinations, so we should define a method for selecting "the most relevant sets of thoughts". For this purpose, we need some definitions.

Definitions *Let E_{Lc} be a theory of L_c and i and j be 2 integers such that $0 < i \leq j$.*

- *A set of thoughts is called a **context**.*
- *A context is said to be **of rank i to j** if all the thoughts in it are of rank i to j. A context of rank i to i is said of rank i.*
- *A context that is verified by at least one model of E_{Lc} is called a **possible** (or a consistent) **context**.*
- *A context that does not check any model of E_{Lc} is called an **impossible** (or an inconsistent) **context**.*
- *An impossible context is called a **minimal impossible context** if each of its strict subsets is possible.*
- *A possible context that has no strict extension that checks E_{Lc} is called a **maximal context**.*
- *A possible context is called the **credible context** if it has no intersection with a minimal impossible context and if all its strict extensions have an intersection with a minimal impossible context.*

In the following, and in accordance with common practice, we invariably use the notions of conjunction of formulae (for example: $c_1 \wedge c_2$) or of set of formulae (for example: $\{c_1, c_2\}$) to designate the same object. A conjunction of thoughts also means a context.

Example *Let a, b and c be three atomic propositions of L_p, and c_1, c_2, c_3, c_4 and c_5 be five thoughts. Consider the following set:*

$$E_{Lc} = \{c_1 \rightarrow a, c_2 \rightarrow \neg a, c_3 \rightarrow b, c_4 \rightarrow \neg b, c_5 \rightarrow c\}$$

$c_1 \wedge c_2$ and $c_3 \wedge c_4$ are the only two minimal impossible contexts. So, c_5 is the credible context, and there are four maximal contexts: $\{c_1, c_3, c_5\}$, $\{c_1, c_4, c_5\}$, $\{c_2, c_3, c_5\}$ and $\{c_2, c_4, c_5\}$.

We see that, for a given theory, there are possibly several maximal contexts (potentially empty) and a single credible context (potentially empty). They are obtained by calculating the minimal impossible contexts in a first step. The different possible combinations of thoughts then produce them.

We use these definitions to define the function that identifies the contexts considered most relevant for semantic interpretation.

Definition *Let E_{Lc} be a theory of L_c, S be a stimulus, i and j be two integers such that $0 < i < j$, and k be the maximal rank of the thoughts of E_{Lc}. The relevant contexts are defined as follows:*

- *calculation of the **maximal epistemic contexts**: if $k < j$ then there is an empty maximal epistemic context, else calculation on $\{E_{Lc}, S\}$ of the maximal contexts of rank j to k,*
- *then enrichment of each maximal epistemic context C, by the credible context of rank i to j-1 on $\{E_{Lc}, S, C\}$.*

*This defines the set of **epistemic contexts**. It is denoted $C_{ELc, S, i, j}$.*

This definition presents the notion of epistemic contexts. They are *the most relevant sets of thoughts*, which meets the need we identified earlier. Other definitions are possible, for example by using the ranks of thoughts more finely. Epistemic contexts are sufficient for the modelling needs presented in this article.

Example *Let a and b be two atomic propositions of L_p, c_1 and c_2 be two thoughts of rank 1, and c_3 and c_4 be two thoughts of rank 2. Consider the following set:*

$$E_{Lc} = \{c_1 \rightarrow a, c_2 \rightarrow \neg b, c_3 \rightarrow c_1, c_4 \rightarrow \neg c_1\}$$

Let i=1 and j=2, and we consider the stimulus is empty. $\{c_3, c_4\}$ is incoherent, so $\{c_3\}$ and $\{c_4\}$ are the two maximal contexts of rank 2. Let's add to each the credible context of rank 1 associated with it to calculate the two epistemic contexts. We obtain:

- *$\{c_1, c_2\}$ is the credible context of rank 1 considering $\{E_{Lc}, S, c_3\}$, so we have $\{c_3, c_1, c_2\}$,*
- *$\{c_2\}$ is the credible context of rank 1 considering $\{E_{Lc}, S, c_4\}$, so we have $\{c_4, c_2\}$.*

We are now able to define the semantic interpretation function of L_c.

Definition *Let E_{Lc} be a theory, S be a stimulus and i and j be two integers such that $0 < i < j$. Considering E_{Lc}, S, i and j, a sentence f, called a piece of **belief**, is said:*

- ***conceivable** if, and only if, there is at least one epistemic context C_1 such that $\{E_{Lc}, S, C_1\} \vDash_{Lp} f$, and there is at least one epistemic context C_2 such that $\{E_{Lc}, S, C_2\} \vDash_{Lp} \neg f$,*
- ***credible** if, and only if, there is at least one epistemic context C_1 such that $\{E_{Lc}, S, C_1\} \vDash_{Lp} f$, and there is no epistemic context C_2 such that $\{E_{Lc}, S, C_2\} \vDash_{Lp} \neg f$,*
- ***improbable** if, and only if, there is at least one epistemic context C_1 such that $\{E_{Lc}, S, C_1\} \vDash_{Lp} \neg f$, and there is no epistemic context C_2 such that $\{E_{Lc}, S, C_2\} \vDash_{Lp} f$,*
- ***not interpretable** in other cases.*

*$\{E_{Lc}, S, C \in C_{ELc, S, i, j}\}$ is called a **semantic perspective**.*

This definition presents the basic semantic interpretation function of L_c. It can be enriched, for example by distinguishing true formulae in all semantic perspectives. This version is sufficient for the modelling needs presented in this article.

Example *Let us return to the epistemic contexts of the previous example. We obtain:*

- *$\{c_3, c_1, c_2\}$: the associated semantic perspective says that a is true and b is false,*
- *$\{c_4, c_2\}$: the associated semantic perspective says that b is false.*

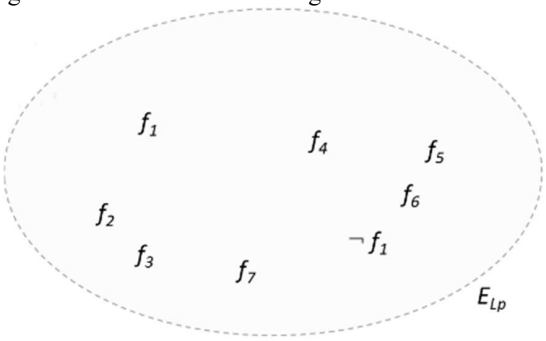
So, according to the semantic vocabulary of L_c, a is credible and b is improbable.

We have just presented the mathematical definition of the semantic interpretation function of L_c. This is not compatible with our natural language habits. Therefore, we will allow ourselves some linguistic shortcuts, for example: a context is said to be a belief or a piece of knowledge, and *vice versa*, a semantic perspective is said to be a perspective, a conceivable expression is said to be true and false, or possible, a credible expression is said to be true, conceivable, or possible, and an improbable expression is said to be false, conceivable, incredible, or impossible. We will use them in a way that does not create confusion.

5 The properties of L_c

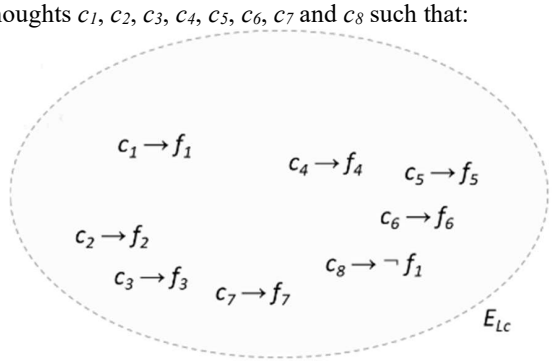
We now present the properties of L_c. We remain on a technical observation and not discuss their relevance. Indeed, each property echoes philosophical concepts and deserves a dedicated article. The debates are rich, and there are as many defenders as detractors. We do not bring new philosophical elements to enrich these exchanges – only a few practical findings that we will share in paragraphs 6 and 7.

To facilitate the understanding of what is to come, we propose to illustrate the principles of L_c with some small diagrams. Consider the following set:



Pic. 3 - A syntactically inconsistent theory of L_p

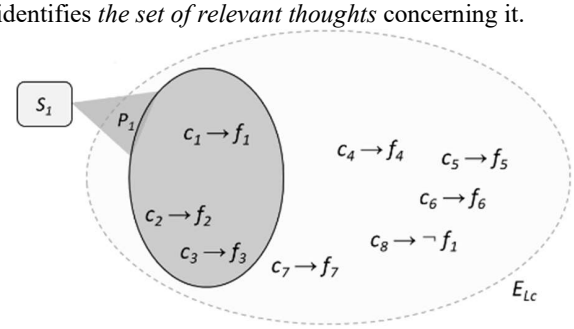
It is syntactically inconsistent, and it has no semantic meaning according to contextual postulate. Let us apply it by enriching the set of atomic propositions with the thoughts c_1, c_2, c_3, c_4, c_5, c_6, c_7 and c_8 such that:



Pic. 4 - A semantically inconsistent theory of L_c

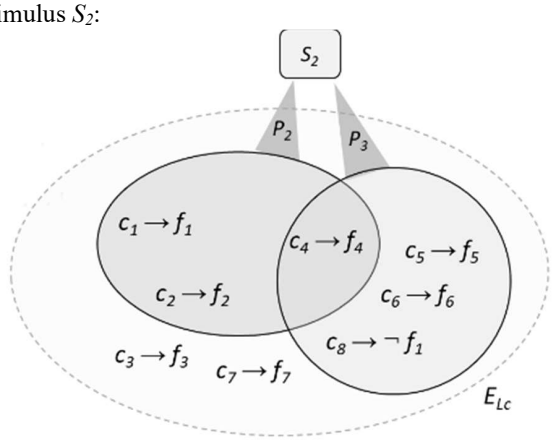
The resulting set E_{Lc} is a kind of dictionary of thoughts: each thought c_i is expressed by a formula f_i. It is syntactically consistent even though the thoughts may be semantically incoherent with each other. This result is obtained at the cost of an absolute uncertainty: all thoughts are potentially false.

To analyse a situation (formalised by a stimulus), L_c identifies *the set of relevant thoughts* concerning it.



Pic. 5 – The semantic interpretation of the stimulus S_1

In this example (presented for illustrative purposes only), the stimulus S_1 generates the semantic perspective P_1. Its epistemic context is $\{c_1, c_2, c_3\}$. c_1, c_2 and c_3 are the most relevant thoughts considering S_1. Assume another stimulus S_2:



Pic. 6 – The semantic interpretation of the stimulus S_2

S_2 is seen through two mutually incoherent semantic perspectives P_2 and P_3.

<u>The syntax production function is monotonic</u>

L_c respects the syntax of the propositional logic and is therefore syntactically monotonic: whatever f and g are contextually well-formed formulae, if a theory E_{Lc} produces f then $\{E_{Lc}, g\}$ produces f. Note that the syntactic interpretation function is mechanically also monotonic.

<u>The semantic interpretation function is non-monotonic</u>

L_c decorrelates the syntactic interpretation function from the semantic interpretation function. Syntax produces a set of formulae according to the rules of the contextualised formalism. Semantics then provides an interpretation by analysing the behaviour of the thoughts in the models of the theory. They are considered as they are produced by the syntactic rules of L_p.

The models of a theory can change if a new piece of knowledge is introduced. So, $C_{ELc,\,i,\,j}$ must be recalculated in this case, and L_c has a non-monotonic semantic:

considering a stimulus S, a formula f can be credible considering $\{E_{Lc}, S\}$ and incredible considering $\{\{E_{Lc}, g\}, S\}$, for g a contextual well-formed formula. We present some examples of use in the following paragraphs.

The semantic is faillibilistic

Non-monotonicity is a matter of completeness of knowledge: a belief that is true in one state may be false in an enriched state. Faillibilism (K. Popper [13]) is a more radical philosophical principle. It assumes that knowledge is impossible: all belief can, at any time, be questioned – and possibly contradicted.

A consequence of contextual postulate is that every proposition (which is not an axiom) is possibly false. To avoid this, the solution is to consider that what is not explicitly false is credible and will remain so until it is explicitly contradicted or challenged. We illustrate this with some examples which we develop in the following paragraphs.

The semantic is perspectivistic

Perspectivism (F. Kaulbach [7]) refers to philosophical doctrines that defend the idea that our perception of reality is composed of the sum of the perspectives we have on it.

In L_c, belief is not the consequence of a global point of view built on the whole of thoughts, but the juxtaposition of several points of view built from distinct subsets of thoughts each considered to be relevant.

Atomic propositions of L_p are attributes and not assertions

In L_c, a mechanical consequence of the application of contextual postulate is that it is not possible to deduce that a proposition is true or false according to L_p. A proposition can only be interpreted in relation to a set of thoughts, called a context. It characterises it. Consider, for example, the sentence:

"*If Tweety is a bird, then it flies*".

Its modelling in predicate logic can be:

$$Bird(Tweety) \rightarrow Fly(Tweety)$$

In L_c, this assertion is modelled by:

$$\{c_1 \rightarrow (Tweety \rightarrow Bird), c_2 \rightarrow (Bird \rightarrow Fly)\}$$

which allows for several readings - for example: *Bird* and *Fly* are attributes of the stimulus *Tweety* if we consider the context $\{c_1, c_2\}$.

Reasoning is introspective

By distinguishing expression and thought, and by modelling a relationship between them, contextual postulate brings a capacity for introspective reasoning to the formal language: the thoughts can reason about themselves using the constraints carried by the sentences that express them.

We have finished with the presentation of L_c. The following sections show how to use its properties to provide answers to the two last difficulties identified in paragraph 3 (the first being covered as we have just observed).

Considering the definition of epistemic contexts, i and j can theoretically take any value. According to the work of J. Pitrat [12], there are probably cognitive thresholds limiting human reasoning abilities. In the rest of this document, we use the thresholds *2* and *3*, which are sufficient to cover the expected level of expressiveness expected in this article.

We will see in paragraph 7 for what we reserve rank 1. And by writing convention, we now note $c_{i,j}$ the thoughts. i singularizes the atomic proposition and j indicates its rank.

6 Modelling an inconsistent information

For example, consider a set of L_p's propositions $\{a, b, c\}$ and let be the following set:

$$E_{Lp} = \{a \rightarrow b, a \rightarrow \neg b, c, a\}$$

It is inconsistent because $E_{Lp} \vdash_{Lp} b \wedge \neg b$. But E_{Lp} has no meaning according to contextual postulate. Let us now place ourselves in the contextual logic framework. Considering the set of thoughts $\{c_{10,2}, c_{20,2}, c_{30,2}, c_{40,2}\}$, we assume the following theory:

$$\begin{aligned} E_{Lc} = \{&c_{10,2} \rightarrow (a \rightarrow b),\\ &c_{20,2} \rightarrow (a \rightarrow \neg b),\\ &c_{30,2} \rightarrow c,\\ &c_{40,2} \rightarrow a\} \end{aligned}$$

Note that thoughts name formulas, which will allow us to carry out introspective reasoning. E_{Lc} is consistent, and there is an incoherence between the three thoughts $c_{10,2}$, $c_{20,2}$, and $c_{40,2}$ because:

$$\{E_{Lc}, c_{10,2}, c_{20,2}, c_{40,2}\} \vdash_{Lp} b \wedge \neg b$$

$\{c_{10,2}, c_{20,2}, c_{40,2}\}$ is a minimal impossible context. So, $\{c_{30,2}\}$ is the only epistemic context. If the stimulus is empty, we obtain one perspective which says $\{c\}$, and $\{a, c\}$ if the stimulus is $\{a\}$.

This is a first result showing the possibility of exploiting inconsistent beliefs in L_c. The solution is to get around the problem by considering that the thoughts $c_{10,2}$, $c_{20,2}$ and $c_{40,2}$ are not credible because they produce an inconsistency.

We now want to address this inconsistency, by modelling that $a \rightarrow b$ (i.e., the thought $c_{10,2}$) is not always true - or, put differently, is sometimes true and sometimes false. Let's use two new thoughts, $c_{11,3}$ and $c_{12,3}$:

$$\begin{aligned} E_{Lc} = \{&c_{10,2} \rightarrow (a \rightarrow b),\\ &\quad \mathbf{c_{11,3} \rightarrow c_{10,2},}\\ &\quad \mathbf{c_{12,3} \rightarrow \neg c_{10,2},}\\ &c_{20,2} \rightarrow (a \rightarrow \neg b),\\ &c_{30,2} \rightarrow c,\\ &c_{40,2} \rightarrow a\} \end{aligned}$$

Considering a is the stimulus, let's calculate the epistemic contexts. There are 2 maximum contexts of rank 3: $\{c_{11,3}\}$ and $\{c_{12,3}\}$. Let us extend each of them to their associated credible contexts of rank 2:

- considering $\{E_{Lc}, a, c_{11,3}\}$, $\{c_{20,2}, c_{40,2}\}$ is the only minimal impossible context, so $\{c_{10,2}, c_{30,2}\}$ is the credible context of the rank 2 in this case,
- considering $\{E_{Lc}, a, c_{12,3}\}$, $\{c_{10,2}\}$ is the only minimal impossible context, and $\{c_{20,2}, c_{30,2}, c_{40,2}\}$ is the credible context in this.

In fine, considering the stimulus $\{a\}$, we obtain two epistemic contexts:

- $\{c_{11,3}, c_{10,2}, c_{30,2}\}$ which says $\{a, b, c\}$ is true,
- $\{c_{12,3}, c_{20,2}, c_{30,2}, c_{40,2}\}$ which says $\{a, \neg b, c\}$ is true.

Taking a as the stimulus, we conclude that c is credible (or true), and that b is conceivable (or true and false). The formalism does this by modelling an epistemic information: the belief $a \rightarrow b$ (i.e., $c_{10,2}$) is true (what $c_{11,3}$ formalises) and false (what $c_{12,3}$ formalises).

We have used a single contradiction $\{c_{11,3}, c_{12,3}\}$ to illustrate our point. If multiple contradictions (two contradictions $\{c_{x1,3}, c_{x2,3}\}$ and $\{c_{y1,3}, c_{y2,3}\}$ for example), the different cases are managed on the maximal epistemic contexts ($\{c_{x1,3}, c_{y1,3}\}$, $\{c_{x1,3}, c_{y2,3}\}$, $\{c_{x2,3}, c_{y1,3}\}$, and $\{c_{x2,3}, c_{y2,3}\}$ with the example).

We obtain by combination the set of relevant perspectives. This is illustrated in the example that we develop at the end of this article.

Comparison with other formalisms

In this section, we point out the major gaps in the treatment of inconsistent or incomplete information between L_c and other non-classical formalisms.

The first and, from our point of view, the main difference is that contextual logic does not quit the syntax of propositional logic. Contrary to what is commonly shared, it is not necessary to add new connectors or to modify the syntactic rules of L_p to model a notion of inconsistency. It's not the only gap.

Paraconsistent and multivalued logics aim to tolerate inconsistencies by escaping the principle of explosion. The approach, theorised by J. Lukaszewicz [10], is either to weaken Aristotle's principles to limit the inferential capacities of language or to add a third truth value to indicate that the piece of knowledge concerned is both true and false.

L_c addresses the issue of uncertainty and inconsistency through its perspectivistic property: it models that something is simultaneously true according to some thoughts and false according to others. L_c is therefore not a paraconsistent or a multivalued formalism: it preserves the syntactic rules of L_p. Therefore, it does not escape the principle of explosion. If one retains a reference context that syntactically produces $f \wedge \neg f$, then it produces any belief g whatsoever.

Inconsistency is accepted in the semantic interpretation of L_c. It remains non-tolerable in its syntax.

Another difference between L_c and other non-classical formalisms is its faillibilistic property: noting that nothing is true, it takes as credible what is possible. This property gives L_c a particular behaviour, which does not allow it to fully capture modal logics or default logics, for example.

Default logic is proposed by R. Reiter [15]. To reason with uncertain information, he extends production rules by expressions of the form ($a : b / c$) which read:

If a is true and if b is possible then c is produced

In L_c, the thought that b is possible "generates the thought b". However, related to R. Reiter's syntax, L_c's expressiveness is limited to normal default rules, of the form ($a : b / b$) [8].

Modal epistemic logics extend the expressiveness of languages by adding a new connector for reasoning about the quality of the interpretative value. The most widely used epistemic modal connector is the alethic connector $\Box$. $\Box f$ usually expresses that f is necessary, and its dual $\neg\Box\neg f$, denoted $\Diamond f$, that f is possible. The language relies on the semantics of possible worlds of S.A. Kripke [9] to benefit from a syntactic interpretation function of $\Box$.

We have proposed a relationship between modal epistemic logics and L_c [8]. This requires an evolution of the definition of epistemic context, using ranks to capture the imbrications of the monadic connector (rank i for $\Box$, rank $i+1$ for $\Box\Box$, etc.). It models the sets $\{\Box f\}$ and $\{\Diamond f, \Diamond\neg f\}$, but the set restricted to $\{\Diamond f\}$ is interpreted as $\{\Box f\}$: f is considered necessary if the possibility of its opposite is not explicitly expressed.

The behaviour of L_c is equivalent to adding a default rule to the K system: $\{\Diamond f : \Box f / \Box f\}$. In the framework of Kripke's semantics, this expression can be understood as:

If I know an accessible world in which f is true, and if I do not know an accessible world in which ¬ f is true, then I consider that f is true in all accessible worlds.

Paraconsistent, default and modal epistemic logics deal with the issue of incoherence by evolving the syntactic capabilities of the language. D. Batens proposes another approach [3]. He considers that there are several reasoning strategies, and that the solution consists in choosing the one that is best adapted to the situation. These are adaptive logics. For example, let be the following set of formulae:

$$E_{La} = \{\neg p, \neg q, p \vee q, p \vee r, q \vee r\}$$

It is incoherent, and therefore explosive in the context of propositional logic because $\{\neg p, \neg q\}$ contradicts $p \vee q$. If one adopts a strategy favouring reliable reasoning, it is not possible to deduce r: it would be unwise to conclude anything using the first three formulae. However, if we choose a strategy that minimises abnormalities, and assume that at least two of the first three formulae are true, then r is produced.

In contextual logic, the set becomes:

$$E_{Lc} = \{c_{1,2} \rightarrow \neg p, c_{2,2} \rightarrow \neg q, c_{3,2} \rightarrow p \vee q, c_{4,2} \rightarrow p \vee r, c_{5,2} \rightarrow q \vee r\}$$

Assume that the stimulus is empty. $\{c_{4,2}, c_{5,2}\}$ is the reference context because $\{c_{1,2}, c_{2,2}, c_{3,2}\}$ is a minimal impossible context. As far as we know, r is not interpretable. Using epistemic contexts that retain the maximal credible contexts at rank 2 is therefore a prudent strategy.

So, L_c is not an adaptive logic. Both formalisms have the capacity to adapt their semantic interpretation to local characteristics: L_c chooses to use or not a thought depending on the stimulus. But its principle is not to adapt its reasoning according to the typology of the situation. It

uses a unique analysis strategy, based on the definition of epistemic contexts.

We end this comparative section with the circumscription logic of J. McCarthy [11]. It consists in extending the set of atomic propositions by some atomic propositions that indicate the epistemic character of a formula. For example, "$a \rightarrow b$ is true except in atypical cases" is modelled by $((a \rightarrow b) \vee abnormal)$.

The atomic proposition *abnormal* carries the exceptional behaviours when needed. The models of the theory are then analysed to select those that minimise the abnormalities.

This approach, which consists in seeking a solution by enriching the set of atomic propositions and then analysing the models of the theory, is most certainly the closest to ours. We have shown that contextual logic can capture its expressive capacity by adapting the definition of epistemic contexts to meet the minimality criterion [8].

However, beyond this result, the choice to minimise abnormalities seems reasonable but can easily be questioned with use cases. This difficulty is shared with adaptive logics, or more generally with the concept of epistemic rooting proposed by P. Gardenfors and D. Makinson [6].

Indeed, these methods suppose the existence of an order relation (on pieces of knowledge or on reasoning strategies) which would oversee selecting the information in case of incoherence. In L_c, syntactic consistency is guaranteed. It is therefore not necessary to manage this in the formalism.

7 Modelling a predicate information

L_p sees a proposition as a whole, which is given a universal value. It is necessary to decompose this whole when we wish to use a singular value. To this end, predicate logic meets this need by allowing the desired relationship to be modelled directly in the elementary proposition.

It then becomes possible to model that *Socrates is a man*, and to deduce that *Socrates is mortal* because *a man is mortal*:

$$Man(Socrates)$$
$$Man(Socrates) \rightarrow Mortal(Socrates)$$

This syllogism uses the link between *Man* and *Socrates* to deduce the association with *Mortal*. In this context, G. Frege [5] theorised the notion of universal quantifier.

As a classical example of use:

$$\forall\, x\ Man(x) \rightarrow Mortal(x)$$

which reads: whatever x is, if x is a man then x is mortal. We note two enrichments with respect to the native modelling capabilities of the propositional logic:

- the notion of the universal quantifier $\forall$. We will come back to the subject of universal connectors or universal quantifier at the paragraph 9,
- the possibility of breaking down an atomic proposition of L_p into several singular instances. In our example, the proposition *he is a man* is modelled by using two distinct units, *Man* and x. The expression *Man(x)* creates a syntactic relationship, which formalises a semantic link, between these two-unit elements. x then allows to create a semantic link between the two propositions *Man(x)* and *Mortal(x)*.

We have seen in paragraph 3 that it is not possible to model a relation between two atomic propositions in L_p because of the symmetric behaviour of connectors. We now present how this point can be solved in L_c.

Suppose in L_p the set of propositions $\{a, b, c, d, e, f\}$ and the following theory:

$$\begin{aligned} E_{Lp} = \{ & a \rightarrow b, \\ & c \rightarrow d, \\ & a \rightarrow \neg c, \\ & c \rightarrow (a \rightarrow e), \\ & c \rightarrow (a \rightarrow f)\} \end{aligned}$$

We want to model that $a \rightarrow e$ is a predicate of c – i.e., c is the subject of $a \rightarrow e$. The problem is that the formula $c \rightarrow (a \rightarrow e)$ is syntactically equivalent to $a \rightarrow (c \rightarrow e)$. Now, consider the following set in L_c:

$$\begin{aligned} E_{Lc} = \{ & c_{10,2} \rightarrow (a \rightarrow b), \\ & c_{20,2} \rightarrow (c \rightarrow d), \\ & c_{30,2} \rightarrow (a \rightarrow \neg c), \\ & \quad \mathbf{c_{40,2} \rightarrow (c \rightarrow c_{41,1})}, \\ & \quad \mathbf{c_{41,1} \rightarrow (a \rightarrow e)}, \\ & c_{50,2} \rightarrow (c \rightarrow (a \rightarrow f))\} \end{aligned}$$

$c_{40,2}$ and $c_{41,1}$ model the predicative piece of knowledge. They break the syntactic symmetry and introduce two new pieces of information: the predicate is syntactically distinguished by the thought $c_{41,1}$, and $c_{40,2}$ says that $c_{41,1}$ is true if the subject c is true. It is important to note that $c_{41,1}$ is of rank 1, so, by definition, it does not belong to an epistemic context. It only appears later, in the contexts in which $c_{40,2}$ is true if c is true.

Let $c \wedge a$ be the stimulus. $\{c_{10,2}, c_{20,2}, c_{40,2}, c_{50,2}\}$ is its epistemic context. The perspective associated says $\{c, a, b, d, e, f, c_{41,1}\}$. The presence of $c_{41,1}$ indicates that $a \rightarrow e$ is a predicate, but the relationship with its subject c is not apparent. It can be found by applying the following method:

- calculate the perspectives of the stimulus c. It says $\{c, \neg a, d, c_{41,1}\}$. The predicate $c_{41,1}$ is obtained by $c_{40,2}$,
- calculate the perspectives of the stimulus a. It says $\{a, \neg c, b\}$. The thought $c_{41,1}$ is not syntactically produced,
- apply the predicates associated with each perspective to the other perspectives. This produces $\{e\}$ by applying $c_{41,1}$ to the perspective of a. e is associated with the stimulus c associated with the applied predicate,
- and finally, calculate the perspectives of the stimulus $c \wedge a$. This produces $\{f\}$.

If the thought of a predicate appears in a perspective, then it expresses that its subject is its stimulus. Unlike in predicate logic, the notion of predicate is not carried by the syntax of L_c. It is interpreted from the semantics of perspectives. This method extends the consumption of epistemic contexts by a recursive function:

The semantics of a composite universe is obtained by crossing the semantics of the objects that compose it.

The semantics of a universe composed of three objects A, B and C can only be partially obtained by analysing the perspectives of $A \wedge B \wedge C$. To obtain a complete perception, we must analyse $\{A\}$, $\{B\}$, $\{C\}$, $\{A, B\}$, $\{A, C\}$, $\{B, C\}$ and $\{A, B, C\}$ separately, and cross-reference the properties associated with these seven different objects.

We call this method the **generalised contextual semantics**. It allows exhaustively capturing the characteristics of each object, of each combination of objects, and to calculate cross-predictive inferences. It seems to produce redundancies. We have not studied whether technical optimizations are possible.

8 An example of application

After presenting the theoretical principles of contextual logic, we propose to develop an example of application to clarify our purpose, and to illustrate the knowledge modelling capabilities of L_c. To do this, we use the example of the bird Tweety, a classical case study in the literature on non-monotonicity and belief revision.

Example *Consider the following knowledge, which we call the E_{NL} (for Natural Language) set:*

Birds and felines are animals (01 and 02). Birds are not felines (03). Animals are diurnal (04). Diurnal animals are not nocturnal (05). Birds fly (06). They are insectivorous (07) and gregarious (08). Felines are carnivorous (09) and solitary (10). Solitaires are not gregarious (11). Insectivores are not carnivorous (12). Swallows, sparrows, ostriches, and owls are birds (13, 14, 15 and 16). Swallows are not sparrows (17), ostriches (18), or owls (19). Sparrows are not ostriches (20) or owls (21). Ostriches are not owls (22). Ostriches do not fly (23). Owls are solitary (24), nocturnal (25), carnivorous (26), and insectivorous (27). Cats and lions are felines (28 and 29). Cats are not lions (30). Cats are nocturnal (31). Lions are gregarious (32). Carnivores are hunters (33). Herbivores are prey for hunters (34). Hunters attack prey (35). If the prey is larger than the hunter, the latter does not attack (36). Ostriches are larger than cats (37) and owls (38).

E_{NL} contains a lot of inconsistent, epistemic, and predicative information - for example: birds fly and do not fly, birds are insectivores and carnivorous, and hunters attack prey (and not the reverse).

We have chosen to take a relatively large knowledge base to model several cross-cases, and we will develop below the example in detail. What follows is therefore quite tedious to read. We apologize to the readers for this. The objective is to give them enough element to reproduce the exercise if they wish.

This knowledge is deemed to escape the syntax of propositional logic. We are however going to show that it is sufficient to model and exploit them.

In a first time, we propose to translate it into the syntax of L_p by the following formulae.

01 $Bird \rightarrow Animal$
02 $Feline \rightarrow Animal$
03 $Bird \rightarrow \neg Feline$
04 $Animal \rightarrow Diurnal)$
05 $Diurnal \rightarrow \neg Nocturnal$
06 $Bird \rightarrow Fly$
07 $Bird \rightarrow Insectivore$
08 $Bird \rightarrow Gregarious$
09 $Feline \rightarrow Carnivore$
10 $Feline \rightarrow Solitary$
11 $Gregarious \rightarrow \neg Solitary$
12 $Insectivore \rightarrow \neg Carnivore$
13 $Swallow \rightarrow Bird$
14 $Sparrow \rightarrow Bird$
15 $Ostrich \rightarrow Bird$
16 $Owl \rightarrow Bird$
17 $Swallow \rightarrow \neg Sparrow$
18 $Swallow \rightarrow \neg Ostrich$
19 $Swallow \rightarrow \neg Owl$
20 $Sparrow \rightarrow \neg Ostrich$
21 $Sparrow \rightarrow \neg Owl$
22 $Ostrich \rightarrow \neg Owl$
23 $Ostrich \rightarrow \neg Fly$
24 $Owl \rightarrow Solitary$
25 $Owl \rightarrow Nocturnal$
26 $Owl \rightarrow Carnivore$
27 $Owl \rightarrow Insectivore$
28 $Cat \rightarrow Feline$
29 $Lion \rightarrow Feline$
30 $Cat \rightarrow \neg Lion$
31 $Cat \rightarrow Nocturnal$
32 $Lion \rightarrow Gregarious$
33 $Carnivore \rightarrow Hunter$
34 $Herbivore \rightarrow (Hunter \rightarrow Prey)$
35 $Hunter \rightarrow (Prey \rightarrow Attack)$
36 $Hunter \rightarrow (Prey \rightarrow (Larger \rightarrow \neg Attack))$
37 $Ostrich \rightarrow (Cat \rightarrow Larger)$
38 $Ostrich \rightarrow (Owl \rightarrow Larger)$

Let us consider that the knowledge is entered in the order in which it appears, according to the following algorithm:

For each formula f_i
  If f_i has a subject, then PK
  Else
    Creating the thought $c_{i0,2}$
    Creating the formula $c_{i0,2} \rightarrow f_i$
    If there is a contradiction, then EM

We assume there is a monitor accompanying the learning. For example, we don't have a grammatical module allowing us to identify the subjects of each sentence. So, the system queries its instructor. Up to formula 34, the instructor's response is that there is none. Indeed, the identification of the subject is not always necessary. Doing it systematically would only make the presentation more cumbersome.

The sentences are processed one after the other (the description of the modules EM and PK will come later).

01 $c_{010.2} \rightarrow (Bird \rightarrow Animal)$
02 $c_{020.2} \rightarrow (Feline \rightarrow Animal)$
..
24 $c_{240.2} \rightarrow (Owl \rightarrow Solitary)$

The formula is integrated, and the system tested its semantic consistency. Until formula 23, there is no question. The formula 24 is then integrated, and a potential inconsistency is detected: owls are birds (16), birds are gregarious (08), owls are solitary (24) and gregarious is not

solitary (11). So, owls do not exist, or they are solitary and not solitary.

We consider that there is no automatic solution to resolve an inconsistency. The properties of L_c make it possible to record it as it arrives, and additional information can possibly bring precision. For this, the system interrogates its instructor. He has three possible answers:

1) he does not know, or he considers it is normal: the semantic inconsistency is accepted, and the system moves on to the next step,
2) he indicates that one of the pieces of knowledge involved in the inconsistency is false. In this case, the system applies the EM module to the indicated formula,
3) he indicates that one of the pieces of knowledge involved in the inconsistency is true and false: it is an alethic information. In this case, birds are gregarious (08) is sometimes true and sometimes false. Then the system applies the EM module to the indicated formula.

The EM module is:

if f_i is false *// Case 2*
Creating the thoughts $c_{i2,3}$
Creating the formula $c_{i2,3} \rightarrow \neg\, c_{i0,2}$
// The combinatorics on the maximal contexts
// of T_m means that $c_{i0,2}$ will no longer be
// retained in the epistemic contexts
else if f_i is true and false *// Case 3*
Creating the thoughts $c_{i1,3}$ et $c_{i2,3}$
Creating the formula $c_{i1,3} \rightarrow c_{i0,2}$
Creating the formula $c_{i2,3} \rightarrow \neg\, c_{i0,2}$

We identify below the formulae that are affected by this module. We will only use the third last case which is the most interesting.

04 *Animal → Diurnal*
06 *Bird → Fly*
07 *Bird → Insectivore*
08 *Bird → Gregarious*
10 *Feline → Solitary*
12 *Insectivore → ¬ Carnivore*
35 *Hunter → (Prey → Attack)*

Let's continue the treatment. We obtain:

01 $c_{010.2} \rightarrow$ *(Bird → Animal)*
02 $c_{020.2} \rightarrow$ *(Feline → Animal)*
03 $c_{030.2} \rightarrow$ *(Bird → ¬ Feline)*
04 $c_{040.2} \rightarrow$ *(Animal → Diurnal)*
$c_{041.3} \rightarrow c_{040.2}$
$c_{042.3} \rightarrow \neg\, c_{040.2}$
05 $c_{050.2} \rightarrow$ *(Diurnal → ¬ Nocturnal)*
..
32 $c_{320.2} \rightarrow$ (*Lion → Gregarious*)
33 $c_{330.2} \rightarrow$ (*Carnivore → Hunter*)

We come to the formula number 34: *Herbivore* → (*Hunter* → *Prey*). Before integration, the system asks if it contains a subject.

Until now, we had not used this possibility so as not to make the presentation unnecessarily heavy. But, within the framework of this formula, the identification of the subject is necessary. So, the answer is yes, for *Herbivore*.

The PK module is then applied:

// f_i is a clause, so it is of type $g \rightarrow h$
// with g is a conjunction of literals
// and h is a disjunction of literals
// g is indicated by the instructor
Creating the thoughts $c_{i0,2}$ et $c_{i3,1}$
Creating the formula $c_{i0,2} \rightarrow g \rightarrow c_{i3,1}$
Creating the formula $c_{i3,1} \rightarrow h$

We identify below the formulae that are affected by this module.

34 *Herbivore → (Hunter → Prey)*
35 *Hunter → (Prey → Attack)*
36 *Hunter → (Prey → (Larger → ¬ Attack))*
37 *Ostrich → (Cat → Larger)*
38 *Ostrich → (Owl → Larger)*

After application to the whole of E_{NL}, we obtain:

01 $c_{010.2} \rightarrow$ *(Bird → Animal)*
02 $c_{020.2} \rightarrow$ *(Feline → Animal)*
03 $c_{030.2} \rightarrow$ *(Bird → ¬ Feline)*
04 $c_{040.2} \rightarrow$ *(Animal → Diurnal)*
$c_{041.3} \rightarrow c_{040.2}$
$c_{042.3} \rightarrow \neg\, c_{040.2}$
05 $c_{050.2} \rightarrow$ *(Diurnal → ¬ Nocturnal)*
06 $c_{060.2} \rightarrow$ *(Bird → Fly)*
$c_{061.3} \rightarrow c_{060.2}$
$c_{062.3} \rightarrow c_{060.2}$
07 $c_{070.2} \rightarrow$ *(Bird → Insectivore)*
$c_{071.3} \rightarrow c_{070.2}$
$c_{072.3} \rightarrow \neg\, c_{070.2}$
08 $c_{080.2} \rightarrow$ *(Bird → Gregarious)*
$c_{081.3} \rightarrow c_{080.2}$
$c_{082.3} \rightarrow \neg\, c_{080.2}$
09 $c_{090.2} \rightarrow$ *(Feline → Carnivore)*
10 $c_{100.2} \rightarrow$ *(Feline → Solitary)*
$c_{101.3} \rightarrow c_{100.2}$
$c_{102.3} \rightarrow \neg\, c_{100.2}$
11 $c_{110.2} \rightarrow$ *(Gregarious → ¬ Solitary)*
12 $c_{120.2} \rightarrow$ *(Insectivore → ¬ Carnivore)*
$c_{121.3} \rightarrow c_{120.2}$
$c_{122.3} \rightarrow \neg\, c_{120.2}$
13 $c_{130.2} \rightarrow$ *(Swallow → Bird)*
14 $c_{140.2} \rightarrow$ *(Sparrow → Bird)*
15 $c_{150.2} \rightarrow$ *(Ostrich → Bird)*
16 $c_{160.2} \rightarrow$ *(Owl → Bird)*
17 $c_{170.2} \rightarrow$ *(Swallow → ¬ Sparrow)*
18 $c_{180.2} \rightarrow$ *(Swallow → ¬ Ostrich)*
19 $c_{190.2} \rightarrow$ *(Swallow → ¬ Owl)*
20 $c_{200.2} \rightarrow$ *(Sparrow → ¬ Ostrich)*
21 $c_{210.2} \rightarrow$ *(Sparrow → ¬ Owl)*
22 $c_{220.2} \rightarrow$ *(Ostrich → ¬ Owl)*
23 $c_{230.2} \rightarrow$ *(Ostrich → ¬ Fly)*
24 $c_{240.2} \rightarrow$ *(Owl → Solitary)*
25 $c_{250.2} \rightarrow$ *(Owl → Nocturnal)*
26 $c_{260.2} \rightarrow$ *(Owl → Carnivorous)*
27 $c_{270.2} \rightarrow$ *(Owl → Insectivore)*
28 $c_{280.2} \rightarrow$ *(Cat → Feline)*
29 $c_{290.2} \rightarrow$ (*Lion → Feline*)
30 $c_{300.2} \rightarrow$ (*Cat → ¬ Lion*)
31 $c_{310.2} \rightarrow$ (*Cat → Nocturnal*)
32 $c_{320.2} \rightarrow$ (*Lion → Gregarious*)
33 $c_{330.2} \rightarrow$ (*Carnivore → Hunter*)
34 $c_{340.2} \rightarrow$ (*Herbivore* → $c_{343.1}$)
$c_{343.1} \rightarrow$ (*Hunter → Prey*)
35 $c_{350.2} \rightarrow$ (*Hunter* → $c_{353.1}$)
$c_{351.3} \rightarrow c_{350.2}$

$c_{352.3} \rightarrow \neg\, c_{350.2}$
$c_{353.1} \rightarrow (Prey \rightarrow Attack)$
36 $c_{360.2} \rightarrow (Hunter \rightarrow c_{363.1})$
$c_{363.1} \rightarrow (Prey \rightarrow (Larger \rightarrow \neg\, Attack))$
37 $c_{370.2} \rightarrow (Ostrich \rightarrow c_{373.1})$
$c_{373.1} \rightarrow (Cat \rightarrow Larger)$
38 $c_{380.2} \rightarrow (Ostrich \rightarrow c_{383.1})$
$c_{383.1} \rightarrow (Owl \rightarrow Larger)$

We have converted E_{NL} into a set E_{Lc} that is a theory of L_p. It is consistent. L_c uses the syntactic rules of propositional logic. Here are some examples obtained by applying the generalised contextual semantic:

- if the stimulus is *Bird*: birds are animals, diurnal, gregarious, insectivore, and fly,
- if the stimulus is *Swallow*: swallows are birds, animals, diurnal, gregarious, insectivore, and fly,
- if the stimulus is *Ostrich*: ostriches are birds, animal, diurnal, gregarious, insectivore, and do not fly,
- if the stimulus is *Owl*: owls are birds, animals, nocturnal, solitary, carnivorous, and fly,
- if the stimulus is *Cat*: cats are feline, animals, diurnal, carnivorous and solitary,
- if the stimulus is *Lion*: lions are feline, animals, diurnal, carnivorous and gregarious,
- if the stimulus is {*Cat*, *Sparrow*}: the sparrow is attacked,
- if the stimulus is {*Cat*, *Owl*, *Sparrow*, *Ostrich*}: the sparrow is in a bad way, but the ostrich can go about its business,
- and if the stimulus is {*Lion*, *Ostrich*}: the ostrich would have some reason to be worried.

Informatic tool

The different steps are described in detail, which probably makes reading tedious at times. The goal is to allow the readers to control and to reproduce the process if they wish.

We use a classical propositional logic solver to calculate the minimum inconsistent contexts, and a classical combinatorial algorithm using the minimum inconsistent contexts to calculate the epistemic contexts.

These tools are common and easily accessible. We hold those which we developed at the disposal of the readers who would like it.

Use case

We use the example of Tweety, which may seem simplistic or even naive. Its first interest is to be understandable by all, while illustrating all the theoretical problems of non-monotonicity and of belief revision.

Any other subject could have done the trick. The syntactic and semantic rules used were defined by the theory, and we did not integrate any behavioural rules specific to the subject.

Modelled information

While strictly preserving the syntactic rules of propositional logic, L_c allows us to define a semantic function on the models of the theories of L_p. It exploits:

- semantically incoherent pieces of knowledge: *Birds fly. Ostriches are birds. Ostriches do not fly* (thoughts $c_{060,2}$, $c_{150,2}$ and $c_{230,2}$),
- predicative pieces of knowledge: *Herbivores are prey for hunters* (thoughts $c_{340,2}$ and $c_{343,1}$),
- and alethic modal pieces of knowledge: *Birds are generally insectivorous* (thoughts $c_{070,2}$, $c_{071,3}$ and $c_{072,3}$).

The example contains several different cases to illustrate their use thanks to the combinatorics calculated for the identification of reference contexts. And we have adjoined a case which simultaneously uses a predicate and an alethic modality: $Hunter \rightarrow (Prey \rightarrow Attack)$ (thought $c_{350,2}$). It is false if the prey is large (thought $c_{360,2}$).

9 Some points of clarification and opening

In the previous paragraphs, we have certainly not answered all the questions that our presentation raises. But there are also some topics that we have deliberately skimmed over so as not to make our remarks totally indigestible. We are not going to develop them, for the same reason. But, for readers who wish to explore these concepts in more depth, we offer to list them below, in a thought-provoking format.

The notion of subject

We do not use the notion of subject on all formulas. In absolute terms, this should have been done in the first sentence. But the result would have been unreadable, without any contribution to what we wished to demonstrate.

In fact, this remark is not insignificant. It raises some questions as: are we saying the same thing with $a \rightarrow b$ and $\neg\, b \rightarrow \neg\, a$? The question is about the meaning of connectors. It has been extensively studied, notably by J. Lukaszewicz [10].

Without elaborating on the subject, let us point out to readers interested in this topic that L_c proposes an answer. $c \rightarrow f$ says that if c is true then f is true. But if c is false, then f can be true or false. The fact that a thought is false does not imply that the expression that describes it is false in the sense of syntactic interpretation.

Thoughts

They are indeed *silent* atomic propositions. They appear completely automatically in the syntax. In the algorithms of paragraph 10, the communication interface with the instructor does not see them, and only acts through the sentences of L_p.

Conjunctive normal form

To expand the example, we have chosen to associate a new thought with each clause of the conjunctive normal form of the theory. This has an impact on the semantic interpretation: for the same stimulus, the interpretation of the set $\{c_1 \rightarrow f, c_2 \rightarrow g\}$ may be different from that which would be obtained on the set $\{c \rightarrow f \wedge g\}$.

Our choice has no theoretical basis. It is pragmatic: the conversion of a set of formulae into its conjunctive normal form is achieved by a linear algorithm, and each clause

identifies a unique formula and therefore a unique thought. We have chosen the simplest technical solution.

Epistemic context

Concerning the semantic interpretation function, we propose the definition of epistemic contexts to identify the relevant sets of thoughts. Other definitions are possible.

Another method, perhaps more purist, would have consisted in giving the different possible definitions and comparing their mathematical properties. It requires a lot of methodical work that we have not done.

Universal connector

A difference between predicate logic and contextual logic is that the latter has no universal quantifier.

In fact, L_c natively models a form of quantifier. For example, assume a knowledge base consisting of a single piece of information: $\{c \rightarrow (Bird \rightarrow Fly)\}$. This says that *Birds fly*. Given this piece of knowledge, it would be the same to say *All bird fly*.

L_c is faillibilistic. In this context, asserting that a state is universal does not make sense: a state is universal as long as it is not contradicted. In the same vein, epistemic modalities are not expressed by a dedicated connector, but are deduced from a semantic interpretation.

Stimulus

Contextual logic introduces the notion of stimulus. Its concept is necessary to generate a possible reaction. We use it in the different examples as an imposed external event *f*. Relations with facts are carried by thoughts, according to an anchoring principle that remains to be defined.

10 Conclusion

Contextual logic is obtained by applying the contextual postulate on propositional logic. It proposes to formalise a relation between thoughts and languages. This brings the possibility of modelling an introspective reasoning.

The principle of proof is the foundation of mathematical philosophy. But introspective reasoning automatically generates the impossibility to demonstrate that something is true. Faced with the need for decidability, L_c uses the principle of non-refutability, which is backed by faillibilism and perspectivism. These are some old and still open subjects (H. Albert [1] and W. Quine [14] for example). Our work proposes to reconcile the different theories, by an answer based on a "semantic of uncertainty" modelled by a "certain syntax".

The historical ambition of logic is to model the process of human reasoning. This article, by presenting an application of the contextual postulate, performs a first clearing. It addresses, or rather it flies over, some basic concepts of AI, mixing mathematics, computer science, philosophy, and cognitive science. We are especially aware that this presentation leaves many open questions.

The syntactic dimension of formal languages is a vast field of work. For many years, it has been the subject of substantial studies, and many discoveries probably remain to be made in this area. Our contribution is to show that the syntax of the simplest known formal language, namely propositional logic, is sufficient to model complex reasoning which until now was deemed to escape it.

Contextual logic has a non-monotonic semantic and exploits some inconsistent information and some predicative forms. We obtain this by remaining in the syntax of L_p. By presenting this result, we invite to study the possibilities of enriching the semantic interpretation capacities of formal language theories by revisiting the definitions and meanings associated with atomic propositions.

References

[1] H. Albert, "Traktat über kritische Vernunft", Utb Fuer Wissenschaft, 1968

[2] D. Andler, "Introduction aux sciences cognitives", Gallimard, 2004

[3] D. Batens, "Une caractérisation générale des logiques adaptatives", Logique & Analyse 173–174–175, 2001

[4] R. Descartes, "Méditations métaphysiques. Objections et Réponses (I à VI)", édition sous la direction de Jean-Marie Beyssade et de Denis Kambouchner, éd. Gallimard Tel (2018), 1641

[5] G. Frege, "Grundgesetze der Arithmetik", 1903

[6] P. Gardenfors, D. Makinson, "Relations between the logic of theory change and nonmonotonic logic", in André Fuhrmann and Michael Morreau, eds. The Logic of Theory Change, LNAI-465, p. 185-205, 1991

[7] F. Kaulbach, "Philosophie des Perspektivismus: Wahrheit und Perspektive bei Kant, Hegel und Nietzsche", Mohr Siebeck, Tübingen, 1990

[8] A. Kohler, "Proposition d'une structure de représentation de la connaissance pour les raisonnements non classiques", Université d'Aix-Marseille I, France, thèse de doctorat en informatique, 1995

[9] S.A. Kripke, Saul Aaron, "Semantical considerations on modal logic", Reference and Modality, L. linsky (éditeur), Oxford University Press, London, 1971

[10] J. Lukaszewicz, "Many-Valued Systems of Propositional Logic", eds. S. Mc Call, polish Logic, Oxford University Press, Oxford, 1967

[11] J. McCarthy, "Circumscription - A Form of Non-Monotonic Reasonnig", Artificial Intelligence 13, pp. 27-39, 1980

[12] J. Pitrat, "Méta-connaissance, Futur de l'Intelligence Artificielle", Hermes, 1990

[13] K. Popper, "The Logic of Scientific Discovery", Routledge, coll. Routledge Classics, 1934

[14] W. Quine, "From a logical point of view", Harvard Univiersity Press, Cambridge, Massachussets and London, England, 1980

[15] R. Reiter, "A logic for default reasoning", Artificial Intelligence 13, 1980

[16] P. Siegel, "Représentation et utilisation de la connaissance en calcul propositionnel", Université de Luminy, Thèse d'état, 1987

[17] A. Thayse et co-auteurs, "Approche logique de l'Intelligence Artificielle", DUNOD, 1990

[18] L. Wittgenstein, "Tractatus logico-philosophicus", traduction Gilles Gaston Granger, éd. Gallimard Tel (1993), 1921

Actes JIAF 2022

Fonctions de croyances interprétées sur la logique de Belnap–Dunn

Marta Bílková[1] **Sabine Frittella**[2]
Daniil Kozhemiachenko[2] **Ondrej Majer**[3]
Sajad Nazari[2]

[1] Czech Academy of Sciences, Institute of Computer Science, Prague, République Tchèque
[2] INSA Centre Val de Loire, Univ. Orléans, LIFO EA 4022, France
[3] Czech Academy of Sciences, Institute of Philosophy, Prague, République Tchèque
bilkova@cs.cas.cz sabine.frittella@insa-cvl.fr
daniil.kozhemiachenko@insa-cvl.fr majer@flu.cas.cz
sajad.nazari@insa-cvl.fr

Dans ce document, nous introduisons les résultats de [4, 3] où nous définissons une extension de la logique de Belnap–Dunn permettant de raisonner avec des informations probabilistes incohérentes et incomplètes en utilisant des fonctions de croyance et de plausibilité. Ce framework permet de raisonner avec les fonctions de croyance et de les combiner même en présence d'informations fortement contradictoires tout en gardant la trace des sujets sur lesquels les différentes sources d'informations sont en désaccord. Nous présentons aussi deux calculs sains et complets de cette logique. Le premier basé sur des inégalités linéaires proche de celui présenté dans [5]. Le second via une logique modale à deux couches inspirée par [1].

Raisonner avec des informations incomplètes et contradictoires. Chaque jour, nous prenons des décisions basées sur divers types d'informations. Celles-ci peuvent être classiques (par exemple, il pleut ou il ne pleut pas), incomplètes (nous *n'avons aucune information* nous permettant de savoir si la représentation décimale de π contient deux mille 9 d'affilée) ou contradictoires (nous avons *à la fois des preuves* de l'efficacité et de l'inefficacité de la thérapie du miroir dans le traitement de la douleur fantôme).

La logique de Belnap–Dunn (BD) [2] a été introduite spécifiquement pour raisonner sur des informations incomplètes et/ou contradictoires. Cette logique a quatre valeurs de vérité $\{t, b, n, f\}$ représentant l'information qu'un ordinateur pourrait avoir concernant une déclaration.

— t signifie "dit seulement vrai".
— f signifie "dit seulement faux".
— b (ou 'both') signifie "dit à la fois vrai et faux".
— n (ou 'neither') signifie "ne dit ni vrai ni faux".

Le langage de BD est

$$\mathscr{L}_{\mathsf{BD}} \ni \phi \coloneqq p \in \mathtt{Prop} \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi.$$

Les formules sont interprétées sur les BD-modèles $\mathfrak{M} = \langle W, v^+, v^- \rangle$ où W est un ensemble non vide et $v^+, v^- : \mathtt{Prop} \to \mathcal{P}(W)$. Soient $\phi, \phi' \in \mathscr{L}_{\mathsf{BD}}$, un état $w \in W$ satisfait positivement (resp. négativement) une formule, noté $w \vDash^+ \phi$ (resp. $w \vDash^- \phi$), dans les conditions suivantes :

$$\begin{array}{cc} w \vDash^+ p \text{ ssi } w \in v^+(p) & w \vDash^- p \text{ ssi } w \in v^-(p) \\ w \vDash^+ \neg\phi \text{ ssi } w \vDash^- \phi & w \vDash^- \neg\phi \text{ ssi } w \vDash^+ \phi \end{array}$$
$$w \vDash^+ \phi \wedge \phi' \text{ ssi } w \vDash^+ \phi \text{ et } w \vDash^+ \phi'$$
$$w \vDash^- \phi \wedge \phi' \text{ ssi } w \vDash^- \phi \text{ ou } w \vDash^- \phi'$$
$$w \vDash^+ \phi \vee \phi' \text{ ssi } w \vDash^+ \phi \text{ ou } w \vDash^+ \phi'$$
$$w \vDash^- \phi \vee \phi' \text{ ssi } w \vDash^- \phi \text{ et } w \vDash^- \phi'$$

On dénote l'interprétation positive et l'interprétation négative d'une formule : $|\phi|^+ \coloneqq \{w \in W \mid w \vDash^+ \phi\}$ et $|\phi|^- \coloneqq \{w \in W \mid w \vDash^- \phi\}$. Un séquent $\phi \vdash_{\mathsf{BD}} \chi$ est *valide sur* $\mathfrak{M} = \langle W, v^+, v^- \rangle$ ssi $\forall w \in W$, on a : si $w \vDash^+ \phi$, alors $w \vDash^+ \chi$, et si $w \vDash^- \chi$, alors $w \vDash^- \phi$. Un séquent $\phi \vdash_{\mathsf{BD}} \chi$ est *universellement valide* ssi il est valide sur chaque modèle.

Probabilités non-standards. Dans [6], les auteurs introduisent la notion de BD-modèles probabilistes $\mathfrak{M} = \langle W, \mu, v^+, v^- \rangle$, avec $\langle W, v^+, v^- \rangle$ un BD-modèle et $\mu : \mathcal{P}(W) \to [0, 1]$ une mesure de probabilité. Cela permet de

définir une mesure de probabilité sur les formules de la logique BD : $\mathtt{p}^+_\mu(\varphi) = \mu(|\varphi|^+)$, $\mathtt{p}^-_\mu(\varphi) = \mu(|\varphi|^-)$. Comme $\mathtt{p}^+_\mu$ et $\mathtt{p}^-_\mu$ sont liées : $\mathtt{p}^-_\mu(\varphi) = \mu(|\varphi|^-) = \mu(|\neg\varphi|^+) = \mathtt{p}^+_\mu(\neg\varphi)$, il suffit de travailler avec $\mathtt{p}^+_\mu$. [6] montre que la fonction $\mathtt{p}^+_\mu$ satisfait les propriétés (i) $0 \leq \mathtt{p}(x) \leq 1$, (ii) si $\varphi \vdash_{\mathsf{BD}} \psi$, alors $\mathtt{p}(\varphi) \leq \mathtt{p}(\psi)$, (iii) $\mathtt{p}(\varphi \vee \psi) = \mathtt{p}(\varphi) + \mathtt{p}(\psi) - \mathtt{p}(\varphi \wedge \psi)$, et que pour chaque fonction $\mathtt{p}$ sur $\mathscr{L}_{\mathsf{BD}}$ vérifiant (i)–(iii), il existe un modèle probabiliste $\langle W, \mu, v^+, v^- \rangle$ tel que $\mathtt{p}(\varphi) = \mu(|\varphi|)$. Cela nous permet de considérer (i)–(iii) comme une axiomatisation des fonctions de probabilité sur BD. Les fonctions $\mathtt{p} : \mathscr{L}_{\mathsf{BD}} \to \mathbb{R}$ satisfiant (i)–(iii) sont appelées des probabilités non-standards. Dans [3], nous fournissons un cadre logique (sémantique et axiomatisation), basé sur [4], qui permet de raisonner à la fois avec les probabilités non-standards et leurs généralisations sous forme de fonctions de croyance et de plausibilité.

Fonctions de croyances sur les formules de Belnap–Dunn. Nous introduisons les modèles de Dempster-Shafer $\mathscr{M} = \langle S, \mathcal{P}(S), \mathtt{bel}, v^+, v^- \rangle$ où $\langle S, v^+, v^- \rangle$ est un BD-modèle et $\mathtt{bel}$ est une fonction de croyance sur $\mathcal{P}(S)$. $\mathtt{bel} : \mathcal{P}(S) \to [0,1]$ est une fonction de croyance si $\mathtt{bel}$ est une fonction croissante telle que $\mathtt{bel}(\emptyset) = 0$ et $\mathtt{bel}(S) = 1$ et pour chaque $k \geq 1$ et chaque $a_1, \ldots, a_k \in \mathcal{L}$, on a

$$\mathtt{bel}\left(\bigcup_{1 \leq i \leq k} a_i\right) \leq \sum_{\substack{J \subseteq \{1,\ldots,k\} \\ J \neq \emptyset}} (-1)^{|J|+1} \cdot \mathtt{bel}\left(\bigcap_{j \in J} a_j\right).$$

Rappelons qu'une *logique* est un tuple $\mathsf{L} = \langle \mathscr{L}, \vdash \rangle$ avec $\mathscr{L}$ un langage sur $\{\circ_1, \ldots, \circ_n\}$ et $\vdash \subseteq \mathcal{P}(\mathscr{L}) \times \mathscr{L}$. Une *algèbre de Lindenbaum de* L est un tuple $\langle \mathscr{L}/_{\dashv\vdash}, \bullet_1, \ldots, \bullet_n \rangle$ où pour chaque $i \in \{1, \ldots, n\}$ et chaque $\phi, \phi' \in \mathscr{L}$, on a $[\phi \circ_i \phi'] = [\phi] \bullet_i [\phi']$ avec $[\phi]$ la classe d'équivalence de ϕ pour $\dashv\vdash$. Soit $\mathcal{L}_{\mathsf{BD}}$ l'algèbre de Lindenbaum pour la logique BD sur l'ensemble des variables propositionnelles $\mathtt{Prop}$ (i.e. l'algèbre de De Morgan libre sur $\mathtt{Prop}$). On dénote $\mathtt{bel}^+_{\mathscr{M}} : \mathcal{L}_{\mathsf{BD}} \to [0,1]$ and $\mathtt{bel}^-_{\mathscr{M}} : \mathcal{L}^{op}_{\mathsf{BD}} \to [0,1]$ les fonctions telles que, pour chaque $\varphi \in \mathcal{L}_{\mathsf{BD}}$,

$$\mathtt{bel}^+_{\mathscr{M}}(\varphi) = \mathtt{bel}(|\varphi|^+) \text{ et } \mathtt{bel}^-_{\mathscr{M}}(\varphi) = \mathtt{bel}(|\varphi|^-).$$

Cette définition permet d'interpréter les fonctions de croyances sur les algèbres de De Morgan au lieu des algèbres de Boole : i.e. sur les treillis distributifs munis d'une négation involutive satisfaisant les lois de De Morgan. Une conséquence directe de cette interprétation se retrouve dans le comportement de la règle de Dempster–Shafer pour combiner des fonctions de croyances. Rappelons qu'une fonction de croyance $\mathtt{bel} : \mathcal{P}(S) \to [0,1]$ peut être représentée de façon équivalente par sa fonction de masse $\mathtt{m} : \mathcal{P}(S) \to [0,1]$ telle que $\mathtt{bel}(X) = \sum_{Y \subseteq X} \mathtt{m}(Y)$ et $\sum_{X \in \mathcal{P}(S)} \mathtt{m}(X) = 1$. La règle de Dempster–Shafer agrège l'information provenant de deux fonctions de masses $\mathtt{m}_1$ et $\mathtt{m}_2$ via la fonction de masse $\mathtt{m}_{1\oplus 2} : \mathcal{P}(S) \to [0,1]$ telle que

$$\mathtt{m}_{1\oplus 2}(X) = \begin{cases} 0 \text{ si } X = \emptyset \\ \dfrac{\sum\{\mathtt{m}_1(X_1) \cdot \mathtt{m}_2(X_2) \mid X_1 \cap X_2 = X\}}{\sum\{\mathtt{m}_1(X_1) \cdot \mathtt{m}_2(X_2) \mid X_1 \cap X_2 \neq \emptyset\}} \text{ sinon.} \end{cases}$$

Sur les algèbres de De Morgan, la règle reste inchangée, il suffit d'interpréter $\emptyset$ comme $\bot$ et $\cap$ comme $\wedge$. La différence est que sur les algèbres de De Morgan, $p \wedge \neg p = \bot$ n'est pas un axiome. Ainsi, il devient possible d'agréger les fonctions de masse $\mathtt{m}_1$ et $\mathtt{m}_2$ telles que $\mathtt{m}_1(p \wedge q) = 1$ et $\mathtt{m}_2(\neg p \wedge q) = 1$ en gardant l'information qu'elles sont en conflit concernant p. On obtiendra $\mathtt{m}_{1\oplus 2}(p \wedge \neg p \wedge q) = 1$. C'est-à-dire les opinions représentées par $\mathtt{m}_1$ et $\mathtt{m}_2$ sont en contradiction au sujet de p, mais en accord au sujet de q. Travailler sur les algèbres de De Morgan et donc sur la logique de BD permet de représenter de façon fine les informations contradictoires et donc de savoir sur quels sujets exactement des sources sont en désaccord plutôt que d'ignorer le conflit.

Nous travaillons en ce moment sur les différentes façons de définir les fonctions de plausibilité et leurs interprétations. Cela s'avère plus compliqué. Il est, par exemple, possible de définir $\mathtt{pl}^+_{\mathscr{M}}(\phi) := 1 - \mathtt{bel}^+_{\mathscr{M}}(\neg\phi)$, mais, comme $\neg$ n'est pas la négation classique, il est possible d'avoir $\mathtt{pl}^+_{\mathscr{M}}(\phi) < \mathtt{bel}^+_{\mathscr{M}}(\phi)$, contrairement au cas classique où $\mathtt{bel}(X)$ et $\mathtt{pl}(X)$ fournissent respectivement la borne inférieure et supérieure de la probabilité qu'un élément appartienne à X.

Références

[1] Baldi, P., P. Cintula et C. Noguera: *Classical and Fuzzy Two-Layered Modal Logics for Uncertainty : Translations and Proof-Theory*. Int. Journal of Computational Intelligence Systems, 13(1) :988–1001, 2020.

[2] Belnap, N.D.: *How a Computer Should Think*. Dans Omori, H. et H. Wansing (rédacteurs) : *New Essays on Belnap-Dunn Logic*, tome 418 de *Synthese Library (Studies in Epistemology, Logic, Methodology, and Philosophy of Science)*. Springer, Cham, 2019.

[3] Bílková, M., S. Frittella, D. Kozhemiachenko, O. Majer et S. Nazari: *Reasoning with belief functions over Belnap–Dunn logic*. preprint, arXiv :2203.01060, 2022.

[4] Bílková, M., S. Frittella, O. Majer et S. Nazari: *Belief based on inconsistent information*. Dans *Int. Workshop on Dynamic Logic*, pages 68–86. Springer, 2020.

[5] Fagin, R., J. Y. Halpern et N. Megiddo: *A logic for reasoning about probabilities*. Information and Computation, 87 :78–128, 1990.

[6] Klein, D., O. Majer et S. Rafiee Rad: *Probabilities with gaps and gluts*. Journal of Philosophical Logic, 50(5) :1107–1141, octobre 2021.

Actes JIAF 2021

Advanced languages of terms for ontologies

Philippe Balbiani[1] **Çiğdem Gencer**[1,2]

[1]Toulouse Institute of Computer Science Research
CNRS-INPT-UT3, Toulouse University
Toulouse, France
[2]Faculty of Arts and Sciences
Istanbul Aydın University
Istanbul, Turkey

philippe.balbiani@irit.fr cigdem.gencer@irit.fr cigdemgencer@aydin.edu.tr

Abstract

This paper is about the integration in a unique formalism of knowledge representation languages such as those provided by description logic languages and rule-based reasoning paradigms such as those provided by logic programming languages. We aim at creating an hybrid formalism where description logics constructs are used for defining concepts that are given as arguments to the predicates of the logic programs.

Résumé

Dans cet article, nous considérons le problème de l'intégration dans un unique formalisme des langages de représentation des connaissances comme ceux de la logique de description et des paradigmes de raisonnement à base de règles comme ceux de la programmation en logique. Notre objectif est la création d'un formalisme hybride dans lequel les connecteurs de la logique de description sont utilisés pour définir des concepts qui sont donnés en argument des prédicats de la programmation en logique.

1 A short introduction

A crucial issue in the development of the semantic web is the possibility to combine rule-based systems and ontologies. There exists already several types of such combination [Eiter *et al.* (2011a), Eiter *et al.* (2011b), Levy and Rousset (1998), Motik and Rosati (2017)]. These approaches either build rules on top of ontologies allowing rule-based systems to use the vocabulary specified in ontologies, or build ontologies on top of rules supplementing ontological definitions by rules. None of them completely answer to the question of the combination of logic programming with description logics that we are seeking for : an hybrid formalism where description logics constructs are used for defining concepts that are given as arguments to the predicates of the logic programs. In this paper, we develop such an hybrid formalism.

This paper is organized as follows. A case study motivating the combination of logic programming with description logics that we are seeking for is presented in Section 2. In Sections 3 and 4, we introduce the syntax and the semantics of our hybrid formalism. Decision problems are presented in Sections 5, 6 and 7. In Section 8, we introduce examples. A research program is presented in Section 9.

2 A case study

Examining role-based access control and organization-based access control, we present a case study motivating the combination of logic programming with description logics that we are seeking for.

Access of subjects to objects in a computer system are permitted in accordance with a security policy embodied in an access control database. Many computer systems use the access control matrix model to represent security policies [Lampson (1974)].

Formally, an access control matrix is a structure consisting of a set of subjects (users, processes, etc), a set of objects (files, tables, etc) and binary relations $(p_i)_{i\in I}$ between objects and subjects giving to subjects the permission to access objects. In this setting, asserting that subject a possesses permission p_i on object b comes down to asserting that p_i holds for b and a.

Access control with a lot of subjects is space-consuming. To reduce the cost of security, within the context of role-based access control (RBAC), it has been proposed that access control administrators treat sets of subjects as instances of a concept called role [1] [Sandhu *et al.* (1996)]. Formally, an RBAC-structure consists of a set of subjects, a set of objects, a set of roles, a binary relation r between subjects and roles defining the roles of subjects and binary relations $(p_i)_{i\in I}$ between objects and roles giving to roles the permission to access objects. In this setting, asserting that subject a has role A comes down to asserting that r holds for a and A, whereas asserting that role A possesses permission p_i on object b comes down to asserting that p_i holds for b and A. It is possible to refine the RBAC model by including the concept of role hierarchy which allows permissions to be inherited through it. This hierarchy is specified by means of assertions of the form $A'\sqsubseteq A''$ where A' and A'' are roles. To put it simply, the idea behind RBAC is the following : in a computer system, subject a possesses a permission p on object b if and only if there are roles $A_0, \ldots, A_m$ such that r holds for a and A_0, for all positive integers $i\leq m$, $A_{i-1}\sqsubseteq A_i$ has been asserted and p holds for b and A_m.

RBAC with a lot of objects is space-consuming. To reduce the cost of security, within the context of organization-based access control (OrBAC), it has been proposed that RBAC administrators treat sets of objects as instances of a concept called view [Abou El Kalam *et al.* (2003)]. Formally, an OrBAC-structure consists of a set of subjects, a set of objects, a set of roles, a set of views, a binary relation r between subjects and roles defining the roles of subjects, a binary relation v between objects and views defining the views of objects and binary relations $(p_i)_{i\in I}$ between views and roles giving to roles the permission to access views. In this setting, asserting that object b has view B comes down to asserting that v holds for b and B, whereas asserting that role A possesses permission p_i on view B comes down to asserting that p_i holds for B and A. It is possible to refine the OrBAC model by including the concept of view hierarchy which allows permissions to be inherited through it. This hierarchy is specified by means of assertions of the form $B'\sqsubseteq B''$ where B' and B'' are views. To put it simply, the idea behind OrBAC is the following : in a computer system, subject a possesses a permission p on object b if and only if there are roles $A_0, \ldots, A_m$ and there are views $B_0, \ldots, B_n$ such that r holds for a and A_0 and v holds for b and B_0, for all positive integers $i\leq m$, $A_{i-1}\sqsubseteq A_i$ has been asserted and for all positive integers $j\leq n$, $B_{j-1}\sqsubseteq B_j$ has been asserted and p holds for B_n and A_m.

It is a great pity that neither RBAC, nor OrBAC allow atomic assertions of the form $p_i(D, C)$ where C and D are, respectively, Boolean combinations of roles and Boolean combinations of views. By using assertions of that form, one may more succinctly define more precise access control policies. For instance, to say that subjects having the role A but not having the role A' possess a permission p_i on objects having the view B but not having the view B', one can simply assert that p_i holds for $B \wedge \neg B'$ and $A \wedge \neg A'$ instead of asserting that p_i holds for B'' and A'' where A'' is a new role such that for all subjects a, $r(a, A'')$ if and only if $r(a, A)$ and not $r(a, A')$ and B'' is a new view such that for all objects b, $v(b, B'')$ if and only if $v(b, B)$ and not $v(b, B')$.

Finally, it is also a great pity that neither RBAC, nor OrBAC allow conditional assertions of the form $p_i(D, C)\leftarrow p_j(D', C')$. By using conditional assertions of that form, one may more succinctly define more precise access control policies. For instance, to say that subjects having the role C possess a permission p_i on objects having the view D if subjects having the role C' possess a permission p_j on objects having the view D', one can simply say that $p_i(D, C)\leftarrow p_j(D', C')$. This is particularly interesting when p_j does not denote a permission, but an obligation corresponding to the permission denoted by p_i [2]. In that case, a conditional assertion like $p_i(D, C)\leftarrow p_j(D, C)$ expresses the deontic rule saying that subjects having the role C possess the permission p_i on objects having the view D if subjects having the role C possess the corresponding obligation p_j on objects having the view D.

Knowledge representation languages such as those provided by description logic languages [Baader *et*

1. The roles in RBAC should not be mistaken for the roles in description logics. In RBAC security policies, roles correspond to sets of subjects, whereas in description logic frames, roles correspond to binary relations.

2. We are assuming the deontic principle saying that permissions are implied by their corresponding obligations [Parent and van der Torre (2018)].

al. (2003)] (allowing expressions of the form $C \sqsubseteq D$ where C and D are complex concepts) and rule-based reasoning paradigms such as those provided by logic programming languages [Gabbay *et al.* (1998), Lloyd (1987)] (allowing expressions of the form $\alpha \leftarrow \beta_1, \ldots, \beta_n$ where α, β_1, ..., β_n are atoms) are well-known and widely used in Computer Science and Artificial Intelligence. Their integration in a unique formalism would be a natural solution for many application problems requiring the following features : allowing rule-based systems to use the vocabulary specified in ontologies and supplementing ontological definitions by rules. Hybrid knowledge bases are the main approaches proposed so far. They integrate some aspects of description logic and some aspects of logic programming [Eiter *et al.* (2011a), Eiter *et al.* (2011b), Levy and Rousset (1998), Motik and Rosati (2017)]. Nevertheless, they hardly address all aspects of our aim : the development of an hybrid formalism where description logics constructs are used for defining concepts that are given as arguments to the predicates of the logic programs.

3 Syntax

We introduce the syntax of our hybrid formalism.

Complex concepts Let **VAR** be a countable set of *variable concepts* (with typical members denoted X, Y, etc). Let **CON** be a countable set of *constant concepts* (with typical members denoted A, B, etc) and **ROL** be a countable set of *constant roles* (with typical members denoted R, S, etc). The set of *complex concepts* (with typical members denoted C, D, etc) is defined by the rule [3]

— $C ::= X \mid A \mid \top \mid (C \sqcap D) \mid \exists R.C$,

where X ranges over **VAR**, A ranges over **CON** and R ranges over **ROL**. We adopt standard rules for omission of the parentheses. A complex concept C is **VAR**-*free* if C contains no occurrence of a variable concept. A complex concept C is **ROL**-*free* if C contains no occurrence of a constant role. For all $k \in \mathbb{N}$, the concept construct $(\exists R.)^k$ is inductively defined as follows for each $R \in$ **ROL** :

— if $k=0$ then $(\exists R.)^k C ::= C$,

— otherwise, $(\exists R.)^k C ::= \exists R.(\exists R.)^{k-1} C$.

3. The set of complex concepts we define here is the one of description logic $\mathcal{EL}$ [Baader and Nutt (2003)]. Most of our definitions can be easily adapted to cases where other description logics are considered instead of description logic $\mathcal{EL}$ [Baader and Lutz (2007), Baader and Nutt (2003), Calvanese and De Giacomo (2003)].

Substitutions A *substitution* is a function from **VAR** to the set of all complex concepts almost everywhere equal to the identity function [Baader and Snyder (2001)]. To apply a substitution σ to a complex concept C amounts to replace each occurrence in C of a variable concept $X \in$ **VAR** by the corresponding complex concept $\sigma(X)$.

Inclusions and equations *Concept inclusions* are expressions of the form $C \sqsubseteq D$ (read "C is contained in D") for each complex concepts C, D. *Concept equations* are expressions of the form $C = D$ (read "C is equal to D") for each complex concepts C, D.

Clauses Let **PRE** be a countable set of *predicate symbols* (with typical members denoted p, q, etc). For all $p \in$ **PRE**, let $\mathtt{ar}(p)$ be the *arity* of p. An *atom* is an expression of the form $p(C_1, \ldots, C_{\mathtt{ar}(p)})$ (read "p holds for C_1, ..., $C_{\mathtt{ar}(p)}$") where p is a predicate symbol and C_1, ..., $C_{\mathtt{ar}(p)}$ are complex concepts. *Clauses* are expressions of the form $\alpha_1, \ldots, \alpha_m \leftarrow \beta_1, \ldots, \beta_n$ (read "if β_1, ..., β_n then either α_1, ..., or α_m") where α_1, ..., α_m, β_1, ..., β_n are atoms. *Definite clauses* are clauses of the form $\alpha \leftarrow \beta_1, \ldots, \beta_n$, *unit clauses* are clauses of the form $\alpha \leftarrow$ and *definite goals* are clauses of the form $\leftarrow \beta_1, \ldots, \beta_n$.

Assertions Let **IND** be a countable set of *individual constants* (with typical members denoted a, b, etc). A *concept assertion* is an expression of the form $C{:}a$ (read "a belongs to C") where C is a **VAR**-free complex concept and a is an individual constant. A *role assertion* is an expression of the form $R{:}(a, b)$ (read "a is R-related to b") where $R \in$ **ROL** and a and b are individual constants.

Deductive ontologies A *T-box* is a finite set of concept inclusions and concept equations. A *program* is a finite set of clauses. An *A-box* is a finite set of concept assertions and role assertions. A *deductive ontology* is a triple $(\mathcal{T}, \Pi, \mathcal{A})$ consisting of a T-box $\mathcal{T}$, a program Π and an A-box $\mathcal{A}$.

4 Semantics

We introduce the semantics of our hybrid formalism [4].

4. In this paper, for all sets E, $\mathcal{P}(E)$ denotes the set of all subsets of E, $E^\star$ denotes the set of all tuples of elements of E and for all $k \in \mathbb{N}$, E^k denotes the set of all k-tuples of elements of E.

Frames and var-interpretations The semantics is defined in terms of *frames,* i.e. structures (W, K, Rel) where W is a nonempty set, $K : \mathbf{CON} \longrightarrow \mathcal{P}(W)$ and $Rel : \mathbf{ROL} \longrightarrow \mathcal{P}(W \times W)$. In a frame (W, K, Rel), for all $R \in \mathbf{ROL}$,

— the *R-image* of a subset S of W is the set of all $t \in W$ such that there exists $s \in S$ such that $Rel(R)(s, t)$,

— the *R-pre-image* of a subset T of W is the set of all $s \in W$ such that there exists $t \in T$ such that $Rel(R)(s, t)$,

— the *domain* of R is the set of all $s \in W$ such that there exists $t \in W$ such that $Rel(R)(s, t)$,

— the *range* of R is the set of all $t \in W$ such that there exists $s \in W$ such that $Rel(R)(s, t)$.

Obviously, in a frame (W, K, Rel), for all $R \in \mathbf{ROL}$, the domain of R is the R-pre-image of W and the range of R is the R-image of W. A *var-interpretation* on a frame (W, K, Rel) is a function $V : \mathbf{VAR} \longrightarrow \mathcal{P}(W)$. For all frames (W, K, Rel), the value of the complex concept C with respect to a var-interpretation V on (W, K, Rel) is the subset $\|C\|_V$ of W defined by

— $\|X\|_V = V(X)$,

— $\|A\|_V = K(A)$,

— $\|\top\|_V = W$,

— $\|C \sqcap D\|_V = \|C\|_V \cap \|D\|_V$,

— $\|\exists R.C\|_V = \{s \in W :$ there exists $t \in W$ such that $Rel(R)(s, t)$ and $t \in \|C\|_V\}$.

Obviously, $\|C\|_V$ does not depend on V when C is **VAR**-free. In that case, $\|C\|_V$ will be denoted $\|C\|$.

Pre-interpretations A *pre-interpretation* on a frame (W, K, Rel) is a function $I : \mathbf{PRE} \longrightarrow \mathcal{P}(\mathcal{P}(W)^\star)$ such that for all $p \in \mathbf{PRE}$, $I(p) \subseteq \mathcal{P}(W)^{\mathtt{ar}(p)}$. For all frames (W, K, Rel) and for all pre-interpretations I on (W, K, Rel), the *value* of an atom $p(C_1, \ldots, C_{\mathtt{ar}(p)})$ with respect to a var-interpretation V on (W, K, Rel) is the element $|p(C_1, \ldots, C_{\mathtt{ar}(p)})|^I_V$ in $\{0, 1\}$ such that

— if $I(p)$ contains $(\|C_1\|_V, \ldots, \|C_{\mathtt{ar}(p)}\|_V)$ then $|p(C_1, \ldots, C_{\mathtt{ar}(p)})|^I_V = 1$,

— otherwise, $|p(C_1, \ldots, C_{\mathtt{ar}(p)})|^I_V = 0$.

Ind-interpretations An *ind-interpretation* on a frame (W, K, Rel) is a function $g : \mathbf{IND} \longrightarrow W$. For all frames (W, K, Rel) and for all ind-interpretations g on (W, K, Rel), the *value* of a concept assertion $C{:}a$ is the element $|C{:}a|^g$ in $\{0, 1\}$ such that

— if $\|C\|$ contains $g(a)$ then $|C{:}a|^g = 1$,

— otherwise, $|C{:}a|^g = 0$,

and the *value* of a role assertion $R{:}(a, b)$ is the element $|R{:}(a, b)|^g$ in $\{0, 1\}$ such that

— if $Rel(R)$ contains $(g(a), g(b))$ then $|R{:}(a, b)|^g = 1$,

— otherwise, $|R{:}(a, b)|^g = 0$.

Models For all T-boxes $\mathcal{T}$, a $\mathcal{T}$*-model* (or a *model* of $\mathcal{T}$) is a frame (W, K, Rel) such that for all var-interpretations V on (W, K, Rel),

— for all concept inclusions $C \sqsubseteq D$ in $\mathcal{T}$, $\|C\|_V \subseteq \|D\|_V$,

— for all concept equations $C = D$ in $\mathcal{T}$, $\|C\|_V = \|D\|_V$.

For all deductive ontologies $(\mathcal{T}, \Pi, \mathcal{A})$, a $(\mathcal{T}, \Pi, \mathcal{A})$-*model* (or a *model* of $(\mathcal{T}, \Pi, \mathcal{A})$) is a structure (W, K, Rel, I, g) consisting of a $\mathcal{T}$-model (W, K, Rel), a pre-interpretation I on (W, K, Rel) and an ind-interpretation g on (W, K, Rel) such that for all var-interpretations V on (W, K, Rel),

— for all clauses $\alpha_1, \ldots, \alpha_m \leftarrow \beta_1, \ldots, \beta_n$ in Π, if $|\beta_1|^I_V = 1$, …, $|\beta_n|^I_V = 1$ then either $|\alpha_1|^I_V = 1$, …, or $|\alpha_m|^I_V = 1$,

— for all concept assertions $C{:}a$ in $\mathcal{A}$, $|C{:}a|^g = 1$,

— for all role assertions $R{:}(a, b)$ in $\mathcal{A}$, $|R{:}(a, b)|^g = 1$.

Notice that in a model (W, K, Rel, I, g) of a deductive ontology $(\mathcal{T}, \Pi, \mathcal{A})$, for all var-interpretations V on (W, K, Rel),

— for all definite clauses $\alpha \leftarrow \beta_1, \ldots, \beta_n$ in Π, if $|\beta_1|^I_V = 1$, …, $|\beta_n|^I_V = 1$ then $|\alpha|^I_V = 1$,

— for all unit clauses $\alpha \leftarrow$ in Π, $|\alpha|^I_V = 1$,

— for all definite goals $\leftarrow \beta_1, \ldots, \beta_n$ in Π, either $|\beta_1|^I_V = 0$, …, or $|\beta_n|^I_V = 0$.

5 Correspondence theory

We briefly present the correspondence theory of our hybrid formalism.

Although of limited expressive power, concept constructs can be used for characterizing classes of frames. As observed by [Baader and Lutz (2007), Schild (1991)], description logic languages are modal languages in disguise. Therefore, the following relationships that can be easily established for all frames (W, K, Rel) will not come as a surprise :

(1) (W, K, Rel) is a model of $X \sqsubseteq \exists R.\top$ if and only if $Rel(R)$ is serial [5],

(2) (W, K, Rel) is a model of $\exists R.\top \sqsubseteq X$ if and only if $Rel(R)$ is empty,

(3) (W, K, Rel) is a model of $X \sqsubseteq \exists R.X$ if and only if $Rel(R)$ is reflexive [6],

(4) (W, K, Rel) is a model of $\exists R.X \sqsubseteq X$ if and only if $Rel(R)$ is included in the identity relation on W,

5. That is to say, for all $s \in W$, there exists $t \in W$ such that $Rel(R)(s, t)$.

6. That is to say, for all $s \in W$, $Rel(R)(s, s)$.

(**5**) (W, K, Rel) is a model of $\exists R.X \sqsubseteq \exists R.\exists R.X$ if and only if $Rel(R)$ is dense [7],

(**6**) (W, K, Rel) is a model of $\exists R.\exists R.X \sqsubseteq \exists R.X$ if and only if $Rel(R)$ is transitive [8],

(**7**) (W, K, Rel) is a model of $\exists R.\exists R.\top \sqsubseteq X$ if and only if the R-pre-image of the R-pre-image of W is empty,

(**8**) (W, K, Rel) is a model of $\exists R.X = \exists S.X$ if and only if $Rel(R)$ is equal to $Rel(S)$,

(**9**) (W, K, Rel) is a model of $A \sqcap B \sqsubseteq X$ if and only if $K(A)$ and $K(B)$ do not intersect,

(**10**) (W, K, Rel) is a model of $A \sqsubseteq B$ if and only if $K(A)$ is included in $K(B)$,

(**11**) (W, K, Rel) is a model of $\exists R.X \sqsubseteq A$ if and only if the domain of R is included in $K(A)$,

(**12**) (W, K, Rel) is a model of $\exists R.X \sqsubseteq \exists R.(X \sqcap A)$ if and only if the range of R is included in $K(A)$.

Within our setting, elementary conditions — like "$Rel(R)$ is serial", "$Rel(R)$ is empty", etc — are first-order conditions that can be expressed as sentences in a function-free first-order language with equality based on a set of unary predicate symbols in one-to-one correspondence with **CON** and a set of binary predicate symbols in one-to-one correspondence with **ROL**. As a result, the following decision problems are of interest :

— **deciding elementary definability (DED)**

input : a T-box $\mathcal{T}$,

output : determine whether there exists an elementary condition F such that for all frames (W, K, Rel), F holds in (W, K, Rel) if and only if (W, K, Rel) is a model of $\mathcal{T}$,

— **deciding concept definability (DCD)**

input : an elementary condition F,

output : determine whether there exists a T-box $\mathcal{T}$ such that for all frames (W, K, Rel), (W, K, Rel) is a model of $\mathcal{T}$ if and only if F holds in (W, K, Rel),

— **deciding elementary equivalence (DEE)**

input : a T-box $\mathcal{T}$ and an elementary condition F,

output : determine whether for all frames (W, K, Rel), (W, K, Rel) is a model of $\mathcal{T}$ if and only if F holds in (W, K, Rel).

DED, **DCD** and **DEE** stem from the corresponding definability problems in modal logics [Chagrov and Chagrova (2006)]. It is not known whether **DED**, **DCD** and **DEE** are decidable [9].

6 Deciding inclusions and equations

We present decision problems about concept inclusions and concept equations.

Let $\mathcal{T}$ be a T-box.

A concept inclusion $C \sqsubseteq D$ is a *logical consequence* of $\mathcal{T}$ (denoted $\mathcal{T} \models C \sqsubseteq D$) if for all $\mathcal{T}$-models (W, K, Rel) and for all var-interpretations V on (W, K, Rel), $\|C\|_V \subseteq \|D\|_V$. A concept equation $C = D$ is a *logical consequence* of $\mathcal{T}$ (denoted $\mathcal{T} \models C = D$) if for all $\mathcal{T}$-models (W, K, Rel) and for all var-interpretations V on (W, K, Rel), $\|C\|_V = \|D\|_V$. As a result, the following decision problems are of interest :

— **deciding concept inclusions (DCI)**

input : a concept inclusion $C \sqsubseteq D$,

output : determine whether $\mathcal{T} \models C \sqsubseteq D$,

— **deciding concept equations (DCE)**

input : a concept equation $C = D$,

output : determine whether $\mathcal{T} \models C = D$.

If $\mathcal{T}$ is **VAR**-free then **DCI** and **DCE** are in **P** [Baader (2003)] [10]. Otherwise, it is not known whether **DCI** and **DCE** are decidable.

7 Deciding consequences and answers

We present decision problems about logical consequences and correct answers.

Let $(\mathcal{T}, \Pi, \mathcal{A})$ be a deductive ontology.

A clause $\alpha_1, \ldots, \alpha_m \leftarrow \beta_1, \ldots, \beta_n$ is a *logical consequence* of $(\mathcal{T}, \Pi, \mathcal{A})$ (denoted $(\mathcal{T}, \Pi, \mathcal{A}) \models \alpha_1, \ldots, \alpha_m \leftarrow \beta_1, \ldots, \beta_n$) if for all $(\mathcal{T}, \Pi, \mathcal{A})$-models (W, K, Rel, I, g) and for all var-interpretations V on (W, K, Rel), if $|\beta_1|^I_V = 1, \ldots, |\beta_n|^I_V = 1$ then either $|\alpha_1|^I_V = 1, \ldots$, or $|\alpha_m|^I_V = 1$. Notice that a definite clause $\alpha \leftarrow \beta_1, \ldots, \beta_n$ is a logical consequence of $(\mathcal{T}, \Pi, \mathcal{A})$ if and only if for all $(\mathcal{T}, \Pi, \mathcal{A})$-models (W, K, Rel, I, g) and for all var-interpretations V on (W, K, Rel), if $|\beta_1|^I_V = 1, \ldots, |\beta_n|^I_V = 1$ then $|\alpha|^I_V = 1$, a unit clause $\alpha \leftarrow$ is a logical consequence of $(\mathcal{T}, \Pi, \mathcal{A})$ if and only if if for all $(\mathcal{T}, \Pi, \mathcal{A})$-models (W, K, Rel, I, g) and for all var-interpretations V on (W, K, Rel), $|\alpha|^I_V = 1$ and a definite goal $\leftarrow \beta_1, \ldots, \beta_n$ is a logical consequence

7. That is to say, for all $s, t \in W$, if $Rel(R)(s, t)$ then there exists $u \in W$ such that $Rel(R)(s, u)$ and $Rel(R)(u, t)$.

8. That is to say, for all $s, t \in W$, if there exists $u \in W$ such that $Rel(R)(s, u)$ and $Rel(R)(u, t)$ then $Rel(R)(s, t)$.

9. Description logic languages being modal languages in disguise [Baader and Lutz (2007), Schild (1991)], the undecidability of **DED**, **DCD** and **DEE** are immediate consequences of Chagrova's Theorems [Chagrov and Chagrova (2006)] when description logic $\mathcal{ALC}$ is considered instead of description logic $\mathcal{EL}$.

10. See [Donini (2003)] when other description logics are considered instead of description logic $\mathcal{EL}$.

of $(\mathcal{T}, \Pi, \mathcal{A})$ if and only if for all $(\mathcal{T}, \Pi, \mathcal{A})$-models (W, K, Rel, I, g) and for all var-interpretations V on (W, K, Rel), either $|\beta_1|^I_V$=0, ..., or $|\beta_n|^I_V$=0. As a result, the following decision problems are of interest :

— **deciding definite clauses (DDC)**

input : a definite clause $\alpha \leftarrow \beta_1, \ldots, \beta_n$,

output : determine whether $(\mathcal{T}, \Pi, \mathcal{A}) \models \alpha \leftarrow \beta_1, \ldots, \beta_n$,

— **deciding unit clauses (DUC)**

input : a unit clause $\alpha \leftarrow$,

output : determine whether $(\mathcal{T}, \Pi, \mathcal{A}) \models \alpha \leftarrow$,

— **deciding definite goals (DDG)**

input : a definite goal $\leftarrow \beta_1, \ldots, \beta_n$,

output : determine whether $(\mathcal{T}, \Pi, \mathcal{A}) \models \leftarrow \beta_1, \ldots, \beta_n$.

A substitution σ is a *correct answer* for the definite goal $\leftarrow \beta_1, \ldots, \beta_n$ with respect to $(\mathcal{T}, \Pi, \mathcal{A})$ if for all $(\mathcal{T}, \Pi, \mathcal{A})$-models (W, K, Rel, I, g) and for all var-interpretations V on (W, K, Rel), $|\sigma(\beta_1)|^I_V$=1, ..., $|\sigma(\beta_n)|^I_V$=1. As a result, the following decision problem is of interest :

— **deciding correct answers (DCA)**

input : a definite goal $\leftarrow \beta_1, \ldots, \beta_n$,

output : determine whether there exists a correct answer for $\leftarrow \beta_1, \ldots, \beta_n$ with respect to $(\mathcal{T}, \Pi, \mathcal{A})$.

DDC, **DUC**, **DDG** and **DCA** stem from the corresponding derivability problems in logic programming [Gabbay *et al.* (1998), Lloyd (1987)]. It is not known whether **DDC**, **DUC**, **DDG** and **DCA** are decidable [11].

8 Examples

We introduce examples.

An example about unbounded recursion Let $(\mathcal{T}_1, \Pi_1, \mathcal{A}_1)$ be the deductive ontology where

— $\mathcal{T}_1$ is the empty T-box,

— Π_1 is the program containing the following clauses :

— $p(\top) \leftarrow$,

— $p(\exists R.X) \leftarrow p(X)$,

— $\mathcal{A}_1$ is the empty A-box.

Models of $\mathcal{T}_1$ are arbitrary frames. Models of $(\mathcal{T}_1, \Pi_1, \mathcal{A}_1)$ are structures (W, K, Rel, I, g) consisting of an arbitrary frame (W, K, Rel), a pre-interpretation I on (W, K, Rel) such that for all subsets U of W,

— W is in $I(p)$,

— if U is in $I(p)$ then the R-pre-image of U is in $I(p)$,

and an arbitrary ind-interpretation g on (W, K, Rel). It follows that for all complex concepts C, $(\mathcal{T}_1, \Pi_1, \mathcal{A}_1) \models p(C) \leftarrow$ if and only if there exists $k \in \mathbb{N}$ such that $\mathcal{T}_1 \models C = (\exists R.)^k \top$.

An example about bounded recursion Let $(\mathcal{T}_2, \Pi_2, \mathcal{A}_2)$ be the deductive ontology where

— $\mathcal{T}_2$ is the T-box containing the following concept inclusion :

— $\exists R.\exists R.\top \sqsubseteq X$,

— Π_2 is the program containing the following clauses :

— $p(\top) \leftarrow$,

— $p(\exists R.X) \leftarrow p(X)$,

— $\mathcal{A}_2$ is the empty A-box.

Models of $\mathcal{T}_2$ are frames (W, K, Rel) such that [12]

— the R-pre-image of the R-pre-image of W is empty.

Models of $(\mathcal{T}_2, \Pi_2, \mathcal{A}_2)$ are structures (W, K, Rel, I, g) consisting of a frame (W, K, Rel) such that the R-pre-image of the R-pre-image of W is empty, a pre-interpretation I on (W, K, Rel) such that for all subsets U on W,

— W is in $I(p)$,

— if U is in $I(p)$ then the R-pre-image of U is in $I(p)$,

and an arbitrary ind-interpretation g on (W, K, Rel). It follows that for all complex concepts C, $(\mathcal{T}_2, \Pi_2, \mathcal{A}_2) \models p(C) \leftarrow$ if and only if either $\mathcal{T}_2 \models C = \top$, or $\mathcal{T}_2 \models C = \exists R.\top$, or $\mathcal{T}_2 \models C = \exists R.\exists R.\top$.

An example about non-unifiability Let $(\mathcal{T}_3, \Pi_3, \mathcal{A}_3)$ be the deductive ontology where

— $\mathcal{T}_3$ is the empty T-box,

— Π_3 is the program containing the following clauses :

— $p(X) \leftarrow q(X), r(X)$,

— $q(\exists Q.X) \leftarrow$,

— $r(\exists R.X) \leftarrow$,

— $\mathcal{A}_3$ is the empty A-box.

Models of $\mathcal{T}_3$ are arbitrary frames. Models of $(\mathcal{T}_3, \Pi_3, \mathcal{A}_3)$ are structures (W, K, Rel, I, g) consisting of an arbitrary frame (W, K, Rel), a pre-interpretation I on (W, K, Rel) such that for all subsets U of W,

— if U is in $I(q)$ and U is in $I(r)$ then U is in $I(p)$,

— the Q-pre-image of U is in $I(q)$,

— the R-pre-image of U is in $I(r)$,

and an arbitrary ind-interpretation g on (W, K, Rel). It follows that for all complex concepts C,

11. When description logic $\mathcal{ALC}$ is considered instead of description logic $\mathcal{EL}$, the undecidability of **DDC**, **DUC**, **DDG** and **DCA** can be easily proved by means of reductions from the undecidability of the reachability problem in Minsky machines.

12. See the 7th item in Section 5.

$(\mathcal{T}_3, \Pi_3, \mathcal{A}_3) \models q(C) \leftarrow$ if and only if there exists a complex concept D such that $\mathcal{T}_3 \models C = \exists Q.D$ and for all complex concepts C, $(\mathcal{T}_3, \Pi_3, \mathcal{A}_3) \models r(C) \leftarrow$ if and only if there exists a complex concept D such that $\mathcal{T}_3 \models C = \exists R.D$. Moreover, for no complex concept C, $(\mathcal{T}_3, \Pi_3, \mathcal{A}_3) \models p(C) \leftarrow$.

An example about unifiability Let $(\mathcal{T}_4, \Pi_4, \mathcal{A}_4)$ be the deductive ontology where

- $\mathcal{T}_4$ is the T-box containing the following concept equation :
  - $\exists Q.X = \exists R.X$,
- Π_4 is the program containing the following clauses :
  - $p(X) \leftarrow q(X), r(X)$,
  - $q(\exists Q.X) \leftarrow$,
  - $r(\exists R.X) \leftarrow$,
- $\mathcal{A}_4$ is the empty A-box.

Models of $\mathcal{T}_4$ are frames (W, K, Rel) such that [13]

- $Rel(Q) = Rel(R)$.

Models of $(\mathcal{T}_4, \Pi_4, \mathcal{A}_4)$ are structures (W, K, Rel, I, g) consisting of a frame (W, K, Rel) such that $Rel(Q) = Rel(R)$, a pre-interpretation I on (W, K, Rel) such that for all subsets U of W,

- if U is in $I(q)$ and U is in $I(r)$ then U is in $I(p)$,
- the Q-pre-image of U is in $I(q)$,
- the R-pre-image of U is in $I(r)$,

and an arbitrary ind-interpretation g on (W, K, Rel). It follows that for all complex concepts C, $(\mathcal{T}_4, \Pi_4, \mathcal{A}_4) \models q(C) \leftarrow$ if and only if there exists a complex concept D such that $\mathcal{T}_4 \models C = \exists Q.D$ and for all complex concepts C, $(\mathcal{T}_4, \Pi_4, \mathcal{A}_4) \models r(C) \leftarrow$ if and only if there exists a complex concept D such that $\mathcal{T}_4 \models C = \exists R.D$. Moreover, for all complex concepts C, $(\mathcal{T}_4, \Pi_4, \mathcal{A}_4) \models p(C) \leftarrow$ if and only if there exists a complex concept D such that $\mathcal{T}_4 \models C = \exists Q.D$ and $\mathcal{T}_4 \models C = \exists R.D$. Of course, since for all $\mathcal{T}_4$-models (W, K, Rel), $Rel(Q) = Rel(R)$, for all complex concepts D, $\mathcal{T}_4 \models \exists Q.D = \exists R.D$.

An example about OrBAC Let an OrBAC security policy be made up of the finite sets I, J, M and N, the binary relations PERM, COMP, OPTI and PROH between I and J, the binary relation $\mathsf{HasRole}$ between I and M, the binary relation $\mathsf{HasView}$ between J and N, the binary relation $\mathsf{HasAuthorityOn}$ on M and the binary relation $\mathsf{ContainsAsSubpart}$ on N. The finite sets I, J, M and N are, respectively, the set of all roles, the set of all views, the set of all subjects and the set of all objects [14]. The binary relations PERM, COMP, OPTI and PROH between I and J correspond to the following assertions :

- "the security policy permits the i-th role to access the j-th view" for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{j} \in \mathsf{J}$ such that $\mathsf{PERM}(\mathsf{i}, \mathsf{j})$,
- "the security policy makes it compulsory for the i-th role to access the j-th view" for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{j} \in \mathsf{J}$ such that $\mathsf{COMP}(\mathsf{i}, \mathsf{j})$,
- "the security policy makes it optional for the i-th role to access the j-th view" for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{j} \in \mathsf{J}$ such that $\mathsf{OPTI}(\mathsf{i}, \mathsf{j})$,
- "the security policy prohibits the i-th role from accessing the j-th view" for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{j} \in \mathsf{J}$ such that $\mathsf{PROH}(\mathsf{i}, \mathsf{j})$.

It may happen that subjects have roles and objects have views. In this respect, the binary relation $\mathsf{HasRole}$ between I and M and the binary relation $\mathsf{HasView}$ between J and N correspond to the following assertions :

- "the m-th subject has the i-th role" for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{m} \in \mathsf{M}$ such that $\mathsf{HasRole}(\mathsf{i}, \mathsf{m})$,
- "the n-th object has the j-th view" for each $\mathsf{j} \in \mathsf{J}$ and for each $\mathsf{n} \in \mathsf{N}$ such that $\mathsf{HasView}(\mathsf{j}, \mathsf{n})$,

It may happen that subjects have authority on other subjects and objects contain other objects as subpart. In this respect, the binary relation $\mathsf{HasAuthorityOn}$ on M and the binary relation $\mathsf{ContainsAsSubpart}$ on N correspond to the following assertions :

- "the m-th subject has authority on the m'-th subject" for each $\mathsf{m}, \mathsf{m}' \in \mathsf{M}$ such that $\mathsf{HasAuthorityOn}(\mathsf{m}, \mathsf{m}')$,
- "the n-th object contains the n'-th object as subpart" for each $\mathsf{n}, \mathsf{n}' \in \mathsf{N}$ such that $\mathsf{ContainsAsSubpart}(\mathsf{n}, \mathsf{n}')$.

To express the above assertions, we will use the constant concepts $\mathsf{Subject}$, Object, $\mathsf{A_i}$ for each $\mathsf{i} \in \mathsf{I}$ and $\mathsf{B_j}$ for each $\mathsf{j} \in \mathsf{J}$, the constant roles $\mathsf{hasAuthorityOn}$ and $\mathsf{containsAsSubpart}$, the predicate symbols perm, comp, opti and proh of arity 2 and the individual constants $\mathsf{a_m}$ for each $\mathsf{m} \in \mathsf{M}$ and $\mathsf{b_n}$ for each $\mathsf{n} \in \mathsf{N}$. Let $(\mathcal{T}_5, \Pi_5, \mathcal{A}_5)$ be the deductive ontology where

- $\mathcal{T}_5$ is the T-box containing the following concept inclusions :
  - $(\mathbf{T_1})$ $\mathsf{Subject} \sqcap \mathsf{Object} \sqsubseteq X$,
  - $(\mathbf{T_2})$ $\mathsf{A_i} \sqsubseteq \mathsf{Subject}$ for each $\mathsf{i} \in \mathsf{I}$,
  - $(\mathbf{T_3})$ $\mathsf{B_j} \sqsubseteq \mathsf{Object}$ for each $\mathsf{j} \in \mathsf{J}$,
  - $(\mathbf{T_4})$ $\exists \mathsf{hasAuthorityOn}.X \sqsubseteq \mathsf{Subject} \sqcap \exists \mathsf{hasAuthorityOn}.(X \sqcap \mathsf{Subject})$,
  - $(\mathbf{T_5})$ $\exists \mathsf{containsAsSubpart}.X \sqsubseteq \mathsf{Object} \sqcap \exists \mathsf{containsAsSubpart}.(X \sqcap \mathsf{Object})$,
- Π_5 is the program containing the following clauses :
  - $(\mathbf{DP_1})$ $\mathsf{perm}(\mathsf{A_i}, \mathsf{B_j}) \leftarrow$ for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{j} \in \mathsf{J}$ such that $\mathsf{PERM}(\mathsf{i}, \mathsf{j})$,

13. See the 8th item in Section 5.
14. See Section 2 for a definition of the words "roles" and "views" within the context of OrBAC.

($\mathbf{DP_2}$) $\mathsf{comp}(\mathsf{A_i}, \mathsf{B_j}) \leftarrow$ for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{j} \in \mathsf{J}$ such that $\mathsf{COMP}(\mathsf{i}, \mathsf{j})$,

($\mathbf{DP_3}$) $\mathsf{opti}(\mathsf{A_i}, \mathsf{B_j}) \leftarrow$ for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{j} \in \mathsf{J}$ such that $\mathsf{OPTI}(\mathsf{i}, \mathsf{j})$,

($\mathbf{DP_4}$) $\mathsf{proh}(\mathsf{A_i}, \mathsf{B_j}) \leftarrow$ for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{j} \in \mathsf{J}$ such that $\mathsf{PROH}(\mathsf{i}, \mathsf{j})$,

($\mathbf{DP_5}$) $\mathsf{perm}(X, Y) \leftarrow \mathsf{comp}(X, Y)$,

($\mathbf{DP_6}$) $\mathsf{opti}(X, Y) \leftarrow \mathsf{proh}(X, Y)$,

— $\mathcal{A}_5$ is the A-box containing the following assertions :

($\mathbf{AB_1}$) $\mathsf{Subject}{:}\mathsf{a_m}$ for each $\mathsf{m} \in \mathsf{M}$,

($\mathbf{AB_2}$) $\mathsf{Object}{:}\mathsf{b_n}$ for each $\mathsf{n} \in \mathsf{N}$,

($\mathbf{AB_3}$) $\mathsf{A_i}{:}\mathsf{a_m}$ for each $\mathsf{i} \in \mathsf{I}$ and for each $\mathsf{m} \in \mathsf{M}$ such that $\mathsf{HasRole}(\mathsf{i}, \mathsf{m})$,

($\mathbf{AB_4}$) $\mathsf{B_j}{:}\mathsf{b_n}$ for each $\mathsf{j} \in \mathsf{J}$ and for each $\mathsf{n} \in \mathsf{N}$ such that $\mathsf{HasView}(\mathsf{j}, \mathsf{n})$,

($\mathbf{AB_5}$) $\mathsf{hasAuthorityOn}{:}(\mathsf{a_m}, \mathsf{a_{m'}})$ for each $\mathsf{m}, \mathsf{m'} \in \mathsf{M}$ such that $\mathsf{HasAuthorityOn}(\mathsf{m}, \mathsf{m'})$,

($\mathbf{AB_6}$) $\mathsf{containsAsSubpart}{:}(\mathsf{b_n}, \mathsf{b_{n'}})$ for each $\mathsf{n}, \mathsf{n'} \in \mathsf{M}$ such that $\mathsf{ContainsAsSubpart}(\mathsf{n}, \mathsf{n'})$.

($\mathbf{T_1}$) says that the set denoted by Subject and the set denoted by Object are disjoint. ($\mathbf{T_2}$) says that the set denoted by $\mathsf{A_i}$ is included in the set denoted by Subject for each $\mathsf{i} \in \mathsf{I}$. ($\mathbf{T_3}$) says that the set denoted by $\mathsf{B_j}$ is included in the set denoted by Object for each $\mathsf{j} \in \mathsf{J}$. ($\mathbf{T_4}$) says that the domain and the range of the binary relation denoted by hasAuthorityOn is included in the set denoted by Subject. ($\mathbf{T_5}$) says that the domain and the range of the binary relation denoted by containsAsSubpart is included in the set denoted by Object. Indeed, models of $\mathcal{T}_5$ are frames (W, K, Rel) such that [15]

— $K(\mathsf{Subject})$ and $K(\mathsf{Object})$ do not intersect,
— $K(\mathsf{A_i})$ is included in $K(\mathsf{Subject})$ for each $\mathsf{i} \in \mathsf{I}$,
— $K(\mathsf{B_j})$ is included in $K(\mathsf{Object})$ for each $\mathsf{j} \in \mathsf{J}$,
— the domain and range of hasAuthorityOn are included in $K(\mathsf{Subject})$,
— the domain and range of containsAsSubpart are included in $K(\mathsf{Object})$.

($\mathbf{DP_1}$), ($\mathbf{DP_2}$), ($\mathbf{DP_3}$) and ($\mathbf{DP_4}$) express the assertions corresponding to the relations PERM, COMP, OPTI and PROH. ($\mathbf{DP_5}$) is the deontic principle saying that every compulsory access is permitted. ($\mathbf{DP_6}$) is the deontic principle saying that every prohibited access is optional. ($\mathbf{AB_1}$) says that $\mathsf{a_m}$ denotes a subject for each $\mathsf{m} \in \mathsf{M}$. ($\mathbf{AB_2}$) says that $\mathsf{b_n}$ denotes an object for each $\mathsf{n} \in \mathsf{N}$. ($\mathbf{AB_3}$) and ($\mathbf{AB_4}$) are the concept assertions corresponding to the relations HasRole and HasView. ($\mathbf{AB_5}$) and ($\mathbf{AB_6}$) are the role assertions corresponding to the relations HasAuthorityOn and ContainsAsSubpart. Within the context of this example, for all $\mathsf{m} \in \mathsf{M}$ and for all $\mathsf{n} \in \mathsf{N}$, we will say that

— the security policy permits the m-th subject to access the n-th object if and only if for all models (W, K, Rel, I, g) of $(\mathcal{T}_5, \Pi_5, \mathcal{A}_5)$ and for all **VAR**-interpretations V on (W, K, Rel), there exists a complex concept C and there exists a complex concept D such that $g(\mathsf{a_m}) \in \|C\|_V$, $g(\mathsf{b_n}) \in \|D\|_V$ and $|\mathsf{perm}(C, D)|_V^I = 1$,
— the security policy makes it compulsory for the m-th subject to access the n-th object if and only if for all models (W, K, Rel, I, g) of $(\mathcal{T}_5, \Pi_5, \mathcal{A}_5)$ and for all **VAR**-interpretations V on (W, K, Rel), there exists a complex concept C and there exists a complex concept D such that $g(\mathsf{a_m}) \in \|C\|_V$, $g(\mathsf{b_n}) \in \|D\|_V$ and $|\mathsf{comp}(C, D)|_V^I = 1$,
— the security policy makes it optional for the m-th subject to access the n-th object if and only if for all models (W, K, Rel, I, g) of $(\mathcal{T}_5, \Pi_5, \mathcal{A}_5)$ and for all **VAR**-interpretations V on (W, K, Rel), there exists a complex concept C and there exists a complex concept D such that $g(\mathsf{a_m}) \in \|C\|_V$, $g(\mathsf{b_n}) \in \|D\|_V$ and $|\mathsf{opti}(C, D)|_V^I = 1$,
— the security policy prohibits the m-th subject from accessing the n-th object if and only if for all models (W, K, Rel, I, g) of $(\mathcal{T}_5, \Pi_5, \mathcal{A}_5)$ and for all **VAR**-interpretations V on (W, K, Rel), there exists a complex concept C and there exists a complex concept D such that $g(\mathsf{a_m}) \in \|C\|_V$, $g(\mathsf{b_n}) \in \|D\|_V$ and $|\mathsf{proh}(C, D)|_V^I = 1$.

To illustrate the expressive power of concept constructs, the following clauses can be added to Π_5 :

($\mathbf{DP_7}$) $\mathsf{comp}(\exists \mathsf{hasAuthorityOn}.X, Y) \leftarrow \mathsf{perm}(X, \exists \mathsf{containsAsSubpart}.Y)$,

($\mathbf{DP_8}$) $\mathsf{proh}(X, \exists \mathsf{containsAsSubpart}.Y) \leftarrow \mathsf{opti}(\exists \mathsf{hasAuthorityOn}.X, Y)$.

($\mathbf{DP_7}$) says that if the security policy permits the set denoted by X to access the set of objects containing as subpart objects of the set denoted by Y then the security policy makes it compulsory for the set of subjects having authority on subjects of the set denoted by X to access the set of objects denoted by Y. ($\mathbf{DP_8}$) says that if the security policy makes it optional for the set of subjects having authority on subjects of the set denoted by X to access the set denoted by Y then the security policy prohibits the set denoted by X from accessing the set of objects containing as subpart objects of the set denoted by Y. With our without ($\mathbf{DP_7}$) and ($\mathbf{DP_8}$), accesses of subjects to objects should be neither both permitted and prohibited, nor both com-

15. See the 9th, 10th, 11th and 12th items in Section 5.

pulsory and optional [16]. For this reason, it is of the utmost importance to check whether one of the following conditions holds :

— there exists a correct answer for the definite goal $\leftarrow \mathsf{perm}(X, Y), \mathsf{proh}(X, Y)$,

— there exists a correct answer for the definite goal $\leftarrow \mathsf{comp}(X, Y), \mathsf{opti}(X, Y)$.

It is also of the utmost importance to check whether there exists $\mathsf{m} \in \mathsf{M}$ and there exists $\mathsf{n} \in \mathsf{N}$ such that one of the following conditions holds :

— the security policy both permits the m-th subject to access the n-th object and prohibits the m-th subject from accessing the n-th object,

— the security policy both makes it compulsory for the m-th subject to access the n-th object and makes it optional for the m-th subject to access the n-th object.

9 A research program

We present a research program [17].

Turing-completeness Our hybrid formalism can be seen as a programming language. It is not known whether it is Turing-complete. When description logic $\mathcal{ALC}$ is considered instead of description logic $\mathcal{EL}$, the Turing-completeness of our hybrid formalism can be easily proved by means of a reduction from the Turing-completeness of Minsky machines. Hence, the following item in our research program :

($\mathbf{RP_1}$) separate the description logics that do give rise to a Turing-complete hybrid formalism from the description logics that do not.

In particular, find simple and natural conditions on concept inclusions, concept equations and clauses such that deductive ontologies satisfying them give rise to a Turing-complete hybrid formalism.

Tractability The success of the logic programming languages comes from the fact that it is relatively easy to define Turing-incomplete restrictions of clauses that can be used as a domain-specific language taking advantage of efficient algorithms developed for them [Dantsin *et al.* (2001), Gottlob and Papadimitriou (2003)]. Thus, the following item in our research program :

($\mathbf{RP_2}$) for the description logics that do not give rise to a Turing-complete hybrid formalism, separate those that do give rise to a hybrid formalism tractable in polynomial time from those that do not.

In particular, find simple and natural conditions on concept inclusions, concept equations and clauses such that deductive ontologies satisfying them give rise to a hybrid formalism tractable in polynomial time.

Declarative and fixpoint semantics In logic programming, the declarative semantics of programs is given by the usual semantics of first-order logic. It is defined in terms of Herbrand interpretations [Gabbay *et al.* (1998), Lloyd (1987)]. In this setting, given a program, the main result is the standard characterization of its Herbrand models as the pre-fixpoints of some continuous mapping associated to it. Consequently, the following item in our research program :

($\mathbf{RP_3}$) develop the declarative and fixpoint semantics of our hybrid formalism.

In particular, given a deductive ontology, characterize its Herbrand models as the pre-fixpoints of some continuous mapping associated to it.

Procedural semantics In logic programming, the refutation procedure of interest is called SLD-resolution where an inference step is based on the unifiability between the selected atom in a given definite goal and the left side of a variant of a definite clause in a given program. Hence, the following item in our research program :

($\mathbf{RP_4}$) develop the procedural semantics of our hybrid formalism.

In particular, considering the unification problem in description logics with empty T-boxes [Baader and Morawska (2009)], adapt the related unification algorithms to the context of our hybrid formalism [18].

16. In most logical models of deontic systems, if it is prohibited to a subject from accessing some object then it is not permitted that this subject accesses that object and if it is made optional for a subject to access some object then it is not made compulsory that this subject accesses that object [Parent and van der Torre (2018)].

17. As can be seen from its presentation, this research program covers different aspects of description logics and logic programming : recursion theory with ($\mathbf{RP_1}$), computational complexity with ($\mathbf{RP_2}$), model theory and fixpoint theory with ($\mathbf{RP_3}$), automated deduction with ($\mathbf{RP_4}$), non-monotonic reasoning with ($\mathbf{RP_5}$) and ontology engineering techniques with ($\mathbf{RP_6}$) and ($\mathbf{RP_7}$). Needless to say, to carry out it, one must neither work in isolation, nor lose sight of the possible applications of the hybrid formalism developed in this paper. In other respect, with respect to expressivity, one must also compare this formalism to the main approaches proposed so far. These approaches include the above-mentioned hybrid knowledge bases [Eiter *et al.* (2011a), Eiter *et al.* (2011b), Levy and Rousset (1998), Motik and Rosati (2017)]. They also include approaches such as the existential rule framework [Bienvenu *et al.* (2020), Mugnier and Thomazo (2014)].

18. The computability of the unification problem with arbitrary T-boxes is not known. In other respect, when description logic $\mathcal{ALC}$ is considered instead of description logic $\mathcal{EL}$, the computability of the unification problem either with empty T-boxes, or with arbitrary T-boxes is not known too.

In this respect, the tools and techniques developed in [Alizadeh *et al.* (2021), Balbiani and Gencer (2017), Balbiani *et al.* (2021), Gencer (2002), Gencer and de Jongh (2008), Sofronie-Stokkermans (2007)] might be useful.

Negation By using conditional assertions of the form $\alpha_1, \ldots, \alpha_m \leftarrow \beta_1, \ldots, \beta_n, \mathtt{not}(\gamma_1), \ldots, \mathtt{not}(\gamma_o)$ where α_1, ..., α_m, β_1, ..., β_n, γ_1, ..., γ_o are atoms, one may write more expressive deductive ontologies. For instance, in our example about security policies, the deontic principle saying that every non-prohibited access is permitted and the deontic principle saying that every non-compulsory access is optional can be expressed by the following conditional assertions :

— $\mathsf{perm}(X, Y) \leftarrow \mathtt{not}(\mathsf{proh}(X, Y))$,
— $\mathsf{opti}(X, Y) \leftarrow \mathtt{not}(\mathsf{comp}(X, Y))$.

In logic programming, the declarative semantics of a program containing, possibly, negation in the right side of clauses is given by the so-called answer set semantics. It is defined in terms of stable models [Erdem (2002), Lifschitz (2008)]. In this setting, the question of the existence of stable models for a given program is of the utmost interest. Thus, the following item in our research program :

(**RP$_5$**) develop the answer set semantics of our hybrid formalism when programs contain, possibly, negation in the right side of their clauses.

Forgetting Forgetting is an ontology engineering technique. It is achieved by eliminating from a given ontology a subset of its signature in such a way that all logical consequences up to the remaining signature are preserved. This form of knowledge compilation has important applications when an engineer who designs an ontology formulated in some language wants to import content from an existing ontology formulated in a richer language. As a result, ontology forgetting has attracted increasing attention and several algorithms have been developed to support it [Chen *et al.* (2019), Del-Pinto and Schmidt (2019)]. Consequently, the following item in our research program :

(**RP$_6$**) develop the ontology engineering technique of forgetting for our hybrid formalism.

Modularization Modularization is an ontology engineering technique. Its importance is the result of the fact that large and monolithic ontologies are difficult to handle, whereas smaller and modular ontologies are easier to understand and use. With a view to collaboratively developing ontologies and merging independently developed ontologies into a single and reconciled one, ontology modularization has attracted increasing attention and several algorithms have been developed to support it [Ben Abbès *et al.* (2012), Cuenca Grau *et al.* (2007)]. Hence, the following item in our research program :

(**RP$_7$**) develop the ontology engineering technique of modularization for our hybrid formalism.

10 Last words

Our idea of an hybrid formalism where description logics constructs are used for defining concepts that are given as arguments to the predicates of the logic programs has only one ancestor : the formalism developed in [Büttner and Simonis (1987)]. In this formalism, Boolean constructs are used for defining expressions that are given as arguments to the predicates of the logic programs, allowing clauses of the form [19]

— $\mathsf{adder}(X, Y, Z, T, U \vee V) \leftarrow \mathsf{halfAdder}(X, Y, W, U)$, $\mathsf{halfAdder}(W, Z, T, V)$,
— $\mathsf{halfAdder}(X, Y, X \oplus Y, X \wedge Y) \leftarrow$.

where X, Y, Z, T, U, V and W denote propositional variables, $\vee$, $\oplus$ and $\wedge$ denote the Boolean constructs of, respectively, disjunction, exclusive disjunction and conjunction and adder and halfAdder are predicate symbols of, respectively, arity 5 and arity 4. Obviously, the Boolean expressions $U \vee V$, $X \oplus Y$ and $X \wedge Y$ used in these clauses can be seen as **ROL**-free complex concepts when description logic $\mathcal{ALC}$ is considered instead of description logic $\mathcal{EL}$.

Knowledge representation languages such as those provided by description logic languages and rule-based reasoning paradigms such as those provided by logic programming languages are well-known and widely used in Computer Science and Artificial Intelligence. Therefore, it is quite amazing that their integration in a unique formalism similar to the formalism proposed by [Büttner and Simonis (1987)] has not been put forward during the last 30 years. A narrow-minded explanation would consist of saying that this lack of interest is the result of the lack of importance of hybrid formalisms such as the one introduced in this paper. The case study presented in Section 2 and the example about OrBAC introduced in Section 8 indicate that this lack of interest might just be the result of a lack of imagination. Indeed, we believe that it is time to give space to advanced languages of terms for ontologies as introduced in Sections 3 and 4, to consider the decision problems presented in Sections 5, 6 and 7 and to address the research program presented in Section 9.

19. See Section 4 in [Büttner and Simonis (1987)].

Acknowledgement

Special acknowledgement is granted to Stéphane Demri, Martín Diéguez, Esra Erdem, Andreas Herzig, Rosalie Iemhoff, George Metcalfe, Mojtaba Mojtahedi, Linh Anh Nguyen, Christophe Ringeissen, Maryam Rostamigiv, Renate Schmidt and Tinko Tinchev for their suggestions.

References

[**Abou El Kalam *et al.* (2003)**] Abou El Kalam, A., El Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miège, A., Saurel, C., Trouessin, G. 'Organization based access control'. In : *Proceedings POLICY 2003.* IEEE (2003) 120–131.

[**Alizadeh *et al.* (2021)**] Alizadeh, M., Ardeshir, M., Balbiani, P., Mojtahedi, M. 'Unification types in Euclidean modal logics'. *Logic Journal of the IGPL* doi.org/10.1093/jigpal/jzab036.

[**Baader (2003)**] Baader, F. 'Terminological cycles in a description logic with existential restrictions'. In : *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence.* Morgan Kaufmann (2003) 325–330.

[**Baader *et al.* (2003)**] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge University Press (2003).

[**Baader and Lutz (2007)**] Baader, F., Lutz, C. 'Description logic'. In : *Handbook of Modal Logic.* Elsevier (2007) 757–819.

[**Baader and Morawska (2009)**] Baader, F., Morawska, B. 'Unification in the description logic $\mathcal{EL}$'. In : *Rewriting Techniques and Applications.* Springer (2009) 350–364.

[**Baader and Nutt (2003)**] Baader, F., Nutt, W. 'Basic description logics'. In : *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge University Press (2003) 47–100.

[**Baader and Snyder (2001)**] Baader, F., Snyder, W. 'Unification theory', In : *Handbook of Automated Reasoning,* Elsevier (2001) 439–526.

[**Balbiani and Gencer (2017)**] Balbiani, P., Gencer, Ç. 'Unification in epistemic logics'. *Journal of Applied Non-Classical Logics* **27** (2017) 91–105.

[**Balbiani *et al.* (2021)**] Balbiani, P., Gencer, Ç., Rostamigiv, M., Tinchev, T. 'About the unification type of $\mathbf{K} + \Box\Box\bot$'. *Annals of Mathematics and Artificial Intelligence* doi.org/10.1007/s10472-021-09768-w.

[**Ben Abbès *et al.* (2012)**] Ben Abbès, S., Scheuermann, A., Meilender, T., d'Aquin, M. 'Characterizing modular ontologies'. In : *Workshop on Modular Ontologies (WoMO) 2012.* CEUR-WS.org (2012) 16–27.

[**Bienvenu *et al.* (2020)**] Bienvenu, M., Leclère, M., Mugnier, M.-L., Rousset, M.-C. 'Reasoning with ontologies'. In : *A Guided Tour of Artificial Intelligence Research.* Springer (2020) 185–215.

[**Büttner and Simonis (1987)**] Büttner, W., Simonis, H. 'Embedding Boolean expressions into logic programming'. *Journal of Symbolic Computation* **4** (1987) 191–205.

[**Calvanese and De Giacomo (2003)**] Calvanese, D., De Giacomo, G. 'Expressive description logics'. In : *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge University Press (2003) 178–218

[**Chagrov and Chagrova (2006)**] Chagrov, A., Chagrova, L. 'The truth about algorithmic problems in correspondence theory'. In : Advances in Modal Logic. Vol. 6. College Publications (2006) 121–138.

[**Chen *et al.* (2019)**] Chen, J., Alghamdi, G., Schmidt, R., Walther, D., Gao, Y. 'Ontology extraction for large ontologies via modularity and forgetting'. In : *Proceedings of the* 10*th International Conference on Knowledge Capture.* ACM (2019) 45–52.

[**Cuenca Grau *et al.* (2007)**] Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U. 'A logical framework for modularity of ontologies'. In : Proceedings of the 20th International Joint Conference on Artificial Intelligence. Morgan Kaufmann (2007) 298–303.

[**Del-Pinto and Schmidt (2019)**] Del-Pinto, W., Schmidt, R. 'ABox abduction via forgetting in ALC'. In : Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence. AAAI (2019) 2768–2775.

[**Dantsin *et al.* (2001)**] Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A. 'Complexity and expressive power of logic programming'. *ACM Computing Surveys* **33** (2001) 374–425.

[**Donini (2003)**] Donini, F. 'Complexity of reasoning'. In : *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge University Press (2003) 101–141.

[**Eiter *et al.* (2011a)**] Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H. 'Combining

answer set programming with description logics for the semantic web'. *Artificial Intelligence* **172** (2011) 1495–1539.

[**Eiter *et al.* (2011b)**] Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R. 'Well-founded semantics for description logic programs in the semantic web'. *ACM Transactions on Computational Logic* **12** (2011) article 11.

[**Erdem (2002)**] Erdem, E. *Theory and Applications of Answer Set Programming.* Doctoral thesis of the University of Texas at Austin (2002).

[**Gabbay *et al.* (1998)**] Gabbay, D., Hogger, C., Robinson, J. *Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 5 : Logic Programming.* Oxford University Press (1998).

[**Gencer (2002)**] Gencer, Ç. 'Description of modal logics inheriting admissible rules for **K**4'. *Logic Journal of the IGPL* **10** (2002) 401–411.

[**Gencer and de Jongh (2008)**] Gencer, Ç., de Jongh, D. 'Unification in extensions of **K**4'. *Logic Journal of the IGPL* **17** (2009) 159–172.

[**Gottlob and Papadimitriou (2003)**] Gottlob, G., Papadimitriou, C. 'On the complexity of single-rules datalog queries'. *Information and Computation* **183** (2003) 104–122.

[**Lampson (1974)**] Lampson, B. 'Protection'. *Operating Systems Review* **8** (1974) 18–24.

[**Levy and Rousset (1998)**] Levy, A., Rousset, M.-C. 'Combining Horn rules and description logics in CARIN'. *Artificial Intelligence* **104** (1998) 165–209.

[**Lifschitz (2008)**] Lifschitz, V. 'What is answer set programming'. In : *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence.* Association for the Advancement of Artificial Intelligence (2008) 1594–1597.

[**Lloyd (1987)**] Lloyd, J. *Foundations of Logic Programming.* Springer (1987).

[**Motik and Rosati (2017)**] Motik, B., Rosati, R. 'Reconciling description logics and rules'. *Journal of the ACM* **57** (2010) article 30.

[**Mugnier and Thomazo (2014)**] Mugnier, M.-L., Thomazo, M. 'An introduction to ontology-based query answering with existential rules'. In : *Reasoning Web.* Springer (2014) 245–278.

[**Parent and van der Torre (2018)**] Parent, X., van der Torre, L. *Introduction to Deontic Logic and Normative Systems.* College Publications (2018).

[**Sandhu *et al.* (1996)**] Sandhu, R., Coyne, E., Feinstein, H., Youman, C. 'Role-based access control models'. *IEEE Computer* **29** (1996) 38–47.

[**Schild (1991)**] Schild, K. 'A correspondence theory for terminological logics : preliminary report'. In : *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence.* Morgan Kaufmann (1991) 466–471.

[**Sofronie-Stokkermans (2007)**] Sofronie-Stokkermans, V. 'On unification for bounded distributive lattices'. *ACM Transactions on Computational Logic* **8** (2007) article 12.