Actes des 14es Journées d'Intelligence Artificielle Fondamentale

Zied Bouraoui, Sylvie Doutre

AfIA

Association française
pour l'Intelligence Artificielle

JIAF

Journées d'Intelligence Artificielle Fondamentale

PFIA 2020

Table des matières

Éditorial

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 14èmes Journées d'Intelligence Artificielle Fondamentale (JIAF 2020) ont eu lieu du 29 juin au 3 juillet 2020, dans le cadre de la Plate-Forme Intelligence Artificielle (PFIA). Les éditions précédentes se sont déroulées à 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, Sylvie Doutre

Comité de programme

Président

— Zied Bouraoui (CRIL, Univ Artois & CNRS)
— Sylvie Doutre (IRIT, Université Toulouse Capitole)

Membres

— Francesco Belardinelli (IBISC, Université d'Évry)
— Elise Bonzon (LIPADE, Université Paris Descartes)
— Tristan Cazenave (LAMSADE, Université Paris Dauphine)
— Nadia Creignou (LIS, Aix-Marseille Université)
— Tiago de Lima (CRIL, Lens)
— Jérôme Euzenat (LIG, INRIA)
— George Katsirelos (MIAT, INRA)
— Sébastien Konieczny (CRIL, CNRS)
— Jérôme Lang (LAMSADE, Université Paris Dauphine)
— Jean Lieber (LORIA, INRIA)
— Pierre Marquis (CRIL, Université d'Artois)
— Marie-Laure Mugnier (LIRMM, Université de Montpellier)
— Amedeo Napoli (LORIA, CNRS)
— Odile Papini (LIS, Aix-Marseille Université)
— Laurent Perrussel (IRIT, Universite Toulouse 1 Capitole)
— Sophie Pinchinat (IRISA, INRIA)
— Stéphanie Roussel (ONERA)
— Serena Villata (I3S, CNRS)
— Christel Vrain (LIFO, Université d'Orléans)
— Bruno Zanuttini (GREYC, UNICAEN)

Actes JIAF 2020

Mesure d'Impact et Interprétabilité des Sémantiques Graduées en Argumentation Abstraite

Jérôme Delobelle Serena Villata

Université Côte d'Azur, Inria, CNRS, I3S, Sophia-Antipolis, France
jerome.delobelle@inria.fr villata@i3s.unice.fr

Résumé

L'argumentation, dans le domaine de l'Intelligence Artificielle, est un formalisme permettant aussi bien de raisonner avec des informations contradictoires que de modéliser un échange d'arguments entre un ou plusieurs agents. Dans ce but, de nombreuses sémantiques ont été définies, dont les sémantique graduées qui ont pour but d'attribuer un degré d'acceptabilité à chaque argument en fonction des critères choisis (e.g. nombre d'attaquants, de défenseurs, qualité des attaquants, ...). Bien que le nombre de ces sémantiques continue de croître, il n'existe actuellement aucune méthode permettant d'expliquer les résultats renvoyés par ces sémantiques. Dans cet article, nous étudions l'interprétabilité de ces sémantiques en mesurant, pour chaque argument, l'impact des autres arguments sur son degré d'acceptabilité. Nous définissons une nouvelle propriété et montrons que le score d'un argument retourné par une sémantique graduée qui satisfait cette propriété peut également être calculé en agrégeant l'impact des autres arguments sur celui-ci. Ce résultat permet de fournir, pour chaque argument présent dans un système d'argumentation, un classement entre les arguments du plus impactant au moins impactant en fonction d'une sémantique graduée donnée.

Abstract

Argumentation, in the field of Artificial Intelligence, is a formalism allowing to reason with contradictory information as well as to model an exchange of arguments between one or several agents. For this purpose, many semantics have been defined with, amongst them, gradual semantics aiming to assign an acceptability degree to each argument. Although the number of these semantics continues to increase, there is currently no method allowing to explain the results returned by these semantics. In this paper, we study the interpretability of these semantics by measuring, for each argument, the impact of the other arguments on its acceptability degree. We define a new property and show that the score of an argument returned by a gradual semantics which satisfies this property can also be computed by aggregating the impact of the other arguments on it. This result allows to provide, for each argument in an argumentation framework, a ranking between arguments from the most to the least impacting ones w.r.t. a given gradual semantics.

1 Introduction

La question de l'interprétation des résultats obtenus par les méthodes d'intelligence artificielle (IA) fait l'objet d'une attention croissante, tant au sein de la communauté de l'IA que de la part d'un public plus large. En particulier, la capacité d'interpréter la logique derrière les résultats (par exemple, les classifications, les décisions) renvoyés par un agent intelligent artificiel est d'une importance capitale afin d'assurer une interaction transparente entre deux entités et ainsi accomplir des tâches coopératives dans les meilleures conditions. Selon Miller [16], l'*interprétabilité* peut être définie comme étant le degré de compréhension que possède un observateur sur la (ou les) cause(s) d'un résultat. Un algorithme, un programme ou une décision est dit interprétable s'il est possible d'identifier les éléments ou les caractéristiques qui amènent au résultat retourné. Ce terme ne doit pas être confondu avec le terme *explication* qui est la réponse à une question commençant par le terme intérogatif "pourquoi" ou avec le terme *justification* qui explique pourquoi un résultat est bon ou mauvais, mais ne vise pas nécessairement à donner une explication sur le processus menant au résultat. Malgré les nombreuses approches (formelles et empiriques) [15, 14, 20, 12] pour aborder le problème de l'interprétabilité des systèmes intelligents artificiels, celui-ci reste un problème de recherche ouvert. Comme le soulignent Mittelstadt et al. [17], l'argumentation artificielle [4] peut jouer un rôle important dans le traitement de cette question ouverte, grâce à sa caractéristique interne permettant de combiner la prise de décision avec les arguments pour et les arguments contre menant à une certaine décision.

Dans cet article, nous cherchons à étudier, d'un point de vue formel, la manière de traduire la notion d'interprétabilité en argumentation abstraite afin que les raisons conduisant à l'acceptabilité d'un argument ou d'un ensemble d'arguments dans un système d'argumentation puissent être explicitement données. Cette question de recherche peut ainsi être décomposer en deux sous-questions :

i) Comment définir et caractériser formellement la notion d'*impact* d'un argument par rapport à l'acceptabilité des autres arguments dans le système ?

ii) Quel est le rôle de cet impact dans le processus d'interprétation de l'acceptabilité des arguments dans le système ?

Pour répondre à ces questions, nous faisons le choix d'étudier la famille des sémantiques graduées [8, 6] en nous focalisant sur deux sémantiques présentant des caractéristiques différentes afin de pouvoir montrer la généralité de notre approche à caractériser la notion d'impact. Plus particulièrement, nous nous focalisons sur la sémantique *h-categorizer* initialement introduite par Besnard et Hunter [7] et la *counting sémantique* de Pu et al. [19]. Pour ces deux approches, l'acceptabilité d'un argument est représentée par un *degré d'acceptabilité* restreint à l'intervalle [0, 1] ce qui diffère des sémantiques à base d'extensions de Dung [13] où les arguments sont soit (entièrement) *acceptés* soit *rejetés*. De manière générale, nous considérons que l'impact d'un certain argument (ou d'un ensemble d'arguments) sur le degré d'acceptabilité d'un autre argument peut être mesuré en calculant la différence entre le degré d'acceptabilité actuel de l'argument et son degré d'acceptabilité lorsque le premier argument (ou l'ensemble d'arguments) est supprimé. Nous étudions les propriétés formelles de cette notion d'impact instanciée à travers ces deux sémantiques graduées pour les systèmes d'argumentation abstraits cycliques et acycliques. Enfin, nous montrons que l'étude de l'impact d'un argument sur les autres arguments nous permet de répondre à certains besoins principaux en termes d'interprétabilité des résultats.

Le papier est organisé comme suit. Dans la section 2, nous commençons par rappeler quelques définitions de base concernant l'argumentation abstraite et les sémantiques graduées pour ensuite introduire la sémantique *h-categorizer* [7] et la *counting sémantique* [19]. La section 3 traite de la notion d'impact d'un argument (ou d'un ensemble d'arguments) dans un système d'argumentation et de ces propriétés formelles alors que la section 4 se concentre sur la propriété d'impact équilibré (*balanced impact)*. Dans la section 5, nous proposons d'utiliser cette notion d'impact pour aider à interpréter le résultat retourné par les sémantiques graduées. Enfin, dans la section 6, nous discutons des autres travaux traitant de l'interprétabilité ou de la notion d'impact en argumentation avant de conclure dans la section 7.

2 Preliminaries

Dung [13] formalise l'argumentation à travers un système d'argumentation abstrait dans lequel aucune hypothèse sur la nature des éléments qu'il contient n'est faite. Un système d'argumentation est composé d'un ensemble d'arguments abstraits et d'une relation de conflit entre ces arguments.

Définition 1 (AF) *Un **système d'argumentation (abstrait)** (AF) est un couple $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ où $\mathcal{A}$ est un ensemble fini et non-vide d'entités (abstraites) appelées **arguments** et $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ est une relation binaire sur $\mathcal{A}$, appelée **relation d'attaque**. Pour deux arguments $x, y \in \mathcal{A}$, la notation $(x, y) \in \mathcal{R}$ signifie que x attaque y.*

D'un point de vue mathématique, un système d'argumentation peut être vu comme un graphe dirigé où les nœuds représentent les arguments et les flèches représentent les attaques entre ces arguments.

Définition 2 (Ensemble non-attaqué d'argument) *Soit $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation. L'ensemble d'arguments $X \subseteq \mathcal{A}$ est non-attaqué si $\forall x \in X, \nexists y \in \mathcal{A} \backslash X$ tel que $(y, x) \in \mathcal{R}$.*

Introduisons maintenant différentes notions liées aux graphes que nous utiliserons tout au long de cet article.

Notation 1 *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation et $x, y \in \mathcal{A}$. Une séquence d'arguments $\langle x_0, \ldots, x_n \rangle$ dans $\mathcal{A}$ est un **chemin** de longueur n (le nombre d'attaque dont il est composé) de y vers x ssi $x_0 = x$, $x_n = y$ et pour tout $i \in \{0, 1, \ldots, n-1\}$, $(x_{i+1}, x_i) \in \mathcal{R}$. Un **cycle** est un chemin de x vers x et une **boucle** est un cycle de longueur 1. Soit $\mathcal{R}_n(x)$ l'ensemble des arguments à l'origine d'un chemin de longueur n vers l'argument x. Ainsi, un argument $y \in \mathcal{R}_n(x)$ est un attaquant (resp. défenseur) direct de x si $n = 1$ (resp. $n = 2$). Plus généralement, y est un **attaquant** (resp. **défenseur**) de x si n est impair (resp. pair).*

Exemple 1 *Sur le système d'argumentation AF (voir Figure 1), il est possible d'observer :*

- *un chemin $\langle c, b, a \rangle$ de longueur 2 de c vers a tandis que $\langle d, a, b \rangle$ n'est pas un chemin,*
- *b et d sont les attaquants directs de a,*
- *c et e sont les défenseurs directs de a,*
- *b, d, f et g sont les attaquants a,*
- *c et e sont les défenseurs de a,*
- *$\{e, f, g\}$ (dans le rectangle rouge) est un ensemble non-attaqué d'arguments,*

— $\{a, b, d\}$ *(dans le rectangle bleu) n'est pas un ensemble non-attaqué d'arguments car b est attaqué par c et d est attaqué par e.*

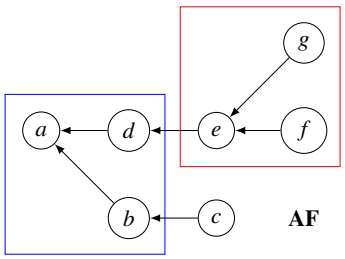


FIGURE 1 – Un système d'argumentation AF

Afin d'évaluer les arguments d'un systèmes d'argumentation de Dung, de nombreuses familles de sémantiques ont été introduites dans la littérature. Nous renvoyons le lecteur à [5, 10, 1] pour un aperçu complet des familles de sémantiques existantes en argumentation abstraite et des différences entre ces approches (e.g., définition, résultat, application). Parmi ces sémantiques, une sémantique graduée attribue à chaque argument dans un système d'argumentation un score, appelé *degré d'acceptabilité*, variant selon les critères choisis. Ce degré appartient souvent à l'intervalle $[0, 1]$.

Définition 3 (Sémantique graduée)
Une sémantique graduée est une fonction $\mathcal{S}$ qui associe à chaque système d'argumentation $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ une fonction $\mathtt{Deg}_F^{\mathcal{S}} : \mathcal{A} \to [0, 1]$. Ainsi, $\mathtt{Deg}_F^{\mathcal{S}}(x)$ représente le degré d'acceptabilité de l'argument $x \in \mathcal{A}$.

2.0.1 Sémantique h-categorizer

La sémantique h-categorizer a initialement été introduite par Besnard et Hunter [7] pour les systèmes d'argumentation acycliques avant que Pu et al. [18] prouvent l'existence et l'unicité de cette solution pour tout système d'argumentation. La fonction *categorizer* attribue un score à chaque argument qui capture la force relative d'un argument en tenant compte de la force de ses attaquants, qui elle-même tient compte de la force de ses attaquants, et ainsi de suite.

Définition 4 *Soit $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation. La fonction **categorizer** $\mathtt{Deg}_F^{Cat} : \mathcal{A} \to]0, 1]$ est définie telle que $\forall x \in \mathcal{A}$,*

$$\mathtt{Deg}_F^{Cat}(x) = \begin{cases} 1 & si\ \mathcal{R}_1(x) = \emptyset \\ \frac{1}{1+\sum_{y \in \mathcal{R}_1(x)} \mathtt{Deg}_F^{Cat}(y)} & sinon \end{cases}$$

2.0.2 Counting sémantique

La counting sémantique [19] permet de classer les arguments selon leur nombre d'attaquants et de défenseurs respectifs. Afin d'attribuer une valeur à chaque argument, Pu et al. considèrent un système d'argumentation comme un jeu de dialogue entre les partisans d'un argument x (c'est-à-dire les défenseurs de x) et les opposants de x (c'est-à-dire les attaquants de x). L'idée est qu'un argument est plus acceptable s'il possède beaucoup d'arguments provenant de ces partisans et peu d'arguments provenant des opposants. Formellement, un AF donné est converti en une matrice $M_{n \times n}$ (où n est le nombre d'arguments dans AF) qui correspond à la matrice d'adjacente de AF (ce qui est possible car un AF est un graphe dirigé). Le produit matriciel de k copies de M, désigné par M^k, représente, pour tous les arguments de AF, le nombre de défenseurs (si k est pair) ou d'attaquants (si k est impair) situés au début d'un chemin de longueur k. Enfin, une norme matricielle N est appliquée à M (e.g., $\|M\|_\infty$) afin de garantir la convergence, et un facteur d'atténuation α est utilisé pour avoir un traitement plus raffiné sur différentes longueurs d'attaquants et de défenseurs (i.e., les lignes d'attaquants/défenseurs plus courtes ont plus d'impact sur l'argument ciblé).

Définition 5 (Modèle de comptage) *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation avec $\mathcal{A} = \{x_1, \ldots, x_n\}$, $\alpha \in]0, 1[$ un facteur d'atténuation et $k \in \mathbb{N}$. Le vecteur colonne à n dimensions v sur $\mathcal{A}$ à l'étape k est définie par,*

$$v_\alpha^k = \sum_{i=0}^{k} (-1)^i \alpha^i \tilde{M}^i \mathcal{I}$$

où $\tilde{M}$ est la matrice normalisée telle que $\tilde{M} = M/N$ avec N comme norme matricielle et $\mathcal{I}$ le vecteur colonne à n dimensions contenant uniquement des 1.
Le modèle de comptage de F est $v_\alpha = \lim_{k \to +\infty} v_\alpha^k$. Le score de $x_i \in \mathcal{A}$ est le i^{me} composent de v_α, dénoté par $\mathtt{Deg}_F^{CS}(x_i)$.

3 Mesure d'impact

L'impact d'un argument sur un autre argument peut être mesuré en calculant la différence entre le moment où cet argument existe et celui où il est supprimé. Pour capturer cette notion de suppression, nous avons besoin, dans un premier temps, de définir un opérateur de restriction qui supprime un ensemble d'arguments du système d'argumentation initial par rapport à un argument donné (i.e., l'argument ciblé de l'impact). Ces changements ont également un impact direct sur la relation d'attaque car les attaques directement liées aux arguments supprimés (attaquants comme attaqués) sont automatiquement supprimées également.

Définition 6 *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation, $X \subseteq \mathcal{A}$ et $y \in \mathcal{A}$. L'**opérateur de restriction** $\ominus$ est défini comme suit $\mathsf{F} \ominus_y X = \langle \mathcal{A}', \mathcal{R}' \rangle$, où*

— $\mathcal{A}' = \mathcal{A} \backslash (X \backslash \{y\})$;
— $\mathcal{R}' = \{(x, z) \mid (x, z) \in \mathcal{R}$ *et* $x, z \in \mathcal{A} \backslash X\}$.

Dans la section 3.1, nous allons commencer par formaliser la manière dont l'impact d'un ensemble non-attaqué d'arguments sur un argument donné est calculé avant de le généraliser pour tous les ensembles d'arguments dans la section 3.2.

3.1 Impact d'un ensemble non-attaqué d'arguments

L'impact d'un ensemble non-attaqué d'arguments X sur le degré d'acceptabilité d'un argument y peut être mesuré en calculant la différence entre le degré d'acceptabilité actuel de y et son degré d'acceptabilité lorsque X est supprimé.

Définition 7 *Soient* $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ *un système d'argumentation,* $y \in \mathcal{A}$ *et* $X \subseteq \mathcal{A}$ *un ensemble non-attaqué d'arguments. Soit* $\mathcal{S}$ *une sémantique graduée. L'**impact** de* X *sur* y *est défini comme suit :*

$$\mathtt{Imp}_F^{\mathcal{S}}(X, y) = \mathtt{Deg}_F^{\mathcal{S}}(y) - \mathtt{Deg}_{F \ominus_y X}^{\mathcal{S}}(y)$$

Globalement, cette définition est déjà implicitement présente dans les formules utilisées par les sémantiques graduées existantes. La preuve en est qu'il est possible de calculer le score d'un argument en combinant son score de base ($\mathtt{Deg}^{\mathcal{S}}_{F \ominus_y \mathcal{A}}(y)$) et l'impact de l'ensemble des arguments du système d'argumentation sur cet argument ($\mathtt{Imp}^{\mathcal{S}}_{\mathsf{F}}(\mathcal{A}, y)$).

Proposition 1 *Soient* $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ *un système d'argumentation et* $y \in \mathcal{A}$*. Pour toute sémantique graduée* $\mathcal{S}$*,*

$$\mathtt{Deg}_F^{\mathcal{S}}(y) = \mathtt{Deg}_{F \ominus_y \mathcal{A}}^{\mathcal{S}}(y) + \mathtt{Imp}_F^{\mathcal{S}}(\mathcal{A}, y)$$

Preuve: *Ce résultat peut être facilement déduit en appliquant notre définition de l'impact (Définition 7) vu que* $\mathcal{A}$ *est, par définition, non-attaqué :*

$$\begin{aligned}\mathtt{Deg}_F^{\mathcal{S}}(y) &= \mathtt{Deg}_{F \ominus_y \mathcal{A}}^{\mathcal{S}}(y) + \mathtt{Imp}_F^{\mathcal{S}}(\mathcal{A}, y) \\ &= \mathtt{Deg}_{F \ominus_y \mathcal{A}}^{\mathcal{S}}(y) + \mathtt{Deg}_F^{\mathcal{S}}(y) - \mathtt{Deg}_{F \ominus_y \mathcal{A}}^{\mathcal{S}}(y) \\ &= \mathtt{Deg}_F^{\mathcal{S}}(y)\end{aligned}$$

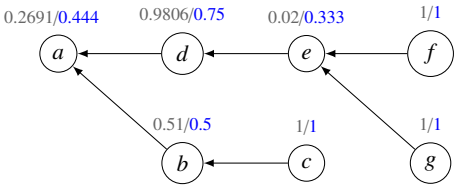
□

Exemple 2 *En appliquant cette formule sur le système d'argumentation de la Figure 2, nous obtenons :*

(CS) $\mathtt{Deg}_F^{CS}(a) = \mathtt{Deg}_{F \ominus_a \mathcal{A}}^{CS}(a) + \mathtt{Imp}_F^{CS}(\mathcal{A}, a) = 1 + (\mathtt{Deg}_F^{CS}(a) - \mathtt{Deg}_{F \ominus_a \mathcal{A}}^{CS}(a)) = 1 - 0.7309 = 0.2691$

(Cat) $\mathtt{Deg}_F^{Cat}(a) = \mathtt{Deg}_{F \ominus_a \mathcal{A}}^{Cat}(a) + \mathtt{Imp}_F^{Cat}(\mathcal{A}, a) = 1 + (\mathtt{Deg}_F^{Cat}(a) - \mathtt{Deg}_{F \ominus_a \mathcal{A}}^{Cat}(a)) = 1 - 0.556 = 0.444$

Mesurer uniquement l'impact des ensembles non-attaqués d'arguments peut avoir du sens pour des applications comme les plateformes de débat en ligne où les gens débattent avec des arguments sur un sujet donné. Un débat peut être formalisé par un AF qui a, dans de nombreux cas, une structure en forme d'arbre ou d'étoile, montrant que plusieurs sous-débats existent. Par exemple, les arguments pour ou contre le régime végétalien peuvent être divisés en plusieurs catégories comme son impact environnemental, son impact sur la santé, ses effets psychologiques, etc. Vérifier l'impact de ces différentes catégories (c'est-à-dire les sous-arbres de l'AF) sur le sujet implique de mieux connaître l'influence de chacune de ces catégories sur le débat.



$\mathtt{Imp}_{\mathsf{F}}^{\mathcal{S}}(X, a)$	CS	Cat
$X = \mathcal{A}$	-0.7309	-0.556
$X = \{e, f, g\}$	0.0138	0.044
$X = \{b, c\}$	-0.2547	-0.127
$X = \{f, g\}$	-0.2215	-0.056
$X = \{c\}$	0.2353	0.0808

FIGURE 2 – En haut de la figure, un système d'argumentation avec, au-dessus de chaque argument, les degrés d'acceptabilité retournés par la counting sémantique (avec $\alpha = 0.98$) et la sémantique h-categorizer [CS/Cat]. En bas de la figure, le tableau contient l'impact de certains ensembles non-attaqués d'arguments sur le degré d'acceptabilité de l'argument a.

3.2 Impact général

En l'état actuel, la formule de l'impact (Définition 7) ne peut pas être utilisée pour un ensemble attaqué d'arguments. En appliquant, par exemple, cette formule pour calculer l'impact de $\{e\}$ sur l'argument a de la Figure 2, cela revient en fait à calculer l'impact de $\{e, f, g\}$ sur a. En effet, en supprimant l'argument e, les chemins de f et g (les attaquants directs de e) vers a sont également supprimés, ce qui implique de prendre également en compte l'impact de f et g sur a.
Afin de calculer l'impact de tout ensemble d'arguments X sur un argument y, nous proposons de considérer le degré d'acceptabilité de y lorsque les arguments de X sont les plus forts (c'est-à-dire lorsque leurs attaquants directs sont supprimés) afin d'éviter de prendre en compte l'impact des arguments qui influencent également ces arguments. Le fait que ces arguments soient attaqués (et/ou défendus) sera pris en compte lors du calcul de l'impact de ces attaquants

sur y.

Définition 8 (Impact) *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation, $y \in \mathcal{A}$ et $X \subseteq \mathcal{A}$. Soit $\mathcal{S}$ une sémantique graduée. L'**impact** de X sur y est :*

$$\texttt{Imp}_F^{\mathcal{S}}(X, y) = \texttt{Deg}^{\mathcal{S}}_{F \ominus_y (\bigcup_{x \in X} \mathcal{R}_1(x))}(y) - \texttt{Deg}^{\mathcal{S}}_{F \ominus_y X}(y)$$

Cette définition généralise la Définition 7 car si $\bigcup_{x \in X} \mathcal{R}_1(x) = \emptyset$ (signifiant que X est non-attaqué) alors les deux formules sont équivalentes. Étant donné que le degré d'acceptabilité d'un argument est compris entre 0 et 1 (voir Définition 3), la valeur de l'impact d'un ensemble d'arguments sur un argument s'inscrit dans l'intervalle $[-1, 1]$.

Proposition 2 *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation, $y \in \mathcal{A}$ et $X \subseteq \mathcal{A}$. Soit $\mathcal{S}$ une sémantique graduée. Nous avons $\texttt{Imp}_F^{\mathcal{S}}(X, y) \in [-1, 1]$.*

Preuve: *Selon la Définition 3, le degré d'acceptabilité d'un argument $x \in \mathcal{A}$ appartient à l'intervalle [0,1]. Ainsi, nous avons $\texttt{Deg}^{\mathcal{S}}_{F \ominus_y (\bigcup_{x \in X} \mathcal{R}_1(x))}(y) \in [0, 1]$ et $\texttt{Deg}^{\mathcal{S}}_{F \ominus_y X}(y) \in [0, 1]$. Par conséquent, en appliquant les règles d'opérations arithmétiques classiques sur les intervalles, nous avons $\texttt{Imp}_F^{\mathcal{S}}(X, y) \in [0 - 1, 1 - 0] = [-1, 1]$.* □

Trois grandes catégories d'impact peuvent être définies, à savoir : *positif*, *négatif* et *neutre*.

Définition 9 *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation, $y \in \mathcal{A}$ et $X \subseteq \mathcal{A}$. Soit $\mathcal{S}$ une sémantique graduée. Nous disons que :*

- *X a un **impact positif** sur y si $\texttt{Imp}_F^{\mathcal{S}}(X, y) > 0$,*
- *X a un **impact négatif** sur y si $\texttt{Imp}_F^{\mathcal{S}}(X, y) < 0$,*
- *X a un **impact neutre** sur y si $\texttt{Imp}_F^{\mathcal{S}}(X, y) = 0$.*

Notons que le fait qu'un ensemble d'arguments ait un impact spécifique (positif, négatif ou neutre) sur un argument ne signifie pas que tous les arguments appartenant à cet ensemble ont également cet impact spécifique sur ce même argument. En effet, nous pouvons voir que sur l'AF de la Figure 2, quand CS est utilisée, l'ensemble $\{e, f, g\}$ a un impact positif sur a alors que e est le seul argument à avoir un impact positif sur a (f et g ont un impact négatif).

Trois notations permettant de sélectionner les arguments ayant un impact positif, négatif ou neutre sur un autre argument peuvent alors être définies et seront utilisées dans la partie interprétabilité (Section 5).

Notation 2 *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation et $y \in \mathcal{A}$. Soit $\mathcal{S}$ une sémantique graduée.*

- $\texttt{I}^+_{\mathcal{S}}(y) = \{x \in \mathcal{A} \mid \{x\}$ *a un impact positif sur* $y\}$
- $\texttt{I}^-_{\mathcal{S}}(y) = \{x \in \mathcal{A} \mid \{x\}$ *a un impact négatif sur* $y\}$
- $\texttt{I}^=_{\mathcal{S}}(y) = \{x \in \mathcal{A} \mid \{x\}$ *a un impact neutre sur* $y\}$

Exemple 3 *Calculons l'impact de chaque argument de l'AF de la Figure 2 sur l'argument a lorsque CS est utilisée ($\alpha = 0,98$). Détaillons d'abord l'impact de e sur a :* $\texttt{Imp}_F^{CS}(\{e\}, a) = \texttt{Deg}^{CS}_{F \ominus_a \{f,g\}}(a) - \texttt{Deg}^{CS}_{F \ominus_a \{e\}}(a) = 0.4906 - 0.25530 = 0.2353$.
Pour les autres arguments, nous obtenons $\texttt{Imp}_F^{CS}(\{a\}, a) = 0$, $\texttt{Imp}_F^{CS}(\{b\}, a) = \texttt{Imp}_F^{CS}(\{d\}, a) = -0.49$, $\texttt{Imp}_F^{CS}(\{c\}, a) = 0.2353$ *et* $\texttt{Imp}_F^{CS}(\{f\}, a) = \texttt{Imp}_F^{CS}(\{g\}, a) = -0.1108$.
Ainsi, $\texttt{I}^+_{CS}(a) = \{c, e\}$, $\texttt{I}^-_{CS}(a) = \{b, d, f, g\}$ *et* $\texttt{I}^=_{CS}(a) = \{a\}$.

4 Propriété Impact Équilibré

La définition d'une nouvelle sémantique graduée est souvent accompagnée d'une évaluation axiomatique de celle-ci [2, 6]. Ces axiomes sont utilisés pour comparer les sémantiques graduées entre elles mais aussi pour mieux comprendre le comportement d'une sémantique graduée donnée dans des situations spécifiques. Plusieurs axiomes traitent notamment du rôle et de l'impact d'un argument ou d'une attaque sur un argument "cible". Ils ont pour but de répondre à des questions comme : Est-ce l'attaque d'un argument "tue" (cf. Killing property [2]) ou juste affaiblit (cf. Weakening property [2]) la cible de l'attaque ? Cependant, l'aspect binaire de ces axiomes présente quelques limites. Par exemple, deux sémantiques peuvent considérer qu'une attaque affaiblit sa cible (toutes deux satisfont alors la propriété Weakening) mais avec des niveaux d'affaiblissement différents. Malheureusement, cette distinction ne peut être capturée avec ces axiomes. Par exemple, CS et h-categorizer satisfont tous les deux la propriété Weakening. Cependant, le calcul de l'impact de b et c sur a dans les trois AF de la Figure 3 avec la sémantique h-categorizer montre que leur impact sur a est moins important lorsqu'ils attaquent ensemble ($\texttt{Imp}^{\text{Cat}}_{\text{F}_3}(\{b, c\}, a) = -0.667$) que lorsqu'ils attaquent a séparément ($\texttt{Imp}^{\text{Cat}}_{\text{F}_1}(\{b\}, a) + \texttt{Imp}^{\text{Cat}}_{\text{F}_2}(\{c\}, a) = -0.5 + -0.5 = -1$). A l'inverse, avec la counting sémantique, les deux donnent le même résultat : $\texttt{Imp}^{\text{CS}}_{\text{F}_3}(\{b, c\}, a) = -0.98 = -0.49 + -0.49 = \texttt{Imp}^{\text{CS}}_{\text{F}_1}(\{b\}, a) + \texttt{Imp}^{\text{CS}}_{\text{F}_2}(\{c\}, a)$.

Pour capturer cette idée, nous définissons une nouvelle propriété, appelée Impact Équilibré [1] (BI), qui stipule que la somme de l'impact de deux ensembles d'arguments, qui n'ont aucun argument en commun, sur un argument y doit être égale à l'impact de l'union de ces deux ensembles d'arguments sur y.

Propriété 1 (Impact Équilibré (BI))
Une sémantique graduée $\mathcal{S}$ satisfait la propriété Impact Équilibré si et seulement si pour tout $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ et

1. Balanced Impact en anglais

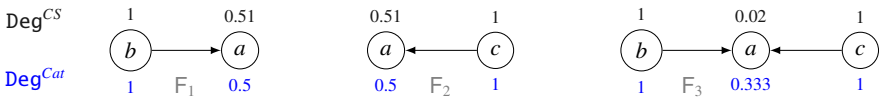


FIGURE 3 – Trois systèmes d'argumentation F_1, F_2, F_3 montrant la différence d'impact entre la counting sémantique (CS) et la sémantique h-categorizer (Cat).

$X, Z \subseteq \mathcal{A}$ t.q. $X \cap Z = \emptyset$,

$$\texttt{Imp}_F^{\mathcal{S}}(X, y) + \texttt{Imp}_F^{\mathcal{S}}(Z, y) = \texttt{Imp}_F^{\mathcal{S}}(X \cup Z, y)$$

Vérifions maintenant si les observations déduites sur l'exemple de la Figure 3 concernant BI peuvent être généralisées à tous les systèmes d'argumentation.

Proposition 3 *La counting sémantique satisfait la propriété Impact Équilibré (BI).*

Preuve: *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ un système d'argumentation et $a, b \in \mathcal{A}$. Commençons par réécrire, d'une manière différente mais équivalente, la formule permettant de calculer le degré d'acceptabilité d'un argument lorsque la counting sémantique est utilisée. Comme décrit dans [19], l'idée principale de l'approche "est de compter le nombre d'attaquants et de défenseurs pour chaque argument". Au lieu d'utiliser une approche matricielle, il est possible de calculer le degré d'acceptabilité d'un argument en combinant l'ensemble des chemins menant à cet argument. Nous désignons par $P_k(b, a)$ l'ensemble des chemins de b vers a avec une longueur $k \geq 0$. Ce qui nous donne la formule suivante :*

$$\texttt{Deg}_F^{CS}(a) = 1 + \sum_{b \in \mathcal{A}} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(b, a)|$$

En appliquant la définition de l'impact en utilisant cette formule, on peut noter que les chemins restants sont ceux dont l'argument au début du chemin appartient à l'ensemble dont l'impact doit être évalué. Donc si $\mathsf{F}' = \mathsf{F} \ominus_a (\bigcup_{x \in X} \mathcal{R}_1(x))$ et $\mathsf{F}'' = \mathsf{F} \ominus_a X$, alors nous avons :

$$\begin{aligned}
&\texttt{Imp}_F^{CS}(X, a) = \texttt{Deg}_{F'}^{CS}(y) - \texttt{Deg}_{F''}^{CS}(a) \\
&= 1 + \textstyle\sum_{y \in Arg(F')} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(y, a)| - 1 - \\
&\qquad \textstyle\sum_{z \in Arg(F'')} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(z, a)| \\
&= \textstyle\sum_{x \in X} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(x, a)|
\end{aligned}$$

Calculons maintenant l'impact de $Z \subseteq \mathcal{A}$ et $X \cup Z$ (avec $X \cap Z = \emptyset$) sur a.
Pour Z :

$$\texttt{Imp}_F^{CS}(Z, a) = \sum_{z \in Z} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(z, a)|$$

Pour $X \cup Z$:

$$\texttt{Imp}_F^{CS}(X \cup Z, a) = \sum_{y \in X \cup Z} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(y, a)|$$

En combinant les deux formules, nous obtenons $\texttt{Imp}_F^{CS}(X, a) + \texttt{Imp}_F^{CS}(Z, a) = \sum_{x \in X} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(x, a)| + \sum_{z \in Z} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(z, a)|$. Comme l'intersection de X et Z est vide, nous pouvons dire que $\texttt{Imp}_F^{CS}(X, a) + \texttt{Imp}_F^{CS}(Z, a) = \sum_{y \in X \cup Z} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha^k}{N} |P_k(y, a)|$. Par conséquent, $\texttt{Imp}_F^{CS}(X, a) + \texttt{Imp}_F^{CS}(Z, a) = \texttt{Imp}_F^{CS}(X \cup Z, a)$. □

Proposition 4 *La sémantique h-categorizer ne satisfait pas la propriété Impact Équilibré (BI).*

Preuve: *Le système d'argumentation F_3 de la Figure 3 est un contre-exemple montrant que la sémantique h-categorizer ne satisfait pas Impact Équilibré (BI) car $\texttt{Imp}_{F_3}^{Cat}(\{b, c\}, a) = -\frac{2}{3} \neq -\frac{1}{3} = -\frac{1}{6} + -\frac{1}{6} = \texttt{Imp}_{F_3}^{Cat}(\{b\}, a) + \texttt{Imp}_{F_3}^{Cat}(\{c\}, a)$.* □

Cette propriété permet ainsi de distinguer les sémantiques qui répartissent de manière équilibrée et indépendante l'impact des arguments sur un argument cible. Autrement dit, pour les sémantiques qui satisfont cette propriété, l'impact d'un argument (e.g., un attaquant direct) sur un argument donné sera le même s'il est le seul à posséder un chemin vers cet argument ou s'il en existe plusieurs (e.g., s'il est l'unique attaquant direct ou s'il en existe plusieurs). En effet, sur l'exemple de la Figure 3, puisque CS satisfait BI alors l'argument b (idem pour c) possède le même impact sur a lorsqu'il est le seul attaquant (système d'argumentation F_1) que lorsque c attaque a également (système d'argumentation F_3). Ce n'est pas le cas par contre avec la sémantique h-categorizer (qui ne satisfait pas BI) qui, même si elle satisfait la propriété Weakening (i.e., plus le nombre d'attaquants directs est grand plus le degré d'acceptabilité de l'argument cible est faible), montre que l'impact des arguments l'attaquant directement est plus faible individuellement que lorsque le nombre d'attaquants directs augmente ($\texttt{Imp}_{F_1}^{Cat}(\{b\}, a) = -\frac{1}{2}$ alors que $\texttt{Imp}_{F_3}^{Cat}(\{b\}, a) = -\frac{1}{6}$). Attention, cela ne signifie pas que les sémantiques ne satisfaisant pas BI ne sont pas des bonnes sémantiques. Comme expliqué dans le papier

de Bonzon et al. [9], une propriété peut être désirable dans un contexte précis et pas dans un autre. Une étude plus poussée serait intéressante afin d'identifier les contextes où BI est désirable ou non.

Il est intéressant de noter que grâce à cet équilibre, il est possible de calculer le score d'un argument en utilisant une sémantique graduée qui satisfait BI à partir de l'impact que chaque argument de l'AF possède sur cet argument. En effet, comme expliqué dans la section 3.1, le score d'un argument y dépend de son score initial (c'est-à-dire $\mathtt{Deg}^{\mathcal{S}}_{\mathsf{F}\ominus_y\mathcal{A}}(y)$) et de l'impact de l'ensemble des arguments du système d'argumentation ($\mathtt{Imp}^{\mathcal{S}}_{\mathsf{F}}(\mathcal{A}, y)$). Cependant, grâce à la propriété Impact Équilibré, il est possible de diviser $\mathtt{Imp}^{\mathcal{S}}_{\mathsf{F}}(\mathcal{A}, y)$ en plusieurs parties, chacune d'entre elles représentant l'impact d'un argument individuel du système d'argumentation sur y. Définissons-le d'abord formellement pour les systèmes d'argumentation acycliques.

Proposition 5 *Soient $\mathsf{F} = \langle \mathcal{A}, \mathcal{R}\rangle$ un système d'argumentation acyclique et $y \in \mathcal{A}$. Soit $\mathcal{S}$ une sémantique graduée qui satisfait BI. Le score de y peut être calculé comme suit :*

$$\mathtt{Deg}^{\mathcal{S}}_{\mathsf{F}}(y) = \mathtt{Deg}^{\mathcal{S}}_{F\ominus_y\mathcal{A}}(y) + \sum_{x\in\mathcal{A}} \mathtt{Imp}^{\mathcal{S}}_{\mathsf{F}}(\{x\}, y)$$

Preuve: *Soient $\mathcal{A} = \{y, y_1, y_2, \ldots, y_n\}$ et $\mathcal{S}$ une sémantique graduée qui satisfait BI. Cette décomposition peut être faite via la formule de la Proposition 1 :*

$$\mathtt{Deg}^{\mathcal{S}}_{F}(y) = \mathtt{Deg}^{\mathcal{S}}_{F\ominus_y\mathcal{A}}(y) + \mathtt{Imp}^{\mathcal{S}}_{F}(\mathcal{A}, y)$$

Comme $\mathcal{S}$ satisfait BI, il est possible de décomposer $\mathtt{Imp}^{\mathcal{S}}_{F}(\mathcal{A}, y)$ en plusieurs sous-ensembles chacun composé d'un argument seul dans $\mathcal{A}$. En effet, comme l'ensemble des ensembles contenant chacun un des arguments de $\mathcal{A}$ est une partition de $\mathcal{A}$ où chaque ensemble est indépendant des autres, nous pouvons écrire : $\mathtt{Imp}^{\mathcal{S}}_{F}(\mathcal{A}, y) = \mathtt{Imp}^{\mathcal{S}}_{F}(\{y\}, y) + \mathtt{Imp}^{\mathcal{S}}_{F}(\{y_1\}, y) + \ldots + \mathtt{Imp}^{\mathcal{S}}_{F}(\{y_n\}, y) = \sum_{x\in\mathcal{A}} \mathtt{Imp}^{\mathcal{S}}_{\mathsf{F}}(\{x\}, y)$. *Donc, nous avons* $\mathtt{Deg}^{\mathcal{S}}_{\mathsf{F}}(y) = \mathtt{Deg}^{\mathcal{S}}_{F\ominus_y\mathcal{A}}(y) + \sum_{x\in\mathcal{A}} \mathtt{Imp}^{\mathcal{S}}_{\mathsf{F}}(\{x\}, y)$. □

Exemple 4 *Calculons le score de a dans l'AF de la Figure 2 en utilisant l'impact de chaque argument lorsque CS est utilisée :* $\mathtt{Deg}^{cs}_{\mathsf{F}}(a) = 1 + (\mathtt{Imp}^{cs}_{F}(\{a\}, a) + \mathtt{Imp}^{cs}_{F}(\{b\}, a) + \mathtt{Imp}^{cs}_{F}(\{c\}, a) + \mathtt{Imp}^{cs}_{F}(\{d\}, a) + \mathtt{Imp}^{cs}_{F}(\{e\}, a) + \mathtt{Imp}^{cs}_{F}(\{f\}, a) + \mathtt{Imp}^{cs}_{F}(\{g\}, a)) = 1 + (0 - 0.49 + 0.2353 - 0.49 + 0.2353 - 0.1108 - 0.1108) = 0.2691$.

Afin de généraliser cette définition pour tout système d'argumentation, une étape de prétraitement est nécessaire. En effet, la suppression d'un argument dans un cycle supprime aussi bien son impact mais également celui des autres arguments du cycle. Comme la méthode fonctionne pour les systèmes d'argumentation acycliques, nous proposons de transformer un système d'argumentation cyclique en un système d'argumentation acyclique infinie [2] axée sur un argument donné. Ainsi, comme le montre la Figure 4, nous obtenons un AF en forme d'arbre où le nœud racine est l'argument cible (ici il s'agit de l'argument a_0), ses nœuds parents sont ses attaquants directs, les nœuds parents de ses nœuds parents sont ses défenseurs directs, et ainsi de suite. L'Algorithme 1 détaille ce mécanisme de transformation appelé ACY.

Algorithm 1: Fonction de transformation ACY

Data: $\mathsf{F} = \langle \mathcal{A} = \{x_1, \ldots, x_n\}, \mathcal{R}\rangle$ et $x_1 \in \mathcal{A}$ l'argument cible.
Result: $\mathsf{F}' = \langle \mathcal{A}', \mathcal{R}'\rangle$ l'AF infinie acyclique de F

```
C = {x_1}; A' = {x_1^0}; R' = ∅    // x_1^0 est appelé "the universal sink vertex" de F'
for tout argument x_i dans C do
    C = C \ {x_i}
    m_1 ← valeur maximale de m parmi x_i^m ∈ A'
    for tout argument x_j dans R_1(x_i) do
        C = C ∪ {x_j}
        if x_j^0 ∉ A' then
            A' = A' ∪ x_j^0
            R' = R' ∪ (x_j^0, x_i^{m_1})
        else
            m_2 ← (valeur maximale de m parmi x_j^m ∈ A') + 1
            A' = A' ∪ x_j^{m_2}
            R' = R' ∪ (x_j^{m_2}, x_i^{m_1})
```

Nous pouvons maintenant utiliser la transformation d'un système d'argumentation F pour définir l'impact de tout argument x sur un argument donné y comme la somme de l'impact de tous les sous-arguments de x ($x^0, x^1, \ldots$) sur y^0 dans $\mathsf{ACY}_y(\mathsf{F})$.

Définition 10 *Soit $\mathsf{F} = \langle \mathcal{A}, \mathcal{R}\rangle$ un système d'argumentation avec $y \in \mathcal{A}$. Soient $\mathsf{F}' = \mathsf{ACY}_y(\mathsf{F})$ et $\mathcal{X} = \{x^0, x^1, \ldots\}$ les sous-arguments de $x \in \mathcal{A}$ dans F'. Soit $\mathcal{S}$ une sémantique graduée qui satisfait BI. L'impact de x sur y est 0 si $\mathcal{X} = \emptyset$, sinon il est défini comme suit :*

$$\mathtt{Imp}^{\mathcal{S}}_{\mathsf{F}}(\{x\}, y) = \sum_{x^i\in\mathcal{X}} \mathtt{Imp}^{\mathcal{S}}_{\mathsf{F}'}(\{x^i\}, y^0)$$

Cette nouvelle définition de l'impact peut ensuite être utilisée dans la Définition 5 pour calculer le score d'un argument donné pour tout type d'AF.

2. En pratique, le score de chaque argument est calculé en utilisant une approche de point fixe. Si la fonction utilisée dans la sémantique graduée converge, le nombre d'itérations nécessaire à la convergence peut également être utilisé pour définir la profondeur maximale de l'AF qui aura alors une forme d'arbre.

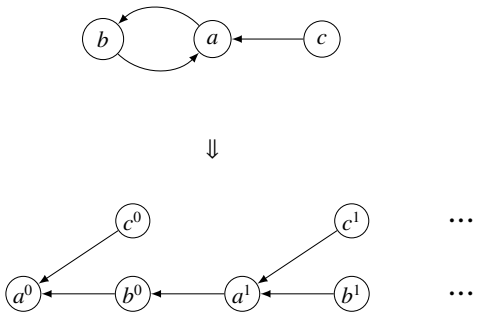


FIGURE 4 – AF cyclique transformé en AF acyclique infini

Exemple 5 *Étant donné l'AF de la Figure 4, l'impact de b sur a est* $\texttt{Imp}^{cs}_{\mathsf{F}}(\{b\}, a) = \texttt{Imp}^{cs}_{\mathsf{ACY}_a(\mathsf{F})}(\{b^0\}, a^0) + \texttt{Imp}^{cs}_{\mathsf{ACY}_a(\mathsf{F})}(\{b^1\}, a^0) + \ldots \simeq -0.63$. *Nous avons également* $\texttt{Imp}^{cs}_{\mathsf{F}}(\{c\}, a) \simeq -0.63$ *et* $\texttt{Imp}^{cs}_{\mathsf{F}}(\{a\}, a) \simeq 0.3$.
Nous obtenons ainsi $\texttt{Deg}^{cs}_{\mathsf{F}}(a) \simeq 0.04 = 1 + 0.3 - 0.63 - 0.63 = 1 + \sum_{x \in \{a,b,c\}} \texttt{Imp}^{cs}_{\mathsf{F}}(\{x\}, a)$.

5 Interprétabilité des sémantiques graduées

L'un des objectifs de l'interprétabilité des sémantiques graduées est d'identifier les éléments qui ont un impact sur le degré d'acceptabilité (retourné par une sémantique graduée donnée) pour chaque argument d'un AF. La définition 9 nous permet d'évaluer si un argument a un impact positif, négatif ou neutre sur le degré d'acceptabilité d'un argument. Il nous permet ainsi de répondre à un certain nombre de questions concernant l'impact de certains arguments sur les autres, comme l'illustre l'exemple suivant avec le système d'argumentation F de la Figure 5 :

Q : *Quels arguments ont un impact positif sur a dans* F *lorsque CS est utilisée ?*
A : c et e ont un impact positif sur a. $\texttt{I}^{+}_{cs}(a) = \{c, e\}$

Q : *Quels arguments ont un impact positif sur a dans* F *lorsque CS est utilisée ?*
A : b, d, f et g ont un impact négatif sur x.
$\texttt{I}^{-}_{cs}(y) = \{b, d, f, g\}$

Q : *Quels arguments ont un impact neutre sur a dans* F *lorsque CS est utilisée ?*
A : a n'a pas d'impact sur lui-même. $\texttt{I}^{=}_{cs}(y) = \{a\}$

Comme le montre la Figure 6, il est possible d'utiliser une représentation plus visuelle permettant de déterminer rapidement quelle est la proportion de l'impact de chaque argument sur un argument donné. Cette représentation peut éventuellement être utilisée comme support afin d'expliquer les résultats à un agent demandant plus d'informations sur la façon dont les degrés d'acceptabilité sont calculés ou pour aider (en plus des axiomes existants pour les sémantiques graduées) l'agent ayant choisi d'utiliser cette sémantique à mieux comprendre et évaluer si la sémantique répond à ses attentes ou non.

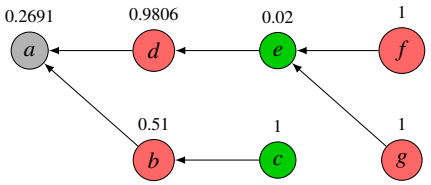


FIGURE 5 – Un système d'argumentation F avec, au-dessus de chaque argument, le score retourné par la counting sémantique (avec $\alpha = 0.98$). Les arguments qui ont un impact positif, négatif et neutre sur l'argument a sont coloriés respectivement en vert, rouge et gris.

Grâce aux valeurs d'impact (voir Définition 8), il est ainsi possible de fournir, pour chaque argument, un classement (pré-ordre total) entre les arguments d'un système d'argumentation partant de celui (ou ceux) ayant le plus d'impact positif à celui (ou ceux) ayant le plus d'impact négatif étant donnée une sémantique graduée.

Définition 11 (Classement d'impact) *Soient* $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ *un système d'argumentation et* $\mathcal{S}$ *une sémantique graduée. Le* ***classement d'impact*** $\succeq^{\mathcal{S}}_{y}$ *sur* $\mathcal{A}$ *par rapport à* $y \in \mathcal{A}$ *est défini tel que* $\forall x, z \in \mathcal{A}$,

$$x \succeq^{\mathcal{S}}_{y} z \text{ ssi } \texttt{Imp}^{\mathcal{S}}_{F}(\{x\}, y) \geq \texttt{Imp}^{\mathcal{S}}_{F}(\{z\}, y)$$

Ce classement nous permet de sélectionner, pour chaque argument, les arguments ayant l'impact positif et l'impact négatif le plus grand, s'ils existent.

Définition 12 *Soient* $\mathsf{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ *un système d'argumentation et* $\mathcal{S}$ *une sémantique graduée. Les arguments ayant le plus d'impact positif (resp. négatif) sur le degré d'acceptabilité de* $y \in \mathcal{A}$ *sont définis comme suit :*

$$PI^{\mathcal{S}}_{F}(y) = argmax_{x \in \texttt{I}^{+}_{\mathcal{S}}(y)} |\{z \in \texttt{I}^{+}_{\mathcal{S}}(y) \mid x \succeq^{\mathcal{S}}_{y} z\}|$$
$$NI^{\mathcal{S}}_{F}(y) = argmax_{x \in \texttt{I}^{-}_{\mathcal{S}}(y)} |\{z \in \texttt{I}^{-}_{\mathcal{S}}(y) \mid z \succeq^{\mathcal{S}}_{y} x\}|$$

Exemple 6 *Considérons le système d'argumentation représenté par la Figure 2. Le classement d'impact de l'argument a, lorsque CS est utilisée, est :*

$$c \simeq^{cs}_{a} e \succ^{cs}_{a} a \succ^{cs}_{a} f \simeq^{cs}_{a} g \succ^{cs}_{a} b \simeq^{cs}_{a} d$$

Par conséquent, nous obtenons $PI^{cs}_{F}(a) = \{c, e\}$ *et* $NI^{cs}_{F}(a) = \{b, d\}$.

En plus de permettre une meilleure compréhension des scores attribuées à chaque argument, ces informations pourront également être utilisées pour élaborer des stratégies au cours d'un débat. Par exemple, si une personne

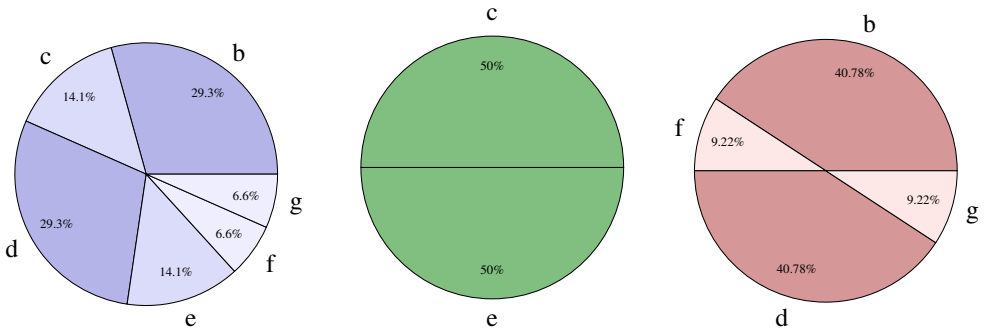


FIGURE 6 – Graphiques circulaires représentant la proportion d'impact de chaque argument sur l'argument a dans l'AF de la Figure 5. Le graphique de gauche prend en compte tous les arguments, tandis que le graphique du milieu (resp. de droite) ne prend en compte que les arguments qui ont un impact positif (resp. négatif) sur a.

veut défendre un point de vue (c'est-à-dire augmenter le degré d'acceptabilité d'un argument), elle peut identifier le ou les arguments ayant un impact négatif sur cet argument et chercher des solutions pour les attaquer en introduisant certains contre-arguments soigneusement choisis.

6 Travaux Connexes

L'interprétabilité a déjà été étudiée dans le contexte des sémantiques à base d'extension en argumentation formelle. Fan et Toni [14] ont d'abord fourni une méthode permettant d'expliquer pourquoi les arguments ont été jugés acceptables par la sémantique admissible en termes d'arguments les défendant, avant de formaliser les explications pour les arguments qui ne sont, cette fois, pas jugés acceptables par cette même sémantique à l'aide d'un arbre de conflit [15]. Bien que les sémantiques à base d'extension et les sémantiques graduées partagent le même objectif (c'est-à-dire évaluer les arguments), les deux approches restent différentes (voir la discussion dans [10] pour plus de détails). Par conséquent, l'étude de la notion d'interprétabilité pour ces deux familles de sémantiques diffère également.

En ce qui concerne les sémantiques graduées, Amgoud et al. [3] ont introduit le concept de mesure de contribution pour évaluer l'intensité de chaque attaque dans un système d'argumentation. La mesure de contribution utilisée dans ces travaux est la valeur de Shapley [21]. Cependant, seule une famille spécifique de sémantiques graduées est considérée dans cette approche (c'est-à-dire celles qui satisfont les propriétés d'indépendance syntaxique et de monotonie comme la sémantique h-categorizer). De plus, contrairement à notre méthode qui vérifie l'impact de tous les arguments dans le système d'argumentation, leur méthode ne mesure que la contribution des attaques directes sur un argument ce qui est cohérent pour la famille de sémantique étudiée dans ce travail, mais ce n'est pas nécessairement le cas pour toutes les sémantiques existantes (comme CS).

7 Conclusion

Dans cet article, nous avons présenté un cadre formel pour interpréter les résultats des sémantiques graduées en argumentation abstraite. Plus précisément, nous avons examiné la sémantique h-categorizer et la counting sémantique, et nous avons formellement étudié la notion d'impact d'un argument (ou d'un ensemble d'arguments) sur le degré d'acceptabilité d'un autre argument pour tous les types de systèmes d'argumentation (à la fois ceux cycliques et ceux acycliques). L'impact des arguments sur le degré d'acceptabilité des autres arguments est ensuite utilisé à la fois pour interpréter la logique derrière le classement qui en résulte, mais également afin de mieux comprendre les raisons pour lesquelles attaquer un argument plutôt qu'un autre peut être un choix stratégique plus judicieux.

Deux pistes de recherches faisant suite à ces travaux peuvent être envisagées. Tout d'abord, dans cet article, nous étudions la notion d'impact des systèmes d'argumentation à la Dung où les arguments sont reliés entre eux par la seule relation d'attaque. Cependant, il sera intéressant d'étendre notre analyse formelle également aux cadres d'argumentation bipolaires où les arguments sont liés entre eux par la relation d'attaque mais également par une relation de *support* [11]. L'étude de la notion d'impact dans ce cadre sera d'une importance capitale pour de nombreuses applications pratiques. Deuxièmement, nous prévoyons d'étendre notre analyse aux autres sémantiques graduées proposées dans la littérature afin d'avoir un aperçu global des propriétés de la notion d'impact sur ces sémantiques.

Références

[1] Amgoud, Leila: *A Replication Study of Semantics in Argumentation*. Dans *Proceedings of the 28th International Joint Conference on Artificial Intelligence, (IJCAI'19)*, pages 6260–6266, 2019.

[2] Amgoud, Leila et Jonathan Ben-Naim: *Axiomatic Foundations of Acceptability Semantics*. Dans *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR'16)*, pages 2–11, 2016.

[3] Amgoud, Leila, Jonathan Ben-Naim et Srdjan Vesic: *Measuring the Intensity of Attacks in Argumentation Graphs with Shapley Value*. Dans *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, pages 63–69, 2017.

[4] Atkinson, Katie, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo Ricardo Simari, Matthias Thimm et Serena Villata: *Towards Artificial Argumentation*. AI Magazine, 38(3) :25–36, 2017.

[5] Baroni, Pietro, Martin Caminada et Massimiliano Giacomin: *An introduction to argumentation semantics*. The Knowledge Engineering Review, 26(4) :365–410, 2011.

[6] Baroni, Pietro, Antonio Rago et Francesca Toni: *From fine-grained properties to broad principles for gradual argumentation : A principled spectrum*. Int. J. Approx. Reasoning, 105 :252–286, 2019.

[7] Besnard, Philippe et Anthony Hunter: *A logic-based theory of deductive arguments*. Artificial Intelligence, 128(1-2) :203–235, 2001.

[8] Bonzon, Elise, Jérôme Delobelle, Sébastien Konieczny et Nicolas Maudet: *A Comparative Study of Ranking-based Semantics for Abstract Argumentation*. Dans *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*, pages 914–920, 2016.

[9] Bonzon, Elise, Jérôme Delobelle, Sébastien Konieczny et Nicolas Maudet: *A Parametrized Ranking-Based Semantics for Persuasion*. Dans *Proceedings of the 11th International Conference on Scalable Uncertainty Management, (SUM'17)*, pages 237–251, 2017.

[10] Bonzon, Elise, Jérôme Delobelle, Sébastien Konieczny et Nicolas Maudet: *Combining Extension-Based Semantics and Ranking-Based Semantics for Abstract Argumentation*. Dans *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR'18)*, pages 118–127, 2018.

[11] Cayrol, Claudette et Marie-Christine Lagasquie-Schiex: *Bipolarity in argumentation graphs : Towards a better understanding*. Int. J. Approx. Reasoning, 54(7) :876–899, 2013.

[12] Cyras, Kristijonas, David Birch, Yike Guo, Francesca Toni, Rajvinder Dulay, Sally Turvey, Daniel Greenberg et Tharindi Hapuarachchi: *Explanations by arbitrated argumentative dispute*. Expert Syst. Appl., 127 :141–156, 2019.

[13] 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) :321–358, 1995.

[14] Fan, Xiuyi et Francesca Toni: *On Computing Explanations in Argumentation*. Dans *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, (AAAI'15)*, pages 1496–1502, 2015.

[15] Fan, Xiuyi et Francesca Toni: *On Explanations for Non-Acceptable Arguments*. Dans *Theory and Applications of Formal Argumentation - Third International Workshop, (TAFA'15)*, pages 112–127, 2015.

[16] Miller, Tim: *Explanation in artificial intelligence : Insights from the social sciences*. Artificial Intelligence, 267 :1–38, 2019.

[17] Mittelstadt, Brent D., Chris Russell et Sandra Wachter: *Explaining Explanations in AI*. Dans *Proceedings of the Conference on Fairness, Accountability, and Transparency, (FAT*'19)*, pages 279–288, 2019.

[18] Pu, Fuan, Jian Luo, Yulai Zhang et Guiming Luo: *Argument Ranking with Categoriser Function*. Dans *Proceedings of the 7th International Conference on Knowledge Science, Engineering and Management, (KSEM'14)*, pages 290–301, 2014.

[19] Pu, Fuan, Jian Luo, Yulai Zhang et Guiming Luo: *Attacker and Defender Counting Approach for Abstract Argumentation*. Dans *Proceedings of the 37th Annual Meeting of the Cognitive Science Society, (CogSci'15)*, 2015.

[20] Rago, Antonio, Oana Cocarascu et Francesca Toni: *Argumentation-Based Recommendations : Fantastic Explanations and How to Find Them*. Dans *Proceedings of the 27th International Joint Conference on Artificial Intelligence, (IJCAI'18)*, pages 1949–1955, 2018.

[21] Shapley, Lloyd S.: *A value for n-person games*, page 31–40. Cambridge University Press, 1988.

Actes JIAF 2020

A Resilient Behavior Approach Based on Non-monotonic Logic

José Luis Vilchis Medina[1] **Pierre Siegel**[2]
Vincent Risch[2] **Andrei Doncescu**[3]

[1]LIRMM, France
[2]Aix-Marseille Univ, Université de Toulon, CNRS, LIS, France
[3]LAAS, CNRS, France
jose-luis.vilchis-medina@lirmm.fr pierre.siegel@lis-lab.fr
vincent.risch@lis-lab.fr andrei.doncescu@laas.fr

This article was accepted and published at the 18th Mexican International Conference on Artificial Intelligence (MICAI) in October 2019.

Résumé

Cet article, décrit une approche de représentation d'un système résilient, qui a la capacité d'absorber les perturbations et de surmonter les crises. Nous présentons KOSA qui est un cadre composé d'un ensemble de connaissances décrivant des objectifs, des états et des actions, le tout étant liés par un ensemble de règles. Le lien est exprimé par une *théorie des défauts*. En premier lieu, la résilience est définie comme une relation entre les états et les objectifs. Ensuite, à partir d'un état courant, on calcule des *extensions* de la théorie des défauts qui représentent des états futurs possibles, dont un est choisi. Une trajectoire est une suite d'états qui va d'un état initial à un objectif. Les états sont calculés à partir d'extensions. La trajectoire est *résiliente*, si partant d'un état initial, quelque soient les perturbations, il existe toujours une trajectoire qui aboutit à l'objectif. Nous présentons comme exemple, les décisions de pilotage d'un avion. Finalement, un comportement théorique discret du modèle est décrit. Une définition de distance entre les extensions est introduite.

Abstract

In this article we present an approach for representing a resilient system which has the capability of absorb perturbations and overcome a disaster. A framework called KOSA is depicted, which contains a set of knowledge describing objectives, states and actions, linked by a set of rules. This link is expressed by a default theory. First, we define resilience as a relation among states and objectives. Secondly, from a given state, extensions are calculated, which provides information where to go to the future state. A trajectory is a sequence of states from an initial state to a target. The states are calculated from extensions. The trajectory is resilient, if starting from an initial state, no matter the perturbations, there is always a trajectory that leads to the objective. Examples of piloting an airplane are concerned through this paper. Eventually, we present a discrete theoretical behavior of the model. Finally the notion of distance among extensions is introduced.

1 Introduction

In every approaches, resilience always concerns the capability for a system to absorb perturbations and overcome a disaster [1] [2]. For example, nature exhibits many resilient behaviors : flock of birds, school of fish, ecological disasters... This behavior allows flocks of birds and schools of fishes to survive in an uncertainty environment, looking for food in order to preserve the species despite of predators. Ecological disasters such as seaquakes, tornados and earthquakes involve a resilient behavior since in all the cases after disasters occurred, elements of the nature will find an equilibrium point [12, 8, 1, 4]. The pioneer of the definition and use of resilience was Holling [6], who explained primarily how ecosystems are able to find equilibrium points after natural disasters, without taking into account all the variables involved in the process. This definition was then extended to other areas such as psychology and engineering [21, 22].

1.1 Resilience : State of the Art

Here, we consider Holling's definition about resilience. He defined it regarding four properties : *reorganization*, *exploration*, *release* and *conservation* [6, 7]. Resilience

1. https ://en.wikipedia.org/wiki/Resilience
2. https ://dictionary.cambridge.org/search/english/direct/ ?q=resilience

was recently studied from a logical point of view : non-monotonic logic [13] was used to describe, mainly, two of the four properties defined by Holling : *exploration* and *conservation* (mostly, to *explore* solutions and *conserve* consistency when facing of perturbations).

From this view, we discuss some questions that motivated this study : *What is the relation among states and objectives ? Is there a resilient behavior ? Are there resilient trajectories ?*

In this article, we introduce a model which allows to understand the property of resilience and the interactions among the elements of a system. Regarding the first question about the relation among states and objectives, this is a non-monotonic relation. Thanks to non-monotonic logic, we can formalize contradictories states, when perturbations occurred, and conserve the consistency of the knowledge according to the objectives. Regarding the resilient behavior and resilient trajectories, we need define some concepts. In general physics, a trajectory is defined as the successions of the positions of a body in a framework [3, 2]. In here, a trajectory will be defined as successions of jumps among fixed points. A fixed-point [5] is a solution of an equation or a system of equations. These fixed points are obtained using defaults from a default theory. This default theory is the core of default logic [18] that is a type of non-monotonic logic [19]. Through this study, examples of piloting an airplane are going to be explained. There are cases that involve contradictories situations, e.g. emergency landing, bad human decisions, system failures...[16, 11, 17]. Recently, fatal accidents have occurred with 737 airplanes operated by Boieng [3]. On the one hand, software in MCAS [4] was not resilient to adapt the physical modifications of the engines in order to optimize fuel consummation. On the other hand, pilots on board also were not resilient to overcome the situation since they were not trained to solve the bad measures displayed on the cockpit. In summary, these problems were the result of two non-resilient systems, because of a system failure and incomplete information.

1.2 Classical Logic

Logic is a particular way of thinking, it studies the formal principles of inference. This is logical consequences from given axioms. Formal systems, e.g., propositional, predicate, modal... are symbolic constructions in a particular language which allows to study inference [19]. Propositional logic is defined as least set of expressions satisfying : $\top$ (true) and $\bot$ (false) are formulas, and if A and B are formulas : $\neg A$ (not A), $A \wedge B$ (A and B), $A \vee B$ (A or B) and $A \rightarrow B$ (A implies B). First-Order Logic (FOL) or predicate logic is an extension of propositional logic that includes universal and existential quantifiers, $\forall$ and $\exists$, respectively. Predicates are denoted by P, Q, R... FOL is very expressive, so it is very natural to formalize sentences. For instance, We can formalize the next sentence : "all airplanes land on wheels", with the following rule :

$$\forall y, Airplane(y) \rightarrow Land\_on\_wheels(y) \quad (1)$$

But we also know that some floatplanes are airplanes that do not land on wheels and some airplanes use skis to land on ice or snow. So, we have the following rules :

$$\forall y, Ski\_airplane(y) \rightarrow Airplane(y) \quad (2)$$

$$\forall y, Ski\_airplane(y) \rightarrow \neg Land\_on\_wheels(y) \quad (3)$$

We can see that formalizations (1) and (3) are contradictory. This is because classical logic, such as FOL, is monotonic. This property is very important in the world of mathematics, because it allows to describe lemmas previously demonstrated. But this property cannot be applied to uncertain, incomplete information or exceptions. In such situations, we would expect adding new information or set of formulas to a model, the set of consequences of this model to be reduced. Formally the property of monotony is : $A \vdash w$ then $A \cup B \vdash w$. The problem leads directly to the general representation of common sense knowledge. By moving to non-monotonic framework, we can carry out the principle of explosion and nevertheless reach a conclusion.

1.3 Default Logic

It is one of the best known formalization for default reasoning, founded by Raymond Reiter. This kind of formalization allows to infer arguments based on partial and/or contradictory information as premises [18]. A default theory is a pair $\Delta = (D, W)$, where D is a set of defaults and W is a set of formulas in FOL. A default d is : $\frac{A(X):B(X)}{C(X)}$, where $A(X), B(X), C(X)$ are well-formed formulas. $A(X)$ are the *prerequisites*, $B(X)$ are the *justifications* and $C(X)$ are the *consequences*. Where $X = (x_1, x_2, x_3, \ldots, x_n)$ is a vector of free variables (non-quantified). Intuitively a default means,"if $A(X)$ is true, and there is no evidence that $B(X)$ might be false, then $C(X)$ can be true". The use of defaults implies the generation of sets containing the consequences of these defaults, called extensions. An extension can be seen as a set of beliefs of acceptable alternatives. Formally, an extension of a default theory Δ is the smallest set E of logical formulas for which the following property holds : If d is a default of D, whose the prerequisite is in E, and the negation of its justification is not in E, then the consequent of d is in E [18].

Definition 1. Let $\Delta = (D, W)$, an *extension* E of Δ is define :

3. https ://www.boeing.com/commercial/737max/737-max-software-updates.page

4. Maneuvering Characteristics Augmentation System

— $E = \bigcup_{i=0}^{\infty} E_i$ with :
— $E_0 = W$ and,
— for $i > 0 : E_{i+1} = Th(E_i) \cup \{C(X) \mid \frac{A(X):B(X)}{C(X)} \in D, A(X) \in E_i, \neg B(X) \notin E\}$

Where as $Th(E_i)$ is the set of formulas have derived from E_i. A default is said to be normal when defaults have the form : $\frac{A(X):C(X)}{C(X)}$. The main result regarding normal defaults theories is that at least one extension is always guaranteed. The original version of the definition of an extension is difficult to compute in practice. Since it involves checking that $\neg B \notin E$ while E is not yet calculated. In the case of normal defaults, E is an extension of Δ if and only if : we replace $\neg B(X) \notin E$ by $\neg C(X) \notin E_i$. Regarding the rules (1) and (3), we can refine the sentence *"all airplanes land on wheels"* by "*generally, airplanes land on wheels*". Having a default theory that is composed of $D = \{\frac{Airplanes(y):Land\_on\_wheels(y)}{Land\_on\_wheels(y)}\}$, and a knowledge about airplanes :

$$W = \{Floatplane(y) \rightarrow Airplane(y), \\ Floatplane(y) \rightarrow \neg Land\_on\_wheels(y)\}$$

Using D, we can note that the prerequisite $Airplane(y)$ is true and the justification $Land\_on\_wheels(y)$ is inconsistent with W, because of :

$$Floatplane(y) \rightarrow \neg Land\_on\_wheels(y),$$

then we cannot conclude that floatplanes land on wheels. But we know that some floatplanes have wheels, formally, $W \cup \{Floatplane\_wheels(y) \rightarrow Airplane(y)\}$. With this a new information, the prerequisite of D is true and the justification is consistent, hence we can conclude that there are floatplanes that have wheels and land on wheels.

2 KOSA

KOSA is an acronym for Knowledge-Objectives-States-Actions framework. It is a formalization in default logic which allows to study the property of resilience. KOSA is a theory which use non-temporal logic to describe its evolution. This theory is used to describe a resilience system.

Definition 2. A *KOSA theory* is a default theory $\Delta = (D, W)$, where $W = (R \cup I)$:

— D is a set of defaults which represents uncertain rules. It contains actions : $\frac{A(x):B(x)}{C(x)}$, and perturbations : $\frac{:C(x)}{C(x)}$,
— R is a set of formulas in FOL which represents certain rules,
— I is a set of grounded literals which represents the state of a system, thereafter we will say that I is a state.

$D = \{d_1, d_2, d_3, \cdots\},\ R = \{r_1, r_2, r_3, \cdots\},\ I = \{i_1, i_2, i_3, \cdots\}$

Therefore a *KOSA theory*, $\Delta = (D, (R \cup I))$, has two types of knowledge : *static* and *dynamic*. D and R are *static*, when KOSA evolves these rules do not change. On the other hand, I is the dynamic system.

Definition 3. A *transition* is a change of state, $\Delta = (D, (R \cup I)) \rightsquigarrow \Delta' = (D, (R \cup I'))$. Considering that D and R are static, the evolution occurs when I changes. Hence a transition amounts to $I \rightsquigarrow I'$.

Example. In the context of piloting an airplane, a transition $I \rightsquigarrow I'$ can be as follows :

$$D = \{\frac{emergency : Land()}{Land()}, \frac{\neg obstacle : Land()}{Land()} \cdots, \\ \frac{: Yoke()}{Yoke()}, \frac{: \neg Motor()}{\neg Motor()} \cdots\} \\ R = \{Aircraft() \rightarrow Flight(), \cdots\}$$

We have D and R that are fixed rules, and I will change, for instance :

$$I = \{Altitude(50), Compass(north), AirSpeed(80), \cdots\}$$

$$I' = \{Altitude(80), Compass(west), AirSpeed(70), \cdots\}$$

Definition 4. A *perturbation* is a modification of some values of I (Fig.1 represents a perturbation that can trigger a transition).

In practice, perturbations can have many causes, e.g. when the pilot pulls the yoke (yoke's position changes), the wind changes (airspeed changes), instructions are given by the control tower (state of flight changes)... In a real system, perturbations occur very often, e.g., airplane's position changes even if all parameters are stables [5].

$$\begin{array}{c} \wr \\ I \rightsquigarrow I' \end{array}$$

FIGURE 1 – A vertical arrow represents a perturbation that can trigger a transition.

Definition 5. A *trajectory* $T = \{I = i_0, i_1, i_2, \cdots, i_{n-1}, i_n = I'\}$ is a sequence of states, for all $i_k \in I$, $R \cup i_k$ is consistent with $0 \leq k \leq n$. (Fig.2 is a trajectory T with some perturbations). A *long-term objective* I' is the last element of a sequence T, and *intermediates objectives* are i_k with $0 \leq k < n$.

For the moment we can consider that *short-terms objectives* are *intermediates objectives* with a reduced number of states. Further on, we will detail the formal definition of *short-term objectives*.

5. In fact, there are two types of disturbances, internal (pilot pulls the yoke) and external (changes in the environment). We just mention them but we are not going to detail them because of place unavailable.

$$T = \{I = i_0, i_1, i_2, \cdots, i_{n-1}, i_n = I'\}$$

FIGURE 2 – A trajectory T with some perturbations.

Example. During a take-off, an airplane should have above stall speed as a *short-term objective*. Once take-off is done, he should climb to increase in altitude, which is another *short-term objective*. He will maintain this objective until he reaches a specific altitude and keep on, a *long-term objective*. To summarize, piloting an airplane is following different objectives and changing them depending of the perturbations.

Definition 6. Let Δ a *KOSA theory*, T is a *trajectory* of Δ and I' an objective of T (that means I' is the last element of T).

— $K = (\Delta, T)$ is resilient, if for all *perturbations* on T there exists $K' = (\Delta, T')$, such that I' is the objective of T'.
— Δ is resilient, if for all trajectories T, $K = (\Delta, T)$ is resilient.

Considering that all the parameters of the states can be modified. We have that Δ is resilient if $K = (\Delta, T)$ can reach an objective I' of T from a perturbed state I_p, passing from I_p to I' with $W = (R \cup I)$ consistent. We give a method to find a trajectory T using default logic. Consider both $K = (\Delta, T)$ and an objective I' of T. Given a perturbation in the current state I (perturbed state I_p), this will trigger a calculation of extension E. Selecting the best extension E it will be possible to reach I'. For that, we consider that each default has a ponderation with different criteria (these could be importance, security, legislation, ...), e.g. $d_x = [C_1, C_2, \ldots, C_m]$ with $C_m \in [0, 1, 2, \ldots, 100]$. Then, $E = \{d_1, d_2 \ldots\} = \{[d_{1C_1}, d_{1C_2}, \ldots], [d_{2C_1}, d_{2C_2}, \ldots] \ldots\}$, that means a default d_1 has more than two ponderations criteria d_{1C_1} and d_{1C_2}, a default d_2 has more than two ponderations criteria d_{2C_1} and d_{2C_2}, and so on. We can see that each extension has multiple criteria, we need to separate each criterion for each extension. For this we are inspired by previous research [20, 13]. For a given E with two default d_1, d_2 and two criteria C_1, C_2, a normalization can be as follows :

$$|d_{1C_1}| = \frac{d_{1C_1}}{d_{1C_1}} + \frac{d_{1C_2}}{d_{1C_1}} + \cdots + \frac{d_{1C_m}}{d_{1C_1}}$$
$$|d_{2C_2}| = \frac{d_{2C_1}}{d_{1C_2}} + \frac{d_{2C_2}}{d_{1C_2}} + \cdots + \frac{d_{2C_m}}{d_{1C_2}}$$
$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$
$$|d_{xC_m}| = \frac{d_{xC_1}}{d_{1C_m}} + \frac{d_{xC_2}}{d_{1C_m}} + \cdots + \frac{d_{xC_m}}{d_{1C_m}}$$

If we continue the normalization for all E calculated, we will have an array of extensions E and normalization criteria. Applying an opportunistic principle [9, 10] which in decision theory is a minimax function, we obtain a solution E_n. This E_n is the best solution to reach I'. In this way we have the transition from I_p to I'. However, if there are more perturbations there will be *intermediate objectives*, resulting a trajectory $T = \{I_p = i_0, i_1, i_2, \cdots, i_{n-1}, i_n = I'\}$. To be more realistic, we can considerate a system with an *long-term objective* I' interacting through an uncertainty environment, a perturbed state I_p triggers a computation of extensions E at moment S_p, $E = \{I_0, I_1, I_3, I_5\}$, then an extension I_1 is chosen using the same principle as before in this section. At some moment, perturbation ζ_1 occurs and extensions are computed one more time : $E = \{I_1, I_4, I_5, I_6\}$, and I_6 is selected. This process occurs every moment a perturbation ζ occurs. In this sense, an objective I' is the concatenation among states I_k and perturbations ζ, a trajectory can be as follows : $T_\star = \{I_1 \cdot \zeta_0 \cdot I_6 \cdot \zeta_1 \cdot I_3 \cdot \zeta_2 \cdot I_6 \cdot \zeta_3 \cdot I_5 \cdot \zeta_4 \cdot I_4 \cdot \zeta_5 \cdot I_1 \cdots\}$. Depending on the force of ζ different trajectories can be generate, for instance : $T_\triangle = \{I_5 \cdot \zeta_0 \cdot I_4 \cdot \zeta_1 \cdot I_3 \cdot \zeta_2 \cdot I_6, \zeta_3 \cdot I_5 \cdot \zeta_4 \cdot I_4 \cdot \zeta_5 \cdot I_6 \cdots\}$ (Fig.3 represents the evolution of trajectories).

The different between $T_\star$ and $T_\triangle$ is the magnitudes of the forces ζ, that's means if a trajectory is longer then ζ has a great impact and vice-versa. In practice, grounded states I are made at each interval of time. This depends of sampling time of the system.

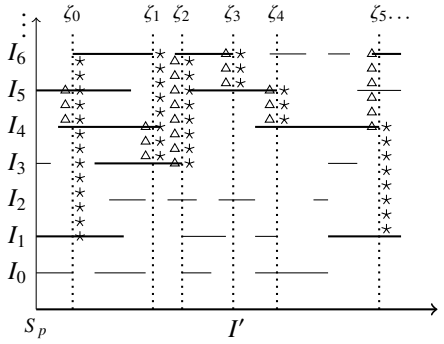


FIGURE 3 – Evolution of trajectories $T_\star$ and $T_\triangle$.

Definition 7. Let $K = (\Delta, T)$ resilient where Δ is a KOSA theory and T is a trajectory of Δ. There is a *strong* or *safe* resilience on T.

— *strong* resilience is the ability of $K = (\Delta, T)$ to reach an objective I' (an objective I' is the last element of T) regardless of the perturbations it suffers,
— *safe* resilience is the ability of $K = (\Delta, T)$ to transform a final objective I' to an intermediate objective I'', in order to maintain in good conditions the elements of a physical system.

Example. In aviation, pilots in a twin-engine aircraft can

land with a single one because the other suffered damage in mid-flight [6], this can be considerate as a *strong* resilience.

Example. When an electric motor rotates certain revolutions per minute and at some point it has a fault. The motor could demand more current to maintain the revolutions. Thanks to new technology this kind of electrical systems include protection systems. In case of a fault, it should enter a *safe* resilience so as not to damage resistors and transistors due to this excess current.

$$T = \{I = i_0, i_1, i_2, \cdots, i_{n-1}, i_n = I'\}$$
$$\{i'_2, \cdots, i'_{n-1}, i'_n = I'' = I'\}$$

FIGURE 4 – A trajectory T with a perturbation on i_1 and the transformation of it with the same objective $I' = I''$.

2.1 Minsky's Model

This is a model that was created by Marvin Minsky [14, 15]. The principle of this model lies on the fact of having three fundamental parts. First, a current state in which a situation develops, second at state on which we want to be. Finally, the difference between both states. The difference are the necessary stages to reach the desired state (Fig. 5 represents Minsky model). The principle of Minsky model

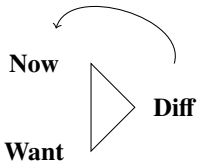


FIGURE 5 – Minsky model.

is introduced in $K = (\Delta, T, I')$. This will allow to have a measure of distance among *intermediates objectives*. That is, for a given state I and a *long-term objective* I', this gives a distance to another nearby objective I_k.

Definition 8. For a given state I, a short-term objective is the closest state I_k where there are fewer disturbances.

The purpose of a distance is to know about the shape of the trajectory T in Δ. To carry out this hypothesis, we include an axis that represents the current states I and another axis for the *long-term objectives*, I'.

6. https ://www.cbsnews.com/news/small-plane-makes-emergency-landing-on-new-jersey-beach-today-2019-06-01/

Definition 9. Vertical axis (want-axis) contains the objectives I'. Horizontal axis (now-axis) is composed of the states I (Fig. 6 represents the axis).

Remark. A point on want-axis is an objective that is accessible through a trajectory T.

Proposition 1. The radius $\overline{p}$ of an extension E is the sum of its ponderations, considering the intersection of now–want axis as the origin.

Démonstration. From a given $K = (\Delta, T, I')$ where Δ is a KOSA theory $\Delta = (D, W = (R \cup I))$ each default d_x in D has criteria defined by $d_x = [C_1, C_2, \ldots, C_m]$ with $C_m \in [0, 1, 2, \ldots, 100]$.
Then, defaults $E = \{d_1, d_2 \ldots\} = \{[d_{1C_1}, d_{1C_2}, \ldots], [d_{2C_1}, d_{2C_2}, \ldots] \ldots\}$ [18]. To obtain the radius $\overline{p}$ of a E we sum the values of each poderation. For $n > 0$, we have the radius for all defaults E_n computed :

$$E_0 = \sum \{d_1, d_2 \ldots d_x\} = \overline{p}$$
$$E_1 = \sum \{d'_1, d'_2 \ldots d'_x\} = \overline{p'}$$
$$\vdots \quad \vdots \quad \vdots \quad \vdots$$
$$E_n = \sum \{d^n_1, d^n_2 \ldots d^n_x\} = \overline{p^n}$$

The representation of the radius $\overline{p^n}$ can be seen in Fig. 6 the representation of the radius of $E_{0,1,2}$ and a fixed objective I'. □

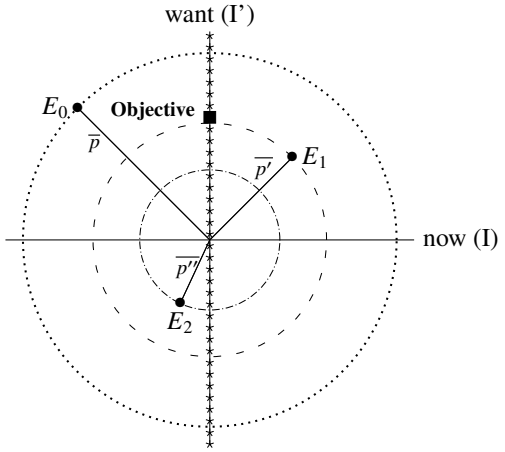


FIGURE 6 – Radius $\overline{p}$ of defaults $E_{0,1,2}$ and a fixed objective I' (black rectangle).

3 Discussion and Conclusion

The importance of using a non-monotonic logic, particulary default logic, is to be able to find consistent solutions.

We presented an approach for representing a resilient system which has the capability of absorb *perturbations* and overcome a disaster. A *KOSA theory* $\Delta = (D, W = (R \cup I))$ is defined with the purpose of study a resilient behavior. It is a default theory which use not temporal logic to describe its evolution. We proved that it exists a resilient trajectory T, for any perturbation (incomplete, partial and contradictory informations). Considering that a perturbed state I_p can be inconsistent. This *trajectory* is a sequence of states, $T = \{I = i_0, i_1, i_2, \cdots, i_{n-1}, i_n = I'\}$ with $W = (R \cup I)$ consistent. A *long-term objective* I' is the last element of a sequence T, and *intermediates objectives* are i_k with $0 \leq k < n$. The introduction of Minsky model to our *KOSA theory* is presented, thanks to this we could have a first step to study the shape of the *trajectories* T. Also the notion of distance among *extensions* is introduced.

To answer the questions that motivated this investigation : *What is the relation among states and objectives ?* We can say that the relation among states and objectives is non-monotonic. *Are there resilient trajectories ?* We demonstrated that all Δ is resilient, if $K = (\Delta, T)$ can reach an objective I' of T from a perturbed state I_p, passing from I_p to I' with $W = (R \cup I)$ consistent. *Is there a resilient behavior ?* We demonstrated that exists a resilient behavior, if all trajectories of Δ are resilient. We presented a discrete theoretical behavior of the trajectories.

The main objective of this research was to conduct a purely logical study. *KOSA theory* does not use learning techniques to infer conclusions which will be interesting for the future, e.g. it could be the use of this type of method to learn the necessary rules to achieve an objective. This study provides the basis for generalizing ***Definition 6***. In which one could consider finding the universe of resilient *trajectories* for any *perturbation*. That is, no matter what the *perturbation* is, we could find the universe of *trajectories* to achieve the desired objective.

A practical application [13] without resilience property was performed on an embedded computer which calculates the extensions for the stabilization of a motorized glider.

Acknowledgments

I would like to extend my thanks to the people who contributed their criticisms and comments in the development of this article, either directly or indirectly.

Références

[1] Bellman, Richard: *Stability theory of differential equations*. Courier Corporation, 2008.

[2] Benenson, Walter, John W Harris, Horst Stöcker et Holger Lutz: *Handbook of physics*. Springer Science & Business Media, 2006.

[3] Chepyzhov, Vladimir V et Mark I Vishik: *Attractors for equations of mathematical physics*, tome 49. American Mathematical Soc., 2002.

[4] Folke, Carl, Stephen R Carpenter, Brian Walker, Marten Scheffer, Terry Chapin et Johan Rockström: *Resilience thinking : integrating resilience, adaptability and transformability*. Ecology and society, 15(4), 2010.

[5] Granas, Andrzej et James Dugundji: *Fixed point theory*. Springer Science & Business Media, 2013.

[6] Holling, Crawford S: *Resilience and stability of ecological systems*. Annual review of ecology and systematics, 4(1) :1–23, 1973.

[7] Holling, Crawford S: *Understanding the complexity of economic, ecological, and social systems*. Ecosystems, 4(5) :390–405, 2001.

[8] Holling, Crawford Stanley: *Engineering resilience versus ecological resilience*. Engineering within ecological constraints, 31(1996) :32, 1996.

[9] Janis, Irving L et Leon Mann: *Decision making : A psychological analysis of conflict, choice, and commitment.* Free press, 1977.

[10] Kahneman, Daniel et Amos Tversky: *Prospect theory : An analysis of decision under risk*. Dans *Handbook of the fundamentals of financial decision making : Part I*, pages 99–127. World Scientific, 2013.

[11] Li, Guohua, Susan P Baker, Jurek G Grabowski et George W Rebok: *Factors associated with pilot error in aviation crashes*. Aviation, space, and environmental medicine, 72(1) :52–58, 2001.

[12] Lyapunov, Aleksandr Mikhailovich: *The general problem of the stability of motion*. International journal of control, 55(3) :531–534, 1992.

[13] Medina, José Luis Vilchis, Pierre Siegel, Vincent Risch et Andrei Doncescu: *Intelligent and Adaptive System based on a Non-monotonic Logic for an Autonomous Motor-glider*. Dans *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 442–447. IEEE, 2018.

[14] Minsky, Marvin: *A framework for representing knowledge*. 1974.

[15] Minsky, Marvin: *The emotion machine*. New York : Pantheon, 56, 2006.

[16] Oster, Clinton V, John S Strong et C Kurt Zorn: *Why airplanes crash : Aviation safety in a changing world*. Oxford University Press on Demand, 1992.

[17] Oster, Clinton V, John S Strong et Kurt Zorn: *Why Airplanes Crash : Causes of Accidents Worldwide*. rapport technique, 2010.

[18] Reiter, Raymond: *A logic for default reasoning*. Artificial intelligence, 13(1-2) :81–132, 1980.

[19] Russell, Stuart J et Peter Norvig: *Artificial intelligence : a modern approach*. Malaysia ; Pearson Education Limited,, 2016.

[20] Toulgoat, Isabelle, Pierre Siegel et Andrei Doncescu: *Modelling of submarine navigation by nonmonotonic logic*. Dans *2011 International Conference on Broadband and Wireless Computing, Communication and Applications*, pages 447–454. IEEE, 2011.

[21] Walker, Brian, Crawford S Holling, Stephen R Carpenter et Ann Kinzig: *Resilience, adaptability and transformability in social–ecological systems*. Ecology and society, 9(2), 2004.

[22] Wu, Gang, Adriana Feder, Hagit Cohen, Joanna J Kim, Solara Calderon, Dennis S Charney et Aleksander A Mathé: *Understanding resilience*. Frontiers in behavioral neuroscience, 7 :10, 2013.

Actes JIAF 2020

On the Convergence of Swap Dynamics to Pareto-Optimal Matchings

Felix Brandt **Anaëlle Wilczynski**

Technische Universität München

brandtf@in.tum.de wilczyns@in.tum.de

Résumé

On étudie dans cet article la possibilité d'atteindre des couplages optimaux au sens de Pareto par le biais d'échanges entre paires d'agents dans des problèmes d'appariement en présence d'un couplage initial. On considère trois problèmes d'appariement particuliers : le problème d'allocation de maisons dans lequel à chaque agent doit être affecté exactement un objet, ainsi que le problème des mariages stables et le problème des colocataires, dans lesquels les agents sont couplés avec d'autres agents. Dans ce cadre, on examine trois différentes notions d'échanges améliorants entre paires d'agents. Tandis qu'il peut être vérifié en temps polynomial si un couplage Pareto-optimal peut être atteint lorsque les échanges sont améliorants relativement à des paires bloquantes d'agents, vérifier si toutes les séquences d'échanges mènent à un couplage Pareto-optimal est difficile. Alternativement, on prouve que ces deux problèmes sont difficiles lorsque l'on considère des échanges dans lesquels ce sont les agents qui échangent qui doivent améliorer leur situation. Ce résultat confirme et étend une conjecture faite par Damamme et al. [16] qui ont, de plus, montré que dans le problème particulier d'allocation de maisons, toute séquence d'échanges améliorants converge vers une allocation Pareto-optimale sous des préférences unimodales. On montre dans cet article que, dans les problèmes de mariages stables et de colocataires, l'unimodalité des préférences n'est pas suffisante pour obtenir un tel résultat de convergence. En revanche, la restriction encore plus forte des préférences 1-Euclidiennes le permet.

Abstract

We study whether Pareto-optimal stable matchings can be reached via pairwise swaps in one-to-one matching markets with initial assignments. We consider housing markets, marriage markets, and roommate markets as well as three different notions of swap rationality. Our main results are as follows. While it can be efficiently determined whether a Pareto-optimal stable matching can be reached when defining swaps via blocking pairs, checking whether this is the case for *all* such sequences is computationally intractable. When defining swaps such that all involved agents need to be better off, even deciding whether a Pareto-optimal stable matching can be reached via *some* sequence is intractable. This confirms and extends a conjecture made by Damamme et al. [16] who have furthermore shown that convergence to a Pareto-optimal matching is guaranteed in housing markets with single-peaked preferences. We show that in marriage and roommate markets, single-peakedness is not sufficient for this to hold, but the stronger restriction of one-dimensional Euclidean preferences is.

1 Introduction

One-to-one matchings, where individuals are matched with resources or other individuals, are omnipresent in everyday life. Examples include the job market, assigning offices to workers, pairing students in working groups, and online dating. The formal study of matching procedures leads to challenging algorithmic problems while being of immediate practical interest [25, 28]. One typically distinguishes between three different types of abstract one-to-one matching settings. In *housing markets* [32], each agent is matched with an object (usually referred to as a house). In *marriage markets* [21], agents are partitioned into two groups—say, males and females—and each member of one group is matched with an agent from the other group. Finally, in *roommate markets* [21], all agents belong to the same group and each agent is matched with another agent. By supposing that agents are rational and want to maximize their satisfaction, individual agreements may naturally occur among them and especially, for realistic reasons, between small groups of agents. An important question is then whether sequences of such individual agreements can lead to socially optimal outcomes. In many applications, it is reasonable to assume that there is an initial assignment

because agents already live in a house, are engaged in a relationship, and are employed by a company [1, 29]. Under these assumptions, we focus on atomic agreements which require the least coordination: *pairwise swaps*.

We consider three different types of individual rationality for pairwise swaps. In housing markets, there is only one meaningful notion of swap rationality: two agents will only exchange objects if both of them are better off. By contrast, when matching agents with each other, one could require that all four agents involved in a swap or only two of them are better off. The latter requirement allows for two kinds of swap rationality: two agents who exchange their match are better off (e.g., a company and its subsidiary exchange employees without asking their consent) or two agents who decide to form a new pair are better off (e.g., two lovers leave their current partners to be together).

Social optimality in settings with ordinal preferences like that of matching markets is measured in terms of Pareto-optimality. We therefore study whether there exists a sequence of pairwise swaps that results in a Pareto-optimal matching that does not allow for further swaps (and hence is called stable). Whenever all sequences of pairwise swaps are of this kind, we say that the given type of swap dynamics converges.

It turns out that in all three types of matching markets and all three notions of swap rationality, it may not be possible to reach a Pareto-optimal stable matching from the initial assignment. We prove that deciding whether this is the case is NP-hard for two types of swap rationality while it can be solved in polynomial time for swaps based on blocking pairs. However, for all types of rationality, checking convergence is co-NP-hard. On the other hand, we show that when preferences are one-dimensional Euclidean—a natural but demanding restriction—swap dynamics for two types of swap rationality will always converge.

2 Related Work

Damamme et al. [16] investigated the dynamics of individually rational pairwise swaps in housing markets, where two agents are better off by exchanging their objects. Recently, variants of this problem that further restrict the agents' interactions using underlying graph structures have been examined [22, 23, 31].

In marriage and roommate markets, most of the literature focuses on deviations based on blocking pairs, where two agents decide to leave their old partners in order to be matched with each other. Blocking pairs are best known for their role in the definition of stability [21], but some papers also studied the dynamics of blocking pair swaps [2, 30]. The notion of exchange stability, where two agents agree to exchange their partners, has been investigated in both roommate markets [8, 13] and marriage markets [14]. We consider both types of swaps, i.e., blocking pair swaps and exchange rational swaps, but focus on the study of dynamics that reach Pareto-optimal matchings.

Perhaps closest to our work is a result by Damamme et al. [16] who proved that swap dynamics always converge to a Pareto-optimal matching in housing markets under single-peaked preferences. However, they left open the computational problem of deciding whether a Pareto-optimal stable matching can be reached for unrestricted preferences and conjectured this problem to be intractable. We solve this problem and extend it to marriage and roommate markets. Moreover, we prove that their convergence result for housing markets under single-peaked preferences does not extend to marriage and roommate markets, but can be restored when restricting preferences even further.

3 The Model

We are given a set N of agents $\{1, \dots, n\}$ and a set O of objects $\{o_1, \dots, o_n\}$ such that $|N| = |O| = n$. Each agent $i \in N$ has strict ordinal preferences, given by a linear order $\succ_i$, over a set A_i of alternatives to be matched with. In the matching markets we consider, A_i is either a subset of the set of agents N or the set of objects O. A tuple of preference relations $\succ = (\succ_1, \dots, \succ_n)$ is called a *preference profile*.

3.1 Matching Markets

In this article, we are considering three different settings where the goal is to match the agents either with objects—like in housing markets—or with other agents—like in marriage or roommate markets. In all cases, we assume that there is an initial matching. More formally,

- a *housing market* consists of a preference profile where $A_i = O$ for all $i \in N$, and an initial endowment given as a bijection $\mu : N \to O$,
- a *marriage market* consists of a preference profile where $N = W \cup M$ with $W \cap M = \emptyset$, $A_i = M$ for all $i \in W$ and $A_i = W$ for all $i \in M$, and an initial matching given as a bijection $\mu : W \to M$, and
- a *roommate market* consists of a preference profile with even n and $A_i = N \setminus \{i\}$ for all $i \in N$, and an initial matching given as an involution $\mu : N \to N$ such that $\mu(i) \neq i$ for all $i \in N$.

When allowing for indifferences as well as unacceptabilities in the preferences, the three settings form a hierarchy: housing markets are marriage markets where the "objects" are indifferent between all agents, and marriage markets are roommate markets where all agents of the same type are considered unacceptable. In this paper, however, we do not make either assumption and therefore these inclusion relationships do not hold.

The key question studied in this paper is whether Pareto-optimal matchings can be reached from the initial matching via local modifications. A matching is *Pareto-optimal* if

there is no other matching μ' such that for every agent i, $\mu'(i) \succeq_i \mu(i)$ and for at least one agent j, $\mu'(j) \succ_j \mu(j)$.

3.2 Preference Restrictions

We consider three restricted preference domains: single-peaked preferences [12], globally-ranked preferences [4, 6] and their common subdomain of one-dimensional Euclidean preferences [15]. A preference profile $\succ$ is *single-peaked* if there exists a linear order $>$ over the alternatives in $A := \bigcup_{i \in N} A_i$ such that for each agent i and each triple of alternatives $x, y, z \in A_i$ with $x > y > z$ or $z > y > x$, $x \succ_i y$ implies $y \succ_i z$. A preference profile $\succ$ is *globally-ranked* (we also speak about *correlated markets* [6]) if there exists a global order $>$ over all possible pairs in the matching market such that for every agent i and any two alternatives $x, y \in A_i$, $x \succ_i y$ iff $\{i, x\} > \{i, y\}$. Globally-ranked preferences impose no restriction in a housing market (the agents are matched with objects which do not express preference), but may capture in other markets the idea that each pair of agents generates an absolute profit and thus each agent prefers the agents with who she can get a better profit. A preference profile $\succ$ is *one-dimensional Euclidean (1-Euclidean)* if there exists an embedding $E : N \cup O \to \mathbb{R}$ on the real line such that for every agent i and any two alternatives $x, y \in A_i$, $x \succ_i y$ iff $|E(i) - E(x)| < |E(i) - E(y)|$.

One-dimensional Euclidean preferences form a subdomain of single-peaked preferences because every 1-Euclidean preference profile is singled-peaked for the linear order $>$ given by $x > y$ iff $E(x) > E(y)$. However, a single-peaked preference profile may not be 1-Euclidean, therefore the inclusion is strict. Moreover, one-dimensional Euclidean preferences form a subdomain of globally-ranked preferences: from a 1-Euclidean preference profile, a global ranking over all possible pairs can be extracted by sorting all pairs according to the Euclidean distance on the embedding E between the two partners. Reversely, a globally-ranked preference profile may not be 1-Euclidean, therefore the inclusion is strict. We know that 1-Euclidean preferences are both globally-ranked and single-peaked. However, the reverse is not true: a globally-ranked and single-peaked preference profile may not be 1-Euclidean, even in markets matching agents with each other. We omit the examples due to space restrictions.

While assuming that all agents have 1-Euclidean preferences certainly represents a strong restriction, there are nevertheless some applications where this assumption is not unreasonable. For example, in job markets, preferences could be 1-Euclidean because employees prefer one workplace to another if it is closer to their home, or when forming pairs of students for the realization of a project, a student could prefer to be matched with a student who is the most productive as the same hours as her.

Note that all considered preference restrictions are recognizable in polynomial time: there exist polynomial time algorithms for checking single-peakedness [10, 20] or the satisfaction of the 1-Euclidean property [18, 19, 26]. Moreover, checking whether a preference profile is globally-ranked boils down to checking the acyclicity of the directed graph defined on all possible pairs and where there is an arc from a pair $\{i, j\}$ to a pair $\{i, k\}$ if and only if $k \succ_i j$ [4]; this can be done in polynomial time.

3.3 Rational Swaps

We study sequences of matchings in which two pairs of the current matching are permuted. More formally, we assume that a swap w.r.t. two agents (i, j) transforms a matching μ into a matching μ' where agents i and j have exchanged their matches, i.e., $\mu'(i) = \mu(j)$ and $\mu'(j) = \mu(i)$, while the rest of the matching remains unchanged, i.e., $\mu'(k) = \mu(k)$ for every $k \notin \{i, j, \mu(i), \mu(j)\}$.

We furthermore require these swaps to be *rational* in the sense that they result from an agreement among agents, and thus make the agents involved in the agreement better off.

The most natural notion of rationality is exchange-rationality, which requires that the two agents who exchange their matches are better off [8]. A swap w.r.t. agents (i, j) from matching μ is *exchange rational (ER)* if the agents who exchange their matches are better off, i.e.,

$$\mu(j) \succ_i \mu(i) \text{ and } \mu(i) \succ_j \mu(j). \qquad (ER\text{-swap})$$

Exchange-rationality is the only meaningful notion of swap rationality in housing markets because only one side of the market has preferences. However, several notions of rationality emerge in marriage and roommate markets, where agents are matched with each other. One could demand that only two of the agents who agree to form a new pair need to be better off. This notion of rational swaps is based on the classic idea of *blocking pairs*, which forms the basis of the standard notion of stability [21]. A swap w.r.t. agents (i, j) from matching μ between agents is *blocking pair (BP) rational* if one of the new pairs in μ' forms a blocking pair, where both agents are better off, i.e.,

$$\Big[\mu(j) \succ_i \mu(i) \text{ and } i \succ_{\mu(j)} j\Big] \quad \text{or} \quad \Big[\mu(i) \succ_j \mu(j) \text{ and } j \succ_{\mu(i)} i\Big]. \qquad (BP\text{-swap})$$

We refer to a *BP*-swap by mentioning the associated blocking pair $((i, \mu(j))$ or $(j, \mu(i)))$. The old partners of the blocking pair are also assumed to be matched together. [1]

Finally, in marriage and roommate markets, a stronger notion of rationality is that of a *fully rational swap*, which makes all four involved agents better off. A swap w.r.t.

1. Once the old partners are alone, they have an incentive to form a new pair. Roth and Vande Vate [30] therefore decompose *BP*-swaps into two steps. We do not explicitly consider these steps in order to always maintain a perfect matching like, e.g., Knuth [27].

agents (i, j) from matching μ is *fully rational (FR)* if all four agents involved in the swap are better off, i.e.,

$$\mu(j) \succ_i \mu(i), \quad \mu(i) \succ_j \mu(j), \quad j \succ_{\mu(i)} i, \text{ and } \quad i \succ_{\mu(j)} j. \tag{FR-swap}$$

Note that for marriage and roommate markets, an *FR*-swap w.r.t. pair of agents (i, j) from a matching μ is an *ER*-swap w.r.t. pair (i, j) or $(\mu(i), \mu(j))$ and also a *BP*-swap w.r.t blocking pair $(i, \mu(j))$ or $(j, \mu(i))$. We thus obtain the following implications:

$$BP\text{-swap} \quad \Leftarrow \quad FR\text{-swap} \quad \Rightarrow \quad ER\text{-swap}$$

The different types of swap rationality are illustrated in the following example.

Example 1. *Consider a roommate market with six agents. The preferences of the agents are given below, where the initial assignment is marked with frames.*

$$\begin{array}{llllllllll} 1: & 4 & \succ & \boxed{3} & \succ & 6 & \succ & 5 & \succ & 2 \\ 2: & 3 & \succ & 1 & \succ & \boxed{4} & \succ & 6 & \succ & 5 \\ 3: & 6 & \succ & 2 & \succ & \boxed{1} & \succ & 5 & \succ & 4 \\ 4: & 5 & \succ & 1 & \succ & 3 & \succ & \boxed{2} & \succ & 6 \\ 5: & 2 & \succ & \boxed{6} & \succ & 4 & \succ & 1 & \succ & 3 \\ 6: & 4 & \succ & 3 & \succ & 1 & \succ & 2 & \succ & \boxed{5} \end{array}$$

The swap w.r.t. pair $(1, 2)$*, which matches Agent* 1 *with Agent* 4 *and Agent* 2 *with Agent* 3*, is an FR-swap because every involved agent is better off. Hence, this is also an ER-swap for pair* $(1, 2)$ *or* $(3, 4)$ *because they both prefer to exchange their partner. It is also a BP-swap for blocking pair* $(2, 3)$ *or* $(1, 4)$ *because they both prefer to be together than with their current partner.*

The swap w.r.t. pair $(1, 6)$ *is a BP-swap for blocking pair* $(3, 6)$ *because Agent* 3*, the old partner of Agent* 1*, prefers to be with Agent* 6*, as well as Agent* 6 *who prefers* 3 *to her old partner* 5*. This is not an ER-swap (and hence not an FR-swap) because neither the agents in pair* $(1, 6)$ *nor in pair* $(3, 5)$ *want to exchange their partners.*

The swap w.r.t. pair $(4, 6)$ *is an ER-swap for* $(4, 6)$ *because Agent* 4 *prefers the current partner of* 6*, i.e., Agent* 5*, to her current partner and* 6 *prefers the current partner of* 4*, i.e., Agent* 2*, to her current partner. This is not a BP-swap (and hence not an FR-swap) because it matches Agent* 4 *with Agent* 5*, who prefers to stay with Agent* 6*, and Agent* 6 *with Agent* 2*, who prefers to stay with Agent* 4*.*

Stability can now be defined according to the different notions of rational swaps. A matching μ is *σ-stable*, for $\sigma \in \{FR, ER, BP\}$, if no σ-swap can be performed from matching μ. A sequence of σ-swaps, for $\sigma \in \{FR, ER, BP\}$, corresponds to a sequence of matchings $(\mu^0, \mu^1, \ldots, \mu^r)$ such that a σ-swap transforms each matching μ^t into matching μ^{t+1} for every $0 \leq t < r$. Then, matching μ is *σ-reachable* from initial matching μ^0 if there exists a sequence of σ-swaps $(\mu^0, \mu^1, \ldots, \mu^r)$ such that $\mu^r = \mu$. When the context is clear, we omit σ and the initial matching μ^0.

A *σ-dynamics* is defined according to initial matching μ^0 and a type σ of rational swaps. The σ-dynamics is *finite* if all associated sequences of σ-swaps terminate in a σ-stable matching, and it is said to *converge* if it is finite for every initial matching μ^0.

We consider the following decision problems related to the convergence of dynamics to a Pareto-optimal matching.

$\exists$-σ-ParetoSequence / $\forall$-σ-ParetoSequence	
Input:	Matching market, type σ of rational swaps
Question:	Does there exist a sequence of σ-swaps terminating in a Pareto-optimal σ-stable matching? / Do all sequences of σ-swaps terminate in a Pareto-optimal σ-stable matching?

In order to tackle these two questions, we also study the stability and convergence properties of the considered dynamics in the three types of matching markets.

4 Exchange Rational Swaps

In housing markets, every *ER*-swap represents a Pareto improvement. Since the number of agents and objects is finite, *ER*-dynamics always converges and the existence of *ER*-stable matchings is guaranteed (simply because every Pareto-optimal matching happens to be *ER*-stable). However, it may be impossible to reach a Pareto-optimal matching from a given matching by only applying *ER*-swaps.

Proposition 1. *ER-dynamics may not converge to a Pareto-optimal matching in housing markets.*

Proof. Consider a housing market with n agents. The preferences of the agents are given below, where the initial assignment is marked with frames and $[\ldots]$ denotes an arbitrary order over the rest of objects.

$$\begin{array}{lllll} 1: & (o_1) & \succ & \boxed{o_2} & \succ \ [\ldots] \\ 2: & (o_2) & \succ & \boxed{o_3} & \succ \ [\ldots] \\ 3: & (o_3) & \succ & \boxed{o_4} & \succ \ [\ldots] \\ & & \cdots & & \\ n: & (o_n) & \succ & \boxed{o_1} & \succ \ [\ldots] \end{array}$$

Observe that no *ER*-swap is possible in this instance, therefore the initial matching (framed objects) is the unique *ER*-reachable matching. However, there exists a unique Pareto-optimal matching (circled objects), and this matching is different from the initial one. Note that, in such an instance, even if exchanges involving up to $n-1$ agents are allowed, the Pareto-optimal matching will not be reached: the only *ER*-exchange would involve all the n agents. □

Nevertheless, it is known that *ER*-dynamics always converges to a Pareto-optimal matching in housing markets when the agents' preferences are single-peaked [16].

In marriage and roommate markets, an *ER*-stable matching may not exist, even for single-peaked preferences [8, 13]. However, it turns out that, for globally-ranked preferences, an *ER*-stable matching always exists, and, moreover, the convergence to such a matching is guaranteed.

Proposition 2. *ER-dynamics always converges in marriage / roommate markets for globally-ranked preferences.*

Proof. Denote by $>$ the global order over all possible pairs such that the preferences of the agents are globally-ranked with respect to this global order. Define as $f : \mu \to \mathbb{R}$ the potential function which assigns to each matching the sum of ranks in order $>$ of all the assigned pairs in the matching, i.e., $f(\mu) = \sum_{\{i,j\} s.t. \mu(i)=j} rank_>(\{i, j\}))$ with $rank_>$ the function which gives the rank of the pairs in order $>$. Now consider a sequence of *ER*-swaps given by the sequence of matchings $(\mu^0, \mu^1, \ldots, \mu^r)$. Between each matchings μ^t and μ^{t+1}, with $0 \leq t < r$, an *ER*-swap is performed, say w.r.t. pair (i, j) of agents. That means, by definition of an *ER*-swap, that agents i and j prefer to exchange their partners in μ^t, and thus, $\mu^t(j) \succ_i \mu^t(i)$ and $\mu^t(i) \succ_j \mu^t(j)$. This implies, by correlation of the preferences, that $\{i, \mu^t(j)\} > \{i, \mu^t(i))\}$ and $\{j, \mu^t(i)\} > \{j, \mu^t(j)\}$. But agents i and $\mu^t(j)$ are matched in μ^{t+1}, as well as agents j and $\mu^t(i)$. Since the rest of the pairs remains unchanged between μ^t and μ^{t+1}, we get that $f(\mu^{t+1}) < f(\mu^t)$. Because the number of different matchings is finite, we can conclude that *ER*-dynamics always converges. □

In general, an *ER*-stable matching may not be Pareto-optimal, thus convergence to a Pareto-optimal matching is not guaranteed even when an *ER*-stable matching exists (note that deciding the existence of an *ER*-stable matching is NP-hard in marriage and roommate markets [13, 14]).

Proposition 3. *ER-dynamics may not converge to a Pareto-optimal matching, in marriage and roommate markets, even when an ER-stable matching exists and for globally-ranked preferences.*

Proof. Consider a marriage market with three women and three men. The preferences are given below and the initial assignment is marked with frames.

w_1 :	(m_1)	$\succ$	$\boxed{m_2}$	$\succ$	m_3	m_1 :	(w_1)	$\succ$	$\boxed{w_3}$	$\succ$	w_2
w_2 :	(m_2)	$\succ$	$\boxed{m_3}$	$\succ$	m_1	m_2 :	(w_2)	$\succ$	$\boxed{w_1}$	$\succ$	w_3
w_3 :	(m_3)	$\succ$	$\boxed{m_1}$	$\succ$	m_2	m_3 :	(w_3)	$\succ$	$\boxed{w_2}$	$\succ$	w_1

No *ER*-swap is possible from initial matching μ^0 (framed agents), therefore μ^0 is the unique *ER*-reachable matching. However, there is another matching (circled agents) which is the unique Pareto-optimal matching. Note that this preference profile is globally-ranked with respect to, e.g., the global order $\{w_1, m_1\} > \{w_2, m_2\} > \{w_3, m_3\} > \{w_1, m_2\} > \{w_2, m_3\} > \{w_3, m_1\} > \{w_1, m_3\} > \{w_2, m_1\} > \{w_3, m_2\}$.

Now, consider a roommate market with six agents. Preferences of the agents are given below, where the initial partner of each agent is marked with frames and [...] denotes an arbitrary order over the rest of the agents.

1 :	(3)	$\succ$	$\boxed{2}$	$\succ$	[...]	4 :	(6)	$\succ$	$\boxed{3}$	$\succ$	[...]
2 :	(5)	$\succ$	$\boxed{1}$	$\succ$	[...]	5 :	(2)	$\succ$	$\boxed{6}$	$\succ$	[...]
3 :	(1)	$\succ$	$\boxed{4}$	$\succ$	[...]	6 :	(4)	$\succ$	$\boxed{5}$	$\succ$	[...]

No *ER*-swap is possible from initial matching μ^0 (framed agents), thus μ^0 is the unique *ER*-reachable matching. However, there is another matching (circled agents) which is the unique Pareto-optimal matching. This preference profile is globally-ranked w.r.t., e.g., the global order $\{4, 6\} > \{1, 3\} > \{3, 4\} > \{2, 5\} > \{1, 2\} > \{5, 6\} > [\ldots]$. □

Note that the above preference profiles are not 1-Euclidean. In fact, they are not even single-peaked. Again, more positive results can be obtained by restricting the domain of admissible preferences.

Proposition 4. *Every ER-stable matching is Pareto-optimal when preferences are single-peaked in marriage and roommate markets.*

Proof. Let μ be an *ER*-stable matching. For any two agents i and j (in N for roommate markets, or both in either W or M for marriage markets) it holds that $\mu(i) \succ_i \mu(j)$ or $\mu(j) \succ_j \mu(i)$. Suppose there is another matching μ' such that $\mu'(i) \succeq_i \mu(i)$ for every $i \in N$ and there exists $j \in N$ such that $\mu'(j) \succ_j \mu(j)$. Then, there exists a Pareto improving cycle from μ to μ' along agents $(n_1, \ldots, n_k)$ such that each agent n_i, $1 \leq i \leq k$, is matched in μ' with agent $\mu(n_{(i \bmod k)+1})$. For marriage markets, the agents in $(n_1, \ldots, n_k)$ are restricted by definition to only one side of the market, but it impacts both sides since the agents exchange agents of the other side. But there is no problem of preferences of the matched agents because no agent is worse off in μ' compared to μ. The same holds for roommate markets. Since μ is *ER*-stable, it holds that $k > 2$. However, for single-peaked preferences, one can prove, by following the same proof by induction as Damamme et al. [16], that a Pareto improving cycle of any length cannot occur, contradicting the fact that μ is Pareto dominated. □

Propositions 2 and 4 allow us to conclude that sequences of *ER*-swaps will always terminate in Pareto-optimal matchings when preferences are both single-peaked and globally-ranked, like in 1-Euclidean preferences.

Corollary 1. *ER-dynamics always converges to a Pareto-optimal matching in marriage and roommate markets for 1-Euclidean preferences.*

For more general preferences, an interesting computational question is whether, given a preference profile and an initial assignment, a Pareto-optimal matching can be reached via *ER*-swaps. In the context of housing markets, the complexity of this question was mentioned as an open problem by Damamme et al. [16]. It turns out that this problem is computationally intractable for all kinds of matching markets, even for globally-ranked preferences.

Theorem 1. ∃-ER-ParetoSequence *is* NP*-hard in housing, marriage, and roommate markets even for globally-ranked preferences.*

Proof. For the case of housing markets, we perform a reduction from 2P1N-SAT, a variant of SAT known to be NP-complete [33], where the goal is to decide the satisfiability of a CNF propositional formula where each variable appears exactly twice as a positive literal and once as a negative literal. The idea of the proof is close to the one given by Gourvès et al. [22] for proving NP-hardness of determining whether a given object is reachable by a given agent. From an instance of 2P1N-SAT with formula φ on m clauses $C_1, \ldots, C_m$ and p variables $x_1, \ldots, x_p$, we build a housing market $(N, O, \succ, \mu^0)$ as follows.

For each clause C_j, with $1 \leq j \leq m$, we construct two clause-agents in N denoted by A_j and A'_j and two clause-objects in O denoted by a_j and a'_j such that $\mu^0(A_j) = a_j$ and $\mu^0(A'_j) = a'_j$. For each variable x_i, with $1 \leq i \leq p$, we construct six literal-agents in N corresponding to two copies of each literal, namely agents Y_i^ℓ and Z_i^ℓ who correspond to the ℓ^{th} ($\ell \in \{1, 2\}$) positive occurrence of variable x_i in formula φ, denoted by x_i^ℓ, and $\overline{Y_i}$ and $\overline{Z_i}$ who correspond to the negative occurrence of variable x_i in formula φ, denoted by $\overline{x_i}$; we also create their associated literal-objects y_i^ℓ, z_i^ℓ, $\overline{y_i}$ and $\overline{z_i}$ such that $\mu^0(Y_i^\ell) = y_i^\ell$, $\mu^0(Z_i^\ell) = z_i^\ell$, $\mu^0(\overline{Y_i}) = \overline{y_i}$ and $\mu^0(\overline{Z_i}) = \overline{z_i}$. The literal-agents are divided in two sets, denoted by Y and Z, which correspond to the original agents and their copy, respectively, i.e., $Y := \bigcup_{1 \leq i \leq p} \{Y_i^1, Y_i^2, \overline{Y}_i\}$ and $Z := \bigcup_{1 \leq i \leq p} \{Z_i^1, Z_i^2, \overline{Z}_i\}$. Three additional agents B, T and T' are created in N, with their initial assigned objects denoted by b, t and t', respectively.

The preferences are given below for each $1 \leq i \leq p$ and $1 \leq j < m$ ([...] is an arbitrary order over the rest of the objects, $\{\mathcal{Y}_j\}$ is an arbitrary order over the literal-objects in $\bigcup_{1 \leq i \leq p} \{y_i^1, y_i^2, \overline{y_i}\}$ associated with literals of clause C_j and $cl(\ell_i)$ is the index of the clause in which literal ℓ_i appears).

T : $t' \succ \{\mathcal{Y}_1\} \succ \boxed{t} \succ [\ldots]$
A_j : $a'_j \succ \{\mathcal{Y}_{j+1}\} \succ t \succ \{\mathcal{Y}_j\} \succ \boxed{a_j} \succ [\ldots]$
A_m : $b \succ t \succ \{\mathcal{Y}_m\} \succ \boxed{a_m} \succ [\ldots]$
Y_i^1 : $z_i^1 \succ a_{cl(\overline{x_i})} \succ a_{cl(x_i^1)} \succ \overline{y_i} \succ \boxed{y_i^1} \succ [\ldots]$
Y_i^2 : $z_i^2 \succ y_i^1 \succ a_{cl(x_i^2)} \succ \overline{y_i} \succ \boxed{y_i^2} \succ [\ldots]$
$\overline{Y_i}$: $\overline{z_i} \succ y_i^2 \succ \boxed{\overline{y_i}} \succ [\ldots]$
B : $t \succ \boxed{b} \succ [\ldots]$

T' : $a'_m \succ \{\mathcal{Y}_1\} \succ \boxed{t'} \succ [\ldots]$
A'_j : $a_j \succ \{\mathcal{Y}_j\} \succ a'_m \succ \{\mathcal{Y}_{j+1}\} \succ \boxed{a'_j} \succ [\ldots]$
A'_m : $a_m \succ \{\mathcal{Y}_m\} \succ \boxed{a'_m} \succ [\ldots]$
Z_i^1 : $y_i^1 \succ \overline{y_i} \succ a_{cl(x_i^1)} \succ a_{cl(\overline{x_i})} \succ \boxed{z_i^1} \succ [\ldots]$
Z_i^2 : $y_i^2 \succ \overline{y_i} \succ y_i^1 \succ a_{cl(x_i^2)} \succ \boxed{z_i^2} \succ [\ldots]$
$\overline{Z_i}$: $\overline{y_i} \succ y_i^2 \succ \boxed{\overline{z_i}} \succ [\ldots]$

We claim that formula φ is satisfiable if and only if the matching assigning to each agent her best object is reachable (this is the only Pareto-optimal matching). The global idea is that the only way to reach this Pareto-optimal matching is to make object t reach agent A_m by first giving to each clause-agent A_j, via *ER*-swaps, a literal-object in $\{\mathcal{Y}_j\}$, objects associated with the literals of clause C_j. Once object t reaches clause-agent A_m, each agent except A'_m exchanges with her prime version agent (agents Z_i^1, Z_i^2 and $\overline{Z_i}$ are the prime versions of agents Y_i^1, Y_i^2 and $\overline{Y_i}$, respectively, and B is the prime version of A_m), and then the prime agents make among them the reverse sequence of swaps of the initial one where the goal was to make object t reach A_m, leading to the Pareto-optimal matching. By construction of the preferences among the literal-agents, once a literal-object associated with a positive (resp., negative) literal of a variable has been chosen to go with a clause-agent A_j, no literal-object associated with a negative (resp., positive) literal of this variable can reach a clause-agent. The details of the equivalence are omitted due to space restrictions.

To adapt the proof to the case of marriage and roommate markets, we now consider the objects in O as agents. More precisely, we build a marriage market $(N, \succ, \mu^0)$ where $N = M \cup W$ with $W = \{T, T', B, \{A_j, A'_j\}_{1 \leq j \leq m}, \{Y_i^1, Y_i^2, \overline{Y_i}, Z_i^1, Z_i^2, \overline{Z_i}\}_{1 \leq i \leq p}\}$ and $M = \{t, t', b, \{a_j, a'_j\}_{1 \leq j \leq m}, \{y_i^1, y_i^2, \overline{y_i}, z_i^1, z_i^2, \overline{z_i}\}_{1 \leq i \leq p}\}$. The preferences of women over men are the same as the preferences of agents over objects previously described, and the preferences of men over women are described below for $1 \leq j < m$ and $1 \leq i \leq p$. Notation $\{\mathcal{Y}_j\}$ (resp., $\{\mathcal{Z}_j\}$) denotes an arbitrary order over the literal-agents in Y (resp., Z) that are associated with a literal of clause C_j where each "negative" literal-agent $\overline{Y}_i$ (resp., $\overline{Z}_i$) is replaced by agent Y_i^1 (resp., Z_i^1), A_0 stands for T, A'_0 for T', and [...] is an arbitrary order over the rest of the women.

t : $B \succ A_m \succ \cdots \succ A_1 \succ \boxed{T} \succ [\ldots]$
a_j : $A'_j \succ \{\mathcal{Z}_j\} \succ \{\mathcal{Y}_j\} \succ \boxed{A_j} \succ [\ldots]$
a_m : $A'_m \succ \{\mathcal{Z}_m\} \succ \{\mathcal{Y}_m\} \succ \boxed{A_m} \succ [\ldots]$
y_i^1 : $Z_i^1 \succ Z_i^2 \succ Y_i^2 \succ A'_{cl(x_i^1)} \succ A'_{cl(x_i^1)-1} \succ A_{cl(x_i^1)-1} \succ A_{cl(x_i^1)} \succ \boxed{Y_i^1} \succ [\ldots]$
y_i^2 : $Z_i^2 \succ \overline{Z_i} \succ \overline{Y_i} \succ A'_{cl(x_i^2)} \succ A'_{cl(x_i^2)-1} \succ A_{cl(x_i^2)-1} \succ A_{cl(x_i^2)} \succ \boxed{Y_i^2} \succ [\ldots]$
$\overline{y_i}$: $\overline{Z_i} \succ Z_i^2 \succ Z_i^1 \succ A'_{cl(\overline{x_i})} \succ A'_{cl(\overline{x_i})-1} \succ A_{cl(\overline{x_i})-1} \succ A_{cl(\overline{x_i})} \succ Y_i^1 \succ Y_i^2 \succ \boxed{\overline{Y_i}} \succ [\ldots]$

t' : $T \succ \boxed{T'} \succ [\ldots]$
a'_j : $A_j \succ \boxed{A'_j} \succ [\ldots]$
a'_m : $T' \succ A'_1 \succ \cdots \succ \boxed{A'_m} \succ [\ldots]$
z_i^1 : $Y_i^1 \succ \boxed{Z_i^1} \succ [\ldots]$
z_i^2 : $Y_i^2 \succ \boxed{Z_i^2} \succ [\ldots]$
$\overline{z_i}$: $\overline{Y_i} \succ \boxed{\overline{Z_i}} \succ [\ldots]$
b : $A_m \succ \boxed{B} \succ [\ldots]$

For roommate markets, we consider the same market but without distinguishing between men and women. The only difference in the preferences is that [...] is an arbitrary order over all the rest of the agents. In such a way, there is no incentive to partner with an agent who belongs to the other side of the marriage market. Given these preferences, a swap is rational for one side of the market if and only if it is also rational for the other side (in the constructed roommate market, no *ER*-swap can occur between two agents who were from two different sides in the marriage market). In other words, the set of *ER*-swaps is identical to the set of *FR*-swaps. Hence, the sequences of swaps that may occur are exactly the same as in the proof for housing markets.

Note that one can exhibit a global order over pairs such that the preferences are globally ranked. □

Not surprisingly, for preferences more general than those restricted to the 1-Euclidean domain, recognizing the instances where *ER*-dynamics converges to a Pareto-optimal matching is intractable. The proof is omitted due to space restrictions but the idea is close to the proof of Theorem 1.

Theorem 2. ∀-ER-ParetoSequence *is* co-NP*-hard in housing, marriage and roommate markets even for globally-*

ranked preferences.

In housing markets, the size of a sequence of *ER*-swaps is bounded by $O(n^2)$ because every agent involved in a swap is strictly better off. Thus, since checking the Pareto-optimality of a matching in housing markets can be done in polynomial time [3], we get the following corollary.

Corollary 2. ∃-ER-ParetoSequence *is* NP*-complete and* ∀-ER-ParetoSequence *is* co-NP*-complete in housing markets even for globally-ranked preferences.*

5 Blocking Pair Swaps

BP-swaps cannot occur in housing markets because objects can never be better off. We thus focus in this section on matching markets that match agents with each other.

First, by definition of a blocking pair, any *BP*-stable matching is Pareto-optimal. Moreover, a *BP*-stable matching always exists in marriage markets by the Deferred Acceptance algorithm [21]. However, the convergence to such a state is not guaranteed, even for single-peaked preferences [27]. Nevertheless, there always exists a sequence of *BP*-swaps leading to a stable matching [30]. [2]

In roommate markets, even the existence of a *BP*-stable matching is not guaranteed [21], and actually this is the case even for single-peaked preferences. Nevertheless, checking the existence of a stable matching in a roommate market can be done in polynomial time [24], and there always exists a sequence of *BP*-swaps leading to a stable matching when there exists one [17]. Therefore, by combining these facts with the observation that every *BP*-stable matching is Pareto-optimal, we get the following corollary.

Corollary 3. ∃-BP-ParetoSequence *is solvable in polynomial time in marriage and roommate markets.*

However, in general, determining whether all sequences of *BP*-swaps terminate in a Pareto-optimal matching, i.e., checking convergence of *BP*-dynamics to a Pareto-optimal matching, is hard. This is due to the hardness of checking the existence of a cycle in *BP*-dynamics.

Theorem 3. *Determining whether BP-dynamics can cycle in marriage and roommate markets is* NP*-hard.*

Proof. We perform a reduction from (3,B2)-SAT, a variant of 3-SAT known to be NP-complete [11], where the goal is to decide the satisfiability of a CNF propositional formula with exactly three literals per clause and where each variable appears exactly twice as a positive literal and twice as a negative literal. From an instance of (3,B2)-SAT with formula φ on m clauses $C_1, \dots, C_m$ and p variables $x_1, \dots, x_p$, we build a marriage market ($N = W \cup M, \succ, \mu^0$) as follows. For each clause C_j, with $1 \leq j \leq m$, we create four clause-agents A_j, B_j, Q_j and K_j, where $A_j, Q_j \in W$ and $B_j, K_j \in M$. For each occurrence of variable x_i, with $1 \leq i \leq p$, we create two literal-agents, i.e., agents $Z_i^\ell, D_i^\ell \in W$ and $Y_i^\ell, E_i^\ell \in M$ for the ℓ^{th} positive literal x_i^ℓ of x_i, with $\ell \in \{1,2\}$, and $\overline{Z}_i^\ell, \overline{D}_i^\ell \in W$ and $\overline{Y}_i^\ell, \overline{E}_i^\ell \in M$ for the ℓ^{th} negative literal $\overline{x}_i^\ell$ of x_i, with $\ell \in \{1,2\}$. Denote by A, B, Q, K, D, E, Y and Z the sets of agents associated with the same letter.

The preferences are given below, for $1 \leq i \leq p$, $1 \leq j \leq m$ and $\ell \in \{1,2\}$, with the initial assignment marked with frames. Notation $\{\mathcal{Y}_j\}$ (resp., $\{\mathcal{D}_j\}$, $\{\mathcal{E}_j\}$ and $\{\mathcal{Z}_j\}$) refers to an arbitrary order over the literal-agents in Y (resp., D, E and Z) corresponding to the literals of clause C_j, and $[\dots]$ is an arbitrary order over the rest of the agents of the other type. In general, when a set is given in the preferences, it refers to an arbitrary order over its elements minus the elements of the set already explicitly given in the rest of the preference ranking. Notation $cl(\ell_i)$ refers to the index of the clause in which literal ℓ_i appears. Note that A_0 (resp., B_0) stands for A_m (resp., B_m) and $\{\mathcal{Y}_{m+1}\}$ stands for $\{\mathcal{Y}_1\}$.

A_j : $\{\mathcal{Y}_{j+1}\} \succ \{\mathcal{Y}_j\} \succ \boxed{B_j} \succ B \succ [\dots]$ | B_j : $\boxed{A_j} \succ A \succ \{\mathcal{Z}_{j+1}\} \succ \{\mathcal{Z}_j\} \succ [\dots]$

Z_i^1 : $\overline{Y}_i^1 \succ \overline{Y}_i^2 \succ Y_i^1 \succ \boxed{E_i^1} \succ Y \succ B_{cl(x_i^1)} \succ B_{cl(x_i^1)-1} \succ [\dots]$ | Y_i^1 : $\boxed{D_i^1} \succ \{\mathcal{D}_{cl(x_i^1)}\} \succ Q \succ \overline{Z}_i^1 \succ \overline{Z}_i^2 \succ A_{cl(x_i^1)} \succ A_{cl(x_i^1)-1} \succ Z_i^1 \succ Q_{cl(x_i^1)} \succ [\dots]$

Z_i^2 : $\overline{Y}_i^2 \succ \overline{Y}_i^1 \succ Y_i^2 \succ \boxed{E_i^2} \succ Y \succ B_{cl(x_i^2)} \succ B_{cl(x_i^2)-1} \succ [\dots]$ | Y_i^2 : $\boxed{D_i^2} \succ \{\mathcal{D}_{cl(x_i^2)}\} \succ Q \succ \overline{Z}_i^2 \succ \overline{Z}_i^1 \succ A_{cl(x_i^2)} \succ A_{cl(x_i^2)-1} \succ Z_i^2 \succ Q_{cl(x_i^2)} \succ [\dots]$

$\overline{Z}_i^1$: $Y_i^1 \succ Y_i^2 \succ \overline{Y}_i^1 \succ \boxed{\overline{E}_i^1} \succ Y \succ B_{cl(\overline{x}_i^1)} \succ B_{cl(\overline{x}_i^1)-1} \succ [\dots]$ | $\overline{Y}_i^1$: $\boxed{\overline{D}_i^1} \succ \{\mathcal{D}_{cl(\overline{x}_i^1)}\} \succ Q \succ Z_i^1 \succ Z_i^2 \succ A_{cl(\overline{x}_i^1)} \succ A_{cl(\overline{x}_i^1)-1} \succ \overline{Z}_i^1 \succ Q_{cl(\overline{x}_i^1)} \succ [\dots]$

$\overline{Z}_i^2$: $Y_i^2 \succ Y_i^1 \succ \overline{Y}_i^2 \succ \boxed{\overline{E}_i^2} \succ Y \succ B_{cl(\overline{x}_i^2)} \succ B_{cl(\overline{x}_i^2)-1} \succ [\dots]$ | $\overline{Y}_i^2$: $\boxed{\overline{D}_i^2} \succ \{\mathcal{D}_{cl(\overline{x}_i^2)}\} \succ Q \succ Z_i^2 \succ Z_i^1 \succ A_{cl(\overline{x}_i^2)} \succ A_{cl(\overline{x}_i^2)-1} \succ \overline{Z}_i^2 \succ Q_{cl(\overline{x}_i^2)} \succ [\dots]$

D_i^ℓ : $K_{cl(x_i^\ell)} \succ \boxed{Y_i^\ell} \succ Y \succ [\dots]$ | E_i^ℓ : $Q_{cl(x_i^\ell)} \succ \boxed{Z_i^\ell} \succ Y \succ [\dots]$

$\overline{D}_i^\ell$: $K_{cl(\overline{x}_i^\ell)} \succ \boxed{\overline{Y}_i^\ell} \succ Y \succ [\dots]$ | $\overline{E}_i^\ell$: $Q_{cl(\overline{x}_i^\ell)} \succ \boxed{\overline{Z}_i^\ell} \succ Y \succ [\dots]$

Q_j : $\{\mathcal{Y}_j\} \succ \boxed{K_j} \succ \{\mathcal{E}_j\}$ | K_j : $\{\mathcal{D}_j\} \succ \boxed{Q_j}$

We claim that *BP*-dynamics can cycle if and only if formula φ is satisfiable. The global idea of the reduction is the following. At the initial matching, the only possible *BP*-swaps involve blocking pairs with literal-agents in D and clause-agents in K associated with the same clause. By their swap, a literal-agent in D associated with clause C_j and clause-agent K_j can "unlock" exactly one literal-agent in Y associated with clause C_j who will not be matched with her most preferred agent anymore, and thus could have an incentive to form a blocking pair. By construction of the preferences, the only possibility to get a cycle in *BP*-dynamics is that, for each clause C_j, exactly one literal-agent $\mathtt{Y}_j$ in Y associated with C_j is unlocked and the cycle involves a sequence of blocking pairs $(A_j, \mathtt{Y}_j), (A_j, \mathtt{Y}_{j+1}), (A_{j+1}, \mathtt{Y}_{j+1}), \dots$ (with $j+1$ modulo m) all along the m clauses. For this cycle to occur, the unlocked literal-agents in Y must have been matched with their associated agent in Z. Therefore, two unlocked literal-agents in Y participating in the cycle cannot correspond to opposite literals, otherwise one of them would be matched at a moment of the cycle with an agent in Z corresponding to her opposite literal, and thus would not agree to form a

2. Assuming that the old partners also form a new pair does not alter this result.

blocking pair with a clause-agent. The details of the equivalence are omitted. This proof can be adapted to roommate markets by assuming that, in the preferences, [...] is an arbitrary order over the remaining agents where the agents of the same "type" in the marriage market are ranked last. □

Corollary 4. ∀-BP-PARETOSEQUENCE *is* co-NP*-hard in marriage and roommate markets.*

Nevertheless, when the preferences are globally-ranked, we can always reach a stable matching thanks to *BP*-dynamics in both settings. Indeed, it has been proved that *BP*-dynamics always converges in marriage markets with globally-ranked preferences [7]. In roommate markets, there always exists a unique *BP*-stable matching under 1-Euclidean preferences [9]. We prove that convergence to this matching is guaranteed using a potential function argument, and further, this holds for more general preferences, namely globally-ranked preferences.

Proposition 5. *BP-dynamics always converges in roommate markets for globally-ranked preferences.*

Proof. Denote by $>$ the global order over all possible pairs such that the preferences of the agents are globally-ranked with respect to $>$. Let $d(\mu)$ be the $n/2$-vector of the ranks in $>$ of all the different pairs of μ, i.e., $d(\mu) = (rank_>(\{i,j\}))_{i,j \text{ s.t. } \mu(i)=j}$ with $rank_>$ the function which gives the rank of the pairs in order $>$. Consider a sequence of *BP*-swaps given by the following sequence of matchings $(\mu^0, \mu^1, \ldots, \mu^r)$. Between each pair of matchings μ^t and μ^{t+1} with $0 \leq t < r$, a *BP*-swap is performed, say w.r.t. blocking pair (i,j). By definition of a *BP*-swap, agents i and j prefer to be together than being with their partner in μ^t, so $j = \mu^{t+1}(i) \succ_i \mu^t(i)$ and $i = \mu^{t+1}(j) \succ_j \mu^t(j)$, which implies, by correlation of the preferences, that $\{i,j\} > \{i,\mu^t(i)\}$ and $\{i,j\} > \{j,\mu^t(j)\}$. Therefore, $(rank_>(\{i,j\}), rank_>(\{\mu^t(i),\mu^t(j)\}))$ is lexicographically strictly smaller than $(rank_>(\{i,\mu^t(i)\}), rank_>(\{j,\mu^t(j)\}))$. Since the rest of the pairs remains unchanged between μ^t and μ^{t+1}, it follows that $d(\mu^{t+1})$ is lexicographically strictly smaller than $d(\mu^t)$. The number of different matchings is finite, therefore *BP*-dynamics always converges. □

Since every *BP*-stable matching is Pareto-optimal, we obtain the following corollary.

Corollary 5. *BP-dynamics always converges to a Pareto-optimal matching in marriage and roommate markets when the preferences are globally-ranked.*

6 Fully Rational Swaps

Just as in the case of *ER*-swaps and housing markets, *FR*-swaps always represent Pareto improvements because all involved agents are strictly better off after the swap. Hence, *FR*-stable matchings are guaranteed to exist because every Pareto-optimal matching is *FR*-stable and *FR*-dynamics always converges because the number of agents is finite.

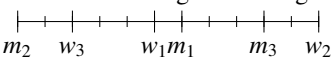
In Section 4, we have shown that *ER*-dynamics always converges to a Pareto-optimal matching when the preferences of the agents are 1-Euclidean. It turns out that this does not hold for *FR*-dynamics.

Proposition 6. *A sequence of FR-swaps may not converge to a Pareto-optimal matching in marriage and roommate markets, even for 1-Euclidean preferences.*

Proof. Consider a marriage market with three women and three men. The preferences are given below, where the initial assignment is marked with frames.

$$\begin{array}{llll|llll} w_1: & (m_1) & \succ \; \boxed{m_3} & \succ \; m_2 & m_1: & (w_1) & \succ \; \boxed{w_3} & \succ \; w_2 \\ w_2: & (m_3) & \succ \; m_1 & \succ \; \boxed{m_2} & m_2: & (w_3) & \succ \; w_1 & \succ \; \boxed{w_2} \\ w_3: & (m_2) & \succ \; \boxed{m_1} & \succ \; m_3 & m_3: & (w_2) & \succ \; \boxed{w_1} & \succ \; w_3 \end{array}$$

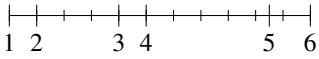
Initial matching μ^0 is the only reachable matching, because there is no *FR*-swap from μ^0. However, there is another matching (circled agents) which is not reachable but which Pareto dominates matching μ^0. The preferences are 1-Euclidean w.r.t. the following embedding on the real line.



Now, consider a roommate market with six agents. The preferences of the agents are given below, where the initial assignment is marked with frames.

$$\begin{array}{llllll|llllll} 1: & (2) & \succ & 3 & \succ & 4 \succ 5 \succ \boxed{6} & 4: & (3) & \succ & 2 & \succ & \boxed{5} \succ 1 \succ 6 \\ 2: & (1) & \succ & \boxed{3} & \succ & 4 \succ 5 \succ 6 & 5: & (6) & \succ & \boxed{4} & \succ & 3 \succ 2 \succ 1 \\ 3: & (4) & \succ & \boxed{2} & \succ & 1 \succ 5 \succ 6 & 6: & (5) & \succ & 4 & \succ & 3 \succ 2 \succ \boxed{1} \end{array}$$

Initial matching μ^0 is the only reachable matching, because there is no *FR*-swap from μ^0. However, there is another matching (circled agents) which is not reachable but which Pareto dominates matching μ^0. The preferences are 1-Euclidean w.r.t. the following embedding on the real line.



□

The proofs of Theorems 1 and 2 only dealt with instances in which *FR*-swaps are identical to *ER*-swaps. We thus immediately obtain hardness of ∃-FR-PARETOSEQUENCE and ∀-FR-PARETOSEQUENCE. An *FR*-swap makes four agents strictly better off and no agent worse off, thus the size of a sequence of *FR*-swaps is bounded by $O(n^2)$. Moreover, the Pareto-optimality of a matching can be checked in polynomial time [5], therefore we get the membership of the problems to NP and co-NP, respectively.

Theorem 4. ∃-FR-PARETOSEQUENCE *is* NP*-complete and* ∀-FR-PARETOSEQUENCE *is* co-NP*-complete in marriage and roommate markets even for globally-ranked preferences.*

7 Conclusion

We have studied the properties of different dynamics of rational swaps in matching markets with initial assignments and, in particular, the question of convergence to a Pareto-optimal matching. For all considered settings, the dynamics may not terminate in a Pareto-optimal matching because *(i)* there is no stable matching, *(ii)* the dynamics does not converge, or *(iii)* the stable matching that is eventually reached is not Pareto-optimal. An overview of our results is given in Table 1.

Market	Preferences	*ER*-Swaps	*BP*-Swaps	*FR*-Swaps
Housing	General / GR	Conv		
	SP	Pareto [16]		
	1-D	Pareto		
Marriage	General	–	Stable [21]	Conv
	GR	Conv (Prop. 2)	Pareto (Cor. 5)	Conv
	SP	– [13]	Stable	Conv
	1-D	Pareto (Cor. 1)	Pareto	Conv
Roommate	General	–	– [21]	Conv
	GR	Conv (Prop. 2)	Pareto (Cor. 5)	Conv
	SP	– [8]	–	Conv
	1-D	Pareto (Cor. 1)	Pareto	Conv

Pareto ⇒ Conv ⇒ Stable

Table 1 – Summary of the results on the existence of a stable matching (Stable), the guarantee of convergence (Conv) and the guarantee of convergence to a Pareto-optimal matching (Pareto) for the three different matching markets under study, according to different types of rational swaps and under different preference domains (General, globally-ranked (GR), single-peaked (SP), and 1-Euclidean (1-D)). Since Pareto ⇒ Convergence ⇒ Stable, we only mention the strongest result which is satisfied. The only meaningful type of rational swaps in housing markets are exchange-rational swaps; hence, the empty spaces.

Computationally, determining whether there *exists* a sequence of rational swaps terminating in a Pareto-optimal matching is NP-hard for fully rational swaps and exchange rational swaps in all matching markets even for globally-ranked preferences (Theorems 1 and 4). For swaps based on blocking pairs, this problem can be solved efficiently (Corollary 3). However, the convergence to a Pareto-optimal matching, that is whether *all* sequences of swaps terminate in a Pareto-optimal matching, is co-NP-hard to decide (Corollary 4). Not surprisingly, the same hardness result holds for fully rational and exchange rational swaps, even for globally-ranked preferences (Theorems 2 and 4). Our computational results are summarized in Table 2. Even if the existence of a sequence of swaps terminating in a Pareto-optimal matching is not guaranteed for single-peaked preferences in marriage and roommate markets, it would be interesting to know whether this preference restriction is nevertheless sufficient for efficiently solving our computational problems in these markets.

Market	Prefs	*ER*-Swaps		*BP*-Swaps		*FR*-Swaps	
		∃-ParSeq	∀-ParSeq	∃-ParSeq	∀-ParSeq	∃-ParSeq	∀-ParSeq
Housing	General / GR	NP-c. (Cor. 2)	co-NP-c. (Cor. 2)				
	SP	P [16]	P [16]				
Marriage / Roommate	General	NP-h. (Th. 1)	co-NP-h. (Th. 2)	P (Cor. 3)	co-NP-h. (Cor. 4)	NP-c. (Th. 4)	co-NP-c. (Th. 4)
	GR	NP-h. (Th. 1)	co-NP-h. (Th. 2)	P (Cor. 3)	P (Cor. 5)	NP-c. (Th. 4)	co-NP-c. (Th. 4)

Table 2 – Summary of the computational results on the existence (∃-ParSeq) or the guarantee (∀-ParSeq) of sequences of rational swaps terminating in a Pareto-optimal matching for the three different matching markets under study, according to different types of rational swaps and under different preference domains (General, globally-ranked (GR) and single-peaked (SP)). P means polynomial time solvable. The only meaningful type of rational swaps in housing markets are *ER*-swaps; hence, the empty spaces.

The convergence to a Pareto-optimal matching in housing markets for exchange rational dynamics and single-peaked preferences [16] does not hold for more general settings where the "objects" are agents with preferences. However, this convergence is guaranteed under 1-Euclidean preferences in marriage and roommate markets. Hence, the generalization of this convergence result to more general settings requires more structure in the preferences.

A natural extension of this work would be to study meaningful dynamics for hedonic games, where agents form groups consisting of more than two agents.

References

[1] Abdulkadiroğlu, A. et T. Sönmez: *House Allocation with Existing Tenants*. Journal of Economic Theory, 88(2) :233–260, 1999.

[2] Abeledo, H. et U. G. Rothblum: *Paths to marriage stability*. Discrete Applied Mathematics, 63(1) :1–12, 1995.

[3] Abraham, D. J., K. Cechlárová, D. F. Manlove et K. Mehlhorn: *Pareto Optimality in House Allocation Problems*. Dans *Proc. of 16th ISAAC*, tome 3341 de *LNCS*, pages 1163–1175, 2005.

[4] Abraham, D. J., A. Leravi, D. F. Manlove et G. O'Malley: *The stable roommates problem with globally-ranked pairs*. Dans *Proc. of 3rd WINE*, tome 4858 de *LNCS*, pages 431–444. Springer, 2007.

[5] Abraham, D. J. et D. F. Manlove: *Pareto optimality in the Roommates problem*. Rapport technique TR-2004-182, University of Glasgow, Department of Computing Science, 2004.

[6] Ackermann, H., P. W. Goldberg, V. S. Mirrokni, H. Röglin et B. Vöcking: *A Unified Approach to Congestion Games and Two-Sided Markets*. Dans *Proc. of 3rd WINE*, pages 30–41. Springer Berlin Heidelberg, 2007.

[7] Ackermann, H., P. W. Goldberg, V. S. Mirrokni, H. Röglin et B. Vöcking: *Uncoordinated two-sided matching markets*. SIAM Journal on Computing, 40(1) :92–106, 2011.

[8] Alcalde, J.: *Exchange-proofness or divorce-proofness ? Stability in one-sided matching markets*. Review of Economic Design, 1(1) :275–287, 1994.

[9] Arkin, E. M., S. W. Bae, A. Efrat, K. Okamoto, J. S. B. Mitchell et V. Polishchuk: *Geometric stable roommates*. Information Processing Letters, 109(4) :219–224, 2009.

[10] Bartholdi, III, J. et M. Trick: *Stable matching with preferences derived from a psychological model*. Operations Research Letters, 5(4) :165–169, 1986.

[11] Berman, P., M. Karpinski et A. Scott: *Approximation hardness of short symmetric instances of MAX-3SAT*. Rapport technique TR03-049, ECCC, 2003.

[12] Black, D.: *On the Rationale of Group Decision-making*. Journal of Political Economy, 56(1) :23–34, 1948.

[13] Cechlárová, K.: *On the complexity of exchange-stable roommates*. Discrete Applied Mathematics, 116(3) :279–287, 2002.

[14] Cechlárová, K. et D. F. Manlove: *The exchange-stable marriage problem*. Discrete Applied Mathematics, 152(1–3) :109–122, 2005.

[15] Coombs, C. H.: *Psychological scaling without a unit of measurement*. Psychological Review, 57(3) :145–158, 1950.

[16] Damamme, A., A. Beynier, Y. Chevaleyre et N. Maudet: *The Power of Swap Deals in Distributed Resource Allocation*. Dans *Proc. of 14th AAMAS Conference*, pages 625–633. IFAAMAS, 2015.

[17] Diamantoudi, E., E. Miyagawa et L. Xue: *Random paths to stability in the roommate problem*. Games and Economic Behavior, 48(1) :18–28, 2004.

[18] Doignon, J. P. et J. C. Falmagne: *A polynomial time algorithm for unidimensional unfolding representations*. Journal of Algorithms, 16(2) :218–233, 1994.

[19] Elkind, E. et P. Faliszewski: *Recognizing 1-Euclidean preferences : An alternative approach*. Dans *Proc. of 7th SAGT*, pages 146–157. Springer, 2014.

[20] Escoffier, B., J. Lang et M. Öztürk: *Single-Peaked Consistency and its Complexity*. Dans *Proc. of 18th European Conference on Artificial Intelligence (ECAI)*, pages 366–370. IOS Press, 2008.

[21] Gale, D. et L. S. Shapley: *College Admissions and the Stability of Marriage*. The American Mathematical Monthly, 69(1) :9–15, 1962.

[22] Gourvès, L., J. Lesca et A. Wilczynski: *Object Allocation via Swaps along a Social Network*. Dans *Proc. of 26th IJCAI*, pages 213–219. IJCAI, 2017.

[23] Huang, S. et M. Xiao: *Object reachability via swaps along a line*. Dans *Proc. of 33rd AAAI Conference*, pages 2037–2044. AAAI Press, 2019.

[24] Irving, R. W.: *An Efficient Algorithm for the "Stable Roommates" Problem*. Journal of Algorithms, 6(4) :577–595, 1985.

[25] Klaus, B., D. F. Manlove et F. Rossi: *Matching under Preferences*. Dans Brandt, F., V. Conitzer, U. Endriss, J. Lang et A. D. Procaccia (rédacteurs) : *Handbook of Computational Social Choice*, chapitre 14. Cambridge University Press, 2016.

[26] Knoblauch, V.: *Recognizing one-dimensional Euclidean preference profiles*. Journal of Mathematical Economics, 46(1) :1–5, 2010.

[27] Knuth, D. E.: *Mariages stables*. Les Presses de l'Université de Montréal, 1976.

[28] Manlove, D. F.: *Algorithmics of Matching Under Preferences*. World Scientific Publishing Company, 2013.

[29] Morrill, T.: *The roommates problem revisited*. Journal of Economic Theory, 145(5) :1739–1756, 2010.

[30] Roth, A. E. et J. H. Vande Vate: *Random Paths to Stability in Two-Sided Matching*. Econometrica, 58(6) :1475–1480, 1990.

[31] Saffidine, A. et A. Wilczynski: *Constrained Swap Dynamics over a Social Network in Distributed Resource Reallocation*. Dans *Proc. of 11th SAGT*, tome 11059 de *LNCS*, pages 213–225. Springer, 2018.

[32] Shapley, L. S. et H. Scarf: *On cores and indivisibility*. Journal of Mathematical Economics, 1(1) :23–37, 1974.

[33] Yoshinaka, R.: *Higher-Order Matching in the Linear Lambda Calculus*. Dans *Proc. of 16th RTA*, pages 235–249, 2005.

Actes JIAF 2020

United We Stand: Accruals in Strength-based Argumentation

Julien Rossit[1] **Jean-Guy Mailly**[1]
Yannis Dimopoulos[2] **Pavlos Moraitis**[1,3]

[1] LIPADE, Université de Paris, France
[2] Dept. of Computer Science, University of Cyprus, Chypre
[3] Argument Theory, France
julien.rossit@u-paris.fr jean-guy.mailly@u-paris.fr
yannis@cs.ucy.ac.cy pavlos.moraitis@u-paris.fr

Abstract

In this paper, we propose a new abstract argumentation framework where arguments are associated with a strength, *i.e.* a quantitative information which is used to determine whether an attack between arguments succeeds or not. Our *Strength-based Argumentation Framework* (*StrAF*) permits to define acceptability semantics sensitive to the existence of accruals between arguments. Intuitively speaking, the question of accruals arises in situations where several arguments defending the same position (but from different points of view) against another argument fail to individually defeat this argument, but could do it collectively if they combine their strengths. We investigate some of the theoretical and computational properties of our new framework and semantics, and present a reasoning algorithm that is based on a translation of the problem into pseudo-Boolean constraint satisfaction.

Introduction

Argumentation has been a main research topic in artificial intelligence for the last twenty years, with applications in various domains such as decision making [4], automated negotiation [23], reasoning with inconsistent knowledge [13], legal reasoning [11], and multi-agent systems [35]. A lot of works have been proposed, mainly based on the influential argumentation framework (AF) of Dung [24]. An AF is a directed graph where nodes are *arguments* and edges are *attacks* between arguments. The classical acceptability semantics are based on the notion of *extensions*, that are sets of jointly acceptable arguments [7].

Different agents may apply contrasting policies to resolve conflicts, depending on the relative priorities they associate with arguments. These relative priorities may be represented as a relation of preference between arguments [3], or by a relation of preference between the values attached to arguments [10]. In these works, the reasoning process is achieved in two steps : first a *defeat* relation is defined as the combination of the initial attack relation and the preference relation, yielding a defeat graph. Then, an extension semantics is applied on the defeat graph. However, none of these frameworks allows to finely compare arguments w.r.t. a quantitative strength. Intuitively speaking, considering a strength-based framework allows to induce priorities among arguments by merely associating a weight to each argument, and specially does not require to examine each possible pairs of arguments.

Assigning a quantitative information to arguments has been considered in different contexts. First, [15] associates arguments with a numerical value that represents "fuzziness, probability or a preference in general" ; the extensions given by Dung's semantics are then refined depending on the weights of the arguments that belong to them. Then, a different approach consists in evaluating the individual acceptability of arguments based on their intrinsic strength and the attack relation between arguments [2]. Contrary to preference-based argumentation, these weighted argumentation frameworks defined in [15, 2] do not consider a notion of defeat in the reasoning process. Moreover, none of the options presented above permits to consider collective attacks [37], and especially *accrual* of arguments.

The question of accruals of reasons has been addressed in the AI literature (see [41, 39, 37] for examples). Although most of these studies differ from this paper since they settle in a rule-based argumentation context, all corresponding approaches furthermore require to express explicitly *a priori* all sets of accruals to take them into account throughout the reasoning process. As far as we know, the question of how accruals can be detected during the rea-

soning process has not been addressed in the literature.

The aim of this paper is to study this question. More precisely, we propose an approach which :

— allows strength compensations in an argumentation context where attacks may not succeed ;
— detects accruals of arguments during the reasoning process without requiring their explicit elicitation in the model as an input.

To this purpose, we define the *Strength-based Argumentation Framework* which combines in an original way the quantitative strength expressed in weighted argumentation frameworks on one hand, and the notion of defeat relation introduced in the contexts of preference-based and value-based argumentation on another hand. We generalize the notion of defeat to define a notion of *collective defeat*. We use the term of *accrual* to identify a set of arguments that attack a same target. While arguments in this set might not be able to individually defeat their common target, they could achieve the defeat by combining their strength. The representation proposed in this paper allows to compute the strength of an existing accrual, and consequently to decide about the outcome of a conflict between an accrual and an individual argument (or between two accruals) by applying a slightly adapted version of extension semantics.

The paper is organized as follows. Section 1 recalls the basic notions of abstract argumentation. In Section 2, we propose an intuitive framework for representing strength of arguments, then Section 3 introduces accrual of arguments, and provides some theoretical properties. In Section 4, we study the complexity of the reasoning processes for some particular sub-classes of *StrAFs*, and then for the general case. Then we provide a computational approach based on a Pseudo-Boolean translation of the reasoning problem. Finally, Section 5 discusses existing related work, and Section 6 concludes the paper and describes some tracks for future research. Proofs are omitted for space reasons.

1 Preliminaries

An *argumentation framework* (AF), as introduced by Dung in [24], is a pair $\langle \mathcal{A}, \mathcal{R} \rangle$, where $\mathcal{A}$ is a set of *arguments*, and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is an *attack relation*. The relation *a attacks b*, or *b is attacked by a*, is denoted by $(a, b) \in \mathcal{R}$. We focus on finite argumentation frameworks, *i.e.* AFs s.t. $\mathcal{A}$ is a finite set. In [24], different acceptability semantics have been introduced, relying on two basic concepts, namely *defence* and *conflict-freeness*, defined as follows :

Definition 1. *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation system. Let $S \subseteq \mathcal{A}$.*

— *S is conflict-free iff $\nexists\, a, b \in S$ s.t $(a, b) \in \mathcal{R}$.*
— *S defends $a \in \mathcal{A}$ iff $\forall\, b \in \mathcal{A}$, if $(b, a) \in \mathcal{R}$, then $\exists c \in S$ s.t. $(c, b) \in \mathcal{R}$.*

The set of all conflict-free sets of AF is denoted $\mathrm{cf}(AF)$.

The basic idea behind these semantics is the following : for a rational agent, an argument *a* is acceptable if he can defend *a* against all attacks. All the arguments acceptable for a rational agent will be gathered in a so-called *extension*. An extension must satisfy a consistency requirement and must defend all its elements.

Definition 2. *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an AF and $S \in \mathrm{cf}(AF)$.*

— *S is an admissible set iff S defends any element in S.*
— *S is a preferred extension iff S is a $\subseteq$-maximal admissible set.*
— *S is a stable extension iff $S \in \mathrm{cf}(AF)$ and S attacks any argument in $\mathcal{A} \setminus S$.*

The sets of admissible, preferred and stable extensions are respectively denoted by $\mathrm{ad}(AF), \mathrm{pr}(AF)$ *and* $\mathrm{st}(AF)$.

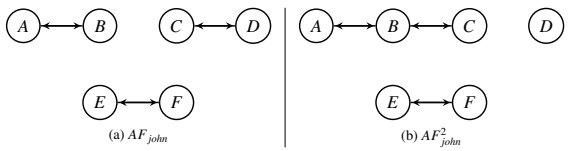
See [24, 7] for more details about extension semantics.

We illustrate that, besides the natural application of AFs to "real argumentation" (*i.e.* debates based on the exchange of arguments, where each argument is a reason for supporting some claim), Dung's AFs can be used to model reasoning problems in presence of conflicting information.

Example 1. *John is a gardener, he has several options for working on the next day : he can maintain the garden of different houses (A, B, C and D), but with some constraints :*

— *He can work at house A between 3 p.m. and 5 p.m. ;*
— *He can work at house B between 4 p.m. and 7 p.m. ;*
— *He can work at house C between 8 a.m. and 10 a.m. ;*
— *He can work at house D between 9 a.m. and 12 a.m ;*
— *He can work at house E between 12 p.m. and 2 p.m ;*
— *He can work at house F between 1 p.m. and 3 p.m ;*

Because of schedule overlapping, houses A and B cannot both be selected by John, and similarly for houses C and D on the one hand, and E and F on the other hand. This situation is represented by AF_{john} given at Figure 1a, were argument X is accepted if John decides to maintain the garden of house X. Now, if the owner of house C tells John that, finally, he cannot come between 8 :00 a.m. and 10 :00 a.m., but rather between 5 :00 p.m. and 7 :00 p.m., we obtain the situation described by AF^2_{john} (Figure 1b).



(a) AF_{john} (b) AF^2_{john}

FIGURE 1 – Both Versions of John's Garden Situation

The (stable) extensions are $\mathrm{st}(AF_{john}) = \{\{A, C, E\}, \{A, C, F\}, \{A, D, E\}, \{A, D, F\}, \{B, C, E\}, \{B, C, F\}, \{B, D, E\}, \{B, D, F\}\}$ *and* $\mathrm{st}(AF^2_{john}) = \{\{A, C, D, E\}, \{A, C, D, F\}, \{B, D, E\}, \{B, D, F\}\}$. *All these extensions can represent reasonable choices of working activities for John.*

2 Strength-based AFs

We have described in the introduction some intuitions on the meaning of the numerical strength associated with arguments. While [15] mentions "fuzziness, probability or a preference in general", or a notion of trust about the arguments, [2] associates the weights to an intrinsic strength, that represents votes given by users [31], a certainty degree of arguments premises [12] or trustworthiness of the source of information [22]. A similar intuitive explanation of weights is given in [36, 38].

Let us now formally introduce the strength-based argumentation framework. Intuitively speaking, a *StrAF* is an argumentation framework where each argument is associated with a weight (here a non-negative integer) which represents its strength :

Definition 3. *A* Strength-based Argumentation Framework *(StrAF) is a triple* $F = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *with* $\mathcal{A}$ *and* $\mathcal{R}$ *(resp.) arguments and attacks, and* $\mathcal{S} : \mathcal{A} \rightarrow \mathbb{N}$ *a* strength function.

In our framework, the higher is the weight, the stronger is the argument this weight is associated with. Let us illustrate this representation with the following example :

Example 2. *Let us continue Example 1. Depending of the size of the garden, John's salary for these works is not the same : houses A, B, C, D, E and F are worth resp.* 40 *Euros,* 60 *Euros,* 40 *Euros,* 80 *Euros,* 40 *Euros and* 60 *Euros. For each argument, the salary that John gets corresponds to the strength of the argument. So, from* AF_{john} *and* AF^2_{john}*, we can define StrAFs* $StrAF_{john} = \langle \mathcal{A}_{john}, \mathcal{R}_{john}, \mathcal{S}_{john} \rangle$ *and* $StrAF^2_{john} = \langle \mathcal{A}_{john}, \mathcal{R}^2_{john}, \mathcal{S}_{john} \rangle$ *described at Figure 2, where* $\mathcal{S}(A) = \mathcal{S}(C) = \mathcal{S}(E) = 40$*,* $\mathcal{S}(B) = \mathcal{S}(F) = 60$ *and* $\mathcal{S}(D) = 80$*. This means that an argument X is stronger than an argument Y if John earns more money when maintaining the garden of house X rather than house Y.*

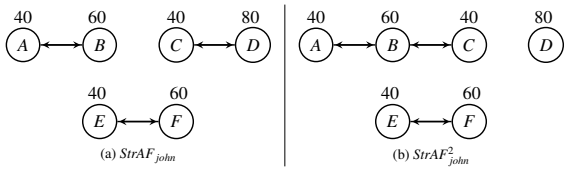


FIGURE 2 – Both Versions of John's Garden Situation

We could adapt Dung-style semantics to *StrAFs* with a notion of defeat relation, similarly to Preference-based AFs and Value-based AFs [3, 10]. Formally, the *defeat* relation Def is defined by $\text{Def} = \{(a, b) \in \mathcal{R} \mid \mathcal{S}(b) \not> \mathcal{S}(a)\}$. Then, the σ extensions can be computed classically from the graph $\langle \mathcal{A}, \text{Def} \rangle$. These defeat-based semantics are relevant when the strength associated with an argument is individual and independent from other arguments, likewise the behavior of PAFs. In our case, the semantics may produce an extension which is not the most desirable outcome, for instance in the case of $StrAF^2_{john}$, the stable semantics gives $\{B, D, F\}$ (that is worth 200 Euros) as the single extension, while $\{A, C, D, F\}$ (that is worth 220 Euros) would be a better option. This is why we propose to define semantics sensitive to the notion of accrual of arguments.

3 Accrual of Arguments

Recent works have tackled the question of "quality versus quantity" : is it worse for an argument to be attacked by only one strong attacker, or to be attacked by plenty of quite weak attackers ? In the context of gradual semantics, this question is materialized by the principles of Quality Precedence and Cardinality Precedence. A Compensation principle is also stated to describe situations where several weak arguments have the same effect on their target as fewer strong arguments [1, 18, 2]. Although the context is not the same (since we consider extension-based semantics), a similar intuition leads to our definition of accruals : several arguments may be individually too weak to defeat their target, but their collective attack may be strong enough to compensate or even exceed the target's strength.

3.1 StrAFs with Accrual

In this paper, we assume that weights associated with arguments are commensurable. Roughly speaking, this means that these weights are comparable from an argument to another, and aggregating them (with *e.g.* a Sum-based operation) makes sense. This kind of assumptions suits well situations where, for instance, weights associated with arguments are provided by an expert or a group of experts sharing a same representation scale, or are regarded as rewards granted for the acceptance of some arguments. An example of the latter is provided by our running example.

We formally define the notion of accrual of arguments :

Definition 4. *Let* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a StrAF. A set* $\kappa \subseteq \mathcal{A}$ *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 c*.*

Intuitively, an accrual is a set of arguments s.t. there exists an argument which is attacked by all arguments in this set. For instance, in $StrAF^2_{john}$ describing John's situation (see Example 2), there is *e.g.* an accrual $\kappa = \{A, C\}$ attacking the argument B.

In the following, we choose to opt for a pessimistic view of attacking an accrual. An accrual is said to be attacked by an argument iff at least one of its arguments is attacked. Then, several definitions of *an accrual attacking another accrual* are possible. In what follows, we choose to focus on the following one, which corresponds again to the pessimistic case : an accrual κ attacks an accrual κ' iff there exists an argument in κ' s.t. all arguments of κ attack this argument. Formally :

Definition 5. *For StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$*, and* $\kappa, \kappa' \subseteq \mathcal{A}$ *two accruals,* κ attacks κ' *iff* $\exists a \in \kappa'$ *s.t.* κ *attacks* a*.*

We then define the *collective strength* associated with an accrual κ, denoted by coval(κ), as the combination of values associated with arguments. Formally :

Definition 6. *Let StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a StrAF and* $\kappa = \{a_1, ..., a_n\} \subseteq \mathcal{A}$ *be an accrual. Then the collective strength associated with* κ *is :*

$$\text{coval}(\kappa) = \text{coval}(\mathcal{S}(a_1), \ldots, \mathcal{S}(a_n))$$

where coval *is an aggregation operator,* i.e. *a mapping from a vector of integers to an integer that satisfies :*

(non-decreasingness) *if* $x_i \geq x'_i$*, then* $\text{coval}(x_1, \ldots, x_i, \ldots, x_n) \geq \text{coval}(x_1, \ldots, x'_i, \ldots, x_n)$*;*

(minimality) $\text{coval}(x_1, \ldots, x_n) = 0$ *iff* $x_1 = \cdots = x_n = 0$*;*

(identity) $\text{coval}(x) = x$*;*

(accrual) $\text{coval}(x_1, \ldots, x_n) \leq \text{coval}(x_1, \ldots, x_n, y)$.

These properties are quite natural for defining the strength of an accrual. Non-decreasingness means that, if $a \notin \kappa$ and $b \notin \kappa$, then the accrual $\kappa \cup \{a\}$ is stronger than the accrual $\kappa \cup \{b\}$ when a is stronger than b. Minimality states that the accumulation of arguments "without" strength (*i.e.* $\mathcal{S}(a_i) = 0$) does not create a collective strength from the void, and that non-null arguments cannot cancel each other's strength when they are taken together. Identity ensures that the "collective" strength of a singleton is exactly the intrinsic strength of the argument in this singleton. These properties are used for defining aggregation operators with other purposes, like defining extensions in AFs with weighted attacks [21] ; they correspond to the basic properties of semirings [14, 16]. Finally, we consider the Accrual property : any set of arguments is at least as strong as all its subsets.

The coval operator may be a classical aggregation function like the sum ($\sum$), the maximum (max) or the weighted sum. Conversely, the product Π does not satisfy the minimality property. Furthermore, one can notice that if we extend our approach to positive real numbers for representing the strength, then the product Π operator does not satisfy the accrual properties if some values belong to the interval $[0, 1]$. However, all the properties are satisfied by Π if only natural or real numbers greater or equal to 1 are allowed.

To accommodate the notion of accrual we extend the semantics of StrAFs to take collective defeat as follows. We say that an argument a is collectively defeated by an accrual κ iff the collective strength associated with κ is greater or equal to the value associated with a.

Definition 7. *Let StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a StrAF,* $a \in \mathcal{A}$*, and* coval *and aggregation operator. Then, an accrual* κ defeats *a w.r.t.* coval*, denoted by* $\kappa \rhd_{\text{coval}} a$*, iff* $\kappa \subseteq \mathcal{A}$ *is an accrual that attacks* a *and* $\text{coval}(\kappa) \geq \mathcal{S}(a)$*. We only write* $\kappa \rhd a$ *when* coval *is clear from the context.*

We can introduce the notion of *an accrual that defeats another accrual* as follows :

Definition 8. *For StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$*,* coval *an aggregation operator and* $\kappa \subseteq \mathcal{A}$*,* $\kappa' \subseteq \mathcal{A}$ *two accruals,* κ defeats κ'*, denoted by* $\kappa \rhd \kappa'$*, iff there exists* $a \in \kappa'$ *s.t.* $\kappa \rhd a$*.*

Roughly speaking, an accrual defeats another accrual if the first accrual induces a defeat against at least one argument of the second accrual.

In [16], the strength of an attack from a set S to a set S' of arguments is defined as the aggregation of strengths associated with all the attacks from any argument in S to any argument in S'. While this may look similar to defeats between accruals, these are actually different concepts. Indeed, since [16] do not associate strength with arguments, but with attacks, there is no comparison between strength associated resp. with S and S'. Moreover, in our approach, defeating any argument of an accrual is enough to consider the whole accrual as defeated. Nevertheless, one can notice that a subset of this defeated accrual can still be an undefeated accrual.

Let us illustrate the notion of collective attacks with our running example.

Example 3. *We continue Example 2. The natural aggregation operator here is* coval $= \sum$*, since the strength of an accrual is the money earned by John for maintaining several gardens (corresponding to the arguments in the accrual). We observe that, in* $StrAF^2_{john}$*, the accruals* $\kappa_1 = \{B\}$ *and* $\kappa_2 = \{A, C\}$ *defeat each other. Indeed,* $\text{coval}(\kappa_1) = \mathcal{S}(B) = 60 \geq \mathcal{S}(A)$*, while* $\text{coval}(\kappa_2) = \mathcal{S}(A) + \mathcal{S}(C) = 80 \geq \mathcal{S}(B)$*.*

3.2 Extension-based Semantics for StrAFS

In [24], different acceptability semantics have been introduced for computing the status of arguments. These are based on two basic concepts, *defence* and *conflict-freeness*, which can be adapted to the context of accruals. The notion of conflict-freeness can be defined into two different senses, the *strong* and the *weak* conflict-freeness.

Definition 9. *For StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$*,* coval *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.* $\kappa_1 \rhd_{\text{coval}} \kappa_2$*.*
- S *defends an element* $a \in \mathcal{A}$ *iff for all* $\kappa_1 \subseteq \mathcal{A}$*, if* $\kappa_1 \rhd_{\text{coval}} a$*, then there exists* $\kappa_2 \subseteq S$ *s.t.* $\kappa_2 \rhd_{\text{coval}} \kappa_1$*.*

Depending on the notion of conflict-freeness that is applied, two versions of acceptability semantics are derived, that are defined formally below.

Definition 10. *Let StrAF* $= \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a StrAF,* coval *an aggregation operator, and* S *a strong (resp. weak) conflict free set of arguments.*

— 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* $a \in \mathcal{A} \backslash S$*, there is an accrual* $\kappa \subseteq S$ *s.t.* $\kappa \rhd a$.

For any semantics $\sigma \in \{\text{cf}, \text{ad}, \text{pr}, \text{st}\}$, the set of strong (resp. weak) σ extension of $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ w.r.t. coval is denoted by $\sigma_S^{\text{coval}}(StrAF)$ (resp. $\sigma_W^{\text{coval}}(StrAF)$). We drop coval from the notations when it is clear from the context.

It is easy to see that a strong extension is a weak extension as well, since the absence of attacks in a set S (*i.e.* strong conflict-freeness) straightforwardly implies the absence of accrual in S defeating elements of S (*i.e.* weak conflict-freeness). Therefore $\sigma_S(StrAF) \subseteq \sigma_W(StrAF)$ for any semantics σ.

We also remark a relation between weak semantics of *StrAFs* and semantics of AFs with collective attacks [37]. Indeed, if $\kappa \rhd a$ holds in a *StrAF*, we can define an AF with collective attacks where the set of arguments κ attacks a (according to the notion of attacks defined in [37]). Thus, weak conflict-freeness (and the weak semantics based on weak conflict-freeness) coincide with the conflict-freeness and semantics for AFs with collective attacks. However, we notice that *StrAFs* allow a more modular representation of argument accrual, since they are only based on individual attacks and arguments strengths, and they do not need to be explicitly defined *a priori*.

Now we prove that Dung's argumentation theory is an instance of our StrAF, as it is shown in the following result, that proves a one to one correspondence between the semantics of AFs and the strong semantics of StrAFs.

Definition 11. *Given* $AF = \langle \mathcal{A}, \mathcal{R} \rangle$*, 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* coval $= \sum$.

Proposition 1. *For* $AF = \langle \mathcal{A}, \mathcal{R} \rangle$*, and* $StrAF_{AF} = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *its associated StrAF, and* coval $= \sum$*,* $\sigma(AF) = \sigma_S(StrAF_{AF})$*, for* $\sigma \in \{\text{cf}, \text{ad}, \text{pr}, \text{st}\}$.

This result holds for any semantics based on conflict-freeness and defence, including those which are out of the scope of this paper. From the above, and the observation that for a *StrAF* associated with a Dung's theory AF, the notions of strong and weak conflict-freeness coincide, we easily obtain the following corollary.

Corollary 1. *For any AF, and* $StrAF_{AF}$ *its associated StrAF,* $\sigma(AF) = \sigma_W(StrAFS)$*, for* $\sigma \in \{\text{cf}, \text{ad}, \text{pr}, \text{st}\}$.

These two results are major tools for providing hardness results in the complexity study of StrAFs (Section 4).

4 Complexity and Algorithms

This section discusses the complexity of various reasoning problems for StrAFs as well as algorithms for solving them. In the rest of the paper, we assume that coval is tractable. Usual aggregation operators (like $\sum$, Π or max) are tractable. We suppose that the reader is familiar with the basic notions of complexity theory ; otherwise, we refer the reader to *e.g.* [6].

4.1 Complexity of Acyclic Frameworks

A $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ is *acyclic* if the relation $\mathcal{R}$ is acyclic. Recall that Dung's acyclic argumentation frameworks have exactly one stable extension. The following example shows that this is not the case for StrAFs under the strong stable semantics.

Example 4. *Let* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a StrAF with* $\mathcal{A} = \{a, b\}$*,* $\mathcal{R} = \{(a, b)\}$*,* $\mathcal{S}(a) = 1$*,* $\mathcal{S}(b) = 2$*. For* coval $= \sum$*, we have* $\text{st}_S^{\Sigma}(StrAF) = \emptyset$*, whereas* $\text{st}_W^{\Sigma}(StrAF) = \{\{a, b\}\}$.

In fact, we prove the existence of a weak stable extension for every acyclic *StrAF*, by providing Algorithm 1. Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and an argument $a \in \mathcal{A}$, the set of attackers of a, denoted by $\Gamma^-_{StrAF}(a)$, is defined as $\Gamma^-_{StrAF}(a) = \{b \mid b \in \mathcal{A} \text{ and } (b, a) \in \mathcal{R}\}$. When *StrAF* is clear from the context we write simply $\Gamma^-(a)$.

```
1 E = ∅;
2 while A ≠ ∅ do
3     E' = {a | a ∈ A s.t. Γ⁻(a) = ∅};
4     E = E ∪ E';
5     A' = {a | a ∈ A, and ∃κ ⊆ E s.t. κ ▷coval a};
6     A = A \ (E ∪ A');
7 end
8 return E;
```

Algorithm 1: Algorithm compute-acyclic-extension($\langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$, coval)

Proposition 2. *Algorithm 1* compute-acyclic-extension *always terminates. The set E returned by the algorithm is the unique weak stable extension of the input acyclic StrAF* $\langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *w.r.t.* coval.

Algorithm 1 is a polynomial time procedure when the set A' can be computed in polynomial time.

Proposition 3. *Let* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be an acyclic StrAF and* coval *an aggregation operator. Computing a weak stable extension of StrAF is polynomial.*

We notice that Algorithm 1 corresponds to the well-known algorithm for computing the grounded extension in

Dung's theory. This is not surprising, since the stable semantics coincide with the single-status grounded semantics for acyclic graphs (as well as any reasonable semantics).

Although we know that strong stable extensions may not exist for acyclic graphs, we show that deciding whether they exist or not is tractable.

Proposition 4. *Let* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be an acyclic StrAF and* coval *an aggregation operator. Deciding whether StrAF has a strong stable extension is polynomial.*

Remark : Since there is a single weak stable extension, and computing this extension is polynomial (Proposition 3), both skeptical and credulous acceptance coincide for weak stable semantics, and are polynomial reasoning tasks. Similarly, Proposition 4 shows that, if a strong stable extension exists for an acyclic StrAF then it can be polynomially computed by Algorithm 1, and thus credulous (and skeptical) acceptance are polynomial as well.

4.2 Complexity of Symmetric Frameworks

Now we consider the case of symmetric StrAFs, *i.e.* frameworks for which $(a, b) \in \mathcal{R}$ iff $(b, a) \in \mathcal{R}$. This corresponds to situations like the one depicted in Example 2 with John's gardens problem. We prove here that, in spite of the specific structure of this subclass of StrAFs, reasoning is at the first level of the polynomial hierarchy (*i.e.* the same complexity as general StrAFs, as shown in Section 4.3).

First, we prove that strong and weak semantics coincide for symmetric *StrAFs*.

Proposition 5. *Let* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a symmetric StrAF and* coval *an aggregation operator.* $\mathrm{cf}_S^{\mathrm{coval}}(StrAF) = \mathrm{cf}_W^{\mathrm{coval}}(StrAF)$.

Proposition 5 implies that $\sigma_S^{\mathrm{coval}}(StrAF) = \sigma_W^{\mathrm{coval}}(StrAF)$, for $\sigma \in \{\mathrm{ad}, \mathrm{pr}, \mathrm{st}\}$, for any symmetric *StrAF* and any coval.

Now we study credulous and skeptical reasoning; we show that deciding whether an argument belongs to some (resp. each) strong (or weak) extension is NP-complete (resp. coNP-complete).

Proposition 6. *For* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *a symmetric StrAF,* coval *an aggregation function,* $a \in \mathcal{A}$ *and* $X \in \{S, W\}$,

- *deciding whether* $\exists E \in \mathrm{st}_X(StrAF)$ *s.t.* $a \in E$ *is* NP*-complete.*
- *deciding whether* $\forall E \in \mathrm{st}_X(StrAF)$, $a \in E$ *is* coNP*-complete.*

Proposition 7. *Given* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *a symmetric StrAF and* coval *an aggregation function, checking whether* $\mathrm{st}_X^{\mathrm{coval}}(StrAF) \neq \emptyset$ *is* NP*-complete, with* $X \in \{S, W\}$.

While verifying the existence of at least one (strong or weak) stable extension is generally hard, even for symmetric StrAFs, we show that this problem is trivial in the case of irreflexive symmetric StrAFs, *i.e.* symmetric StrAFs where no self-attack appear. Indeed, Algorithm 2 is a simple (non-deterministic) polynomial time procedure that returns a strong stable extension of the irreflexive symmetric StrAF given as input. This algorithm proves the existence of strong, and therefore weak as well, stable extensions for every irreflexive symmetric StrAF.

Proposition 8. *Algorithm 2* compute-symmetric-extension *always terminates. Any set E returned by the algorithm is a strong (and weak) stable extension of the irreflexive symmetric* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *w.r.t.* coval.

```
1 E = ∅;
2 while A ≠ ∅ do
3 |   Select a ∈ A s.t. ∄a' ∈ A with S(a') > S(a);
4 |   E = E ∪ {a};
5 |   A = A \ ({a} ∪ {b | b ∈ A, (a, b) ∈ R});
6 end
7 return E;
```

Algorithm 2: Algorithm compute-symmetric-extension($\langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$, coval)

4.3 General Complexity

Now, we study the complexity of reasoning with general *StrAFs*. We first prove that verifying whether a set of arguments is a (strong or weak) stable extension is polynomial.

Proposition 9. *Let* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *be a StrAF and* coval *an aggregation function. For* $E \subseteq \mathcal{A}$*, checking whether* $E \in \mathrm{st}_X^{\mathrm{coval}}(StrAF)$*, with* $X \in \{S, W\}$*, is polynomial.*

Computing stable extensions is a central problem in argumentation. Similar to the case of Dung's theories, this problem is intractable for StrAFs. However, its complexity does not increase for StrAFs.

Proposition 10. *Given* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ *and* coval*, checking whether* $\mathrm{st}_X^{\mathrm{coval}}(StrAF) \neq \emptyset$ *is* NP*-complete, with* $X \in \{S, W\}$.

Finally, we show that credulous and skeptical acceptance are at the first level of the polynomial hierarchy.

Proposition 11. *Given* $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$*,* coval *an aggregation function, and* $a \in \mathcal{A}$*,*

- *deciding whether* $\exists E \in \mathrm{st}_X(StrAF)$ *s.t.* $a \in E$ *is* NP*-complete.*
- *deciding whether* $\forall E \in \mathrm{st}_X(StrAF)$, $a \in E$ *is* coNP*-complete.*

Computing the stable extensions of a StrAF is a hard problem even when coval $= \sum$. The rest of this section presents an algorithm that finds the stable extensions of $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$, with coval $= \sum$, through

pseudo-Boolean constraint satisfaction. A pseudo-Boolean constraint is an inequality of the form $\sum_i w_i v_i \geq k$, where w_i and k are positive integers, and v_i is a Boolean variable. Nowadays, several systems can solve pseudo-Boolean constraint satisfaction problems (*e.g.* `SAT4J` [30], `OpenWBO` [34], `RoundingSat` [26]), many of which draw on the power of efficient SAT solving techniques.

Weak Semantics We start with weak stable semantics. Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and coval $= \sum$, we associate a pseudo-Boolean constraint satisfaction problem $CS_W(StrAF) = (X, C)$, defined as follows :

— $\forall a_i \in \mathcal{A}$, we define a Boolean variable $x_i \in X$;

— the set of constraints C is as follows :

(1) $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, a_2, \ldots, a_n\}$, add the constraint $\mathcal{S}(a_1)\times x_1 + \mathcal{S}(a_2)\times x_2 + \cdots + \mathcal{S}(a_n)\times x_n < \mathcal{S}(a) \times x + (1 - x) \times M^a$, where x is the variable that corresponds to a, and M^a an integer that is explained below.

(2) $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, a_2, \ldots, a_n\}$, add the constraint $\mathcal{S}(a_1) \times x_1 + \mathcal{S}(a_2) \times x_2 + \cdots + \mathcal{S}(a_n) \times x_n \geq (1 - x) \times \mathcal{S}(a)$, where x is the variable that corresponds to a.

Intuitively, x_i is true means that the corresponding argument a_i belongs to an extension. Each solution of the problem corresponds then to one extension of the StrAF.

The value M^a associated with $a \in \mathcal{A}$ in constraint (1) is an integer s.t. $M^a > \sum_{a_i \in \Gamma^-(a)} \mathcal{S}(a_i)$. The aim of M^a is to neutralize constraint (1) when x is false. Indeed, depending on the value of x, (1) can be understood as follows :

(1.1) if $x = 0$, the constraint becomes $\sum_{a_i \in \Gamma^-(a)} \mathcal{S}(a_i) \times x_i < M^a$, which is satisfied regardless of the values assigned to the variables x_i, since $\sum_{a_i \in \Gamma^-(a)} \mathcal{S}(a_i) \times x_i \leq \sum_{a_i \in \Gamma^-(a)} \mathcal{S}(a_i)$ and by construction $\sum_{a_i \in \Gamma^-(a)} \mathcal{S}(a_i) < M^a$. Therefore, in this case constraint (1) is inactive.

(1.2) if $x = 1$, the constraint becomes $\sum_{a_i \in \Gamma^-(a)} \mathcal{S}(a_i) \times x_i < \mathcal{S}(a)$, requiring that $\text{coval}(\Gamma^-(a)) < \mathcal{S}(a)$.

In other words, constraint (1) means that we can accept a only if it is stronger than its accepted attackers ; roughly speaking, these attackers form an accrual that attacks a but cannot defeat it even collectively.

Constraint (2) can be understood as follows :

(2.1) if $x = 0$, the constraint becomes $\mathcal{S}(a_1) \times x_1 + \mathcal{S}(a_2) \times x_2 + \cdots + \mathcal{S}(a_n) \times x_n \geq \mathcal{S}(a)$, *i.e.* $\text{coval}(\Gamma^-(a)) \geq \mathcal{S}(a)$.

(2.2) if $x = 1$, the constraint becomes $\mathcal{S}(a_1) \times x_1 + \mathcal{S}(a_2) \times x_2 + \cdots + \mathcal{S}(a_n) \times x_n \geq 0$, that is trivially satisfied ; the constraint is inactive in this case.

Intuitively, this constraint can be interpreted as the fact that we can reject a only if its accepted attackers can join their respective strength to collectively overtake the strength of a. Roughly speaking, this means that there exists an accrual that defeats a. Note that, when a has no attacker, constraint (2) becomes $0 \geq (1 - x) \times \mathcal{S}(a)$, that can be rewritten into $x \geq 1$. This means that non-attacked arguments must be accepted.

The next proposition shows a direct correspondence between weak stable extensions of *StrAF* and the solutions of the problem $CS_W(StrAF)$. For $E \subseteq \mathcal{A}$, the Boolean assignment ω_E is defined, $\forall x_i \in X$, by $\omega_E(x_i) = 1$ iff $a_i \in E$.

Proposition 12. *A set $E \subseteq \mathcal{A}$ is a weak stable extension of $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ w.r.t. $\sum$ iff ω_E is a solution for $CS_W(StrAF)$.*

Example 5. *Consider $StrAF_{weak} = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ described at Figure 3a. The numerical values close to the nodes represent the strength of arguments* i.e. *the function $\mathcal{S}(\cdot)$. We observe that this StrAF does not admit a strong stable extension, but weak stable extensions can be computed as follows. If we set M^a to 5 in all the constraints for simplicity, we obtain the following set of constraints :*

$$
\begin{aligned}
&3 \times x_3 < 1 \times x_1 + (1 - x_1) \times 5\\
&1 \times x_1 < 2 \times x_2 + (1 - x_2) \times 5\\
&1 \times x_1 + 2 \times x_2 < 3 \times x_3 + (1 - x_3) \times 5\\
&2 \times x_2 < 2 \times x_4 + (1 - x_4) \times 5\\
&3 \times x_3 \geq (1 - x_1) \times 1\\
&1 \times x_1 \geq (1 - x_2) \times 2\\
&1 \times x_1 + 2 \times x_2 \geq (1 - x_3) \times 3\\
&2 \times x_2 + \geq (1 - x_4) \times 2
\end{aligned}
$$

The above set of constraints has the solutions $S_1 = \{x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0\}$, and $S_2 = \{x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0\}$, thus $\text{st}^{\Sigma}_W(StrAF) = \{\{a_1, a_2\}, \{a_2, a_3\}\}$.

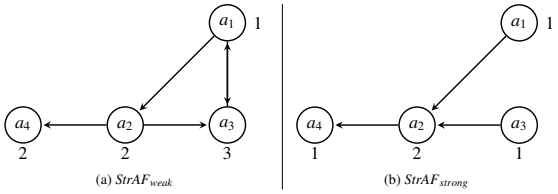


FIGURE 3 – Two Examples of *StrAFs*

Strong Semantics For strong stable semantics, we use the encoding for weak stable semantics as a starting point. We only need to add a constraint for enforcing the strong conflict-freeness. Given $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ and coval $= \sum$, we associate a pseudo-boolean constraint satisfaction problem $CS_S(StrAF) = (X, C)$ as follows :

— $\forall a_i \in \mathcal{A}$, we define a Boolean variable $x_i \in X$;

— the set of constraints C is as follows :

(1) $\forall a \in \mathcal{A}$ with $\Gamma^-(a) = \{a_1, a_2, \ldots, a_n\}$, add the constraint $\mathcal{S}(a_1)\times x_1 + \mathcal{S}(a_2)\times x_2 + \cdots + \mathcal{S}(a_n)\times x_n < \mathcal{S}(a) \times x + (1 - x) \times M^a$, where x corresponds to a, and M^a an integer as explained previously.

(2) $\forall a \in \mathcal{A}$ with $\Gamma^{-}(a) = \{a_1, a_2, \ldots, a_n\}$, add the constraint $\mathcal{S}(a_1)\times x_1 + \mathcal{S}(a_2)\times x_2 + \cdots + \mathcal{S}(a_n)\times x_n \geq (1 - x) \times \mathcal{S}(a)$, where x corresponds to a.

(3) $\forall a_i, a_j \in \mathcal{A}$ s.t. $(a_i, a_j) \in \mathcal{R}$, add $x_i + x_j \leq 1$.

Constraints (1) and (2) have already been explained. Constraint (3) means that two conflicting arguments cannot be jointly accepted : this is (strong) conflict-freeness.

The next proposition shows that there is a direct correspondence between strong stable extensions of *StrAF* and the solutions of the problem $CS_S(StrAF)$.

Proposition 13. *A set $E \subseteq \mathcal{A}$ is strong stable extension of $StrAF = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ w.r.t. $\sum$ iff ω_E is a solution of $CS_S(StrAF)$.*

Example 6. *Consider $StrAF_{strong} = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle$ described at Figure 3b. We observe that if we use the classical defeat-based semantics, neither a_1 nor a_3 can defeat a_2, while our semantics sensitive to accruals give a different result. The computation of strong stable extensions is made as follows. We translate the StrAF into a set of pseudo-Boolean constraints.*

$$
\begin{array}{c}
0 < 1 \times x_1 + (1 - x_1) \times 5 \\
1 \times x_1 + 1 \times x_3 < 2 \times x_2 + (1 - x_2) \times 5 \\
0 < 1 \times x_3 + (1 - x_3) \times 5 \\
2 \times x_2 < 1 \times x_4 + (1 - x_4) \times 5
\end{array}
$$

$$
\begin{array}{c|l}
0 \geq (1 - x_1) \times 1 & x_1 + x_2 \leq 1 \\
1 \times x_1 + 1 \times x_3 \geq (1 - x_2) \times 2 & x_3 + x_2 \leq 1 \\
0 \geq (1 - x_3) \times 1 & x_2 + x_4 \leq 1 \\
2 \times x_2 \geq (1 - x_4) \times 1 &
\end{array}
$$

The three groups correspond resp. to constraints (1), (2) and (3). The only solution of this set of constraints is $S = \{x_1 = x_3 = x_4 = 1, x_2 = 0\}$, that corresponds to the unique strong stable extension $E = \{a_1, a_3, a_4\}$.

We notice that $StrAF_{weak}$ from Example 5 does not have any strong stable extension. Indeed, none of the solutions S_1 and S_2 that we exhibited previously satisfies the strong conflict-freeness constraint.

5 Related Work

Several approaches have been proposed in the AI literature to model the accrual of arguments. Especially, in structured argumentation setting, [39, 32, 33, 28, 40] deal with accruals by adding a new argument that represents the accrual ; these works consider that the arguments members of an accrual should not be taken into consideration on their own. On the other hand, [41] defines a formalism in which accruals are explicitly given, since conflicts are defined in terms of sets of arguments attacking other (sets of) arguments. The notions introduced in these works strongly rely on the internal structure of the arguments, whereas in abstract argumentation, which is the subject matter of our study, this structure is unknown.

Collective attacks have also been studied in abstract argumentation. In [37] the authors proposed a generalization of Dung's abstract framework by presenting an abstract attack relation between sets of arguments and by extending the associated semantics. Similarly, in [19], *n*-ary conflicts are defined between sets of arguments that cannot be jointly accepted, while no explicit conflicts exist between these arguments. Besides, [17] defines collective argumentation frameworks where sets of arguments attack sets of arguments. In [27, 42], combined attacks (*i.e.* several arguments attacking a same target) are modelled through notions of joint attack or accrual patterns. In these works there is no notion of arguments strength (and thus, defeat relation) and then compensation as we do here.

Another framework representing synergies of arguments is proposed in [29]. This framework proposes an extension of the value-based argumentation framework [10] by defining a defeat relation with varied strength. The strength of defeat of a subset A of arguments over another subset B of arguments depends then on the values promoted by A and B. This framework is also different to ours as in our framework the strength is associated with arguments and the collective strength of an accrual is computed through an aggregation operator that aggregates according to different ways the strengths of the individual arguments forming the accrual. Then this collective strength is used in the definition of a collective defeat relation that is used between subsets of arguments.

We have mentioned previously that several works attach a weight to arguments (see *e.g.* [31, 9, 2, 8, 36, 38]). The meaning of these weights can be the votes of users, some notion of trust, or the certainty about the arguments premises. However, all these works strongly differ from ours since they intend to define a ranking or an acceptance degree of arguments. They focus on the individual acceptance of arguments, while the extension-based semantics that we define in this paper correspond to a notion of joint acceptance. Moreover, these works do not study collective defeat and accrual of arguments.

6 Conclusion and Future Work

In this paper, we have proposed an intuitive framework for representing strength of arguments called Strength-based Argumentation Framework, which allows strength compensations in an argumentation context where attacks may not succeed, completed by an approach which detects accruals throughout the reasoning process without requiring the elicitation of all compensatory combinations of arguments as an input. We use the numerical strength of arguments to define extension-based semantics based on a defeat relation, that is the combination of the initial attacks between arguments and the comparison of their respective strengths. We have shown how quantitative strengths can

be aggregated when several arguments attack a same target, making an *accrual of arguments*. The definition of collective defeat has allowed us to define new semantics that lead to interesting extensions that cannot be obtained when the usual Dung's AFs or individual defeat are used.

We have established the complexity of several reasoning problems for special cases (acyclic and symmetric graphcs) as well as general StrAFS. Surprisingly, we notice that, while our framework can model situations that are not captured by Dung's AFs without a blow up of the framework size, the complexity of reasoning does not increase. Table 1 summarizes our results for (strong and weak) stable semantics. P stands for polynomial, while X-c means X-complete where X is either NP or coNP. We say that a problem is trivial when the answer is straightforwardly "yes". We recall that all these results hold under the assumption that coval can be computed in polynomial time.

A technical report available online discusses in details several issues related to StrAFs, and provides full proofs. [1]

	Verif.	Exist.	Cred.	Skep.
Acyc. Strong	P	P	P	P
Acyc. Weak	P	Trivial	P	P
Sym. Strong	P	NP-c	NP-c	coNP-c
Sym. Weak	P	NP-c	NP-c	coNP-c
Gen. Strong	P	NP-c	NP-c	coNP-c
Gen. Weak	P	NP-c	NP-c	coNP-c

TABLE 1 – Complexity for (strong/weak) stable semantics

Our complexity results only concern the (strong and weak) stable semantics. For the case of weak semantics, they remind the complexity results for AFs with collective attacks [25]. We plan to deepen this investigation and consider other semantics as well, especially for the strong versions since the weak versions can be deduced from [25]. Moreover, regarding symmetric StrAFs, we have proven that extension existence is a trivial problem if we only consider irreflexive attack relations, *i.e.* no self-attacking arguments belong to the StrAF. It is known that reasoning with irreflexive symmetric AFs is polynomial [20]. We will investigate this question for irreflexive symmetric StrAFs, and determine whether self-attacks are the source of NP (or coNP) hardness for symmetric StrAFs, as they are in the case of symmetric AFs.

Weak extensions make sense for applications where conflicts are expected between pieces of information, but these pieces of information appear to be not strong enough to actually exclude each other. In other words, evaluation of these conflicts does not lead to reject some of the involved pieces of information which thus can be be accepted together. However, in the case of logic-based argumentation, the loss of (strong) conflict-freeness may lead to problems with the inference defined from the argumentation framework if two arguments belong to the same extension while one of them undercuts the other one. We plan to refine the notion of weak conflict-freeness according to this idea, similarly to the refinement of Preference-based AFs [5].

Finally, we plan to study the logical properties of our accrual-based semantics w.r.t. the properties of coval operators. Different properties may be suited to different applications scenarios, thus indicating which coval operators to choose depending on the application domain or the nature of arguments.

1. https://frama.link/SduY2A0E

Références

[1] Amgoud, L., J. Ben-Naim, D. Doder et S. Vesic: *Ranking Arguments With Compensation-Based Semantics*. Dans *KR'16*, pages 12–21, 2016.

[2] Amgoud, L., J. Ben-Naim, D. Doder et S. Vesic: *Acceptability Semantics for Weighted Argumentation Frameworks*. Dans *IJCAI'17*, pages 56–62, 2017.

[3] 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.

[4] Amgoud, L. et H. Prade: *Using arguments for making and explaining decisions*. Artif. Intell., 173(3-4) :413–436, 2009.

[5] Amgoud, L. et S. Vesic: *Rich preference-based argumentation frameworks*. Int. J. Approx. Reasoning, 55(2) :585–606, 2014.

[6] Arora, S. et B. Barak: *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009, ISBN 978-0-521-42426-4.

[7] Baroni, P., M. Caminada et M. Giacomin: *Abstract Argumentation Frameworks and Their Semantics*. Dans *Handbook of Formal Argumentation*, pages 159–236. College Publications, 2018.

[8] Baroni, P., A. Rago et F. Toni: *How Many Properties Do We Need for Gradual Argumentation ?* Dans *AAAI'18*, pages 1736–1743, 2018.

[9] Baroni, P., M. Romano, F. Toni, M. Aurisicchio et G. Bertanza: *Automatic evaluation of design alternatives with quantitative argumentation*. Arg. & Comp., 6(1) :24–49, 2015.

[10] Bench-Capon, T.: *Value-based Argumentation Frameworks*. Dans *NMR'02*, pages 443–454, 2002.

[11] Bench-Capon, T., H. Prakken et G. Sartor: *Argumentation in Legal Reasoning*. Dans *Argumentation in Artificial Intelligence*, pages 363–382. 2009.

[12] Benferhat, S., D. Dubois et H. Prade: *Argumentative inference in uncertain and inconsistent knowledge bases*. Dans *UAI '93*, pages 411–419, 1993.

[13] Besnard, P. et A. Hunter: *Elements of Argumentation*. MIT Press, 2008.

[14] Bistarelli, S. et F. Gadducci: *Enhancing Constraints Manipulation in Semiring-Based Formalisms*. Dans *ECAI'06*, pages 63–67, 2006.

[15] Bistarelli, S., D. Pirolandi et F. Santini: *Solving Weighted Argumentation Frameworks with Soft Constraints*. Dans *CILC'10*, 2010.

[16] Bistarelli, S., F. Rossi et F. Santini: *A novel weighted defence and its relaxation in abstract argumentation*. Int. J. Approx. Reasoning, 92 :66–86, 2018.

[17] Bochman, A.: *Collective Argumentation and Disjunctive Logic Programming*. J. Log. Comput., 13(3) :405–428, 2003.

[18] Bonzon, E., J. Delobelle, S. Konieczny et N. Maudet: *A Comparative Study of Ranking-Based Semantics for Abstract Argumentation*. Dans *AAAI'16*, pages 914–920, 2016.

[19] Caminada, M. et S. Vesic: *On extended conflict-freeness in argumentation*. Dans *BNAIC'12*, 2012.

[20] Coste-Marquis, S., C. Devred et P. Marquis: *Symmetric Argumentation Frameworks*. Dans *ECSQARU'05*, pages 317–328, 2005.

[21] Coste-Marquis, S., S. Konieczny, P. Marquis et M. A. Ouali: *Weighted Attacks in Argumentation Frameworks*. Dans *KR'12*, 2012.

[22] da Costa Pereira, C., A. Tettamanzi et S. Villata: *Changing One's Mind : Erase or Rewind ?* Dans *IJCAI'11*, pages 164–171, 2011.

[23] Dimopoulos, Y., J.-G. Mailly et P. Moraitis: *Argumentation-based Negotiation with Incomplete Opponent Profiles*. Dans *AAMAS'19*, pages 1252–1260, 2019.

[24] Dung, P. M.: *On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games*. Artificial Intelligence, 77 :321–357, 1995.

[25] Dvorák, W., A. Greßler et S. Woltran: *Evaluating SETAFs via Answer-Set Programming*. Dans *COMMA'18*, pages 10–21, 2018.

[26] Elffers, J. et J. Nordström: *Divide and Conquer : Towards Faster Pseudo-Boolean Solving.* Dans *IJCAI'18*, pages 1291–1299, 2018.

[27] Gabbay, D.: *Semantics for Higher Level Attacks in Extended Argumentation Frames Part 1 : Overview*. Studia Logica, 93(2-3) :357–381, 2009.

[28] Gordon, T.: *Defining argument weighing functions*. 5 :747–773, 2018.

[29] Kaci, S. et C. Labreuche: *Representing Synergy among Arguments with Choquet Integral*. Dans *ECSQARU'13*, pages 302–314, 2013.

[30] Le Berre, D. et A. Parrain: *The Sat4j library, release 2.2*. Journal on Satisfiability, Boolean Modeling and Computation, 7(2-3) :59–6, 2010.

[31] Leite, J. et J. Martins: *Social Abstract Argumentation*. Dans *IJCAI'11*, pages 2287–2292, 2011.

[32] Lucero, M., C. Chesñevar et G. Simari: *On the Accrual of Arguments in Defeasible Logic Programming*. Dans *IJCAI'09*, pages 804–809, 2009.

[33] Lucero, M., C. Chesñevar et G. Simari: *Modelling argument accrual with possibilistic uncertainty in a logic programming setting*. Inf. Sci., 228 :1–25, 2013.

[34] Martins, R., V. Manquinho et I. Lynce: *Open-WBO : A modular MaxSAT solver*. Dans *International Conference on Theory and Applications of Satisfiability Testing*, pages 438–445. Springer, 2014.

[35] McBurney, P., S. Parsons et I. Rahwan (rédacteurs): *the 8th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS'11)*, tome 7543 de *Lecture Notes in Computer Science*. Springer, 2012.

[36] Mossakowski, T. et F. Neuhaus: *Modular Semantics and Characteristics for Bipolar Weighted Argumentation Graphs*. CoRR, abs/1807.06685, 2018.

[37] Nielsen, S. et S. Parsons: *A Generalization of Dung's Abstract Framework for Argumentation : Arguing with Sets of Attacking Arguments*. Dans *ArgMAS'07*, tome 4766, pages 54–73, 2007.

[38] Potyka, N.: *Extending Modular Semantics for Bipolar Weighted Argumentation*. Dans *AAMAS'19*, pages 1722–1730, 2019.

[39] Prakken, H.: *A Study of Accrual of Arguments, with Applications to Evidential Reasoning*. Dans *ICAIL'05*, pages 85–94, 2005.

[40] Prakken, H.: *Modelling Accrual of Arguments in ASPIC+*. Dans *ICAIL'19*, pages 285–297, 2019.

[41] Verheij, B.: *Accrual of Arguments in Defeasible Argumentation*. Dans *2nd Dutch/German Workshop on Nonmonotonic Reasoning*, pages 217–224, 1995.

[42] Villata, S., G. Boella et L. van der Torre: *Argumentation Patterns*. Dans *ArgMAS'11*, pages 133–150, 2011.

Actes JIAF 2020

Logique Modale des Hypothèses, Systèmes Dynamiques Booléens et Réseaux de gênes

Pierre Siegel[1] **Vincent Risch**[1]
Syvain Sené[1] **Andrei Doncescu** [1]

[1] Université publique, France

Acknowledgement This work was mainly funded by our salaries as French State agents, affiliated to Université des Antilles, LAMIA, La Guadeloupe, France (AD) ; Université d'Aix-Marseille, Université de Toulon, CNRS, LIS, UMR 7020, Marseille, France (PS, VR, and SS) ; and secondarily by the project ANR-18-CE40-0002 FANs (SS).

Abstract

Les réseaux génétiques sont des systèmes biologiques qui représentent la façon dont les gènes ou les protéines interagissent dans une cellule. Ils sont particulièrement étudiés dans le cadre des réseaux d'automates et des systèmes dynamiques booléens (BDS). Cet article étudie une représentation des BDS qui utilise la *Logique modale des hypothèses*, notée $\mathcal{H}$. Dans le formalisme des BDS, un réseau génétique peut être représenté par un *graphe d'interaction* (IG), ou par un *graphe de transition* (TG). Chacune de ces représentations met l'accent sur des caractéristiques distinctes. La *dynamique* d'un BDS est caractérisée par une fonction f et un mode de mise à jour μ qui dit comment le BDS évolue dans le temps. Une partie importante des études réalisées sur les BDS s'est concentrée sur l'analyse de leurs configurations stables (ou points fixes) et de leurs cycles stables et instables. La représentation d'un BDS par la logique des défauts, les ASP, ou d'autres formalismes non monotones proches, permet de trouver les configurations stables. Cependant, ces représentations ne sont pas adaptées pour capturer les comportements dynamiques cycliques. Nous introduisons des représentations en $\mathcal{H}$ pour les IG et les TG asynchrones, qui conduit à de nouveaux résultats formels. Ces résultats permettent, en particulier, de discriminer entre les configurations stables, les cycles limites et les cycles instables. Un travail précédent a étudié en détail les IG. Cet article se concentre principalement sur les TG. Les *extensions fantômes* de $\mathcal{H}$, jouent ici un rôle clé.

1 Introduction

From a logical point of view, a biological system can be viewed as a set of interacting elements, let's say entities, whose states change over discrete time. Genetic networks are specific biological systems that represent how the genes (or proteins) of a cell interact with each other for the survival, reproduction, or death of this cell. The study of genetic networks is a source of relevant questions regarding knowledge representation. First, interactions appear as a form of causality. As such, we expect to model it thanks to logical inferences, but of which kind ? The use of classical logic is inadequate in this context because it cannot deal with inconsistencies, whereas what we learn on genetic networks arises often from long and expensive experiments and we know only a small part of the interactions while this knowledge can be revisable, uncertain, contradictory and even false. Moreover, algorithmic complexity is a crucial issue regarding the need to provide algorithms with reasonable calculation times in practice. These questions have been studied in artificial intelligence since the late 1970s, especially by the use of both particular nonmonotonic logics and techniques derived from constraint programming. Notably, default logic [22, 6] as well as answer set programming (ASP) [18] can be used here. For the representation of genetic networks with no cycle, results have been obtained with default logic [12, 11].

Genetic networks have been studied from the end 1960s in the context of automata networks and Boolean dynamical systems (BDSs) as a set of entities mapped to Boolean states. In this framework, it is considered that the expression of one gene modulates the expression of another gene by activation or inhibition. The use of BDSs leads to founding theorems on feedback circuits, simply called circuits hereafter, that create behavioral complexity and rich-

ness [9, 19, 23, 24, 25, 26, 28, 36].

This article deals with the representation of BDSs, using a non-monotonic modal logic called *hypothesis logic* ($\mathcal{H}$) [30, 32] defined in 1993, after a first approach proposed by [3]. In the context of BDSs, a genetic network can be represented by both an *interaction graph* (IG) and a *transition graph* (TG). The relationship between IGs and TGs have been studied since many years in dynamical systems theory, but remains an important open question. Our logic-based approach is a step toward a global clarification of this relationship. Preliminary results on circuits were given in [31, 33].

A *dynamics* of a BDS is characterized by a function f associated with an updating mode μ that organizes the entities updates over time (in this paper, we focus solely on the asynchronous one). Most of the studies done on BDSs have focused on their temporal asymptote, *i.e.*, on the analysis of both their stable configurations (or fixed points) and stable/unstable cycles. If the representation of a BDS by whatever default logic, ASP or other well known non-monotonic formalisms enables us to find fixed points, these representations are not suitable to capture cyclic dynamical behaviors. This is embarrassing because these cycles may represent real fundamental phenomena in living organisms such that the cell cycle [5, 17], the circadian cycle [1, 29], or the cardio-respiratory pace [10]. This possible lack of extensions in default logic has been fully studied in the context of hypothesis logic. As shown in [32, 30], default logic is a fragment of $\mathcal{H}$. In the latter logic, theories always have extensions among which some of them, called *ghost extensions*, have no counterpart in default logic. And it is these ghost extensions that enable $\mathcal{H}$ to discriminate between BDSs stable configurations, stable cycles and unstable cycles.

This article is structured as follows : Section 2 (resp. Section 3) gives the main definitions and notations related to $\mathcal{H}$ (resp. BDSs) ; in Section 4, we present the way to represent BDSs in the hypothesis logic framework and develop the main results of the work showing that the asynchronous asymptotic behaviors such as stable configurations and stable cycles, as well as unstable cycles, are properly captured ; Section 5 gives a brief conclusion on the work led.

2 Hypothesis Logic

Syntax Hypothesis logic $\mathcal{H}$ [30, 32] is a bi-modal logic [4] with two modal operators L and [H]. If f is a formula, the intuitive meaning of Lf is *f is proved/stated*. The dual H of [H] is defined as H$f = \neg[\mathrm{H}]\neg f$. The intuitive meaning of Hf is *f is a hypothesis*, and hence [H]f means *$\neg f$ is not a hypothesis*. The language of $\mathcal{H}$, denoted by $\mathscr{L}(\mathcal{H})$, is defined by the following inductive rules :

- Any formula of first-order logic is in $\mathscr{L}(\mathcal{H})$.
- Whenever f and g are in $\mathscr{L}(\mathcal{H})$, $\neg f$, $(f \wedge g)$, $(f \vee g)$, $(f \rightarrow g)$, Lf, [H]f, Hf are in $\mathscr{L}(\mathcal{H})$ too.

And no other formulas are in $\mathscr{L}(\mathcal{H})$ than those formed by applying these two rules. Operator L has the properties of the modal system T and [H] has those of the modal system K. As a consequence, the inference rules and axiom schemata of $\mathcal{H}$ are :

- All inference rules and axiom schemata of first-order logic.
- (N[H]) : $\vdash f \implies \vdash [\mathrm{H}]f$, the *necessitation rule* for [H].
- (NL) : $\vdash f \implies \vdash \mathrm{L}f$, the *necessitation rule* for L.
- (K[H]) : $\vdash [\mathrm{H}](f \rightarrow g)) \rightarrow ([\mathrm{H}]f \rightarrow [\mathrm{H}]g)$, the *distribution axiom schema* for [H].
- (KL) : $\vdash \mathrm{L}(f \rightarrow g) \rightarrow (\mathrm{L}f \rightarrow \mathrm{L}g)$, the *distribution axiom schema* for L.
- (TL) : $\vdash \mathrm{L}f \rightarrow f$, the *reflexivity axiom schema* for L.

Unlike L, the axiom of reflexivity does not hold for [H]. There are so far no connections between L and [H]. We force this connection by adding the following *link axiom schema* :

- (LI) : $\vdash \neg(\mathrm{L}f \wedge H\neg f)$.

This very weak axiom is one of the bases of $\mathcal{H}$. It means that it is impossible to prove f and to assume the hypothesis $\neg f$ at the same time. Note the following equivalences : $\neg(\mathrm{L}f \wedge \mathrm{H}\neg f) \iff \mathrm{L}f \rightarrow \neg\mathrm{H}\neg f \iff \mathrm{H}\neg f \rightarrow \neg\mathrm{L}f$, where the second (resp. third) formula means that *if we prove f, we cannot assume the hypothesis $\neg f$* (resp. *if we assume the hypothesis $\neg f$, we cannot prove f*).

Semantics As shown in [30], $\mathcal{H}$ has a Kripke semantics with two accessibility relations, R[H] for [H], RL for L. R[H] is the relation of system K and RL is the relation of system T, hence reflexive. The relationship between the two relations, given by the link axiom, is RL $\subseteq$ R[H]. Proofs of completeness, correctness, and compactness for $\mathcal{H}$ are given in [30].

2.1 Hypothesis theories and extensions

As defined above, $\mathcal{H}$ is a monotonic logic. In order to deal with the *revisable* character of usual informations (here of biological nature), a notion of *extension* is added just as in default logic. However, contrary to the latter, three kinds of extensions are considered here, namely stable extensions, ghost extensions and sub-extensions.

Definition 1 *Given $\mathcal{H}$:* • *A* hypothesis theory *is a pair $\mathcal{T} = \{\mathrm{HY}, \mathrm{F}\}$, where* F *is a set of formulas of $\mathcal{H}$ and* HY *is a set of hypotheses.*

• *An* extension *E of $\mathcal{T}$ is a set $E = \mathrm{Th}(\mathrm{F} \cup \mathrm{HY}')$, such that* HY$'$ *is a maximal subset of* HY *consistent with* F.

• *A* sub-extension *E of $\mathcal{T}$ is a set $E = \mathrm{Th}(\mathrm{F} \cup \mathrm{HY}')$, such that* HY$'$ *is a non-maximal subset of* HY *consistent with* F.

• *E is a* stable extension *if it is an extension that satisfies the* coherence property *:* $\forall \mathrm{H}f,\ \neg\mathrm{H}f \in E \implies \mathrm{L}\neg f \in E$. *Thanks to the link axiom schema, we hence get :* $\forall f,\ \mathrm{L}\neg f \in E \iff \neg\mathrm{H}f \in E$.

• *E is a* ghost extension *if it is an extension that satisfies :* $\exists \mathrm{H}f,\ \neg\mathrm{H}f \in E$ *and* $\mathrm{L}\neg f \notin E$. *Hence for a ghost extension, we only get :* $\forall f,\ \mathrm{L}\neg f \in E \implies \neg\mathrm{H}f \in E$.

Definition 2 *Let E be an extension or a sub-extension :*

1. *A propositional variable* $i \in V$ *is* free *in E if* $\mathrm{L}i \notin E$ *and* $\mathrm{L}\neg i \notin E$. *It is* fixed *otherwise.*
2. *The* degree of freedom *of E, denoted by* degree(E) *is the number of free propositional variables that compose it.*

It is shown in [32, 30] that, given a hypothesis theory $\mathcal{T} = \{\mathrm{HY}, \mathrm{F}\}$, if F is consistent then $\mathcal{T}$ has at least one extension. Thus, an extension is obtained by adding one of the largest consistent sets of hypotheses to F while remaining consistent. Intuitively, *E* is stable if whenever it is forbidden to assume the hypothesis f, $\neg f$ is proven. It is a ghost extension otherwise. As shown in [30], stable extensions correspond to the standard extensions of default logic (and to stable models of ASP). Ghost extensions do not have any correspondence in default logic nor in ASP.

2.2 Example : representing genetic networks into $\mathcal{H}$

A *genetic network* represents interactions among genes or proteins in cell [2, 7, 8, 13, 15, 21]. From now one, let us consider that the entity at stake are proteins. In a modeling context, a protein is classically represented by an integer $i \in \{1, \ldots, n\}$. Its *concentration* in a cell is denoted by x_i. In such networks, given a protein i, a set of interactions (or influences) from a set of proteins toward i describes in which conditions the concentration of i evolves. In the most general case, a concentration x_i is a real number. Here, we study the particular case where the concentrations x_i are in $\{0, 1\}$. This simplification may seem crude at first glance, but it actually makes sense because, experimentally, it is almost impossible to get precise concentrations. Genetic networks can be studied with the formalism of *Boolean Dynamical Systems* (BDSs), defined in the following section. To introduce our example, it suffices to know that, for a BDS, the concentration $x_i = 1$ (resp. $x_i = 0$) denotes the presence (resp. the absence) of protein i in the cell. Moreover, to study a BDS from a logical point of view, we just consider the congruence $(x_i = 1) \equiv i$ (resp. $(x_i = 0) \equiv \neg i$), viewing i as a Boolean variable.

One of the interests of hypothesis logic is that this bimodal logic enables us to use three kinds of information : i, Li and Hi. Hence, by combining modalities with negations, we can use $\{i, \mathrm{H}i, \mathrm{H}\neg i, \mathrm{L}i, \mathrm{L}\neg i\}$. Remark that in $\mathcal{H}$, we have : $\mathrm{L}i \neq \neg\mathrm{L}\neg i$, $\neg\mathrm{L}i \neq \mathrm{L}\neg i$, $\mathrm{H}i \neq \neg\mathrm{H}\neg i$ and $\neg\mathrm{H}i \neq \mathrm{H}\neg i$. This increasing of expressiveness allows for a more precise representation of biological networks. Let us consider the genetic network of a cell, and i a protein. Using hypothesis logic, we state that :

• i means that *i is present in the cell and* $\neg i$ *means that it is absent.*

• Li means that *i is produced by the cell (i is being activated) and* $\neg$Li *means that i is not produced (i is not being activated).*

• L$\neg i$ means that *i is destroyed by the cell (i is being inhibited) and* $\neg$L$\neg i$ *means that i is not destroyed (i is not being inhibited).*

• Hi (resp. $\neg$Hi) means that *the cell gives (resp. does not give) the permission for attempting to produce i. In other words, the cell has (resp. has not) the ability to activate i.*

• H$\neg i$ (resp. $\neg$H$\neg i$) means that *the cell gives (resp. does not give) the permission for attempting to destroy i. In other words, the cell has (resp. has not) the ability to inhibit i.*

Regarding the use of $\mathcal{H}$ in this context, the role of an extension appears to gather a maximum of consistent permissions. Note that even if Hi stands for the cell giving permission to attempt the production of i, this production is not mandatory. It can be carried out or not, according to the context (*i.e.*, the set of all interactions in the cell). Similarly H$\neg i$ gives the authorization to destroy i. It is important to note that Li and L$\neg i$ are actually actions (production or destruction of a protein). So there is a difference between L$\neg i$ which says that i is destroyed, and $\neg$Li which says that i is not produced, and hence is weaker. There is a similar distinction between H$\neg i$ and $\neg$Hi (and between Li and $\neg$L$\neg i$; and between Hi and $\neg$H$\neg i$).

Proposition below give some general properties of $\mathcal{H}$, particularly adequate for the modeling of the different states of proteins in a cell.

Proposition 1 *Given i a protein, the following results hold in* $\mathcal{H}$ *:*

1. *(1)* $\mathrm{L}i \rightarrow i$ *and* $\mathrm{L}\neg i \rightarrow \neg i$ (i.e.*, if i is produced (resp. destroyed), then i is present (resp. absent)).*
2. *(2)* $\neg(\mathrm{L}i \wedge \mathrm{H}\neg i)$ *and* $\neg(\mathrm{L}\neg i \wedge \mathrm{H}i)$ *(*i.e.*, it is impossible to produce (resp. destroy) i and to give the permission to destroy (resp. produce) i it at the same time).*
3. *(3)* $\neg(\mathrm{L}i \wedge L\neg i)$ *(*i.e.*, it is impossible to produce and destroy i at the same time).*

Idea of the proof. Axioms of $\mathcal{H}$ are all what is needed. (1)are instances of axiom (T) ; (2) are instances of the linking axiom (LI) ; (3)derives directly from *(1)*.

3 Boolean dynamical systems

A finite *Boolean dynamical system* (BDS) describes the evolution of the interactions in a network of n entities numbered from 1 to n, over discrete time.

Consider $V = \{1, ..., n\}$ a set of n entities. A *configuration* $x = (x_1, \ldots, x_n)$ of the network is an assignment

of a truth value $x_i \in \{0,1\}$ to each element i of V. The set of all configurations (*i.e.*, all interpretations on the logic side), also called the *configuration space*, is denoted by $X = \{0,1\}^n$. A *dynamics* of such a network is modeled *via* both a function f, called the *global transition function*, and an *updating mode* μ that defines how the elements of V are updated over time. More formally, $f : X \to X$ is such that $x = (x_1, \ldots, x_n) \mapsto f(x) = (f_1(x), \ldots, f_n(x))$, where each function $f_i : X \to \{0,1\}$ is *a local transition function* that gives the evolution of the entity i over time. There exists an infinite number of updating modes [1] among which the *parallel* and the *asynchonous* ones remain the most used [14, 35]. The *parallel*, or *perfectly synchronous*, updating mode is such that all the entities of the network are updated at each time step. Conversely, the *asynchronous* updating mode is a non-deterministic variation in which only one entity is updated at a time. In the sequel, we restrict our study to asynchronous dynamics [23, 26].

3.1 Asynchronous transition graphs

Let $X = \{0,1\}^n$ be a space of configuration and $f : X \to X$ the global transition function of a BDS. The *asynchronous dynamics* of f is given by its *asynchronous transition graph* (ATG), $ATG(f)$=$(X, T(f))$, a digraph whose vertex set is the configuration space and arc set is the set of asynchronous transitions such that : $T(f) = \{(x,y) \in X^2 \mid x \neq y, x = (x_1, \ldots, x_i, \ldots, x_n), y = (x_1, \ldots, x_{i-1}, f_i(x), x_{i+1}, \ldots, x_n)\}$. Therefore, if $(x,y) \in T(f)$, x and y differ exactly by one element ; the transition is *unitary*.

Note 1 *By definition, we relate a unique ATG to a given function f. Moreover, by construction of ATGs, $ATG(f) = ATG(g)$ whenever f and g are equivalent (*i.e.*, $f(x) = g(x)$ for every x, or again f and g have the same truth tables). In other words, functions, truth tables and ATGs are isomorphic representations.*

An *orbit* in $ATG(f)$ is a sequence of configurations $(x^0, x^1, x^2, \ldots)$ such that either $(x^t, x^{t+1}) \in T(f)$ or $x^{t+1} = x^t$ if $x^t = f(x^t)$ (*i.e.*, x^t has no successors). A *cycle* of length r is a sequence of configurations $(x^1, \ldots, x^r, x^1)$ with $r \geqslant 2$ whose configurations $x^1, \ldots, x^r$ are all different. From this, we derive what is classically called an asynchronous attractor in dynamical systems. An *attractor* is terminal *strongly connected component* (SCC) of $ATG(f)$, *i.e.*, a SCC with no outward transitions. Among attractors, we distinguish stable configurations from stable cycles. A *stable configuration* is a trivial attractor, *i.e.*, a configuration x such that $\forall i \in V, x_i = f_i(x)$, which implies that $x = f(x)$. A *stable cycle* is a cyclic attractor such that, in $ATG(f)$, $\forall t < r$, x^{t+1} is the unique successor of x^t and x^1 is the unique successor of x^r. If an attractor is neither trivial nor cyclic, it is called a *stable oscillation*. When it is possible to get out from a SCC, this SCC is called an *unstable cycle* or an *unstable oscillation* depending on whether it is cyclic or not. An orbit that reaches a stable configuration stays there endefault logicessly. Similarly, when it reaches a stable cycle, it adopts endefault logicessly a stable oscillating behavior. Notice that in Figures, recurring configurations, *i.e.*, configurations belonging to an attractor, are pictured in gray, and cycles are represented by bold transitions.

1. Infinite, because deterministic updating modes are basically defined as infinite sequences of subsets of nodes of the network.

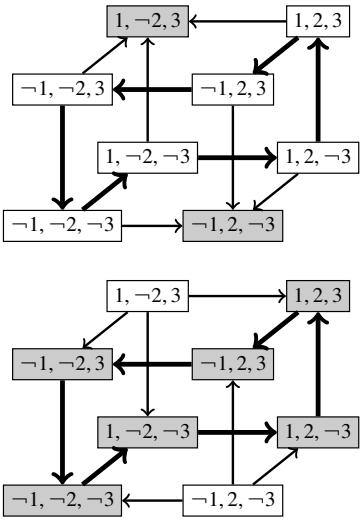


FIGURE 1 – *(top)* $ATG(f)$, and *(bottom)* $ATG(g)$ of fonctions f and g presented in Example 1.

Example 1 Boolean circuits of size 3 Consider $V = \{1,2,3\}$, $x = \{0,1\}^3$ and two functions f and g such that $f(x_1, x_2, x_3) = (\neg x_2, \neg x_3, x_1)$ and $g(x_1, x_2, x_3) = (\neg x_3, x_1, x_2)$. From the definitions of f and g, it is easy to derive their related $ATG(f)$ and $ATG(g)$, pictured in Figure 1. A transition corresponds to one arrow in the picture. There are up to 3 transitions leaving each configuration. Here, $ATG(f)$ has two symmetric stable configurations, $(\neg 1, 2, \neg 3)$ and $(1, \neg 2, 3)$ while all the other configurations belong to an unstable cycle ; $ATG(g)$ has a stable cycle, of length 6. This cycle is stable because there is only one transition leaving from each configuration, which is not the case for the unstable cycle of $ATG(f)$. We will see in Section 4 that the two stable configurations of $ATG(f)$ correspond to two stable extensions of $\mathcal{H}$, and that the stable cycle of $ATG(g)$ corresponds to a set of 6 ghost extension of degree 1. The functions f and g can also be represented by *elementary circuits*, pictured in Figure 3. These ones are special cases of *interaction graphs*, defined below.

Example 2 *Consider function $h(x_1, x_2)$=$(\neg x_1 \vee x_2, x_1 \vee$*

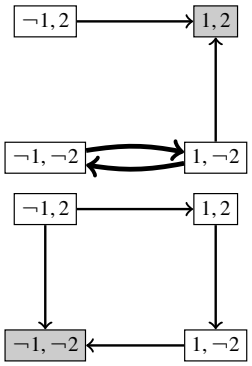


FIGURE 2 – *(top)* ATG of function h presented in Example 2, and *(bottom)* ATG of function k presented in Example 3.

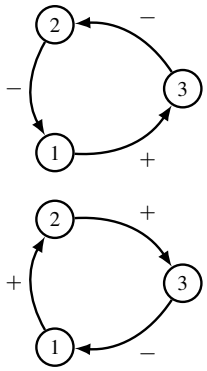


FIGURE 3 – *(top)* Positive circuit : IG associated with $ATG(f)$, and *(bottom)* Negative Circuit : IG associated with $ATG(g)$ of functions f and g introduced in Example 1.

x_2) *whose ATG is pictured on the top in Figure 2. This ATG has a stable state* $(1,2)$ *and an unstable cycle* $\{(\neg 1,2),(\neg 1,\neg 2)\}$. *There is an infinity of possible orbits because one can go follow the unstable cycle indefinitely, before attending* $(1,\neg 2)$ *and then stabilizing in* $(1,2)$.

Example 3 *Consider function* $k(x_1,x_2) = (x_2, x_1 \wedge \neg x_1 \wedge x_2)$, *whose ATG is pictured on the bottom in Figure 2. This ATG has a stable state* $\{\neg 1,\neg 2\}$ *and no cycles.*

3.2 Interaction graphs and circuits

A transition graph is a very precise tool for studying the behavior of a function, but its size is exponential depending on the number of entities. Regarding practical applications, the information is often represented by more compact and more readable graphs of a different type. It is indeed the case with biological data which come from experiments that generally simply yield correlations among gene expressions. For BDSs, it is common to use interaction graphs.

An *interaction graph (IG)* is a signed digraph $G = (V, I)$, where $V = \{1,\dots,n\}$ is the vertex set corresponding to the so called entities, and $I \subseteq V \times S \times V$, with $S = \{-,+\}$ is the arc set corresponding to the so called interactions. An arc $(i,+,j)$ (resp. $(i,-,j)$) is said to be *positive* (resp. *negative*). From a dynamical point of view, the presence of an arc (i,s,j) in an IG means that the value of i affects the value of j, positively or negatively according to s : we say that *i regulates j*.

A *circuit* $C = \{(i_1, s_{(1,2)}, i_2), \dots, (i_k, s_{(k,1)}, i_1)\}$ of size k is *elementary* if all the i_s that compose it are distinct. A circuit is *positive* (resp. *negative*) if it contains an even (resp. an odd) number of negative arcs.

Consider the toy example where j has only one incoming arc from i. In this case, the effect of the regulation is obvious : if the arc is positive (resp. negative), j will take the value (resp. the opposite value) of i after one update. Remark that elementary circuits are regulated this way.

Note 2 *Consider an IG that contains an arc* (i,s,i), i.e., *a loop on i. If* $s = +$ *(resp.* $s = -$*), this arc makes i tend to maintain (resp. negate) its state. It depends of course on whether i admits other in-neighbors than itself or not, and on the positive or negative influence of these eventual neighbors. In the case that i admits no other in-neighbors, it is trivial that i endefault logicessly maintains (resp. negate) its state if* $s = +$ *(resp. if* $s = -$*).*

As mentioned above, an IG $G = (V, I)$ represents the existence of the interactions involved between the entities of V. Specifying the nature of these interactions, and the conditions under which they occur effectively, leads to relate G to a BDS of function f. Then, G is the IG of f and is then denoted by $G(f) = (V, I(f))$. This is done by assigning a local transition function f_i to every $i \in V$ so that $\forall j \in V, \exists x \in \{0,1\}^n,\ f_i(x) \neq f_i(\overline{x}^j) \iff (j,s,i) \in I(f)$, where given $x = (x_1,\dots,x_n)$, $\overline{x}^j = (x_1,\dots,x_{j-1},\neg x_j, x_{j+1},\dots,x_n)$. This specification induces the minimality of $G(f)$ because each arc represents an effective interaction. It is also essential to note that if an IG corresponds to a single BDS/TG the converse is false. Contrary to TGs, an IG only provides static information about which entity acts which other and about the way it does.

Example 4 *Figure 3 pictures the IGs associated with the ATGs of the BDSs defined from f and g in Example 1. Consider the positive circuit associated with f by following the directions of its arcs :*

- *Starting from* x_1=1, *or simply* 1 *using* $\mathcal{H}$, *we get the infinite sequence* $(1,3,\neg 2,1,3,\neg 2,\dots)$.
- *Starting from* x_1=0, *or simply* $\neg 1$ *using* $\mathcal{H}$, *we get the infinite sequence* $(\neg 1,\neg 3,2,\neg 1,\neg 3,\dots)$.

The first (resp. second) dynamical behavior highlights the stable configurations $1, 2, \neg 3$ *(resp.* $\neg 1, \neg 2, 3$*) of* f*. For the negative circuit associated with* g *:*

• *Starting from* x_1=1*, or simply* 1*, we get the infinite sequence :* $(\neg 1, \neg 2, \neg 3, 1, 2, 3, \neg 1, \neg 2, \neg 3, 1, 2, 3, \dots)$.

• *Starting from* x_1=0*, or simply* $\neg 1$*, we get the infinite sequence :* $(1, 2, 3, \neg 1, \neg 2, \neg 3, 1, 2, 3, \neg 1, \neg 2, \neg 3, \dots)$.

In both cases, the observed dynamical behavior highlights the stable cycle of g*.*

3.3 General fundamental results and their biological direct applications

By considering that BDSs are good candidates for qualitatively modeling genetic networks (since established by the seminal papers [14, 35]), the presence of several attractors in their dynamical behaviors allows to model the cellular specialization. Indeed, if a genetic network controls a phenomenon of specialization, the cell will specialize (*i.e.*, will acquire a particular phenotype or a specific physiological function) according to the attractor towards which its underlying BDS evolves. A classical example of direct biological applications is the immunity control in bacteriophage λ, for which both lytic and lysogenic cycles of λ have been modeled in [34]. Another more tricky applications of BDSs in molecular systems biology concerns the floral morphogenesis of the plant *Arabidopsis thaliana* [20, 21]. Its dynamical behavior admits notably four stable configurations that correspond to the genetic expression patterns of the floral tissues, sepals, petal, stamens and carpels. This model has also allowed to formally explain the role of the hormone gibberellin on the floral development [8].

These works and the numerous other ones using BDSs or more general discrete dynamical systems (DDSs) emphasized the essential role of studies aiming at understanding the formal relations between IGs and TGs and their respective properties. They also clearly underlined the essential role of circuits, nowadays known as the behavioral complexity engines in dynamical systems. This comes in particular from Robert who established that, if the IG $G(f)$ of a DDS f is acyclic, then f converges towards a unique stable configuration [27, 28]. Moreover, in [36], basing himself on asynchronous DDS, Thomas conjectured that $G(f)$ of an asynchronous DDS f must contain a positive (resp. negative) circuit, for the latter to admit several stable configurations (resp. a non-trivial attractor such as a stable cycle or a stable oscillation). These two conjectures were proved to be true under the hypothesis of the asynchronous updating mode [23, 24, 25, 26].

Furthermore, in [23], the authors showed that an asynchronous positive (resp. negative) circuit of size n admits two attractors (resp. one attractor), namely two stable configurations x and its dual $\overline{x}$ (resp. a stable cycle of length $2n$). In [31], we obtained these results *via* the translation of the BDS into $\mathcal{H}$.

4 Representing BDS into $\mathcal{H}$

In [31], we studied in detail a translation of both positive and negative circuits into $\mathcal{H}$, which seems to be a first step to us because of their essential role in the regulation of the cell. In the sequel, we extend this translation to any asynchronous BDS. Indeed, our previous approach left formulas of the type $(\mathrm{H}i \wedge \mathrm{H}j) \rightarrow \mathrm{L}k$ out of reach, whereas such formulas are essential for representing the notion of *binding* in genetic networks.

4.1 Syntax representation of BDS

Remind that an asynchronous BDS is characterized by a global transition function $f : X \rightarrow X$ such that $x = (x_1, \dots, x_n) \mapsto f(x) = (f_1(x), \dots, f_n(x))$, where each function $f_i : X \rightarrow \{0, 1\}$ is a local transition function. Also, remind that we consider that each x_i is a propositional variable i, that the assignment x_i is a Boolean value i or $\neg i$, and therefore that each f_i is a Boolean formula.

Definition 3

• *The translation of a local transition function* f_i *into* $\mathcal{H}$ *is given by a set* $TR(f_i)$ *containing two formulas :* $TR(f_i) = \{\mathrm{H}f_i(x) \rightarrow \mathrm{L}i$ *and* $\mathrm{H}\neg f_i(x) \rightarrow \mathrm{L}\neg i\}$.

• *The* translation of a global transition function $f : X \rightarrow X$ of a BDS in $\mathcal{H}$ *is the union of translations* $TR(f_i)$ *for all* $i \in \{1, \dots n\}$ *such that* $TR(f) = \bigcup_{i=1}^{n} TR(f_i(x))$.

From the correspondence given in Note 1, this translation is equivalently the translation obtained for $ATG(f)$ and the truth tables of f.

Example 5 *Consider* $V = \{1, 2, 3\}$*,* $X = \{0, 1\}^3$*, and the global transition function* f *defined as* $f(x_1, x_2, x_3) = (\neg x_2, \neg x_3, x_1)$ *of Example 1.*

• *Function* f_1 *is translated into* $\mathcal{H}$ *by* $TR(f_1) = \{\mathrm{H}2 \rightarrow \mathrm{L}\neg 1, \mathrm{H}\neg 2 \rightarrow \mathrm{L}1\}$.

• *Likewise* f_2 *is translated by* $TR(f_2)$=$\{\mathrm{H}3 \rightarrow \mathrm{L}\neg 2, \mathrm{H}\neg 3 \rightarrow \mathrm{L}2\}$*, and* f_3 *by* $TR(f_3) = \{\mathrm{H}1 \rightarrow \mathrm{L}3, \mathrm{H}\neg 1 \rightarrow \neg \mathrm{L}3\}$.

Therefore we obtain the following global translation : $TR(f)$=$\{\mathrm{H}2 \rightarrow \mathrm{L}\neg 1, \mathrm{H}\neg 2 \rightarrow \mathrm{L}1, \mathrm{H}3 \rightarrow \mathrm{L}\neg 2, \mathrm{H}\neg 3 \rightarrow \mathrm{L}2, \mathrm{H}1 \rightarrow \mathrm{L}3, \mathrm{H}\neg 1 \rightarrow \neg \mathrm{L}3\}$ *that admits two stable extensions :*

• $E1 = Th(TR(f) \cup \{\mathrm{H}1, \mathrm{H}\neg 2, \mathrm{H}3)\})$

• $E2 = Th(TR(f) \cup \{\mathrm{H}\neg 1, \mathrm{H}2, \mathrm{H}\neg 3)\})$ [2].

When developing these extensions, we see that they are equivalent to their simplified forms :

• $E1 = \{\mathrm{H}\neg 1, \mathrm{H}2, \mathrm{H}\neg 3, \mathrm{L}1, \mathrm{L}\neg 2, \mathrm{L}3, \neg \mathrm{H}1, \neg \mathrm{H}\neg 2, \neg \mathrm{H}3, \neg \mathrm{L}\neg 1, \neg \mathrm{L}2, \neg \mathrm{L}\neg 3\}$ *;*

• $E2 = \{\mathrm{H}1, \mathrm{H}\neg 2, \mathrm{H}3, \mathrm{L}\neg 1, \mathrm{L}2, \mathrm{L}\neg 3,$

2. This is shown by attempting to add to $\mathrm{F}(G(f))$ each subset of $\mathrm{HY}(G(f))$ and keeping only those among them that are the maximals ones consistent with $\mathrm{F}(G(f))$. This can be done very quickly using a SAT solver.

$\neg$H$\neg$1, $\neg$H2, $\neg$H$\neg$3, $\neg$L1, $\neg$L$\neg$2, $\neg$L3}.

In order to ease the reading and abusing notations, from now on in the text and in the figures, the extensions will contain only the Li and L$\neg i$ that are true. *So, here,* $E1 = \{$L1, L$\neg$2, L3$\}$ *and* $E2 = \{$L$\neg$1, L2, L$\neg$3$\}$.

We can check that $E1$ *and* $E2$ *are stable extensions (because for all* i, $\neg$H$i \in E1$ *(resp.* $E2$*)* $\implies$ $L\neg i \in E1$ *(resp.* $E2$*) and that* $E2$ *is the mirror of* $E1$.

Example 6 *Consider the global transition function* $g(x_1, x_2, x_3) = (\neg x_3, x_1, x_2)$ *of Example 1. The translation in* $\mathcal{H}$ *leads to the following set of formulas :* $F(G(g)) = \{$H1 $\rightarrow$ L2, H2 $\rightarrow$ L3, H3 $\rightarrow$ L$\neg$1, H$\neg$1 $\rightarrow$ L$\neg$2, H$\neg$2 $\rightarrow$ L$\neg$3, H$\neg$3 $\rightarrow$ L1$\}$. *This allows us to obtain the following* 6 *equivalent ghost extensions :* $E1 = \{$L2, L3$\}$, $E2 = \{$L$\neg$1, L3$\}$, $E3 = \{$L$\neg$1, L$\neg$2$\}$, $E4 = \{$L$\neg$2, L$\neg$3$\}$, $E5 = \{$L1, L$\neg$3$\}$, $E6 = \{$L1, L2$\}$. *The following statements hold :*

• $E1$, ..., $E6$ *are extensions because they are maximal consistent. They are ghost extensions because in each of them there is a* $\neg$Hi *(resp.* $\neg$H$\neg i$*) without* L$\neg i$ *(resp.* $\neg$Hi*).*

• *These extensions are of degree* 1.

• *In [31], we proved that there exists a permutation on the is that allows us to pass from* $E1$ *to* $E2$, ..., $E5$ *to* $E6$. *This permutation represents the stable cycle of* g.

Moreover, there are also two sub-extensions, $E7 = \{1, \neg 2, 3\}$ *and* $E8 = \{\neg 1, 2, \neg 3\}$ *that contain neither* Li *nor* L$\neg i$. *Hence all the is are free and their degree is* 3.

Example 7 *Consider function* k *such that* $k(x_1, x_2) = (x_2, x_1 \wedge \neg x_1 \wedge x_2)$, *whose ATG is pictured in Figure 2.*

• *Function* k_1 *is translated into* $\mathcal{H}$ *by the couple* $TR(k_1) = \{$H2 $\rightarrow$ L$\neg$1, H$\neg$2 $\rightarrow$ L1$\}$.

• *Function* k_2 *is translated by* $TR(k_2) = \{$H$(1 \wedge \neg 1 \wedge 2 \rightarrow$ L2), H$\neg(1 \wedge \neg 1 \wedge 2) \rightarrow$ L$\neg$2$\}$.

Since $\neg(1 \wedge \neg 1 \wedge 2) = \neg 1 \vee 1 \vee \neg 2$, *we finally obtain the following global translation into* $\mathcal{H}$ *for* k *:*

• $TR(k) = \{$H2 $\rightarrow$ L$\neg$1, H$\neg$2 $\rightarrow$ L1, H$(1 \wedge \neg 1 \wedge 2) \rightarrow$ L2), H$(\neg 1 \vee 1 \vee \neg 2) \rightarrow$ L$\neg$2$)\}$, *which admits 3 extensions :*

• *A stable extension* $E1 = Th(TR(k) \cup \{$H$\neg$1, H$\neg$2$\}) = \{$L$\neg$1, L$\neg$2$\}$ *;*

• *Two ghost extensions of degree* 1 *:* $E2 = Th(TR(k) \cup \{$H1, H$\neg$1$\}) = \{$L$\neg$2$\}$, *and* $E3 = Th(TR(k) \cup \{$H2$\}) = \{$L1$\}$.

Note 3 *Function* k *may appear naive, because* $x_1 \wedge \neg x_1 \wedge x_2 = \bot$, *which gives an equivalent translation* $TR(h) = \{$H2 $\rightarrow$ L$\neg$1, H$\neg$2 $\rightarrow$ L1, H$\bot \rightarrow$ L2, H$\top \rightarrow$ L$\neg$2$\}$. *However, one of the aims of this study is also to show that we can deal with functions of any kind, without the need of a pre-processing. The formalism of* $\mathcal{H}$ *and the algorithms, that can in particular use a SAT solver, implicitly make the expected simplifications.*

4.2 Semantic representation of ATGs into $\mathcal{H}$

This section studies the formal relationship between ATGs and their representation into $\mathcal{H}$. It uses *Kripke semantics* [16] such as defined for *normal modal logics* (*i.e.*, the logics that contain at least axiom (K)). We shortly remind here the bases needed for our developments. A *Kripke structure* is a digraph $K = (W, R)$ where the *universe* W is a set $\{w_1, \ldots, w_n\}$ of *worlds* and the *accessibility relation* $R \subseteq W \times W$ is a binary relation among worlds. When $w_j R\, w_k$, w_k is *accessible* from w_j. A *Kripke model* is obtained by assigning in every world a truth value to every proposition i. This makes possible to assign a truth value to all the formulas of the propositional calculus (PC). A world is then mapped to a logical interpretation and hence implicitly to a configuration of a BDS. Formulas other than those of PC are assigned to worlds with the following condition : for all f, Lf is true in a world w_k if and only if f is true in all accessible worlds from w_k. The different axioms that hold in different modal logics depend on the properties of the accessibility relations R. As well known, for the system K, R is any relation, while axiom (TL) holds if and only if R is reflexive. In the following, we give a morphism between ATGs and Kripke models for the modal system T, which allows us to exhibit a morphism from hypothesis theories to ATGs.

Let $V = \{1, \ldots, n\}$ be a set of entities, $X = \{0, 1\}^n$ be a configuration space, $f : X \rightarrow X$ be a function with its associated $ATG(f) = (X, T(f))$. Remind that $T(f)$ is a set of edges corresponding to transitions. We now look at an increased version of $ATG(f)$, namely $ATG^\star(f) = (X, T \cup \circlearrowleft)$ where $\circlearrowleft$ denotes the reflexivity such that (x, x) is a transition of $ATG^\star(f)$ for all $x \in \{0, 1\}^n$, as a Kripke structure whose universe is X and whose accessibility relation is $T(f) \cup \circlearrowleft$. If we consider that any configuration $x \in X$ is a world, we get a Kripke model with $T(f) \cup \circlearrowleft$ as accessibility relation. Therefore, we get an isomorphism between Kripke models and increased ATGs, from which it is trivial to obtain the related ATGs.

In order to obtain a morphism between hypothesis theories and ATGs, we define the concept of a *projection* of an extension or of a sub-extension.

Definition 4 *Consider a sub-extension, or an extension,* E *of* $\mathcal{H}$. *The* projection *of* E *on the system* T *is the set of formulas of* E *that do not contain the operator* H.

Now, if $\mathcal{T} = \{$HY, F$\}$ is an hypothesis theory, and P is the set of the projections of the extensions or of the sub-extensions of $\mathcal{T}$, we obtain a morphism from $\mathcal{T}$ to P, and therefore a morphism from hypothesis theories to Kripke models.

Note 4 *Note that one does not get an isomorphism. Indeed the projections of two different extensions can be equal, and therefore be related to the same Kripke model. Note*

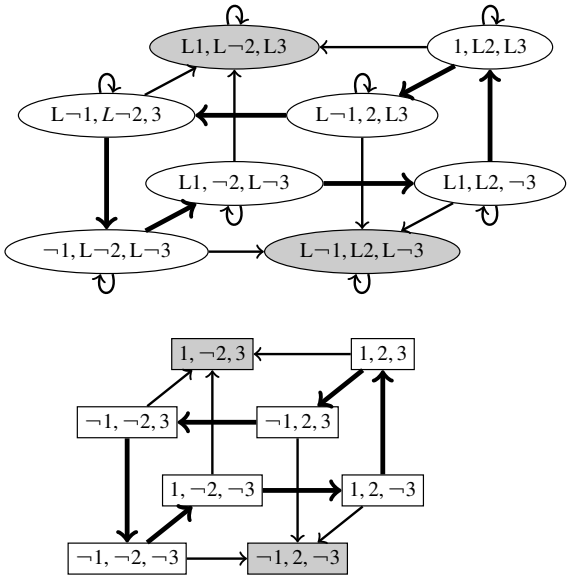


FIGURE 4 – *(top)* Kripke model of function f given in Example 1, and *(bottom)* its associated ATG.

also that, since every edge connects two configurations that differ by one entity in an ATG, two different accessible worlds of the Kripke model differ by one propositional variable.

Example 8 *On the top of Figure 4 is depicted the Kripke model associated with the projection of the ATG of function f given in Example 1. The eight nodes are the worlds, and the arcs express the accessibility relation. The two nodes in gray of the Kripke model,* $\{L1, L\neg 2, L3\}$ *and* $\{L\neg 1, L2, L\neg 3\}$ *represent the two stable extensions from the translation / projection of f. Their degree of freedom is* 0*. The other six nodes are sub-extensions since they are non-maximal. Their degree is* 2*. These six nodes form the unstable cycle of f, represented by bold transitions. We remark that the set of edges of the Kripke model contains the set of edges of the ATG. The missing edges are the loops related to the axiom of reflexivity. We also remark that in order to get the Kripke model from the ATG, it is enough to inject the modality* L *at the right places into the cube, according to the corresponding Kripke frame.*

Example 9 *On the top of Figure 5 is depicted the Krypke model (resp. the ATG) of function g given in Example 1. The six nodes* $\{1, L2, L3\}$*,* $\{L\neg 1, 2, L3\}$*,* $\{L\neg 1, L\neg 2, 3\}$*,* $\{\neg 1, L\neg 2, L\neg 3\}$*,* $\{L1, \neg 2, L\neg 3\}$*,* $\{L1, L2, \neg 3\}$ *represent the ghost extensions of the translation of g. Their degree of freedom is* 1*. The other two nodes* $\{1, \neg 2, 3\}$ *and* $\{\neg 1, 2, \neg 3\}$ *are sub-extensions because they are non-maximal; their degree of freedom is* 3*. The six extensions of the Kripke model correspond to the six configurations of the stable cycle of function g.*

Example 10 *As in the two former examples, Figure 6 depicts both the Kripke model and the ATG of function k given in Example 3. There are three extensions : a stable one* $\{L\neg 1, L\neg 2\}$*, and two ghost ones of degree* 1*, namely* $\{1, L\neg 2\}$ *and* $\{L1, 2\}$*. There is also one sub-extension of degree* 2*,* $(1, 2)$*.*

4.3 Results

Proposition 2 *Let $\mathcal{T}$ be an hypothesis theory, E be an extension or a sub-extension of $\mathcal{T}$, w be its projection and k be the degree of freedom of E. In the Kripke model associated to $\mathcal{T}$, there are exactly k distinct worlds, different from w, reachable from w.*

Proof 1 Since the degree of E is k, there are $\{i_1, .., i_k\}$ propositional variables, free in E. For every $i \in \{i_1, \ldots, i_k\}$, we have both $\neg Li \in E$ and $\neg L\neg i \in E$. Two cases are possible, either $i \in E$ or $\neg i \in E$. If $i \in E$, since $\neg Li \in E$, there exists a world w' accessible from w, and distinct from w, that contains $\neg i$. Regarding the second case, if $\neg i \in E$, since $\neg L\neg i \in E$, there exists a world w'' accessible from w, and distinct from w, that contains i. Therefore, for each $i \in \{i_1, \ldots, i_k\}$, there is a world accessible from w, and distinct from w, that contains the opposite of i. Because w is related to an asynchronous ATG, from Note 4, all these accessible worlds are distinct. Hence there are k distinct worlds reachable from w.

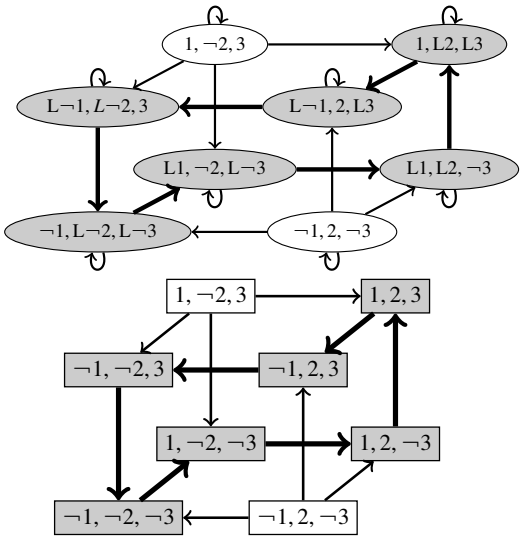


FIGURE 5 – *(top)* Kripke model of function g given in Example 1, and *(bottom)* its associated ATG.

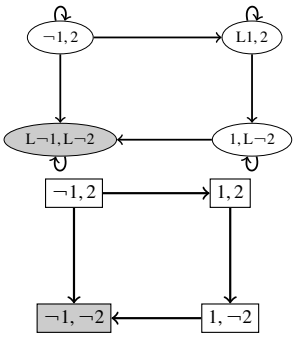


FIGURE 6 – *(top)* Kripke model of function k given in Example 7, and *(bottom)* its associated ATG.

Theorem 1 *Let f be a function, $ATG(f)$ and $TR(f)$ be its associated hypothesis theory. The following holds :*

1. *If $x = \{x_1, \ldots, x_n\}$ is a stable configuration of $ATG(f)$, then there exists an extension E of degree 0, issued from $TR(f)$, that contains $\{Lx_1, \ldots, Lx_n\}$.*
2. *Let E be an extension of degree 0, issued from $TR(f)$, and w the projection of E. If x is the configuration related to w, then x is stable.*

Proof 2 *Each statement is proved separately :*

1. *If x is a stable configuration of $ATG(f)$, no edges can leave from x. By construction of the Kripke model, the same holds for the Kripke world w related to x. Hence the only word accessible from w is w, that is, for any $i \in w$ (resp. $\neg i \in w$), $Li \in w$ (resp. $L\neg i \in w$). Therefore, every i is fixed and the degree of the extension E, issued from $TR(f)$, is 0.*
2. *Let the projection of E be represented by the world w. Since E is of degree 0, from Proposition 2, the only reachable world from w is w. By construction of the Kripke model, the same holds for x. Therefore x is a stable configuration of $ATG(f)$.*

Theorem 2 *Let $ATG(f)$ be the ATG of function f and $TR(f)$ be its associated hypothesis theory. Every stable cycle C of $ATG(f)$ corresponds to a cycle of extensions of degree 1 in $TR(f)$.*

Proof 3 *The proof is similar to that of Theorem 1. Let $C = \{x_1, \ldots, x_k\}$ be a stable cycle of $ATG(f)$, and $W = \{w_1, \ldots, w_\ell\}$ the set of extensions associated with C. By construction of the Kripke model, W is also a cycle of same length as C. Since C is stable, each of its configurations x_i admits only one outward arc. And the same property holds for for w_i,* i.e.*, the degree of w_i is 1. Therefore, all extensions of W are of degree 1.*

Analog theorems were proved in the context of [31]. They correspond to the results given in [23]. With the same arguments as those used for the proofs of the previous theorems, we can show that if a BDS contains an unstable cycle C, it is represented by a set of extensions such that at least one of those is of degree greater than 1. Indeed, if the cycle is unstable, it contains a configuration x of degree greater than 1 and, by construction, the Kripke model associated with the BDS contains the extension E corresponding to x.

5 Conclusion

In [31], we studied in detail a translation of both positive and negative asynchronous circuits into ($\mathcal{H}$). In this paper, we extend this translation to any asynchronous BDS, by showing that hypothesis logic capture some of their essential behavioural, such as stable configurations and stable cycles that are specific attractors and unstable cycles. Of course, these results pave the way to further studies about how hypothesis logic could enable to represent all the dynamical richness of BDSs, by taking for instance into account their stable and unstable oscillations and other known properties related to the orbits.

Our translation allows us to capture all Boolean representations of gene networks. In particular, we can write formulas like $(Hi \wedge Hj) \rightarrow Lk$. These formulas are essential to represent the notion of *binding* in genetic networks.

The lack of space prevents a serious discussion about algorithms in the context of this work. However our approach provides us with hints for simple algorithms that can distinguish stable/unstable states, and as such, provides ways for enumerating all the solutions in a practical way. Eventually, note that an extension is obtained by adding a consistent maximal set of hypotheses. Since it is possible to test whether consistency is preserved when adding each hypothesis, the computation of extensions is non-deterministic but constructive, contrary to default logic and ASP.

A implementation has been written in Prolog. It is based on a translation of ($\mathcal{H}$) into SAT. It allows to calculate, very quickly, all the extensions for small examples. The scaling must be done. This could be done using, very fast modern SAT solvers.

Références

[1] Akman, O. E., S. Watterson, A. Parton, N. Binns, A. J. Millar et P. Ghazal: *Digital clocks : simple Boolean models can quantitatively describe circadian systems*. Journal of The Royal Society Interface, 9 :2365–2382, 2012.

[2] Aracena, J., M. González, A. Zuñiga, M. A. Mendez et V. Cambiazo: *Regulatory network for cell shape changes during Drosophila ventral furrow formation*. Journal of Theoretical Biology, 239 :49–62, 2006.

[3] Besnard, P. et P. Siegel: *Supposition-Based Logic for Automated Nonmontonic Reasoning*. Dans *Proceedings of CADE'88*, tome 310 de *LNCS*, pages 592–601. Springer, 1988.

[4] Chellas, B.: *Modal logic : an introduction*. Cambridge University Press, 1980.

[5] Davidich, M. I. et S. Bornholdt: *Boolean network model predicts cell cycle sequence of fission yeast*. PLoS One, 3 :e1672, 2008.

[6] Delgrande, J. P. et T. Schaub: *Expressing default logic variants in default logic*. Journal of Logic and Computation, 15 :593–621, 2005.

[7] Demongeot, J., A. Elena, M. Noual, S. Sené et F. Thuderoz: *"Immunetworks", intersecting circuits and dynamics*. Journal of Theoretical Biology, 280 :19–33, 2011.

[8] Demongeot, J., E. Goles, M. Morvan, M. Noual et S. Sené: *Attraction basins as gauges of the robustness against boundary conditions in biological complex systems*. PLoS One, 5 :e11793, 2010.

[9] Demongeot, J., M. Noual et S. Sené: *Combinatorics of Boolean automata circuits dynamics*. Discrete Applied Mathematics, 160 :398–415, 2012.

[10] Dergacheva, O., K. J. Griffioen, R. A. Neff et D. Mendelowitz: *Respiratory modulation of premotor cardiac vagal neurons in the brainstem*. Respiratory Physiology & Neurobiology, 174 :102–110, 2010.

[11] Doncescu, A. et P. Siegel: *DNA double-strand break-based nonmonotonic logic*. Dans *Emerging Trends in Computational Biology, Bioinformatics, and Systems Biology*, pages 409–427. Elsevier, 2015.

[12] Doncescu, A., P. Siegel et T. Le: *Representation and efficient algorithms for the study of cell signaling pathways*. Dans *Proceedings of ICAI'14*, pages 504–510. IEEE Computer Society, 2014.

[13] Fauré, A., A. Naldi, C. Chaouyia et D. Thieffry: *Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle*. Bioinformatics, 22 :e124–e131, 2006.

[14] Kauffman, S. A.: *Metabolic stability and epigenesis in randomly constructed nets*. Journal of Theoretical Biology, 22 :437–467, 1969.

[15] Kauffman, S. A., C. Peterson, B. Samuelsson et C. Troein: *Random Boolean network models and the yeast transcriptional network*. PNAS, 100 :14796–14799, 2003.

[16] Kripke, S. A.: *Semantical analysis of modal logic I Normal modal propositional calculi*. Mathematical Logic Quarterly, 9 :67–96, 1963.

[17] Li, F., T. Long, Y. Lu, Q. Ouyang et C. Tang: *The yeast cell-cycle network is robustly designed*. PNAS, 101 :4781–4786, 2004.

[18] Lifschitz, V.: *Action languages, answer sets, and planning*. Dans *The Logic Programming Paradigm : A 25-Year Perspective*, pages 357–373. Springer, 1999.

[19] Melliti, T., M. Noual, D. Regnault, S. Sené et J. Sobieraj: *Asynchronous dynamics of Boolean automata double-cycles*. Dans *Proceedings of UCNC'15*, tome 9252 de *LNCS*, pages 250–262. Springer, 2015.

[20] Mendoza, L. et E. R. Alvarez-Buylla: *Dynamics of the genetic regulatory network for* Arabidopsis thaliana *flower morphogenesis*. Journal of Theoretical Biology, 193 :307–319, 1998.

[21] Mendoza, L., D. Thieffry et E. R. Alvarez-Buylla: *Genetic control of flower morphogenesis in* Arabidopsis thaliana. Bioinformatics, 15 :593–606, 1999.

[22] Reiter, R.: *A logic for default reasoning*. Artificial Intelligence, 13 :81–132, 1980.

[23] Remy, É., B. Mossé, C. Chaouyia et D. Thieffry: *A description of dynamical graphs associated to elementary regulatory circuits*. Bioinformatics, 19 :ii172–ii178, 2003.

[24] Remy, É., P. Ruet et D. Thieffry: *Graphic requirement for multistability and attractive cycles in a Boolean dynamical framework*. Advances in Applied Mathematics, 41 :335–350, 2008.

[25] Richard, A.: *Negative circuits and sustained oscillations in asynchronous automata networks*. Advances in Applied Mathematics, 44 :378–392, 2010.

[26] Richard, A. et J. P. Comet: *Necessary conditions for multistationarity in discrete dynamical systems*. Discrete Applied Mathematics, 155 :2403–2413, 2007.

[27] Robert, F.: *Itérations sur des ensembles finis et automates cellulaires contractants*. Linear Algebra and its Applications, 29 :393–412, 1980.

[28] Robert, F.: *Discrete Iterations : A Metric Study*. Springer, 1986.

[29] Roenneberg, T. et M. Merrow: *The network of time : understanding the molecular circadian system*. Current Biology, 13 :R198–R207, 2003.

[30] Schwind, C. et P. Siegel: *A modal logic for hypothesis theory*. Fundamenta Informaticae, 21 :89–101, 1994.

[31] Siegel, P., A. Doncescu, V. Risch et S. Sené: *Towards a Boolean dynamical system representation into a nonmonotonic modal logic*. Dans *Proceedings of NMR'18*, pages 53–62, 2018.

[32] Siegel, P. et C. Schwind: *Modal logic based theory for nonmonotonic reasonning*. Journal of Applied Non-classical Logic, 3 :73–92, 1993.

[33] Siegel, Pierre, Andrei Doncescu, Vincent Risch et Sylvain Sené: *Vers une representation des systèmes dynamiques booleens en logique des hypotheses*. Dans *Proceedings of JIAF'17*, July 2017.

[34] Thieffry, D. et R. Thomas: *Dynamical behaviour of biological regulatory networks – II. Immunity control in bacteriophage Lambda*. Bulletin of Mathematical Biology, 57 :277–297, 1995.

[35] Thomas, R.: *Boolean formalization of genetic control circuits*. Journal of Theoretical Biology, 42 :563–585, 1973.

[36] Thomas, R.: *On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations*. Dans *Numerical Methods in the Study of Critical Phenomena*, pages 180–193. Springer, 1981.

Actes JIAF 2020

An incomplete knowledge compilation map for conditional preference statements-based languages

Jérôme Mengin

IRIT - Université de Toulouse 3 - France
Jerome.Mengin@irit.fr

Résumé

Les assertions de préférences conditionnelles (CP-statements) permettent de représenter de manière compacte les préférences sur des domaines combinatoires. Elles sont au cœur des CP-nets et de leurs généralisations, et des arbres de préférences lexicographiques. Plusieurs travaux ont abordé la complexité de certaines requêtes liées à ces formalismes (optimisation, dominance en particulier). Cet article étend certains de ces résultats, et s'intéresse à d'autres requêtes (comme l'équivalence), contribuant ainsi à une carte de compilation pour les langages basés sur les assertions de préférences conditionnelles.

Abstract

Conditional preference statements have been used to compactly represent preferences over combinatorial domains. They are at the core of CP-nets and their generalizations, and lexicographic preference trees. Several works have addressed the complexity of some queries (optimization, dominance in particular). We extend in this paper some of these results, and study other queries which have not been addressed so far, like equivalence, thereby contributing to a knowledge compilation map for languages based on conditional preference statements.

1 Introduction

Preference handling is a key component in several areas of Artificial Intelligence, notably for decision-aid systems. Research in Artificial Intelligence has led to the development of several languages that enable compact representation of preferences over complex, combinatorial domains. Some preference models rank alternatives according to their values given by some multivariate function; this is the case for instance with valued constraints [26], additive utilities and their generalizations [24, 10]. Ordinal models like CP nets and their generalisations [5, 29, 8], or lexicographic preferences and their generalisations [21, 27, 30, 4, 11, 18] use sets of conditional preference statements to represent a pre-order over the set of alternatives.

Many problems of interest, like comparing alternatives or finding optimal alternatives, are at least NP-hard for some of these models, which makes these representations difficult to use in some decision-aid systems like configurators, where real-time interaction with a decision maker is needed. One approach to tackle this problem is Knowledge Compilation, whereby a model, or a part of it, is *compiled*, off-line, into another representation which enables fast query answering, even if the compiled representation has a much bigger size. This approach has first been studied in propositional logic: [14, 15] compare how various subsets of propositional logic can succinctly, or not, express some propositional knowledge bases, and the complexity of queries of interest. [13] follow a similar approach to compare extensions of propositional logic which associate real values to models of a knowledge base; [19] provide such a map for value function-based models.

The aim of this paper is to initiate such a compilation map for ordinal models of preferences. Specifically, we compare the expressiveness and succinctness of various languages based on conditional preference statements, and the complexity of several queries of interest for these languages.

The next section recalls some basic definitions about combinatorial domains and pre-orders, and introduces notations that will be used throughout. Section 3 gives an overview of various languages based on conditional preference statements that have been studied in the literature. Section 4 and 5 respectively study expressiveness and succinctness for languages based on conditional preference statements . Sections 6 study the complexity of queries for these languages. Proofs are omitted due to lack of space.

2 Preliminaries

2.1 Combinatorial domain

We consider languages that can be used to represent the preferences of a decision maker over a combinatorial space $\underline{\mathcal{X}}$: here $\mathcal{X}$ is a set of attributes that characterise the possible alternatives, each attribute $X \in \mathcal{X}$ having a finite set of possible values $\underline{X}$; then $\underline{\mathcal{X}}$ denotes the cartesian product of the domains of the attributes in $\mathcal{X}$, its elements are called alternatives. For binary attribute X, we will often denote by $x, \bar{x}$ its two possible values.

For a subset U of $\mathcal{X}$, we will denote by $\underline{U}$ the cartesian product of the domains of the attributes in U, called instantiations of U, or partial instantiations (of $\mathcal{X}$). If v is an instantiation of some $V \subseteq \mathcal{X}$, $v[U]$ denotes the restriction of v to the attributes in $V \cap U$; we say that instantiation $u \in \underline{U}$ and v are compatible if $v[U \cap V] = u[U \cap V]$; if $U \subseteq V$ and $v[U] = u$, we say that v extends u.

Sets of partial instantiations can often be conveniently, and compactly, specified with propositional formulas: the propositions are $X = x$ for every $X \in \mathcal{X}$ and $x \in \underline{X}$, and we use the standard connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence) and $\neg$ (negation). Implicitly, this propositional logic is equipped with a theory that enforces that every attribute has precisely one value from its domain; so, for two distinct values x, x' of attribute X, the formula $X = x \wedge X = x'$ is a contradiction; also, the interpretations are thus in one-to-one correspondence with $\underline{\mathcal{X}}$. If α is such a propositional formula over $\mathcal{X}$ and $o \in \underline{\mathcal{X}}$, we will write $o \models \alpha$ when o satisfies α, that is when, assigning to every literal $X = x$ that appears in α the value true if $o[X] = x$, and the value false otherwise, makes α true.

We assume that the domains of the attributes in $\mathcal{X}$ are disjoint, so that, given a formula α, or a partial instantiation u, we can unambiguously define $\mathsf{Var}(\alpha)$ and $\mathsf{Var}(u)$ as the set of variables, the values of which appear in α and u respectively.

When it is not ambiguous, we will use x as a shorthand for the literal $X = x$; also, for a conjunction of such literals, we will omit the $\wedge$ symbol, thus $X = x \wedge Y = \bar{y}$ for instance will be denoted $x\bar{y}$.

2.2 Preference relations

2.2.1 Preorders

Depending on the knowledge that we have about a decision maker's preferences, given any pair of distinct alternatives $o, o' \in \underline{\mathcal{X}}$, one of the following situations must hold: one may be strictly preferred over the other, or o and o' may be equally preferred, or o and o' may be incomparable.

Assuming that preferences are transitive, such a state of knowledge about the DM's preferences can be characterised by a preorder $\succeq$ over $\underline{\mathcal{X}}$: $\succeq$ is a binary, reflexive and transitive relation; for alternatives o, o', we then write $o \succeq o'$ when $(o, o') \in \succeq$; $o \succ o'$ when $(o, o') \in \succeq$ and $(o', o) \notin \succeq$; $o \sim o'$ when $(o, o') \in \succeq$ and $(o', o) \in \succeq$; $o \bowtie o'$ when $(o, o') \notin \succeq$ and $(o', o) \notin \succeq$. Note that for any pair of alternatives $o, o' \in \underline{\mathcal{X}}$ either $o \succ o'$, or $o' \succ o$, or $o \sim o'$ or $o \bowtie o'$. The relation $\sim$ defined in this way is the *symmetric part* of $\succeq$, it is reflexive and transitive, $\bowtie$ is irreflexive, they are both symmetric. The relation $\succ$ is the *irreflexive part* of $\succeq$, it is what is usually called a strict partial order: it is irreflexive and transitive.

Terminology and notations We say that alternative o *dominates* alternative o' (w.r.t. $\succeq$) if and only if $o \succeq o'$; if $o \succ o'$, then we say that o *strictly dominates* o'. We use standard notations for the complements of $\succ$ and $\succeq$: we write $o \not\succeq o'$ when it is not the case that $o \succeq o'$, and $o \not\succ o'$ when it is not the case that $o \succ$. We will denote by $\preceq$ (respectively $\prec$) the dual of $\succeq$ (resp. of $\succ$), also called its converse or transpose: $o \preceq o'$ if and only if $o' \succeq o$, and $o \prec o'$ iff $o' \succ o$. Note that since $\bowtie$ and $\sim$ are symmetric, they are equal to their dual.

Following [23] we say that alternative o is:

- weakly undominated if there is no $o' \in \underline{\mathcal{X}}$ such that $o' \succ o$;
- undominated if there is no $o' \in \underline{\mathcal{X}}$, $o' \neq o$, such that $o' \succeq_\varphi o$;
- dominating if for every $o' \in \underline{\mathcal{X}}$, $o \succeq_\varphi o'$;
- strongly dominating if for every $o' \in \underline{\mathcal{X}}$ with $o' \neq o$, $o \succ_\varphi o'$.

Note that o is strongly dominating if and only if it is dominating and undominated; and that if o is dominating or undominated, then it is weakly undominated.

2.2.2 Antisymmetric preorders

Some languages designed to represent preferences are associated with a primitive relation which is not a preorder but a strict partial order $\succ$. However, it is possible to define a preorder $\succeq$ as the reflexive closure of $\succ$: that is, $o \succeq o'$ holds when $o \succ o'$ or $o = o'$. The relation $\succeq$ defined in this way is antisymmetric.

3 Languages

3.1 Conditional preference statements

Let us call *conditional preference statement*, or CP statements, over $\mathcal{X}$ any expression of the form $\alpha \,|\, V : w \geq w'$, where α is a propositional formula over $U \subseteq \mathcal{X}$, $w, w' \in \underline{W}$, $w \neq w'$, and U, V, W are disjoint subsets of $\mathcal{X}$. Informally, such a statement represents the piece of knowledge that, when comparing alternatives o, o' that both satisfy α, the one that has values w for W is preferred to the one that has values w' for W, irrespective of the values of the

attributes in V, every other attribute being fixed. We call α the conditioning part of the statement; we call W the swapped attributes, and V the free part.

Conditional preference statements have been studied in many works. They are the basis for CP-nets [7, 5] and their extensions, and have been studied in a more logic-based fashion by e.g. [23] and [29, 28, 31]. In all these works, a syntactic restriction is put on W: it must be the case that $|W| = 1$. Also, [23] do not consider any free part ($V = \emptyset$), and [29, 28, 31] only considers statements with a conjunctive conditioning part (α must be a consistent conjunction of literals).[1]

The semantics of a set φ of conditional preference statements can be defined as follows: consider a pair of alternatives (o, o') such that there is a statement $\alpha \,|\, V : w \geq w' \in \varphi$ with $o[U] = o'[U] \models \alpha$, $o[W] = w$ and $o'[W] = w'$, and such that for every attribute $Y \notin U \cup V \cup W$ it holds that $o[Y] = o'[Y]$; following [31] we say that (o, o') is a *worsening swap*. We also say that the statement $\alpha \,|\, V : w \geq w' \in \varphi$ *sanctions* (o, o'). Let φ^* be the set of all worsening swaps that φ sanctions, and define $\succeq_\varphi$ to be the reflexive and transitive closure of φ^*. [31] proves that $o \succeq_\varphi o'$ if and only $o = o'$ or φ^* contains a finite sequence of worsening swaps $(o_i, o_{i+1})_{0 \leq i \leq k-1}$ with $o_0 = o$ and $o_k = o'$.[2]

The language of the above statements is very expressive: in fact, by considering a set $W = \mathcal{X}$, and $\alpha = \top$ and $V = \emptyset$, it is possible to represent any preorder "in extension" with preference statements of the form $o \geq o'$. Let us call:

- CP the language where formulas are sets of statements of the general form $\alpha \,|\, V : w \geq w'$;

This expressiveness has a cost: we will see that many queries about pre-orders represented by CP-statements are PSPACE-hard for the language CP. Several restrictions / sub-languages have been studied in the literature, we review them below. Note that formulas in CP are not required to verify any form of consistency or completeness; such conditions, will be imposed for some sublanguages defined below.

Notations We write $\alpha : w \geq w'$ when V is empty, and $w \geq w'$ when V is empty and $\alpha = \top$, the formula always true. Note that we reserve the symbol $\geq$ for conditional preference statements, whereas "curly" symbols $\succ$, $\not\succ$, $\succeq$, $\not\succeq$ are used to represent relations over the set of alternatives.

In the remainder of this section, we introduce various restrictions on formulas. Table 1 gives an overview of these restrictions, as well as some complexity results that will be detailed in section 6.

[1] The formula $u \,|\, V : x \geq x'$ is written $u : x > x'[V]$ by [31].

[2] Actually, [31] proves that (o, o') is in the transitive closure of φ^* if and only there is such a worsening sequence from o to o', but adding the reflexive closure to this transitive closure does not change the result, since we can add any pair (o, o) to, or remove it from, any sequence of worsening swaps without changing the validity of the sequence.

3.2 Statement-wise restrictions

Some restrictions apply on the syntactical form of statements allowed; they bear on the size of the set of free variables, or on the size of the set of swapped variables, or on the type of conditioning formulas allowed. Given some language $\mathcal{L} \subseteq \mathsf{CP}$, we define the following restrictions:

- $\mathcal{L}\not\emptyset$ is the restriction of $\mathcal{L}$ to formulas with empty free parts ($V = \emptyset$) for every statement;[3]
- $\mathcal{L}\wedge$ is the restriction of $\mathcal{L}$ to formulas where the condition α of every statement is a conjunction of literals;
- $\mathcal{L}\mathsf{k}$ is the restriction of $\mathcal{L}$ to formulas where the set of swapped attributes contains no more than k attributes ($|W| \leq k$) for every statement; in particular, we call elements of $\mathsf{CP1}$ *unary* statements.

In particular, $\mathsf{CP1}\wedge$ corresponds to the language studied by [31], and $\mathsf{CP1}\not\emptyset$ is the language of generalized CP-nets as defined by [23].

3.3 Graphical restrictions

CP-statements describe some interactions between attributes. Many tractability results on CP-statements based languages require that the graph of these interactions has some "good" properties. In their seminal work, [5] consider that these interaction can be elicited first from a decision-maker, and that this structure can be used to render easier the elicitation of $\mathsf{CP1}$ statements that represent her preferences. However, these interactions can also be extracted *a posteriori*, for any set of CP statements.

The graph defined by [5] is restricted to the case where all CP statements are unary and have no free variables. This definition has been extended by [8, 31] to cover the case of statements with free variables. The following definition is inspired by [31, Def. 15]. Given $\varphi \in \mathsf{CP}$ over set of attributes $\mathcal{X}$, we define the following graphs with sets of vertices $\mathcal{X}$: given $X, Y \in \mathcal{X}$

- $(X,Y) \in D_\varphi^{\mathsf{uncond}}$ if there is some statement $\alpha \,|\, V : w \geq w' \in \varphi$ such that $X \in \mathsf{Var}(\alpha)$ and $Y \in \mathsf{Var}(w) \cup V$;
- for every alternative o, $(X,Y) \in D_\varphi^{\mathsf{cond}}(o)$ if there is some statement $\alpha \,|\, V : w \geq w' \in \varphi$ such that $o \models \alpha$, $X \in \mathsf{Var}(w)$ and $Y \in V$;
- $D_\varphi = D_\varphi^{\mathsf{uncond}} \cup \bigcup_{o \in \underline{\mathcal{X}}} D_\varphi^{\mathsf{cond}}(o)$.

Given some language $\mathcal{L} \subseteq \mathsf{CP}$, we define:

- $\mathcal{L}\not\circlearrowleft$ the restriction of $\mathcal{L}$ to *acyclic* formulas, which are those φ such that D_φ is acyclic;[4];

[3] In the literature, the symbol $\rhd$ is sometimes used to represent an *importance* relation between attributes; and, as explained by [31], statement $\alpha \,|\, V : w \geq w'$ is a way to express that attributes in $\mathsf{Var}(w)$ have more importance those in V (when α is true).

[4] This is *full acyclicity* in [31].

Properties/Restrictions									
Base language				CPnet		CPnet	CPnet	LPT	LTP
Unary swaps / nodes		1	1	(1)	1	(1)	(1)		1
No free variable		$\not\rhd$	$\not\rhd$	($\not\rhd$)		($\not\rhd$)	($\not\rhd$)		
Condition in conjunctive form		$\wedge$		($\wedge$)	$\wedge$	($\wedge$)	($\wedge$)		
Local consistency			$\not\perp^{\mathsf{loc}}$	($\not\perp^{\mathsf{loc}}$)	$\not\perp^{\mathsf{loc}}$	($\not\perp^{\mathsf{loc}}$)	($\not\perp^{\mathsf{loc}}$)		
Local completeness			$\top^{\mathsf{loc}}$	($\top^{\mathsf{loc}}$)		($\top^{\mathsf{loc}}$)	($\top^{\mathsf{loc}}$)		
Acyclicity					cuc	full	polytree		
Queries									
LINEARISABILITY	✘!	✘!	✘!		$\top$	$\top$	$\top$	$\top$	$\top$
EQUIVALENCE	✘!	✘!		✓	✘!	✓	✓	✓	✓
R-COMPARISON, $R \in \{\succeq, \succ, \bowtie\}$	✘!	✘!	✘!	✘!	✘!	✘!	✓	✓	✓
$\sim$-COMPARISON	✘!	✘!	✘!		✓	✓	✓	✓	✓
2-ORDERING	✘!	✘!	✘!		✓	✓	✓	✓	✓
UNDOMINATED CHECK	✓	✓	✓	✓	✓	✓	✓	✓	✓
S. DOM., DOM., W. UNDOM. CHECK	✘!	✘!	✘!			✓	✓	✓	✓
S. DOM. $\exists$, DOM. $\exists$	✘!		✘!			$\top$	$\top$	✓	✓
UNDOMINATED $\exists$	✘!	✘!			$\top$	$\top$	$\top$	$\top$	$\top$
W. UNDOMINATED $\exists$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

Table 1: Language restrictions and complexity of queries: ✓ = indicates that the query can be solved in time polynomial in $|\varphi|+|\mathsf{result}|$; ✘! = no such algorithm unless P= NP, or PSPACE= P; $\top$ = always true for the language.

- $\mathcal{L}\ \not\circlearrowleft^{\mathsf{cuc}}$ the restriction of $\mathcal{L}$ to cuc-acyclic formulas, which are those φ such that for every alternative o, $D_\varphi^{\mathsf{uncond}} \cup D_\varphi^{\mathsf{cond}}(o)$ is acyclic.[5]

Note that D_φ can be computed in polynomial time. Moreover, in the case of $\mathsf{CP1}\not\rhd$, the $D_\varphi^{\mathsf{cond}}(o)$'s are all empty, so cuc-acyclicity reduces to D_φ being acyclic, and D_φ is the set of all (X,Y) such φ contains some statement $u\!:\!y\!>\!y'$ with $X \in \mathsf{Var}(u)$ – which is the definition used by [5].

In the more general case of CP1, checking cuc-acyclicity can be hard [8, Th. 3], [31, Prop. 24].

3.4 Attribute-wise restrictions

It is possible, especially with CP1 statements, to consider restrictions that guarantee some form of completeness and consistency on the conditions that sanction swaps on a given variable X. In other words, the idea is that for every pair of alternatives $o, o' \in \underline{\mathcal{X}}$ such that o and o' are equal except for their value for one attribute, there must be exactly one statement in a CP-net that orders o and o'. These conditions are implicit in CP-nets defined by [5], and have been formally defined by [23] in a slightly more restrictive context (binary attributes) and, in part, by [31].

Definition 1 (Local completeness and local consistency [23, 31]). Let $\varphi \in \mathsf{CP1}$. For every attribute $X \in \mathcal{X}$ and every partial instantiation u, define $\geq_\varphi^{X,u}$ to be the reflexive and transitive closure of the set of all pairs $(x,x') \in \underline{\mathcal{X}}^2$ such that there exists some $\alpha\,|\,V\!:\!x \geq x' \in \varphi$ with $u \models \alpha$; then φ is *locally consistent* if $\geq_\varphi^{X,o}$ is antisymmetric for every attribute $X \in \mathcal{X}$ and every alternative $o \in \underline{\mathcal{X}}$; and φ is *locally complete* if $\geq_\varphi^{X,o}$ is a total preorder for every attribute X and every alternative $o \in \underline{\mathcal{X}}$.

Given some language $\mathcal{L} \subseteq \mathsf{CP}$, we define:

- $\mathcal{L}\ \not\perp^{\mathsf{loc}}$ is the restriction of $\mathcal{L}$ to those formulas that are locally consistent;
- $\mathcal{L}\top^{\mathsf{loc}}$ is the restriction of $\mathcal{L}$ to those formulas that are locally complete.

Note that the problem of checking if a formula of $\mathsf{CP1}\wedge$ is locally consistent is coNP complete [31, Prop. 11]. Locally complete and locally consistent formulas of $\mathsf{CP1}\not\rhd$, that is, formulas of $\mathsf{CP1}\not\rhd\ \not\perp^{\mathsf{loc}}\ \top^{\mathsf{loc}}$, are called *CP-nets* by [23]. However, we recall next the original definition of CP-nets, which is slightly different.

3.5 CP-nets

In their seminal work, [5] define a CP-net over a set of attributes $\mathcal{X}$ to be composed of two elements:

1. a directed graph over $\mathcal{X}$, which should represent *preferential dependencies* between attributes;[6]

[5] This definition is weaker than the one given by [31], who also imposes local consistency as will be defined shortly; it corresponds to the definition of *conditional acyclicity* as given by [8].

[6] Given some pre-order $\succeq$ over $\mathcal{X}$, attribute X is said to be preferentially dependent on attribute Y if there exist $x, x' \in \underline{X}$, $y, y' \in \underline{Y}$, $z \in \underline{\mathcal{X}\ (\{X,Y\})}$ such that $xyz \succeq_\varphi x'yz$ but $xy'z \not\succeq_\varphi xy'z$.

2. a set of conditional preference tables, one for every attribute X: if U is the set of parents of X in the graph, the conditional preference table for X contains exactly $|\underline{U}|$ rules $u{:}\geq$, for every $u \in \underline{U}$, where the $\geq$'s are linear orders over $\underline{X}$.

Therefore, as shown by [31], CP-nets can be seen as sets of unary CP statements in conjunctive form with no free attribute. Specifically, given a CP-net $\mathcal{N}$ over $\mathcal{X}$, define $\varphi_{\mathcal{N}}$ to be the set of all CP statements $u{:}x \geq x'$, for every attribute X, every $u \in \mathsf{Pa}(X)$, every $x, x' \in \underline{X}$ such that x are consecutive values in the linear order $\geq$ specified by the rule $u{:}\geq$ of $\mathcal{N}$. $\mathcal{N}$ being a CP-net enforces a very strong form of local consistency and completeness: it must the case that, for every attribute X with parents U, for every $u \in \underline{U}$, for every $x, x' \in$, the CP-net must explicitly, and uniquely, order ux and ux'.

Thus we call

- CPnet the language that contains all $\varphi_{\mathcal{N}}$, for every CP-net $\mathcal{N}$.

Note that $\mathsf{CPnet} \subseteq \mathsf{CP1}\not\rhd\wedge \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}}$. [8] define TCP-nets as an extension of CP-nets where it is possible to represent tradeoffs, by stating that, under some condition, some attribute is more important than another one. [31] describes how TCP-nets can be transformed, in polynomial times, into equivalent sets of $\mathsf{CP1}\wedge$ statements.

3.6 Semantic restriction

Although the original definition of CP-nets by [7] does not impose it, many works on CP-nets, especially following [5], consider that they are intended to represent a strict partial order, that is, that $\succeq_\varphi$ should be antisymmetric; equivalently, this means that φ^* can be extended to a linear order. We say that a set φ of CP-statements is *linearisable* if φ^* can be extended to a linear order;[7] in this case, several authors define $\succ_\varphi$ to be the transitive closure of φ^*, which leads to the same definition of preorder $\succeq_\varphi$ as ours. But, like [23], we use the same definition for $\succeq_\varphi$ even when φ^* is not acyclic; note that in this case $\succeq_\varphi$ is not antisymmetric.

Note that, if φ is linearisable, then it is locally consistent.

3.7 Lexicographic preference trees

LP-trees generalise lexicographic orders. As an inference mechanism, they are equivalent to search trees used by [6], and formalised by [28, 31]. As a preference representation, and elicitation, language, slightly different definitions for LP-trees have been proposed by [4, 11, 18]. We use here a definition which subsumes the others.

[7] Such sets of CP-statements are often called *consistent* in the standard terminology on CP-nets, but we prefer to depart from this definition which only makes sense when one asserts that φ should indeed represent a strict partial order.

An LP-tree over $\mathcal{X}$ is a rooted tree with labelled nodes and edges, and a set of preference tables; specifically

- every node N is labelled with a set $W \subseteq \mathcal{X}$;
- we denote by $\mathsf{Anc}(N)$ the set of attributes that appear in the nodes above N (excluding those at N), and by $\mathsf{NonInst}(N)$ the set of attributes that appear in the nodes above N that have only one child;
- if N is not a leaf, it can have one child, or $|\underline{W}|$ children;
- in the latter case, the edges that connect N to its children are labelled with the instantiations in $\underline{W}$;
- if N has one child only, the edge that connect N to its child is not labelled;
- a conditional preference table $\mathsf{CPT}(N)$ is associated with N: it contains local preference rules of the form $\alpha{:}\geq$, where $\geq$ is a preorder over $\underline{W}$, and α is a propositional formula over some attributes in $U \subseteq \mathsf{NonInst}(N)$.

We assume that the rules in $\mathsf{CPT}(N)$ define their preorder in extension. Additionally, two constraints guarantee that an LP-tree φ defines a unique preorder over $\underline{\mathcal{X}}$:

- no attribute can appear at more than one node on any branch of φ; and,
- at every node N of φ, for every $u \in \mathsf{NonInst}(N)$, $\mathsf{CPT}(N)$ must contain exactly one rule $\alpha{:}\geq$ such that $u \models \alpha$.

Given an LP-tree φ and an alternative $o \in \underline{\mathcal{X}}$, there is a unique way to traverse the tree, starting at the root, and along edges that are either not labelled, or labelled with instantiations that agree with o, until a leaf is reached. Now, given two distinct alternatives o, o', it is possible to traverse the tree along the same edges as long as o and o' agree, until a node N is reached which is labelled with some W such that $o[W] \neq o'[W]$: we say that N decides $\{o, o'\}$.

In order to define $\succeq_\varphi$ for some LP-tree φ, we define φ^* to be the set of all pairs of distinct alternatives (o, o') such that there is a node N that decides $\{o, o'\}$ and the only rule $\alpha{:}\geq \in \mathsf{CPT}(N)$ with $o[\mathsf{NonInst}(N)] = o'[\mathsf{NonInst}(N)] \models \alpha$ is such that $o[W] \geq o'[W]$. Then $\succeq_\varphi$ is the reflexive closure of φ^*.

Proposition 1 ([3]). *Let φ be an LP-tree over $\mathcal{X}$, then $\succeq_\varphi$ as defined above is a preorder.*

An LP-tree is said to be complete if every attribute appears on every branch, and if every preference rule specifies a linear order; $\succeq_\varphi$ is then a linear order too.

From a semantic point of view, an LP-tree φ is equivalent to the set that contains, for every node N of φ labelled with W , and every rule $\alpha{:}\geq$ in $\mathsf{CPT}(N)$, all CP statements of the form $\alpha \wedge u \,|\, V : w \geq w'$, where

- u is the combination of values given to the attributes in $\mathsf{Anc}(N) - \mathsf{NonInst}(N)$ along the edges between the root and N, and

- $w, w' \in \underline{W}$ such that $w \geq w'$, and
- $V = [\mathcal{X} - (\mathsf{Anc}(N) \cup W)]$.

We define the following languages:

- LPT is the language of LP-trees as defined above; we consider that LPTis a subset of CP.[8]

Note that, using the notations defined above:

- LPTk = LPT $\cap$ CPk is the restriction of LPT where every node has at most k attributes, for every $k \in \mathbf{N}$; in particular, LPT1 is the language of LP-trees with one attribute at each node;
- LPT$\wedge$ = LPT$\cap$CP$\wedge$ is the restriction of LPT where the condition α in every rule at every node is a conjunction of literals.

LP-trees as defined by [28, 4, 25] are sublanguages of LPT1$\wedge$; and those of [18] and [11] are sublanguages of LPT$\wedge$.

4 Expressiveness

A first criterium for comparing languages is expressiveness:

Definition 2. Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that:

- $\mathcal{L}'$ is a sublanguage of $\mathcal{L}$ if $\mathcal{L} \supseteq \mathcal{L}'$; it is a proper sublanguage if $\mathcal{L} \supset \mathcal{L}'$, that is, if $\mathcal{L} \supseteq \mathcal{L}'$ and $\mathcal{L}' \not\supseteq \mathcal{L}$;
- $\mathcal{L}$ is *at least as expressive as* $\mathcal{L}'$, written $\mathcal{L} \sqsupseteq \mathcal{L}'$, if every preorder that can be represented with a formula of $\mathcal{L}'$ can also be represented with a formula of $\mathcal{L}$; we write $\mathcal{L} \sqsupset \mathcal{L}'$ if $\mathcal{L} \sqsupseteq \mathcal{L}'$ but it is not the case that $\mathcal{L}' \sqsupseteq \mathcal{L}$, and say in this case that $\mathcal{L}$ is strictly more expressive than $\mathcal{L}'$.

Note that $\sqsupseteq$ is a preorder, and obviously $\mathcal{L} \supseteq \mathcal{L}'$ implies $\mathcal{L} \sqsupseteq \mathcal{L}'$, but the converse does not hold in general.

Clearly, CP$\not\triangleright$ $\subseteq$ CP and CP$\wedge$ $\subseteq$ CP; however, these three language have the same expressiveness:

Proposition 2. *CP, CP$\wedge$ and CP$\not\triangleright$ can all three represent every preorder, thus CP $\sqsupseteq$ CP$\not\triangleright$ $\sqsupseteq$ CP$\wedge$ $\sqsupseteq$ CP.*

A large body of works on CP-statements since the seminal paper by [6] concentrate on various subsets of CP1. With this strong restriction on the number of swapped variables, CP-theories have a reduced expressiveness.

Example 1 (CP1 $\not\sqsupseteq$ CP). Consider two binary attributes A and B, with respective domains $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. Define preorder $\succeq$ such that $ab \succ \bar{a}\bar{b}$, with the two remaining alternatives being incomparable to the former and to each other. This can be represented in CP with $\varphi = \{ab \geq \bar{a}\bar{b}\}$. But it cannot be represented in CP1, because this would require at least two rules: one to flip the value of A, the other one to flip the value of B; but then there must be one intermediate alternative comparable with ab and $\bar{a}\bar{b}$.

[8] Strictly speaking, for LPT $\subseteq$ CP to hold, we can add the possibility to augment every formula in CP with a tree structure.

Example 2 (CP1 $\not\sqsupseteq$ CP even if restricted to linear orders). Consider two binary attributes A and B, with respective domains $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. Define preorder $\succeq$ such that $ab \succ \bar{a}\bar{b} \succ a\bar{b} \succ \bar{a}b$. This can be represented in CP with $\varphi = \{ab \geq \bar{a}\bar{b}, \bar{a}\bar{b} \geq a\bar{b}, a\bar{b} \geq \bar{a}b\}$. But it cannot be represented in CP1: $\{b : a \geq \bar{a}, \bar{b} : \bar{a} \geq a, a : b \geq \bar{b}, \bar{a} : \bar{b} \geq b\}^* \subseteq \varphi^*$, but this is not sufficient to compare $a\bar{b}$ with $\bar{a}b$. The four remaining formulas of CP1 over these two attributes are $B : a \geq \bar{a}$, $B : \bar{a} \geq a$, $A : b \geq \bar{b}$, $A : \bar{b} \geq b$, adding any of them to φ yields a preorder which would not be antisymmetric.

Forbidding free parts incurs an additional loss in expressiveness:

Example 3 (CP1$\not\triangleright$ $\not\sqsupseteq$ CP1). Consider two binary attributes A and B, with respective domains $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. Define preorder $\succeq$ such that $ab \succ a\bar{b} \succ \bar{a}b \succ \bar{a}\bar{b}$. This can be represented in CP1 with $\varphi = \{B : a \geq \bar{a}, b \geq \bar{b}\}$. But it cannot be represented in CP1$\not\triangleright$, and it cannot be obtained by transitivity from the comparisons that can be expressed in CP1$\not\triangleright$.

However, restricting to conjunctive statements does not incur a loss in expressiveness.

Proposition 3. *CP $\sqsupset$ CP1 $\sqsupset$ CP1$\not\triangleright$, CP1$\wedge$ $\sqsupset$ CP1$\wedge\not\triangleright$, but CP1 $\sqsupseteq$ CP1$\wedge$ $\sqsupseteq$ CP1 and CP1$\not\triangleright$ $\sqsupseteq$ CP1$\wedge\not\triangleright$ $\sqsupseteq$ CP1$\not\triangleright$.*

LP trees Because an LP-tree can be a single node labelled with $\mathcal{X}$, and a single preference rule $\top : \geq$ where $\geq$ can be any preorder, LPT can represent any preorder.

Proposition 4. *LPT $\sqsupseteq$ CP $\sqsupset$ LPT.*

For LP-trees too, limiting the number of attributes per node reduces expressiveness:

Proposition 5. *LPT $\sqsupset$ LPT$\wedge$ $\sqsupseteq$ LPT, LPT $\sqsupset$ LPTk $\sqsupset$ LPT$k\wedge$ $\sqsupseteq$ LPTk.*

5 Succinctness

Another criterium is the relative sizes of formulas that can represent the same preorder in different languages. [12] study the space efficiency of various propositional knowledge representation formalisms. An often used definition of succinctness makes it a particular case of expressiveness:

Definition 3 ([22, 15]). Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that $\mathcal{L}$ is *at least as succinct as* $\mathcal{L}'$, written $\mathcal{L} \leqq \mathcal{L}'$, if there exists a polynomial p such that for every formula $\varphi' \in \mathcal{L}'$, there exists a formula $\varphi \in \mathcal{L}$ that represent the same preorder as φ' and such that $|\varphi| < p(|\varphi'|)$.

With this definition, if $\mathcal{L}' \subseteq \mathcal{L}$ then $\mathcal{L} \leqq \mathcal{L}'$; and if $\mathcal{L} \leqq \mathcal{L}'$ then $\mathcal{L} \sqsupseteq \mathcal{L}'$. In particular, if we have two languages such that $\mathcal{L} \supset \mathcal{L}'$ and $\mathcal{L}' \not\supseteq \mathcal{L}$, then $\mathcal{L} \leqq \mathcal{L}'$ and $\mathcal{L}' \not\leqq \mathcal{L}$, even if there is no *real* succinctness hierarchy between the two, it is just that one is strictly more expressive than the other. Therefore, we introduce the following definition for strict succinctness, more restrictive than taking the strict partial order induced by $\leqq$.

Definition 4. Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages for representing preorders. We say that $\mathcal{L}$ is *strictly more succinct than* $\mathcal{L}'$, written $\mathcal{L} \ll \mathcal{L}'$, if $\mathcal{L} \leqq \mathcal{L}'$ and for every polynomial p, there exists $\varphi \in \mathcal{L}$ such that:

- there exists $\varphi' \in \mathcal{L}'$ such that $\succeq_\varphi = \succeq_{\varphi'}$, but
- for every $\varphi' \in \mathcal{L}'$ such that $\succeq_\varphi = \succeq_{\varphi'}$, $|\varphi'| > p(|\varphi|)$.

With this definition, $\mathcal{L} \ll \mathcal{L}'$ if every formula $\varphi \in \mathcal{L}'$ has an equivalent formula in $\mathcal{L}$ which is "no bigger"[9], and there is at least one sequence of formulas[10] in $\mathcal{L}$ that have equivalent formulas in $\mathcal{L}'$ but necessarily "exponentially bigger". Note that $\mathcal{L} \ll \mathcal{L}'$ implies that $\mathcal{L} \leqq \mathcal{L}'$ and $\mathcal{L}' \not\leqq \mathcal{L}$.

Restricting the conditioning part of the statements to be conjunctions of literals does induce a loss in succinctness, because propositional logic is strictly more succinct than the language of DNFs.

Example 4. Consider $2n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_n, Z$, and let φ contain $2n+2$ unary CP-statements with no free attribute: $(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \ldots \wedge (x_n \vee y_n) : z \geq \bar{z}$, $\neg[(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \ldots \wedge (x_n \vee y_n)] : \bar{z} \geq z$ and $\bar{x}_i \geq x_i$ and $\bar{y}_i \geq y_i$ for every $i \in \{1, \ldots, n\}$. Then $\varphi \in \mathsf{CP1}\not\triangleright \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}}$, but φ is not in conjunctive form. A set of conjunctive CP-statements equivalent to φ has to contain all 2^n statements of the form $\mu_1 \mu_2 \ldots \mu_n : z \geq \bar{z}$ with $\mu_i = x_i$ or $\mu_i = x_i$ for every i.

Restricting to CP-nets induces a further loss in expressiveness, as the next example shows:

Example 5. Consider $n+1$ binary attributes $X_1, X_2, \ldots, X_n, Y$, and let φ be the $\mathsf{CP1}\not\triangleright\wedge$ formula that contains the following statements: $x_i \geq \bar{x}_i$ for $i = 1, \ldots, n$; $x_1 x_2 \ldots x_n : y \geq \bar{y}$; $\bar{x}_i : \bar{y} \geq y$ for $i = 1, \ldots, n$. The size of φ is linear in n. Because preferences for Y depend on all X_i's, a CP-net equivalent to φ will contain, in the table for Y, 2^n CP statements.

Proposition 6. *For every language such that* $\mathsf{CP1}\not\triangleright \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}} \subseteq \mathcal{L} \subseteq \mathsf{CP}$, $\mathcal{L} \ll \mathsf{CP}\wedge \cap \mathcal{L}$. *Moreover,* $\mathsf{CP1}\not\triangleright\wedge \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}} \ll$ *CP-net.*

6 Queries

Linearisability Checking if a given $\varphi \in \mathsf{CP}$ is linearisable, that is, if $\succeq_\varphi$ is antisymmetric, can give some interesting insights into the semantics of φ. The following query has been addressed in many works on CP statements:[11]

LINEARISABILITY Given φ, is φ linearisable?

[5] prove that when its dependency graph D_φ is acyclic, then a CP-net φ is linearisable. This result has been extended by [16, 8, 31], who give weaker graphical conditions that guarantee that a locally consistent set of unary, conjunctive CP statements, that is, a formula of $\mathsf{CP1}\wedge \not\perp^{\mathsf{loc}}$ is linearisable: specifically, every formula of $\mathsf{CP1}\wedge \not\perp^{\mathsf{loc}} \not\circlearrowleft^{\mathsf{cuc}}$ is linearisable. However, checking these conditions is a hard problem. [23, Theorem 3 and 4] prove that LINEARISABILITY is PSPACE-complete for $\mathsf{CP1}\not\triangleright\wedge$, $\mathsf{CP1}\not\triangleright \not\perp^{\mathsf{loc}} \top^{\mathsf{loc}}$.

Comparing theories Checking if two theories yield the same preorder can be useful during the compilation process. We say that two formulas φ and φ' are equivalent if they represent the same preorder, that is, if $\succeq_\varphi$ and $\succeq_{\varphi'}$ are identical; we then write $\varphi \equiv \varphi'$.

EQUIVALENCE Given two formulas φ and φ', are they equivalent?

Consider a formula $\varphi \in \mathsf{CP}$, two alternatives o, o', and let $\varphi' = \varphi \cup \{o \geq o'\}$: clearly $o \succeq_{\varphi'} o'$, thus $\varphi \equiv \varphi'$ if and only if $o \succeq_\varphi o'$. Therefore, if language $\mathcal{L}$ is such that adding CP statement $o \geq o'$ to any of its formulas yields a formula that is still in $\mathcal{L}$, then EQUIVALENCE has to be at least as hard as $\succeq$-COMPARISON for $\mathcal{L}$. This is the case of CP. The problem remains hard for $\mathsf{CP1}\not\triangleright$, because it is hard to check the equivalence, in propositional logic, of the conditions of statements that entail a particular swap $x \geq x'$.

Example 6. Consider three attributes A, B and C with respective domains $\{a, \bar{a}\}$, $\{b, \bar{b}\}$ and $\{c_1, c_2, c_3\}$. Consider two CP statements $s = \bar{a} : c_1 \geq c_2$ and $s' = b : c_2 \geq c_3$, and let $\varphi = \{s, s', a : c_1 \geq c_3\}$. Because of statements s and s' we have $\bar{a}bc_1 \geq_\varphi \bar{a}bc_2 \geq_\varphi \bar{a}bc_3$; also, $abc_1 \geq_\varphi abc_3$ because of statement $a : c_1 \geq c_3$. Hence, for any $u \in \underline{A} \times \underline{B}$, if $u \models a \vee (\bar{a}b)$ then $uc_1 \geq uc_3$. Thus $\varphi \equiv \{s, s'\} \cup \{a \vee (\bar{a}b) : c_1 \geq c_3\} \equiv \varphi \cup \{b : c_1 \geq c_3\}$.

In general, given a propositional language $\mathcal{P}$ we define $\mathcal{P}^\vee$ to be the set of finite disjunctions of formulas in $\mathcal{P}$, and:

- $\mathsf{CP1}\not\triangleright\mathcal{P}$ is $\mathsf{CP1}\not\triangleright$ restricted to those statements such that the condition is in $\mathcal{P}$.

Proposition 7. *Given a propositional language $\mathcal{P}$ closed for conjunction,* EQUIVALENCE *for $\mathcal{P}^\vee$ (in the sense of propositional logic), restricted to consistent formulas, reduces to* EQUIVALENCE *for $\mathsf{CP1}\not\triangleright\mathcal{P}$ restricted to fully acyclic, locally consistent formulas.*

[9] up to some polynomial transformation of the size of φ

[10] one formula for every polynomial p

[11] This query is often called *consistency*

In particular, EQUIVALENCE is NP-hard for $\mathsf{CP1}\not\triangleright\wedge\not\emptyset\not\perp^{\mathsf{loc}}$ restricted to binary attributes, because checking if two propositional, consistent DNFs are equivalent is NP-hard.

However, equivalence is not hard to check for CP-nets, thanks to the existence of a canonical, minimal form: given a CP-net with attributes X and Y such that $X \in \mathsf{Pa}(Y)$, it is easy to check if the preferences that appear in the conditional preference table for Y truly depend on X: if not, the table can be simplified and the edge (X,Y) can be removed. This can be done in polynomial time for all edges (X,Y) of the dependency graph of the CP-net.

For LP-trees too, EQUIVALENCE is easy to check because of the existence of a canonical form: given a node of an LP-tree φ labelled with set of variables S, it is possible to check if it can be split into a "root" node and one or more several children, using an approach like that proposed by [18] for learning an LP-tree from positive examples; this can be done in time polynomial w.r.t. to $|\underline{S}|$, which is itself polynomially bounded by the size of the preference table at S, since we assume that the pre-orders over $\underline{S}$ are given in extension in this table. This procedure can be iterated until no node of the tree can be split. Moreover, if all subtrees of a node are identical, they can be merged into one subtree; applying this in a bottom-up fashion, one obtains the canonical form of the tree; two LP-trees are then equivalent if and only if they have the same canonical form.

Comparing alternatives A basic question, given a formula φ and two alternatives o, o' is: how do o and o' compare, according to φ? Is it the case that $o \succ_\varphi o'$, or $o' \succ_\varphi o$, or $o \bowtie_\varphi o'$, or $o \sim_\varphi o'$? We define the following query, for any relation $R \in \{\succ, \succeq, \sim, \bowtie\}$:

R-COMPARISON Given formula φ, alternatives o, o', is it the case that $oR_\varphi o'$?

For LP-trees, in order to compare alternatives o and o', one only has to traverse the tree from the root downwards until a node that decides the pair is reached, or up to a leaf if no such node is encountered: in this case o and o' are incomparable. Note that checking if a node decides the pair, and checking if a rule at that nodes applies to order them, can both be done in polynomial time.

Proposition 8. *R-COMPARISON is in P for LPT for every $R \in \{\succ, \succeq, \sim, \bowtie\}$.*

The complexity of comparisons has been studied by [5] for CP nets, by [23] for $\mathsf{CP1}\not\triangleright$ and by [31] for $\mathsf{CP1}\wedge$. [23] propose a simple non-deterministic algorithm to prove membership in PSPACE of $\succeq$-COMPARISON; we rewrite the algorithm here for our more general preference statements:

Algorithm : $\succeq$-comparison. Input: o, o', φ

1. Repeat:
   (a) guess o'', $\alpha \,|\, V : w \geq w' \in \varphi$; $Y \leftarrow \mathcal{X} - (U \cup V \cup W)$;
   (b) if $\alpha \,|\, V : w \geq w' \in \varphi$ sanctions (o, o''): $o \leftarrow o''$;
   until $o'' = o'$.

This algorithm only needs space to store two outcomes at any iteration, and checks sanctioning w.r.t. one rule at every iteration. Repeated applications of this algorithm can answer R-COMPARISON queries for $R \in \{\succ, \sim, \bowtie\}$; for instance, to check if $o \bowtie_\varphi o'$, we check that $o \succeq_\varphi o'$ does not hold and that $o' \succeq_\varphi o$ does not hold either. [12]

Tractability of comparisons, except in some trivial cases, comes at a heavy price in terms of expressiveness: the only positive result for $\succeq$-COMPARISON is about CP-nets when the dependency graph is a polytree [5, Theorem 14]; clearly, this entails a positive results for the other comparison queries for this language. The next proposition shows that most comparisons are hard for a vast family of CP statements; it follows from hardness results proved by [5] and [23].

Proposition 9. *R-COMPARISON for $R \in \{\succ, \succeq, \bowtie\}$ is NP hard for the language of fully acyclic CP-nets. $\sim$-COMPARISON is trivial for linearisable CP formulas, but hard for $\mathsf{CP1}\not\triangleright\wedge$, and for the language of linearisable (hence locally consistent) and locally complete $\mathsf{CP1}\not\triangleright$ formulas.*

Optimisation Comparison queries can be used to compute, in a given set $S \subseteq \underline{\mathcal{X}}$, an alternative that is not dominated by any other alternative in S: this can be achieved by asking at most $|S|(|S|-1)/2$ dominance queries (such query can return failure when S contains no such alternative). More generally, given some integer k, we may be interested in finding a subset S' of S that contains k "best" alternatives of S, in the following sense: we say that $S' \subseteq S$ is *weakly undominated* in S if for every $o \in S'$, for every $o' \in S \setminus S'$ it is not the case that $o' \succ_\varphi o$. Note that such a set must exist, because $\succ_\varphi$ is acyclic. [31] proposes a stronger query:

ORDERING Given k, $S \subseteq \underline{\mathcal{X}}$ and φ, find $o_1, o_2, \ldots, o_k \in S$ such that for every $i \in 1, \ldots, k$, for every $o' \in S$, if $o' \succ_\varphi o$ then $o' \in \{o_1, \ldots, o_i\}$.

Note that if $o_1, o_2, \ldots, o_k$ is the answer to such query, if $1 \leq i < j \leq k$, then it can be the case that $o_i \bowtie o_j$, but it is guaranteed that $o_j \not\succ o_i$: in the context of a recommender system for instance, where one would expect alternatives to be presented in order of non-increasing preference, o_i could be safely presented before o_j.

[5] consider a specific case of the above query, when $|S| = 2$: note that given two alternatives $o, o' \in \underline{\mathcal{X}}$ it must be the case that at least one of $o \succ_\varphi o'$ or $o' \succ_\varphi o$ must be false, since $\succ_\varphi$ is irreflexive and transitive.

[12] Recall that NPSPACE = co-NPSPACE = PSPACE .

2-ORDERING Given $o, o' \in \underline{\mathcal{X}}$, return a pair $(o_1, o_2) \in \{(o, o'), (o', o)\}$ such that $o_2 \not\succ_\varphi o_1$.

[5] prove that 2-ORDERING is tractable for acyclic CP-nets. This result can be generalised to cuc-acyclic formulas of CP1$\wedge$:

Proposition 10 (Generalisation of Theorem 5 in [5])**.** 2-ORDERING *and* ORDERING *can be answered in time which is polynomial in the size of φ and the size of S for cuc-acyclic, locally consistent formulas of CP1$\wedge$; and for LPT.*

For more general languages, the query is hard:

Proposition 11. 2-ORDERING *is PSPACE-complete for CP1$\wedge\not\triangleright$, even restricted to locally consistent and locally complete formulas.*

This approach for optimisation is of course not practical when $S = \underline{\mathcal{X}}$, because in this case the size of S is exponential in the number of attributes. When k is fixed, $k = 1$ and $S = \underline{\mathcal{X}}$, the ORDERING query amounts to finding a weakly undominated alternative. Based on the notions of (weakly) undominated / (strongly) dominating alternatives (defined in section 2.2.1), [23] define two types of queries: 1) given φ and o, is o (weakly) undominated, or is it (strongly) dominated? 2) given φ, is there a (weakly) undominated, or a (strongly) dominated, alternative? We call these queries W-UNDOMINATED CHECK, UNDOMINATED CHECK, and so on, for queries of type 1); and W-UNDOMINATED-$\exists$, and so on, for queries of type 2). Note also that there is always at least one weakly undominated alternative (because $\underline{\mathcal{X}}$ is finite), so WEAKLY UNDOMINATED-$\exists$ is trivial (always true).

All these queries are easily shown to be tractable for LPT. The problem UNDOMINATED CHECK is tractable for CP; in fact, the proof, originally given by [5] for CP-nets, and generalised to CP1$\not\triangleright$ by [23], can be generalised further to CP.

Proposition 12. UNDOMINATED CHECK *is in P for CP.*

That all other, dominance related, queries are in PSPACE for CP can be proved using again the algorithm for checking $\succeq$-comparison. Checking for instance that o is *not* undominated can be done by guessing some o' and checking if $o' \succeq_\varphi o$. [23] prove several hardness results:

Proposition 13. *[23] The problems* W. UNDOMINATED CHECK, DOMINATING CHECK, S. DOMINATING CHECK *are PSPACE complete for CP1$\not\triangleright\wedge$. These problems, as well as* DOMINATING $\exists$, S. DOMINATING $\exists$*, are PSPACE complete for CP1$\not\triangleright$, even if restricted to locally consistent and locally complete formulas.* UNDOMINATED $\exists$ *is NP complete for CP1$\not\triangleright\wedge$.*

For CP-nets, [5] give a polytime algorithm that computes the only dominating alternative when the dependency graph is acyclic; in this case, this alternative is also the only strongly dominating one, the only undominated, and the only weakly undominated one, since the CP-net is linearisable.

7 Conclusion

We have not studied here transformations, like conditioning or other forms of projection for instance. Some initial results on projections can be found in [2]. Note that the result of transformations like conditionning for CP1 formulas if often not expressible in CP1. However, this preliminary study shows that, for conditional preference statements, gains in terms of query complexity is not only at the cost of a loss in succinctness, but often at the cost of big losses in expressiveness. This may indicate that the language of conditional preference statement is not an adequate target language for compilation, but that other languages may be more suitable for that. However, existing real-valued languages in general force a complete ordering of the alternatives, thus a target language for the compact representation of possibly incomplete preorders has yet to be defined, possibly using combinations of real-valued formulas, used as multiple criteria, as in the definition of "partial order rationalizable" choice functions by [1]; or as approximation of the preorder represented by a set of CP statements.

Acknowledgments We thank Hélène Fargier for fruitful discussions, anonymous reviewers for their helpful comments and suggestions. This work has benefitted from the AI Interdisciplinary Institute ANITI. ANITI is funded by the French "Investing for the Future – PIA3" program under grant agreement n° ANR-19-PI3A-0004. This work has also been supported by the PING/ACK project of the French National Agency for Researchgrant agreement n° ANR-18-CE40-0011.

References

[1] F. Aleskerov, D. Bouyssou, and B. Monjardet. *Utility Maximization, Choice and Preference*. Springer-Verlag Berlin Heidelberg, 2nd edition, 2007.

[2] P. Besnard, J. Lang, and P. Marquis. Variable forgetting in preference relations over combinatorial domains. In *Proc. IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, 2005.

[3] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and Chattrakul Sombattheera. Learning various classes of models of lexicographic orderings. Tech. report IRIT/RR–2009-21–FR, IRIT, Toulouse, 2009.

[4] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and C. Sombattheera. Learning conditionally lexicographic

preference relations. In Proc. ECAI 2010, pages 269–274. IOS Press, 2010.

[5] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[6] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Preference-based constrained optimization with cp-nets. *Computational Intelligence*, 20(2):137–157, 2004.

[7] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In Proc. UAI 99, pages 71–80. Morgan Kaufmann, 1999.

[8] R. I. Brafman, C. Domshlak, and S. E. Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389–424, 2006.

[9] R. I. Brafman, F. Rossi, D. Salvagnin, K. B. Venable, and Toby Walsh. Finding the next solution in constraint- and preference-based knowledge representation formalisms. In Proc. KR'10. 2010.

[10] D. Braziunas and C. Boutilier. Local utility elicitation in gai models. In Proc. UAI'05, pages 42–49. 2005.

[11] M. Bräuning and E. Hüllermeyer. Learning conditional lexicographic preference trees. In Proc. ECAI 2012 workshop on Preference Learning: Problems and Applications in AI, pages 11–15, 2012.

[12] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Space efficiency of propositional knowledge representation formalisms. *Journal of Artificial Intelligence Research*, 13:1–31, 2000.

[13] S. Coste-Marquis, J. Lang, P. Liberatore, and P. Marquis. Expressive power and succinctness of propositional languages for preference representation. In Proc. KR'04, pages 203–212. AAAI Press, 2004.

[14] A. Darwiche. Compiling knowledge into decomposable negation normal form. In Proc. IJCAI 99, pages 284–289. Morgan Kaufmann, 1999.

[15] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[16] C. Domshlak and R. I. Brafman. CP-nets: Reasoning and consistency testing. In Proc. KR'02, pages 121–132. Morgan Kaufmann, 2002.

[17] Domshlak, C. , Prestwich, S. , Rossi, F. , Venable, K. , Walsh, T. Hard and soft constraints for reasoning about qualitative conditional preferences J. Heuristics 12(4-5), 2006, 263–285

[18] H. Fargier, P. F. Gimenez, and J. Mengin. Learning lexicographic preference trees from positive examples. In Proc. AAAI'18, pages 2959–2966. 2018.

[19] H. Fargier, P. Marquis, A. Niveau, and N. Schmidt. A knowledge compilation map for ordered real-valued decision diagrams. In Proc. AAAI'14, pages 1049–1055. AAAI Press, 2014.

[20] E. C. Freuder, R. Heffernan, R. J. Wallace, and N. Wilson. Lexicographically-ordered constraint satisfaction problems. *Constraints*, 15(1):1–28, 2010.

[21] G. Gigerenzer and D. G. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103(4):650–669, 1996.

[22] G. Gogic, H. A. Kautz, C. H. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In Proc. IJCAI'95, pages 862–869. Morgan Kaufmann, 1995.

[23] J., Jérôme Lang, M. Truszczynski, and N. Wilson. The computational complexity of dominance and consistency in CP-nets. *Journal of Artificial Intelligence Research*, 33:403–432, 2008.

[24] C. Gonzales and P. Perny. Gai networks for utility elicitation. In Proc. KR'04, pages 224–233. AAAI Press, 2004.

[25] J. Lang, J. Mengin, and L. Xia. Voting on multi-issue domains with conditionally lexicographic preferences. *Artificial Intelligence*, 265:18–44, 2018.

[26] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In Proc. IJCAI'95, pages 631–639. Morgan Kaufmann, 1995.

[27] M. Schmitt and L. Martignon. On the complexity of learning lexicographic strategies. *Journal of Machine Learning Research*, 7:55–83, 2006.

[28] N. Wilson. Consistency and constrained optimisation for conditional preferences. In Proc. ECAI'04, pages 888–892. IOS Press, 2004.

[29] N. Wilson. Extending cp-nets with stronger conditional preference statements. In Proc. AAAI'04, pages 735–741. AAAI Press / The MIT Press, 2004.

[30] N. Wilson. An effcient upper approximation for conditional preference. In Proc. ECAI'06. IOS Press, 2006.

[31] N. Wilson. Computational techniques for a simple theory of conditional preferences. *Artificial Intelligence*, 175:1053–1091, 2011.

Actes JIAF 2020

Consistance d'arc souple appliquée aux problèmes DCOP

Rachid Adrdor[1] **Redouane Ezzahir**[2] **Lahcen Koutti**[1]

[1] Département d'informatique, FSA Agadir, B.P 8106, Agadir, Maroc
[2] Département d'informatique, ENSA Agadir, B.P 1136, Agadir, Maroc
rachid.adrdor@edu.uiz.ac.ma r.ezzahir@uiz.ac.ma l.koutti@uiz.ac.ma

Résumé

AFB_BJ$^+$ est un algorithme puissant et efficace pour résoudre des problèmes d'optimisation de contraintes distribués (DCOPs). Dans AFB_BJ$^+$, les agents résolvent un DCOP en affectant leurs variables séquentiellement et en évaluant ces affectations de manière asynchrone. Pendant la recherche, AFB_BJ$^+$ réinitialise les domaines des agents chaque fois qu'il y a une extension d'une nouvelle solution, ce qui conduit à des recherches inutiles. Dans cet article, nous utilisons la consistance d'arc souple (AC*) dans l'algorithme AFB_BJ$^+$ pour éviter ces recherches inutiles en supprimant les valeurs qui ne font pas partie de la solution optimale parmi les valeurs possibles d'un DCOP. Sur le plan expérimental, l'algorithme résultant de cette combinaison, AFB_BJ$^+$-AC*, présente une meilleure performance en termes de charge de communication et d'effort de calcul.

Abstract

AFB_BJ$^+$ is a powerful and efficient algorithm for solving Distributed Constraint Optimization Problems (DCOPs). In AFB_BJ$^+$, agents solve a DCOP by assigning sequentially their variables and evaluating asynchronously these assignments. During the search, AFB_BJ$^+$ resets domains of agents whenever there is an extension of a new solution, which leads to unnecessary searches. In this article, we use the soft arc consistency (AC*) in the AFB_BJ$^+$ algorithm to avoid these unnecessary searches by removing values that are not part of the optimal solution among the possible values of a DCOP. At the experimental level, the algorithm resulting from this combination, AFB_BJ$^+$-AC*, presents a better performance in terms of communication load and computation effort.

1 Introduction

Le cadre des problèmes d'optimisation de contraintes distribués (DCOPs) est utilisé pour formaliser divers problèmes réels de forte combinatoire tels que l'allocation des ressources, le séquençage des voitures, etc. Un DCOP se compose d'un groupe d'agents autonomes, où chaque agent contrôle une partie de ce problème consistant en un sous-ensemble de variables et les contraintes qui les relient. Chacune de ces contraintes spécifie les coûts des combinaisons de valeurs affectées aux variables qu'elles connectent. En général, les contraintes ou les affectations de valeur peuvent être des informations stratégiques qui ne doivent pas être divulguées à d'autres agents [6]. Ainsi, chaque agent ne connaît que les informations de ses variables et les contraintes qui les impliquent. La résolution d'un DCOP est effectuée, de manière distribuée, en affectant des valeurs à ses variables de sorte que la somme des coûts de toutes ses contraintes soit minimisée.

De nombreux algorithmes ont été développés pour résoudre les DCOPs [4]. Synchronous Branch and Bound (SyncBB) est l'algorithme DCOP le plus simple. SyncBB effectue des affectations de manière synchrone et séquentielle. Dans SyncBB, seul l'agent possédant l'affectation partielle courante (CPA ou Current Partial Assignment) effectue une affectation ou un retour en arrière [8].

AFB [5] peut être considéré comme une amélioration de SyncBB où les agents étendent une affectation partielle courante (CPA) tant que la borne inférieure sur son coût ne dépasse pas la borne supérieure globale (c'est-à-dire le coût de la meilleure solution trouvée jusqu'à présent). Dans AFB, chaque extension synchrone d'une CPA est suivie d'une phase asynchrone (FB) permettant d'effectuer une évaluation en avant pour cette CPA. Dans la phase FB, les bornes inférieures sont calculées simultanément par des agents non affectés. Lorsque la borne inférieure de toutes les affectations d'un agent dépasse la borne supérieure globale, il effectue un retour en arrière simple vers l'agent précédent.

Plus tard, l'algorithme AFB a été amélioré en deux stades. Dans le premier stade, le simple retour en arrière a été changé en saut en arrière, ce qui a permis l'introduction de l'algorithme AFB_BJ [5]. Dans le deuxième stade, le saut en arrière a été optimisé en améliorant les bornes

inférieures estimées dans la phase FB, ce qui a donné l'algorithme AFB_BJ$^+$ [12].

Dans l'algorithme AFB_BJ$^+$, la réinitialisation des domaines des agents, chaque fois qu'il y a une extension d'une nouvelle solution, entraîne des recherches inutiles. Dans ce travail, nous utilisons la consistance d'arc souple (AC*) dans l'algorithme AFB_BJ$^+$ pour éviter ces recherches inutiles en supprimant définitivement des domaines d'un DCOP les valeurs sous-optimales.

Cet article est organisé comme suit. La section 2 donne un aperçu du cadre DCOP, de la consistance d'arc souple (AC*) et de l'algorithme AFB_BJ$^+$. La section 3 décrit l'algorithme AFB_BJ$^+$-AC*. La section 4 présente une évaluation expérimentale de l'algorithme AFB_BJ$^+$-AC*. Enfin, nous concluons l'article dans la section 5.

2 Contexte

2.1 DCOP

Le problème d'optimisation des contraintes distribuées (DCOP) [11] est défini par le quadruplet $(\mathcal{A}, \mathcal{X}, \mathcal{D}, C)$, où $\mathcal{A} = \{A_1, A_2, ..., A_k\}$ est un ensemble de k agents, $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ est un ensemble de n variables, où chaque variable x_j est contrôlée par un agent dans $\mathcal{A}$. $\mathcal{D} = \{D_1, D_2, ..., D_n\}$ est un ensemble de n domaines, où D_j est l'ensemble des valeurs possibles auxquelles la variable x_j peut être affectée. Seul l'agent contrôlant une variable peut lui affecter une valeur. $C = \{C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+\} \cup \{C_i : D_i \rightarrow \mathbb{R}^+\}$ est l'ensemble de contraintes souples. Chaque contrainte $C_{ij} \in C$ est définie sur la paire de variables $\{x_i, x_j\} \subseteq \mathcal{X}$. Les variables x_i et x_j sont appelées voisins.

Dans cet article, nous étudions les DCOPs restreints, où chaque agent contrôle une seule variable ($k = n$) et chaque contrainte implique au plus deux variables. Ainsi, nous utilisons les termes agent A_j et variable x_j de manière interchangeable et nous identifions l'agent A_j avec l'indice j de sa variable associée x_j. De plus, tous les agents stockent un ordre total unique $\prec$ appliqué aux agents. Les agents qui existent avant un agent $A_j \in \mathcal{A}$ dans l'ordre total sont les agents de priorité supérieure et ceux qui existent après A_j sont les agents de priorité inférieure. L'ordre $\prec$ divise l'ensemble $\Gamma(x_j)$ des voisins de A_j en voisins de priorité supérieure $\Gamma^-(x_j)$ et en voisins de priorité inférieure $\Gamma^+(x_j)$. Pour plus de clarté, nous supposons que l'ordre total est l'ordre lexicographique $[A_1, A_2, ..., A_n]$. Dans la suite de l'article, nous considérons un agent générique $A_j \in \mathcal{A}$. Ainsi, j est le niveau ou le rang de l'agent A_j. Une affectation pour l'agent A_j est un couple (x_j, v_j), où $v_j \in D_j$. Une affectation partielle courante (CPA) est un ensemble ordonné d'affectations, par exemple, $Y = [(x_1, v_1), \ldots, (x_j, v_j)]$ avec $x_1 \prec \cdots \prec x_j$. L'ensemble de toutes les variables affectées dans Y est noté $var(Y) = \{x_1, x_2, ..., x_j\}$. Soit $Y = [(x_1, v_1), \ldots, (x_i, v_i), \ldots, (x_j, v_j)]$ une affectation partielle courante (CPA), le sous-ensemble de Y incluant toutes les variables jusqu'à x_i est noté par $Y^i = [(x_1, v_1), \ldots, (x_i, v_i)]$.

Procédure 1 : ProjectUnary()

```
1 β ← min_{v_i ∈ D_i} {c_i(v_i)} ;
2 C_{φ_i} ← C_{φ_i} + β ;
3 pour chaque (v_i ∈ D_i) faire
4     c_i(v_i) ← c_i(v_i) − β ;
```

Le coût garanti de Y, $GC(Y)$, est la somme des coûts des contraintes C_{ij} impliquées dans Y (1).

$$GC(Y) = \sum_{C_{ij} \in C} c_{ij}(v_i, v_j) \mid (x_i, v_i),\ (x_j, v_j)\ \in Y \qquad (1)$$

Chaque CPA Y devient une solution si Y inclut les affectations de valeurs de toutes les variables du problème, c-à-d. $var(Y) = \mathcal{X}$. Le but des algorithmes DCOP est d'atteindre une solution avec un coût minimal, c-à-d. $Y^* = \arg\min_Y \{GC(Y) \mid var(Y) = \mathcal{X}\}$.

2.2 Consistance d'arc souple (AC*)

La consistance d'arc souple (AC*) sert à transformer un problème en un autre équivalent et facile à résoudre [9]. Cela se fait en remplissant les conditions représentées par la consistance de nœud (NC*) et la consistance d'arc (AC*).

Soient (x_i, v_i) une affectation de valeur, C_{ij} une contrainte binaire entre x_i et x_j, c_{ij} son coût binaire associé, C_i une contrainte unaire sur x_i, c_i son coût unaire associé, C_ϕ la contrainte d'arité zéro, qui est un entier représentant la borne inférieure de toute solution du problème et $\top$ la borne supérieure globale, qui est un entier représentant le coût inacceptable le plus bas.

Consistance de noeud (NC*) :

— (x_i, v_i) est NC* si $C_\phi + c_i(v_i) < \top$.
— Une variable x_i est NC* si pour chaque valeur $v_i \in D_i$, (x_i, v_i) est NC* et s'il existe une valeur $v_i \in D_i$ qui satisfait $c_i(v_i) = 0$.
— Un problème est NC* si toutes ses variables sont NC*.

Pour transformer un problème donné à un problème NC*, il faut appliquer sur chaque variable x_i de ce problème :

1. Une projection unaire (Proc. 1) qui transfére le coût minimum d'une C_i vers C_ϕ.
2. Un filtrage de son domaine D_i qui supprime toute valeur $v_i \in D_i$ avec $C_\phi + c_i(v_i) \geq \top$.

Consistance d'arc (AC*) :

Procédure 2 : ProjectBinary(x_i, x_j)

pour chaque ($v_i \in D_i$) **faire**
  $\alpha \leftarrow min_{v_j \in D_j} \left\{c_{ij}(v_i, v_j)\right\}$;
  pour chaque ($v_j \in D_j$) **faire**
    $c_{ij}(v_i, v_j) \leftarrow c_{ij}(v_i, v_j) - \alpha$;
  si (A_i *is the current agent*) **alors**
    $c_i(v_i) \leftarrow c_i(v_i) + \alpha$;

— (x_i, v_i) est AC* par rapport à C_{ij} s'il existe une valeur $v_j \in D_j$ qui satisfait $c_{ij}(v_i, v_j) = 0$, la valeur v_j appelée un support de la valeur v_i.
— Une variable x_i est AC* par rapport à C_{ij} si toute valeur $v_i \in D_i$ possède un support dans D_j.
— Un problème est AC* si toutes ses variables sont NC* et AC*.

Pour transformer un problème donné à un problème AC*, il faut appliquer sur chaque variable x_i de ce problème :

1. Une projection binaire (Proc. 2) qui transfère le coût minimum d'une C_{ij} vers C_i.
2. Une projection unaire (Proc. 1) qui transfère le coût minimum d'une C_i vers C_ϕ.
3. Un filtrage de son domaine D_i qui supprime toute valeur $v_i \in D_i$ avec $C_\phi + c_i(v_i) \geq \top$.

AC* peut être étendu pour fonctionner dans un environnement distribué avec quelques modifications. Premièrement, pour obtenir la valeur globale de C_ϕ, chaque agent A_i stocke temporairement sa valeur de contribution dans une variable locale C_{ϕ_i} ($C_\phi = \sum_{A_i \in \mathcal{A}} C_{\phi_i}$) (Proc. 1, ligne 2). Deuxièmement, chaque agent, A_i ou A_j, d'une paire d'agents partageant une contrainte C_{ij}, conserve une copie identique de cette C_{ij}. Pour garder la même copie de C_{ij} dans chaque agent lors de l'application d'AC*, A_i doit simuler l'action d'A_j sur sa copie de C_{ij} et vice versa (Proc. 3, lignes 3, 5), mais un seul d'entre eux transfère ses coûts unaires vers C_ϕ. Cela pour éviter la redondance des coûts transférés vers C_ϕ (Proc. 2, ligne 5) [7].

2.3 Algorithme AFB_BJ+

Chaque agent A_j, exécutant l'algorithme AFB_BJ+, utilise les structures de données suivantes : UB_j, le coût de la meilleure solution trouvée jusqu'à présent ou bien la borne supérieure globale courante. Y, l'affectation partielle courante (CPA). GC, le coût garanti. FC_j, le coût futur. LC_j, le coût local. $LB_k(Y)$, les bornes inférieures reçues des agents $A_k \in \mathcal{A}^+$.

Dans la phase d'initialisation, chaque agent A_j initialise ses structures de données et calcule les coûts futurs minimaux pour chaque valeur v_j de D_j.

Le coût futur d'une valeur v_j ($FC_j(v_j)$), donné par l'équation (2), est la somme des coûts minimaux des contraintes partagées avec les agents $A_k \in \Gamma^+(x_j)$. Chaque agent A_j calcule ces coûts futurs une seule fois et les stocke dans le vecteur $FC_j[\,]$.

$$FC_j(v_j) = \sum_{x_k \in \Gamma^+(x_j)} min_{v_k \in D_k} \left\{c_{jk}(v_j, v_k)\right\} \tag{2}$$

Ensuite, A_j génère une CPA (Y), s'il s'agit du premier agent dans l'ordre statique. Après, A_j ajoute son affectation de valeur (x_j, v_j) à la CPA Y^j et l'envoie via un message **ok?** à l'agent suivant A_{j+1}.

$$ok? : \langle Y^j = [(x_1, v_1), \ldots, (x_j, v_j)],\ GC(Y^j)[\,],\ UB_j \rangle$$

$GC(Y^j)[j]$, le coût garanti de la CPA Y^j au niveau j, donné par l'équation (3), est la somme des coûts des contraintes partagées entre les agents ayant des affectations de valeurs dans la CPA Y^{j-1}.

$$GC(Y^j)[j] = GC(Y^{j-1}) + LC_j(Y^j, v_j)[j-1] \tag{3}$$

$LC_j(Y^j, v_j)[h]$, le coût local d'une valeur v_j au niveau h, donné par l'équation (4), est la somme des coûts des contraintes partagées avec les agents $A_i \in \Gamma^-(x_j)$.

$$LC_j(Y^j, v_j)[h] = \sum_{(x_i, v_i) \in Y^h} c_{ij}(v_i, v_j) \mid i \leq h < j \tag{4}$$

En même temps que l'envoi d'un message **ok?**, l'agent A_j envoie des copies de la CPA, via des messages **fb?**, à tous les agents $A_k \in \mathcal{A}^+$.

$$fb? : \langle Y^j = [(x_1, v_1), \ldots, (x_j, v_j)],\ GC(Y^j)[\,],\ UB_j \rangle$$

Lors de la réception d'un message **ok?**, A_j se comporte de la même manière que l'agent émetteur.

Lors de la réception d'un message **fb?**, A_j calcule une borne inférieure, définie par l'équation (5), pour chaque valeur $v_i \in D_i$ de l'expéditeur A_i et pour chaque niveau h de la CPA reçue Y^i.

$$\begin{aligned} LB_j(Y^i)[h] = \min_{v_j \in D_j} \Big\{ & LC_j(Y^i, v_j)[h] + \\ & \sum_{m=h+1}^{i-1} \min_{v_m \in D_m} \left\{c_{mj}(v_m, v_j)\right\} + \\ & c_{ij}(v_i, v_j) + FC_j(v_j) \Big\} \end{aligned} \tag{5}$$

Une fois que le calcul des bornes inférieures est terminé, l'agent A_j les envoie à l'agent demandeur via un message **lb**.

$$lb : \langle Y^i = [(x_1, v_1), \ldots, (x_i, v_i)],\ lb_j(Y^i)[\,][\,] \rangle$$

Procédure 3 : AC* ()

```
pour chaque (A_k ∈ Γ+) faire
    ProjectBinary(x_j, x_k);
    ProjectBinary(x_k, x_j);
pour chaque (A_k ∈ Γ−) faire
    ProjectBinary(x_k, x_j);
    ProjectBinary(x_j, x_k);
ProjectUnary();
```

Lors de la réception d'un message **lb**, il enregistre les bornes inférieures reçues et calcule une borne inférieure globale, définie par l'équation (6), pour la comparer avec la borne supérieure globale UB_j. Si cette borne inférieure dépasse UB_j, l'agent A_j change son affectation de valeur par une autre dont la borne inférieure ne dépasse pas UB_j si elle existe.

$$LB(Y^j)[i] = GC(Y^j)[i] + \sum_{A_k > A_j} LB_k(Y^j)[i] \qquad (6)$$

Si elle n'existe pas, il tente d'effectuer un retour en arrière en recherchant le dernier agent affecté A_i où $LB(Y)[i-1] < UB_j$. Si un tel agent existe, A_j lui envoie un message **back**.

$$back : \langle Y^i = [(x_1, v_1), \ldots, (x_i, v_i)],\ UB_j \rangle$$

Sinon, A_j déclare la terminaison de son exécution et informe les autres via des messages **stp**.

$$stp : \langle UB_j \rangle$$

L'algorithme AFB_BJ$^+$ indique ensuite que la solution optimale est la meilleure CPA complète trouvée jusqu'à présent.

Procédure 4 : ProcessPruning(msg)

```
DelVals ← msg.DelVals;
pour chaque (A_k ∈ Γ) faire
    pour chaque (a ∈ DelVals[k]) faire
        D_k ← D_k − a;
    ProjectBinary(x_j, x_k);
    ProjectUnary();
    si (D_k is changed) alors
        DelVals[k].nbUnvisitedNbrs.decrement(−1);
    si (DelVals[k].nbUnvisitedNbrs = 0) alors
        DelVals[k].listVals.clear;
C_φ ← max{C_φ, msg.C_φ} + C_φj ; C_φj ← 0;
si (C_φ ≥ UB_j) alors
    broadcastMsg : stp(UB_j);
    end ← true;
CheckPruning();
ExtendCPA();
```

Procédure 5 : CheckPruning()

```
pour chaque (a ∈ D_j) faire
    si (c_j(a) + C_φ ≥ UB_j) alors
        D_j ← D_j − a;
        DelVals[j].listVals.add(a);
si (D_j is changed) alors
    DelVals[j].nbUnvisitedNbrs ← A_j.nbNbrs;
pour chaque (A_k ∈ Γ) faire
    ProjectBinary(x_k, x_j);
si (D_j is empty) alors
    broadcastMsg : stp(UB_j);
    end ← true;
```

3 Algorithme AFB_BJ$^+$-AC*

Dans l'algorithme AFB_BJ$^+$, les domaines des agents sont réinitialisés à chaque extension d'une nouvelle solution. Cela signifie que des recherches inutiles sont effectuées en raison de l'existence de valeurs sous-optimales dans ces domaines. Dans ce papier, nous utilisons la consistance d'arc souple (AC*) comme un outil pour éviter les recherches inutiles en supprimant les valeurs sous-optimales des domaines des agents. Pour ce faire, nous ajoutons quelques modifications à l'algorithme AFB_BJ$^+$ permettant d'appliquer la consistance d'arc souple (AC*), à savoir :

1. L'utilisation de deux copies de contraintes binaires C_{ij}. Une copie pour effectuer les calculs liés à la recherche, incluant les coûts garantis, les coûts locaux, les coûts futurs et les bornes inférieures. L'autre copie C_{ij}^{ac} est dédiée à effectuer les calculs d'AC*, incluant les projections binaires, les projections unaires et la valeur de C_ϕ. L'utilisation de deux copies des contraintes binaires permet à un agent d'éviter la réinitialisation continue de ses structures de données et d'éviter le blocage de la recherche en attendant que les autres agents l'informent des dernières mises à jour effectuées du fait de l'application d'AC*.

2. L'utilisation d'un nouveau coût garanti $GC^*(Y^j)$, défini sur la base de la copie des contraintes binaires C_{ij}^{ac}. $GC^*(Y^j)$, donné par l'équation 7, représente la somme des coûts des contraintes C_i et C_{ij}^{ac} impli-

quées dans la CPA Y^j.

$$GC^*(Y^j) = \sum_{C_i \in Y^j} c_j(v_j) + \sum_{C_{ij}^{ac} \in Y^j} c_{ij}(v_i, v_j) \qquad (7)$$

3. L'utilisation de trois nouvelles procédures pour appliquer AC*. La première procédure est AC*() (Proc. 3), utilisée uniquement dans la phase d'initialisation de l'algorithme AFB_BJ+ pour appliquer AC*. La deuxième procédure est *CheckPruning*() (Proc. 5), utilisée pour supprimer les valeurs sous-optimales et pour conserver la symétrie des contraintes binaires partagées. La troisième procédure est *ProcessPruning*() (Proc. 4), utilisée pour traiter les suppressions de valeurs effectuées et pour réappliquer AC*.
4. Le contenu de certains messages de l'algorithme AFB_BJ+ est modifié. Pour obtenir la valeur globale de C_ϕ et partager les valeurs supprimées, nous utilisons les messages **ok?** et **back**. Le choix des messages **ok?** et **back** pour partager C_ϕ et *DelVals* est dû à leur caractère séquentiel et synchrone. Le coût garanti $GC^*(Y^j)$ est ajouté aux messages **ok?** et **fb?**.

Les modifications ajoutées pour appliquer la consistance d'arc souple (AC*) dans l'algorithme AFB_BJ+ permettent d'introduire un nouvel algorithme appelé AFB_BJ+-AC* [1]. Son fonctionnement est basé sur la suppression des valeurs sous-optimales des domaines des agents d'un DCOP via l'application d'AC* et sur la propagation de ces valeurs supprimées entre les agents voisins pour générer d'autres suppressions de valeurs.

4 Description

Chaque agent A_j, exécutant l'algorithme AFB_BJ+-AC* (Proc. 6), utilise les structures de données suivantes :

— UB_j, le coût de la meilleure solution trouvée jusqu'à présent. Il représente la borne supérieure globale courante.
— v_j^*, la valeur optimale de l'agent A_j.
— Y, l'affectation partielle courante (CPA).
— C_ϕ, la valeur globale de la contrainte d'arité zéro. Elle représente la borne inférieure de toute solution du DCOP à résoudre.
— C_{ϕ_j}, la valeur de contribution de l'agent A_j dans C_ϕ globale , $C_\phi = \sum_{A_j \in \mathcal{A}} C_{\phi_j}$.
— $c_j(v_j)$, le coût unaire d'une valeur $v_j \in D_j$ de l'agent A_j.
— *DelVals*, une liste de n tableaux utilisés pour stocker les valeurs supprimées par chaque agent. Chaque tableau $DelVals[j]$ contient deux éléments, *listVals* qui est la liste des valeurs supprimées par l'agent A_j et *nbUnvisitedNeighbors* qui est le compteur associé à cette liste et représente le nombre de voisins, de l'agent A_j, qui n'ont pas encore traité cette liste.

Procédure 6 : AFB_BJ+-AC* ()

```
1  UB_j ← +∞ ;
2  v_j^* ← empty ;
3  Y ← [] ;
4  GC[i..j − 1] ← [0, ..., 0] ;
5  lb_k[0][v_j] ← min_{v_k∈D_k} {c_jk(v_j, v_k)} ;   (A_k>A_j)∧(v_j∈D_j)
6  mustSendFB ← True ;
7  C_φ ← 0; C_φj ← 0 ;
8  GC*[i..j − 1] ← [0, ..., 0] ;
9  ∀a ∈ D_j, c_j(a) ← 0 ;
10 AC*() ;
11 si (A_j = A_1) alors
12     C_φ ← C_φ + C_φj; C_φj ← 0;
13     CheckPruning() ;
14     ExtendCPA() ;
15 tant que (¬end) faire
16     msg ← getMsg() ;
17     si (msg.UB < UB_j) alors
18         UB_j ← msg.UB ; v_j^* ← v_j ;
19     si (msg.Y is stronger than Y) alors
20         Y ← msg.Y ; GC ← msg.GC ;
21         clear irrelevant lb() ;
22         reset D_j ;
23     suivant (msg.type) faire
24         cas où ok ? faire
25             mustSendFB ← True ;
26             GC* ← msg.GC* ;
27             ProcessPruning(msg) ;
28         cas où back faire
29             Y ← Y^{j−1} ;
30             ProcessPruning(msg) ;
31         cas où fb ? faire
32             GC* ← msg.GC* ;
33             pour chaque (v_j ∈ D_j) faire
34                 cost ← C_φ + GC*(Y^{j−1}) + c_j(v_j) ;
35                 si (cost ≥ UB_j) alors
36                     D_j ← D_j − v_j ;
37             sendMsg : lb (lb_j(Y^i)[], msg.Y) ;   to A_i
38         cas où lb faire
39             lb_k(Y^j) ← msg.lb ;
40             si (lb(Y^j) ≥ UB_j) alors
41                 ExtendCPA() ;
42         cas où stp faire
43             end ← true ;
```

Après l'initialisation des structures de données locales,

Procédure 7 : ExtendCPA()

1 $v_j \leftarrow argmin_{v'_j \in D_j} \left\{ lb(Y \cup (x_j, v'_j)) \right\}$;
2 **si** $(lb(Y \cup (x_j, v_j)) \geq UB_j) \vee (C_\phi + GC^*(Y^{j-1}) + c_j(v_j) \geq UB_j)$ **alors**
3 **pour** $i \leftarrow j-1$ ***to*** 1 **faire**
4 **si** $(lb(Y)[i-1] < UB_j)$ **alors**
5 sendMsg : **back**$_{\text{to } A_i}(Y^i, UB_j, DelVals, C_\phi)$;
  return ;
6 broadcastMsg : **stp**(UB_j) ;
7 $end \leftarrow true$;
8 **sinon**
9 $Y \leftarrow \left\{ Y \cup (x_j, v_j) \right\}$;
10 **si** $(var(Y) = X)$ **alors**
11 $UB_j \leftarrow GC(Y)$;
12 $v^*_j \leftarrow v_j$;
13 $Y \leftarrow Y^{j-1}$;
14 $CheckPruning()$;
15 $ExtendCPA()$;
16 **sinon**
17 sendMsg : **ok ?**$_{\text{to } A_{j+1}}(Y, GC, UB_j, DelVals, C_\phi, GC^*)$;
18 **si** $(mustSendFB)$ **alors**
19 sendMsg : **fb ?**$_{\text{to } A_k, k>j}(Y, GC, UB_j, GC^*)$;
20 $mustSendFB \leftarrow false$;

l'agent A_j commence l'exécution d'AC* (Proc. 6, ligne 10) en appliquant sur sa variable (Proc. 3) une projection binaire puis une projection unaire pour satisfaire AC* et en appliquant sur chacun de ses voisins une autre projection binaire pour garantir la symétrie des contraintes binaires partagées. Puis, s'il s'agit du premier agent dans l'ordre statique considéré, il met à jour C_ϕ en y ajoutant sa contribution C_{ϕ_j}. Il appelle ensuite la procédure *CheckPruning*() (Proc. 5) pour filtrer son domaine D_j en supprimant les valeurs inconsistantes. Après, il appelle la procédure *ExtendCPA*() (Proc. 7) pour commencer la phase de recherche en générant une affectation partielle courante (CPA) Y.

Lors de l'appel de la procédure *CheckPruning*() (Proc. 5), l'agent A_j supprime de son domaine D_j toute valeur $v_j \in D_j$ ayant un coût unaire $c_j(v_j)$ plus C_ϕ dépasse UB_j (Proc. 5, lignes 2 - 3). Ensuite, A_j enregistre ces valeurs supprimées dans la liste $DelVals[j].listVals$ et réinitialise le compteur de ses voisins $DelVals[j].nbUnvisitedNeighbors$ (Proc. 5, lignes 4 - 6). Ce compteur lui permet de savoir combien de voisins restants n'ont pas encore traité sa liste de valeurs supprimées. Après chaque suppression de valeur, l'agent A_j réinitialise ce compteur en lui attribuant son nombre de voisins. Chaque fois que cette liste est traitée par un voisin, ce compteur est décrémenté d'un pas jusqu'à atteindre zéro, ce qui indique que la liste a été traitée par tous les voisins (Proc. 4, lignes 7 - 10). Une fois le filtrage de domaine terminé, A_j effectue une projection binaire pour conserver la symétrie des contraintes binaires partagées (Proc. 5, ligne 8). Enfin, si A_j s'est assuré que son domaine est vide, il arrête son exécution et informe les autres via des messages **stp** (Proc. 5, lignes 9 - 11).

Lors de l'appel de la procédure *ExtendCPA*() (Proc. 7), l'agent A_j affecte la valeur v_j ayant la meilleure borne inférieure à sa variable x_j et ajoute cette affectation de valeur à la CPA Y^j (Proc. 7, ligne 1). Si cette affectation de valeur satisfait l'une des conditions d'inconsistance (Proc. 7, ligne 2), A_j essaie de retourner aux agents précédents en les parcourant un par un jusqu'à trouver le dernier agent A_i dans lequel la borne inférieure de la CPA, Y, dépasse UB_j (Proc. 7, lignes 3 - 5). Si un tel agent existe, A_j lui envoie un message **back**. Sinon, A_j arrête son exécution et informe les autres via des messages **stp** (Proc. 7, lignes 6 - 7). Si l'affectation de valeur (x_j, v_j) est valide, A_j affirme l'extension de la CPA Y avec cette affectation de valeur (Proc. 7, ligne 9). Si la CPA Y devient une affectation complète (Proc. 7, ligne 10), l'agent A_j met à jour UB_j en fonction du coût de la nouvelle solution obtenue (Proc. 7, ligne 11). Ensuite, pour continuer la recherche, l'agent A_j appelle la procédure *CheckPruning*() pour filtrer son domaine D_j en fonction de la nouvelle UB_j obtenue et essaie à nouveau de répéter les étapes de la procédure *ExtendCPA*() (Proc. 7, lignes 11 - 15). Si l'agent A_j n'a pas encore atteint une affectation complète, il envoie la CPA Y étendue à l'agent suivant dans l'ordre (A_{j+1}) via un message **ok?** afin de poursuivre son extension. Puis, il l'envoie aux agents non affectés via des messages **fb?** pour demander leurs estimations à ce sujet (Proc. 7, lignes 17 - 19).

Lors de la réception d'un message **ok?** (Proc. 6, lignes 24 - 27), A_j donne l'autorisation d'envoyer des messages **fb?**. Puis, il met à jour GC^* par celui reçu. Enfin, il appelle la procédure *ProcessPruning*() (Proc. 4) pour traiter les suppressions de valeurs effectuées par ses voisins et pour réappliquer AC*.

$$ok? : \langle Y^j, GC[\,], UB_j, DelVals, C_\phi, GC^*[\,] \rangle$$

Lors de l'appel de la procédure *ProcessPruning*() (Proc. 4), l'agent A_j met à jour sa liste locale de valeurs supprimées $DelVals$ (Proc. 4, ligne 1). Puis, il met à jour les domaines D_k de ses voisins A_k un par un en supprimant toutes les valeurs supprimées par chaque voisin A_k afin de conserver les mêmes domaines de ces agents (Proc. 4, lignes 2 - 4). Ensuite, A_j applique à nouveau AC* (Proc. 4, lignes 5 - 6). Après, A_j décrémente le nombre de voisins restants, $DelVals[k].nbUnvisitedNeighbors$, de l'agent A_k et vérifie ensuite s'il s'agit du dernier voisin visité de cet

agent A_k pour réinitialiser la liste des valeurs supprimées, $DelVals[k].listVals$, de cet agent A_k (Proc. 4, lignes 7 - 10). Ensuite, A_j met à jour la valeur globale de C_ϕ. Si la valeur de C_ϕ dépasse UB_j, l'agent A_j arrête son exécution et informe les autres via des messages **stp** (Proc. 4, lignes 12 - 14). Enfin, A_j appelle la procédure *CheckPruning*() pour filtrer son domaine et essaie d'étendre la CPA Y reçue en appelant la procédure *ExtendCPA*() (Proc. 4, lignes 15 - 16).

Lors de la réception d'un message **fb?**, il met à jour GC^* par celui reçu et filtre son domaine D_j en supprimant, temporairement, toute valeur $v_j \in D_j$ pouvant générer une solution sous-optimale lors de l'extension de la CPA Y reçue (Proc. 6, lignes 34 - 36). Enfin, il calcule les bornes inférieures appropriées à la CPA Y reçue (5) et les envoie à l'expéditeur via un message **lb** (Proc. 6, ligne 37).

$$fb? : \langle Y^j,\ GC[\,],\ UB_j,\ GC^*[\,]\rangle$$

Lors de la réception d'un message **lb**, il enregistre les bornes inférieures reçues (Proc. 6, ligne 39) et vérifie si la nouvelle borne inférieure globale de la CPA Y (6) dépasse UB_j. Dans un tel cas, A_j appelle la procédure *ExtendCPA*() pour changer son affectation de valeur.

$$lb : \langle Y^i,\ lb_j(Y^i)[\,][\,]\rangle$$

Lors de la réception d'un message **back** (Proc. 6, lignes 3 - 5), il efface son affectation de valeur (x_j, v_j) de la CPA Y pour se préparer à une autre affectation de valeur plus appropriée et correspondant au coût de la meilleure solution trouvée jusqu'à présent UB_j. Pour ce faire, A_j appelle la procédure *ProcessPruning*() (Proc. 4).

$$back : \langle Y^i,\ UB_j,\ DelVals,\ C_\phi\rangle$$

4.1 Correction

Théorème 1. *L'algorithme AFB_BJ$^+$-AC* est garanti de se terminer et de calculer la solution optimale.*

Démonstration 1. *L'algorithme AFB_BJ$^+$-AC* fonctionne de la même manière que l'algorithme AFB_BJ$^+$ [12]. La nouveauté réside dans le filtrage de domaine effectué par chaque agent sur la base d'AC* avant chaque affectation de valeur. Ce filtrage permet à chaque agent de supprimer les valeurs inconsistantes de son domaine. Ainsi, pour prouver la validité de l'algorithme AFB_BJ$^+$-AC*, il suffit de prouver que la validité de l'algorithme AFB_BJ$^+$ reste garantie avec l'existence de ces filtrages de domaines (Proc. 7, ligne 2), (Proc. 6, ligne 34) et (Proc. 5, ligne 2). En d'autres termes, il faut prouver que les valeurs supprimées lors de ces filtrages de domaines ne sont pas des valeurs optimales.*

À toutes les étapes, il n'existe aucune redondance de coûts lors des calculs effectués pour appliquer AC*. Cela apparaît à plusieurs niveaux de l'algorithme AFB_BJ$^+$-AC*. Tout d'abord, l'ajout d'une autre projection binaire (Proc. 3, lignes 3, 5) (Proc. 5, ligne 8) permet de garantir la symétrie des contraintes binaires partagées entre les agents. Deuxièmement, le partage continu des valeurs supprimées entre les agents permet de garantir la symétrie des domaines partagés (Proc. 7, lignes 5, 17). Enfin, l'exécution des mises à jour des contraintes et des suppressions de valeurs de manière synchrone permet de prendre en compte tous les coûts sans augmentation ni diminution (Proc. 6, lignes 27, 30).

Les techniques d'AC* ne sont que des transformations de coûts entre les contraintes du problème. La validité de ces transformations a déjà été prouvée par [9]. Ainsi, les valeurs supprimées lors des filtrages de domaines sont des valeurs sous-optimales.

En observant que le nombre de CPAs générées par l'algorithme AFB_BJ$^+$-AC* lors de la recherche est un nombre fini et que l'évaluation de chaque CPA ne peut jamais conduire à une boucle infinie, on peut simplement en déduire que l'algorithme AFB_BJ$^+$-AC* se termine [12].

5 Résultats expérimentaux

Nous avons comparé expérimentalement AFB_BJ$^+$-AC* après et avant l'utilisation d'AC*[1], et avec les algorithmes BnB-Adopt$^+$-AC* [7] et BnB-Adopt$^+$-DP2 [2] en utilisant le simulateur DisChoco 2.0 [13]. Quatre types de DCOPs ont été utilisés dans ces expériences : les problèmes Max-DisCSPs binaires, les DCOPs binaires, les problèmes de planification des réunions et les réseaux de capteurs. La comparaison a été effectuée sur la base du nombre de messages échangés ($msgs$) et le nombre de tests non simultanées de contraintes ($ncccs$) effectués.

5.1 Max-DisCSPs aléatoires binaires

Les problèmes Max-DisCSPs binaires [12] sont des problèmes générés aléatoirement et uniformément en se basant sur un cadre défini par un 4-uplet (n, d, p_1, p_2), où n est le nombre de variables/agents, d est le nombre de valeurs dans chaque domaine d'une variable, p_1 est la probabilité d'existence d'une contrainte entre deux variables et p_2 est la probabilité d'existence d'une violation d'une contrainte.

Nous avons évalué deux classes d'instances de problème Max-DisCSP binaire, classe clairsemée définie par (n = 10, d = 10, p_1 = 0.4, p_2 = 0.6..0.9) et classe dense définie par (n = 10, d = 10, p_1 = 0.7, p_2 = 0.6 .. 0.9). En ce qui concerne p_2, nous avons essayé de changer sa valeur entre 0.6 et 0.9 par pas de 0.1. Le coût de chaque combinaison des valeurs des variables d'une contrainte est choisi parmi l'ensemble $\{0, 1\}$. Pour chaque paire (p_1, p_2), nous avons évalué une moyenne de 50 instances [12].

5.2 DCOPs aléatoires binaires

Les problèmes DCOPs binaires [12] sont des problèmes générés aléatoirement et uniformément en se basant sur un cadre défini par un triplet (n, d, p_1), qui est le même triplet utilisé pour les problèmes Max-DisCSPs binaires, où n est le nombre de variables/agents, d est le nombre de valeurs dans chaque domaine d'une variable, p_1 est la probabilité d'existence d'une contrainte entre deux variables.

Nous avons évalué une classe d'instances de problème DCOP binaire définie par ($n = 10$, $d = 10$, $p_1 = 0.4 .. 0.8$), où nous avons essayé de changer la valeur de p_1 entre 0.4 et 0.8 par pas de 0.1. Le coût de chaque combinaison de valeurs des variables d'une contrainte est choisi parmi l'ensemble $\{0, ..., 100\}$. Pour chaque valeur de p_1, nous avons évalué une moyenne de 50 instances.

5.3 Planification des réunions

La planification des réunions [12] est un problème où chaque personne, parmi un groupe de personnes, cherche à faire des réunions avec une ou plusieurs personnes, de ce groupe, à condition de respecter les calendriers des personnes du groupe.

Chaque problème de planification des réunions est défini par :

— Un ensemble de réunions représentant les variables du problème, chacune se produisant dans un créneau horaire spécifié représentant les valeurs possibles pour cette réunion.
— Un ensemble de personnes représentant les participants aux réunions, chacune ayant son propre calendrier personnel.
— Un ensemble de contraintes, chacune reliant des réunions partageant des participants.

Nous avons évalué 4 cas A, B, C et D, chacun représentant un scénario hiérarchique différent [10]. Pour chaque cas, nous avons évalué une moyenne de 30 instances.

5.4 Réseau de capteurs

Le problème de réseau de capteurs [12, 3] est un problème dans lequel un ensemble de capteurs fixes essaie de suivre un ensemble de cibles mobiles. L'objectif est donc de suivre toutes les cibles à l'aide de ces capteurs. Pour ce faire, tous les capteurs du réseau doivent coopérer pour pouvoir les suivre. Pour que la cible puisse être suivie avec précision, au moins un ensemble de trois capteurs doivent la suivre. Cependant, chaque capteur peut suivre au plus une cible. Ainsi, une solution consiste à affecter trois capteurs distincts à chaque cible.

Chaque problème de réseau de capteurs est défini par :

— Un ensemble de cibles mobiles représentant les variables du problème.

	nccccs				*msg*			
p_2	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9
AFB_BJ$^+$	2890	8116	15702	29573	292	892	1731	3375
AFB_BJ$^+$-AC*	**2700**	**7416**	**14635**	**27713**	**284**	**873**	**1673**	**3176**
BnB-Adopt$^+$-AC*	3872	17860	117168	705734	586	3582	22642	114082
BnB-Adopt$^+$-DP2	4368	17769	85270	202398	616	3434	16528	38474

Table 1 – Nombre total de *msg* envoyés et *nccccs* exécutés sur Max-DisCSPs aléatoires binaires, où $p_1 = 0.4$

	nccccs $\times 10^2$				*msg* $\times 10^2$			
p_2	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9
AFB_BJ$^+$	707	1228	2304	3197	64	114	208	293
AFB_BJ$^+$-AC*	**687**	**1215**	**2272**	**3000**	**62**	**113**	**204**	**273**
BnB-Adopt$^+$-AC*	9918	65595	314164	730672	1134	6862	30127	59581
BnB-Adopt$^+$-DP2	9555	62191	262828	503293	1083	6461	25271	44228

Table 2 – Nombre total de *msg* envoyés et *nccccs* exécutés sur Max-DisCSPs aléatoires binaires, où $p_1 = 0.7$

— Un ensemble de capteurs fixes où chaque combinaison de trois capteurs distincts constitue une valeur possible pour une cible mobile.
— Un ensemble de contraintes, chacune reliant des cibles adjacentes.

Nous avons évalué 4 cas A, B, C et D, chacun représentant un scénario topologique différent [10]. Pour chaque cas, nous avons évalué une moyenne de 30 instances.

5.5 Description

Les deux tables 1 et 2 montrent les résultats des expériences effectuées sur les problèmes Max-DisCSPs aléatoires binaires respectivement dans deux cas différents, le cas d'un graphe dense ($p_1 = 0.7$) où chaque problème implique un grand nombre de contraintes et le cas d'un graphe clairsemé ($p_1 = 0.4$) où chaque problème implique un petit nombre de contraintes. Ces résultats montrent que la performance de l'algorithme AFB_BJ$^+$-AC* est très proche de celle de l'algorithme d'origine AFB_BJ$^+$ dans le cas d'un graphe clairsemé et s'améliore si le graphe devient dense, que ce soit pour le nombre de tests (*nccccs*) ou le nombre de messages (*msg*). En ce qui concerne les algorithmes BnB-Adopt$^+$-AC* et BnB-Adopt$^+$-DP2, leurs résultats montrent une performance faible comparée aux algorithmes AFB_BJ$^+$ et AFB_BJ$^+$-AC*.

La table 3 représente les résultats des expériences effectuées sur les problèmes DCOPs aléatoires binaires. L'algorithme AFB_BJ$^+$-AC* reste le meilleur au niveau de performance par rapport aux autres algorithmes, mais son taux d'amélioration s'accroît lentement par rapport à l'algorithme d'origine AFB_BJ$^+$.

La table 4 affiche les résultats des expériences effectuées sur les problèmes de planification des réunions. La comparaison des résultats montre que l'algorithme AFB_BJ$^+$-AC* a réussi de réduire généralement le nombre de messages et le nombre de tests à près de la moitié par rapport à l'algorithme d'origine AFB_BJ$^+$. En ce qui concerne les algorithmes BnB-Adopt$^+$-AC* et BnB-Adopt$^+$-DP2, la

	$nccc s \times 10^3$					$msg \times 10^3$				
p_1	0.4	0.5	0.6	0.7	0.8	0.4	0.5	0.6	0.7	0.8
AFB_BJ$^+$	33	82	197	251	340	4	9	18	23	30
AFB_BJ$^+$-AC*	**31**	**80**	**189**	**246**	**333**	**3**	**8**	**17**	**22**	**29**
BnB-Adopt$^+$-AC*	618	2974	18507	47831	58861	113	468	2240	4767	4346
BnB-Adopt$^+$-DP2	190	1050	6531	17253	24881	36	174	835	1814	2081

Table 3 – Nombre total de *msg* envoyés et *ncccs* exécutés sur DCOPs aléatoires binaires

	ncccs				*msg*			
case	A	B	C	D	A	B	C	D
AFB_BJ$^+$	5194	5104	2815	2753	383	992	567	659
AFB_BJ$^+$-AC*	**2294**	**2172**	**1134**	**1111**	**290**	**773**	**341**	**373**
BnB-Adopt$^+$-AC*	99687	45329	31749	14051	6503	5935	5063	4592
BnB-Adopt$^+$-DP2	6261	4517	2094	1667	700	794	510	530

Table 4 – Nombre total de *msg* envoyés et *ncccs* exécutés sur des problèmes de planification des réunions

supériorité de l'algorithme AFB_BJ$^+$-AC* est nette et expressive, surtout si on le compare avec l'algorithme BnB-Adopt$^+$-DP2 qui est connu pour sa supériorité dans la résolution de tels problèmes.

La table 5 présente les résultats des expériences effectuées sur les problèmes de réseau de capteurs. Ces résultats montrent que l'algorithme AFB_BJ$^+$-AC* améliore en général la performance de l'algorithme d'origine AFB_BJ$^+$ d'un taux proche de la moitié. En comparant l'algorithme AFB_BJ$^+$-AC* avec les algorithmes BnB-Adopt$^+$-AC* et BnB-Adopt$^+$-DP2, on peut dire, d'après les résultats, que l'algorithme AFB_BJ$^+$-AC* reste le meilleur par rapport à l'algorithme BnB-Adopt$^+$-AC*. Mais, en ce qui concerne l'algorithme BnB-Adopt$^+$-DP2, la supériorité de cet algorithme sur tous les autres algorithmes est très claire, ce qui fait de cet algorithme le pionnier dans la résolution de tels problèmes.

5.6 Discussion

En regardant tous les résultats décrits ci-dessus, on peut en déduire que l'algorithme AFB_BJ$^+$-AC* fonctionne généralement mieux que l'algorithme d'origine AFB_BJ$^+$. Cela est dû aux techniques de consistance d'arc souple (AC*) intégrées qui permettent aux agents de supprimer les valeurs sous-optimales de leurs domaines.

Les résultats obtenus pour les problèmes aléatoires, qu'il s'agisse de Max-DisCSPs binaires ou de DCOPs binaires, montrent une légère amélioration de l'algorithme AFB_BJ$^+$-AC* par rapport à l'algorithme AFB_BJ$^+$. La raison principale est due au comportement d'AC*, dans certaines instances de ces problèmes, qui produit une valeur globale de C_ϕ n'est pas assez proche de la borne supérieure globale UB pour effectuer des suppressions.

En ce qui concerne les problèmes réels, qu'il s'agisse des problèmes de planification des réunions ou de réseau de capteurs, leurs résultats montrent une meilleure performance de l'algorithme AFB_BJ$^+$-AC* que les problèmes aléatoires. Cela est dû au comportement d'AC* dans les

	ncccs				*msg*			
case	A	B	C	D	A	B	C	D
AFB_BJ$^+$	8367	7309	2390	4417	2993	2594	321	1293
AFB_BJ$^+$-AC*	2763	2432	1044	1768	2659	1950	217	960
BnB-Adopt$^+$-AC*	2881	5281	2861	6111	827	1220	742	1775
BnB-Adopt$^+$-DP2	**1000**	**1025**	**1042**	**1208**	**195**	**226**	**187**	**309**

Table 5 – Nombre total de *msg* envoyés et *ncccs* exécutés sur des problèmes de réseau de capteurs

instances de ces problèmes, où la valeur globale de C_ϕ qui s'améliore parallèlement avec la borne supérieure globale UB entraîne des suppressions continues tout au long de la période de recherche.

Cette différence entre les problèmes aléatoires et les problèmes réels dans le comportement des techniques d'AC* dans l'algorithme AFB_BJ$^+$-AC* est due à la structure de chacun d'eux, soit au niveau des contraintes entre leurs variables, soit au niveau des coûts liés à ces contraintes. Les problèmes aléatoires sont basés sur une structure complètement aléatoire, dans laquelle les instances sont générées de manière aléatoire, qu'il s'agisse de contraintes entre leurs variables ou de coûts de ces contraintes. En revanche, les problèmes réels reposent sur une structure bien définie, dans laquelle les instances sont générées sur la base des règles rigides lors de la définition des contraintes et de leurs coûts. L'impact de la structure d'un problème apparaît dans la génération d'une meilleure valeur de C_ϕ et, par la suite, dans la génération de plus de suppressions.

En général, la structure du problème n'est pas le seul facteur qui influence la performance d'un algorithme DCOP, il existe d'autres facteurs qui influencent cette performance. Parmi lesquels, la stratégie d'arrangement des agents du problème, la stratégie de sélection des valeurs dans les agents, la stratégie d'affectation des variables, la densité du graphe de contraintes du problème, etc. Cela ressort clairement des résultats obtenus pour les algorithmes BnB-Adopt$^+$-AC* et BnB-Adopt$^+$-DP2, où la faible performance de ces algorithmes dans les problèmes aléatoires, tels que Max-DisCSPs binaires et DCOPs binaires, par rapport aux algorithmes AFB_BJ$^+$ et AFB_BJ$^+$-AC* est due principalement à la nature asynchrone qui pousse ces algorithmes à utiliser un grand nombre de messages et à effectuer un grand nombre de tests de contraintes pour résoudre un problème. Cela ressort également des résultats obtenus pour la performance de l'algorithme BnB-Adopt$^+$-AC* dans les problèmes de planification des réunions, où la nature asynchrone a une influence plus grande sur la performance de cet algorithme que les techniques d'AC*. La situation est différente dans l'algorithme BnB-Adopt$^+$-DP2 où l'existence de l'heuristique $DP2$ a eu un impact plus important sur sa performance, ce qui est claire dans ses résultats qui sont proches de ceux de l'algorithme AFB_BJ$^+$-AC*. En ce qui concerne les problèmes de réseau de capteurs, les résultats montrent une forte performance de l'algorithme BnB-Adopt$^+$-DP2 par rapport à tous les algo-

rithmes. Ceci est dû principalement à la structuration de pseudo-arbre adoptée par cet algorithme et qui est mieux adaptée à la structure de ce type de problème. De plus, l'heuristique *DP*2 utilisée par cet algorithme a eu un impact significatif sur sa performance en rendant les agents plus précis dans le choix de leurs valeurs.

6 Conclusion

Dans cet article, nous avons présenté l'algorithme AFB_BJ$^+$-AC* qui combine l'algorithme AFB_BJ$^+$ avec les techniques de consistance d'arc souple (AC*). AFB_BJ$^+$-AC* est un algorithme de résolution de DCOPs qui repose sur le mécanisme de recherche adopté par l'algorithme d'origine AFB_BJ$^+$, renforcé par la consistance d'arc souple (AC*) qui permet de supprimer des domaines des agents d'un DCOP les valeurs sous-optimales. Nous avons fait une évaluation de l'efficacité de l'algorithme AFB_BJ$^+$-AC* dans la résolution de différents DCOPs, incluant les problèmes Max-DisCSPs binaires, les DCOPs binaires, la planification des réunions et les réseaux de capteurs. Les résultats obtenus ont montré que la performance de l'algorithme AFB_BJ$^+$-AC* est généralement meilleure que celle de l'algorithme AFB_BJ$^+$, en particulier dans les problèmes réels. L'une des perspectives de recherche est de combiner AC* avec les heuristiques d'arrangement des agents.

Références

[1] Adrdor, Rachid, Redouane Ezzahir et Lahcen Koutti: *Connecting AFB_BJ$^+$ with soft arc consistency*. International Journal of Computing and Optimization, 5 no. 1 :9–20, 2018. https://doi.org/10.12988/ijco.2018.857.

[2] Ali, Syed, Sven Koenig et Milind Tambe: *Preprocessing techniques for accelerating the DCOP algorithm ADOPT*. Dans *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1041–1048. ACM, 2005.

[3] Béjar, Ramón, Carmel Domshlak, Cèsar Fernández, Carla Gomes, Bhaskar Krishnamachari, Bart Selman et Magda Valls: *Sensor networks and distributed CSP : communication, computation and complexity*. Artificial Intelligence, 161(1-2) :117–147, 2005.

[4] Fioretto, Ferdinando, Enrico Pontelli et William Yeoh: *Distributed constraint optimization problems and applications : A survey*. Journal of Artificial Intelligence Research, 61 :623–698, 2018.

[5] Gershman, Amir, Amnon Meisels et Roie Zivan: *Asynchronous forward bounding for distributed COPs*. Journal of Artificial Intelligence Research, 34 :61–88, 2009.

[6] Grinshpoun, Tal, Tamir Tassa, Vadim Levit et Roie Zivan: *Privacy preserving region optimal algorithms for symmetric and asymmetric DCOPs*. Artificial Intelligence, 266 :27–50, 2019.

[7] Gutierrez, Patricia et Pedro Meseguer: *Improving BnB-ADOPT+-AC*. Dans *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 273–280, 2012.

[8] Hirayama, Katsutoshi et Makoto Yokoo: *Distributed partial constraint satisfaction problem*. Dans *International Conference on Principles and Practice of Constraint Programming*, pages 222–236. Springer, 1997.

[9] Larrosa, Javier et Thomas Schiex: *In the quest of the best form of local consistency for weighted CSP*. Dans *IJCAI*, tome 3, pages 239–244, 2003.

[10] Maheswaran, Rajiv T, Milind Tambe, Emma Bowring, Jonathan P Pearce et Pradeep Varakantham: *Taking DCOP to the real world : Efficient complete solutions for distributed multi-event scheduling*. Dans *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 310–317. IEEE Computer Society, 2004.

[11] Modi, Pragnesh Jay, Wei Min Shen, Milind Tambe et Makoto Yokoo: *ADOPT : Asynchronous distributed constraint optimization with quality guarantees*. Artificial Intelligence, 161(1-2) :149–180, 2005.

[12] Wahbi, Mohamed, Redouane Ezzahir et Christian Bessiere: *Asynchronous forward bounding revisited*. Dans *International Conference on Principles and Practice of Constraint Programming*, pages 708–723. Springer, 2013.

[13] Wahbi, Mohamed, Redouane Ezzahir, Christian Bessiere et El Houssine Bouyakhf: *DisChoco 2 : A platform for distributed constraint reasoning*. Proceedings of DCR, 11 :112–121, 2011.

Actes JIAF 2020

Extension possibiliste de la logique de description $\mathcal{EL}$

Rym MOHAMED[1] **Zied BOURAOUI**[2]
Zied LOUKIL[1] **Faiez GARGOURI**[1]

[1] MIRACL Laboratory, ISIMS, Sfax, Tunisia
[2] CRIL, Univ Artois et CNRS, France

rymmohammed2@gmail.com bouraoui@cril.fr
zied.loukil@isims.usf.tn faiez.gargouri@isims.usf.tn

Résumé

Dans de multiples situations, les informations provenant de différentes sources sont souvent affectées d'incertitude et d'imprécision. La représentation de ces informations donne généralement lieu à une base de connaissances stratifiée où chaque information attachée a un certain degré de priorité. Pour raisonner avec de telles connaissances prioritaires d'une manière efficace, nous proposons une extension de $\mathcal{EL}$, une famille de la logique de description légère dans le cadre de la théorie des possibilités. Cette théorie fournit un cadre très naturel pour représenter et résumer en présence d'incertitude ordinale, qualitative ou des préférences et des priorités entre les informations. Nous introduisons d'abord la syntaxe et la sémantique de la logique $\mathcal{EL}$ possibiliste, puis définissons les principales tâches de raisonnement. Nous montrons en particulier que ces tâches restent traitables dans l'extension que nous proposons.

Abstract

In different situations, information coming from different sources are often affected with uncertainty and imprecision. Representing such information generally gives rise to a prioritized (i.e. stratified) knowledge base. To reason with such prioritized knowledge in a principled way, we propose an extension of $\mathcal{EL}$ description logics within possibility theory, which provides a very natural framework to deal with ordinal, qualitative uncertainty, preferences and priorities. We first introduce the syntax and semantics of possibilistic $\mathcal{EL}$, and then provide the main related reasoning tasks. We show in particular that these tasks remain tractable in possibilistic $\mathcal{EL}$.

1 Introduction

Les connaissances structurées définies entre des concepts et des relations entre objets jouent un rôle très important dans de nombreuses applications telles que la recherche d'information [9, 15], le traitement automatique du langage naturel [5, 13], la bio-informatique [12, 16] et le web sémantique. Les ontologies offrent un cadre puissant pour représenter de telles connaissances structurées. Elles sont généralement exprimées à l'aide des logiques de descriptions [1], et exprimées en deux parties : une partie contenant des connaissances génériques, c.-à-d. des relations sémantiques entre les concepts et les relations, et une autre contenant les données, c.-à-d. les instances qui appartiennent à quels concepts (ou relations). Les logiques de descriptions fournissent les bases du langage d'ontologie web *OWL2* [1], et ses profils *OWL2-QL*, *OWL2-EL* et *OWL2-RL*.

Lors de ces dernières années, un intérêt croissant a été consacré à l'utilisation du fragment *OWL2-EL*, qui est basé sur une famille de la logique de description légère appelée $\mathcal{EL}$ [2, 3]. $\mathcal{EL}$ offre un niveau raisonnable d'expressivité lors de l'expression des connaissances ontologiques et garantit la tractabilité du processus du raisonnement en particulier les tâches de subsomption et de vérification d'instance.

Dans différentes applications, l'information est fournie avec incertitude. Pour gérer le problème de l'incertitude, quelques travaux ont été proposés pour étendre les logiques de descriptions dans le cadre de la théorie des probabilités (par exemple [14]), de la logique floue (par exemple [7]) et de la théorie des possibilités (par exemple [6, 8, 10]).

Ce travail préliminaire s'accentue sur la gestion de l'incertitude qualitative, ce type d'incertitude peut être due aux informations fournies par plusieurs sources caractérisées par un pré-ordre total entre elles reflétant leur fiabilité, ou

1. https ://www.w3.org/TR/owl2-overview/

lorsqu'il existe une préférence entre les informations fournies en fonction de leur niveau de priorité. La représentation de ces informations donne généralement lieu à une base de connaissances stratifiée. Pour raisonner avec de telles connaissances de manière efficace, nous proposons dans cet article des méthodes basées sur la théorie des possibilités [11]. Cette théorie offre un cadre naturel pour traiter des incertitudes ordinales, qualitatives, des préférences et des priorités.

La suite de cet article est organisée comme suit : dans la section 2, nous rappelons la logique de description $\mathcal{EL}$, puis nous introduisons les principales notions de la théorie des possibilités. Dans la section 3, nous définissons la syntaxe et la sémantique de l'extension $\mathcal{EL}^+_\perp$ possibiliste, ainsi que les algorithmes nécessaires pour raisonner dans cette extension. L'article sera achevé par une conclusion.

Syntaxe	Sémantique
A	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
r	$r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
a	$a^\mathcal{I} \in \Delta^\mathcal{I}$
$\top$	$\Delta^\mathcal{I}$
$\perp$	$\emptyset$
$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
$\exists r.C$	$\{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I}$ tel que $(x,y) \in r^\mathcal{I}$ et $y \in C^\mathcal{I}\}$
$\{a\}$	$\{a^\mathcal{I}\}$
$r \circ s$	$\{<x,y> \mid \exists z \in \Delta^\mathcal{I}$ tel que $<x,z> \in r^\mathcal{I}$ et $<z,y> \in s^\mathcal{I}\}$

TABLE 1 – La syntaxe et la sémantique des expression de concepts et de rôles .

2 Préliminaires

Dans cette section, nous rappelons $\mathcal{EL}$, puis nous définissons les principales notions de la théorie des possibilités reformulées dans un cadre de logique de descriptions.

2.1 La famille $\mathcal{EL}$ de la logique de descriptions

Syntaxe. Soient N_C, N_R et N_I trois ensembles disjoints avec N_C un ensemble fini de concepts atomiques, N_R un ensemble fini de rôles atomiques et N_I un ensemble fini d'individus. Les expressions de la logique $\mathcal{EL}$ sont construites selon la syntaxe suivante :

$$C, D \rightarrow \top \mid A \mid C \sqcap D \mid \exists r.C$$

avec $A \in N_C$, $r \in N_R$.

Une ontologie (ou base de connaissances) $\mathcal{EL}$ consiste en un ensemble des relations de subsomption de concept (General Concept Inclusion GCI) de la forme $C \sqsubseteq D$, ce qui signifie que C est plus spécifique que D ou simplement C est subsumé par D, un ensemble d'équivalence de concept de la forme $C \equiv D$, qui est une notation pour les deux GCIs $C \sqsubseteq D$ et $D \sqsubseteq C$, un ensemble d'assertions de concept de la forme $C(a)$, avec C un concept atomique et un ensemble d'assertions de rôle de la forme $r(a,b)$ avec r un rôle atomique.

Plusieurs extensions de $\mathcal{EL}$ ont été proposées. Par exemple, $\mathcal{EL}^+$ étend $\mathcal{EL}$ avec l'inclusion de rôles de la forme $s \sqsubseteq r$ et la composition de rôles de la forme $r_1 \circ ... \circ r_n \sqsubseteq s$ où $r \circ s$ est l'expression de composition de rôles. La logique $\mathcal{EL}^+_\perp$ étend $\mathcal{EL}^+$ en permettant l'utilisation du concept vide $\perp$ dans l'expression de la logique de descriptions. La logique de descriptions $\mathcal{ELO}^+_\perp$étend $\mathcal{EL}^+_\perp$ avec l'utilisation de concepts nominaux de la forme $\{a\}$. Enfin, l'extension de $\mathcal{ELO}^+_\perp$ avec des domaines concrets $\mathcal{D} = (\Delta^\mathcal{D}, \mathcal{P}^\mathcal{D})$, avec $\Delta^\mathcal{D}$ un ensemble non-vide et $\mathcal{P}^\mathcal{D}$ un prédicats [2], des restrictions, la transitivité de rôles et la réflexivité de rôles est notée par $\mathcal{EL}^{++}$, et elle est en effet le noyau de la spécification *OWL-EL*. Dans cet article, nous nous concentrons sur $\mathcal{EL}^+_\perp$ car il est le bloc principal du fragment *OWL-EL*, ce fragment est assez expressif pour exprimer par exemple la transitivité de rôle en considérant $r \circ r \sqsubseteq r$ ou les axiomes qui sont disjoint notés par $C \sqcap D \sqsubseteq \perp$. Pour plus de détails sur la famille $\mathcal{EL}$ et ses extensions, nous nous référons à [2–4].

Sémantique. La sémantique de la logique $\mathcal{EL}$ est définie en terme d'interprétations. Une interprétation $\mathcal{I}$ est un couple $\mathcal{I} = (\Delta^\mathcal{I}, .^\mathcal{I})$, où le domaine d'interprétation $\Delta^\mathcal{I}$ est un ensemble non vide. La fonction $.^\mathcal{I}$ affecte un élément $a^\mathcal{I} \in \Delta^\mathcal{I}$ à chaque individu $a^\mathcal{I} \in N_I$, un sous-ensemble $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ à chaque concept atomique $A \in N_C$ et une relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ à chaque rôle atomique $r \in N_R$. Pour plus de détails sur la sémantique des autres concepts et rôles voir tableau 1.

Une interpretation $\mathcal{I}$ est modèle de, (ou satisfait) un axiome terminologique ϕ, avec ϕ de la forme suivante :

— $C \sqsubseteq D$ notée par $\mathcal{I} \models C \sqsubseteq D$, si $C^\mathcal{I} \subseteq D^\mathcal{I}$.
— $r \sqsubseteq s$ notée par $\mathcal{I} \models r \sqsubseteq s$, si $r^\mathcal{I} \subseteq s^\mathcal{I}$.
— $r_1 \circ r_2 \sqsubseteq s$ notée par $\mathcal{I} \models r_1 \circ r_2 \sqsubseteq s$, si $(r_1 \circ r_2)^\mathcal{I} \subseteq s^\mathcal{I}$.

De même pour les assertions, $\mathcal{I}$ est modèle de la formule $C(a)$ notée par $\mathcal{I} \models C(a)$, si $a^\mathcal{I} \in C^\mathcal{I}$. Maintenant $\mathcal{I}$ est modèle de la formule $r(a,b)$ notée par $\mathcal{I} \models r(a,b)$, si $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$. Une interprétation $\mathcal{I}$ est modèle de l'ontologie $\mathcal{O}$, si elle satisfait tous les axiomes de l'ontologie $\mathcal{O}$. On dit que l'ontologie est consistante si et seulement si il existe au moins un modèle de $\mathcal{O}$. Sinon, elle est inconsistante. Un axiome ϕ est impliqué par une ontologie, notée par $\mathcal{O} \models \phi$, si ϕ est satisfait par chaque modèle de $\mathcal{O}$. Un concept C est subsumé par un autre concept D par rapport à une ontologie $\mathcal{O}$, si $\mathcal{O} \models C \sqsubseteq D$. De la même manière pour les assertions, on dit que a est une instance de C par rapport à une ontologie $\mathcal{O}$, si $\mathcal{O} \models C(a)$. Un concept C est considéré

comme insatisfiable pour une ontologie $\mathcal{O}$, si $\mathcal{O} \models C \sqsubseteq \bot$, sinon C est satisfiable.

Dans cet article, nous ne considérons que la partie terminologique de la base de connaissances, car nous traitons le problème d'incohérence qui est lié à cette partie. La principale tâche de raisonnement que nous étudions est la classification. Elle consiste à déterminer toutes les relations de subsomption (et des équivalences) de concepts impliqués entre les concepts atomiques d'une ontologie $\mathcal{O}$, ou les concepts $\top$ et $\bot$. Nous suivons la procédure indiquée dans [2, 3]. Soit $\mathcal{O}$ une ontologie de la logique de description $\mathcal{EL}^+_\bot$, la première étape du raisonnement consiste à transformer l'ontologie $\mathcal{O}$ en forme normale en utilisant un ensemble de règles. Nous rappelons que $\mathcal{O}$ est dit sous forme normale si chacun de ses axiomes a l'une des formes suivantes :

$$A \sqsubseteq B, A_1 \sqcap ... \sqcap A_n \sqsubseteq B, A \sqsubseteq \exists r.B, \exists r.A \sqsubseteq B$$

Avec $A, B \in N_C \cup \{\top, \bot\}$ et $A_i \in N_C$.

Une fois que l'ontologie est sous forme normale, on peut utiliser l'ensemble des règles d'inférences [2, 3] pour faire le raisonnement.

2.2 Théorie des possibilités

Étant donné un langage de description $\mathcal{L}$ et Ω un univers de discours qui contient un ensemble des interprétations de la logique de descriptions, i.e. $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}}) \in \Omega$. Nous introduisons la sémantique de la théorie des possibilités sur les interprétations de la logique de descriptions $\mathcal{EL}$.

Distributions de possibilités. Une distribution de possibilités, notée par π, est le bloc principal de la théorie des possibilités. Une distribution de possibilités est une fonction de Ω dans l'intervalle $[0, 1]$ [2]. Elle attribue à chaque interprétation $\mathcal{I} \in \Omega$ un degré de possibilité $\pi(\mathcal{I}) \in [0, 1]$ reflétant sa compatibilité ou sa cohérence par rapport aux autres connaissances. Les poids pourraient être interprétés de deux façons, i) une interprétation numérique lorsque les valeurs ont un sens réel (par exemple, observation issue d'une expérience). ii) une interprétation ordinale lorsque les valeurs ne reflètent qu'un ordre total entre les différentes informations. Dans cet article, nous nous intéressons à la dernière interprétation, c'est-à-dire le cadre qualitatif. Une interprétation $\mathcal{I}$ est dite parfaitement possible lorsque $\pi(\mathcal{I}) = 1$, et impossible (c.-à-d incohérente) lorsque $\pi(\mathcal{I}) = 0$. Enfin, soient $\mathcal{I}$ et $\mathcal{I}'$ deux interprétations, alors $\mathcal{I}$ est plus cohérente ou plus compatible que $\mathcal{I}'$, si $\pi(\mathcal{I}) > \pi(\mathcal{I}')$.

Les mesures de possibilité et de nécessité. Étant donné une distribution de possibilité π, la théorie des possibilités standard offre deux mesures de 2^Ω dans l'intervalle $[0, 1]$ qui discriminent entre la possibilité et la certitude concernant un événement $M \subseteq \Omega$.
La mesure de possibilité $\Pi(M) = \sup\{\pi(\mathcal{I}) : \mathcal{I} \in M\}$ évalue dans quelle degré M est compatible ou cohérent par rapport aux autres connaissances disponibles exprimées par π.

La mesure de nécessité $N(M) = 1 - \Pi(\bar{M})$, évalue à quel degré l'événement M est certain. Lorsque $N(M) = 1$, nous disons que M est certain. Lorsque $N(M) \in]0, 1[$, nous disons que M est un peu certain. Lorsque $N(M) = 0$ et $N(\bar{M}) = 0$, nous disons qu'il y a une ignorance totale de M.

Soient ϕ un axiome de la logique de descriptions et $Mod(\phi)$ l'ensemble des modèles de ϕ, la mesure de la possibilité et la mesure de la nécessité associées à ϕ sont définies respectivement comme suit :

$$\Pi(Mod(\phi)) = \sup_{\mathcal{I} \in \Omega}\{\pi(\mathcal{I}) : \mathcal{I} \models \phi\},$$

et

$$N(Mod(\phi)) = 1 - \sup_{\mathcal{I} \in \Omega}\{\pi(\mathcal{I}) : \mathcal{I} \not\models \phi\}.$$

avec $\mathcal{I} \models \phi$ est la relation de satisfaction définie dans la section 2

3 La logique de descriptions $\mathcal{EL}^+_\bot$ possibiliste

Dans ce qui suit, nous introduisons la syntaxe et la sémantique de l'extension possibiliste de la logique de descriptions $\mathcal{EL}^+_\bot$, notée par π- $\mathcal{EL}^+_\bot$.

Syntaxe. Soit $\mathcal{O} = \{\phi_i : i = 1, ..., n\}$ une ontologie exprimée en logique de descriptions $\mathcal{EL}^+_\bot$ composée d'un ensemble d'axiomes comme présente dans la section 2. Une ontologie $\mathcal{EL}^+_\bot$ possibiliste, notée par $\mathcal{O}_\pi = \{(\phi_i, \alpha_i) : i = 1, ..., n\}$, consiste en un ensemble fini d'axiomes possibilistes de la forme (ϕ_i, α_i) avec ϕ_i est un axiome standard de la logique $\mathcal{EL}^+_\bot$ et α_i son degré de certitude, ce qui signifie que $N(\phi_i) \geqslant \alpha_i$. Noter que les axiomes avec un degré α_i égale à "0" ne sont pas explicitement représentés dans l'ontologie, c.-à-d sont des tautologies, ce qui permet de les enlever de l'ontologie sans perte d'information. De plus, lorsque tous les degrés sont égaux à 1, $\mathcal{O}_\pi$ coïncide avec une ontologie standard $\mathcal{O}$ de la logique $\mathcal{EL}^+_\bot$. Dans une ontologie possibiliste, le degré de nécessité attaché à un axiome reflète son degré de priorité.

Sémantique. La sémantique d'une ontologie $\mathcal{EL}^+_\bot$ possibiliste est donnée par une distribution de possibilités, notée par $\pi_\mathcal{O}$, définie sur l'ensemble des interprétations de la logique de descriptions présentées dans la Section 2.2, notée $\Omega = \{\mathcal{I}_1, ..., \mathcal{I}_n\}$. La distribution de possibilités attribue à chaque interprétation $\mathcal{I} \in \Omega$ un degré de possibilité $\pi(\mathcal{I}) \in]0, 1]$ qui reflète à quel point l'interprétation satisfaite (voir Section 2) les axiomes de l'ontologie. Plus formellement,

2. En fait, il s'agit d'une fonction de Ω dans une échelle totalement ordonnée. Cette échelle peut être un ensemble fini d'entiers ou l'intervalle unitaire [0, 1] et exprime les connaissances sur le monde réel. En général, on considère l'intervalle [0, 1].

Définition 1 *La distribution de possibilités π_O associée à une ontologie de la logique de descriptions $\mathcal{EL}^+_\perp$, notée par O_π, est définie comme suit :*

$$\forall \mathcal{I} \in \Omega, \pi(\mathcal{I}) = \begin{cases} 1 \ \ si \ \forall(\phi_i, \alpha_i) \in O_\pi, \mathcal{I} \models \phi_i \\ 1 - \max\{\alpha_i : (\phi_i, \alpha_i) \in O_\pi, \mathcal{I} \not\models \phi\} \ sinon. \end{cases}$$

Un avantage important de l'utilisation de la logique possibiliste est qu'elle peut naturellement traiter l'incohérence, en donnant un degré d'incohérence à l'ontologie. Par conséquent, on peut associer à une ontologie un degré d'incoherence qui varie entre 0 et 1 si nous utilisons l'intervalle unitaire]0, 1] pour exprimer les degrés de certitude.

Une interprétation $\mathcal{I}$ est modèle de O_π si elle satisfait tous les axiomes de O_π. Dans ce cas $\pi(\mathcal{I}) = 1$. Cela signifie également que la distribution de possibilités π_O est normalisée. Sinon, si $\mathcal{I}$ n'est pas un modèle de O_π, alors le degré de possibilité $\pi(\mathcal{I})$ dépend de l'axiome ayant le poids maximal qui n'est pas satisfait (falsifié) par l'interprétation, noté $\pi(I) = 1 - \max\{\alpha_i : (\phi_i, \alpha_i) \in O_\pi, \mathcal{I} \not\models \phi_i\}$. Dans ce cas, l'ontologie est incohérente et son degré d'incohérence est $Inc(O_\pi) = 1 - \max_{I \in \Omega} \pi(I)$

Définition 2 *Soit O_π une ontologie possibiliste exprimée en logique de descriptions $\mathcal{EL}^+_\perp$, notée par π-$\mathcal{EL}^+_\perp$, nous définissons l'implication possible comme suit :*

- *Un axiome ϕ est impliqué à partir de O_π, noté par $\pi \models \phi$ si et seulement si $N(\phi) > 0$ où $N(\phi)$ est le degré de nécessité de ϕ calculé à partir de π.*
- *Un axiome ϕ est impliqué à partir de O_π avec un degré de certitude α, noté par $\pi \models (\phi, \alpha)$ si et seulement si $N(\phi) \geqslant \alpha > 0$ où $N(\phi)$ est le degré de nécessité de ϕ calculé à partir de π.*

Dans ce qui suit, nous étudions le raisonnement en π-$\mathcal{EL}^+_\perp$. Nous définissons des algorithmes qui calculent l'implication possible donnée dans la définition 2. Nous nous concentrons sur le problème de subsomption en π- $\mathcal{EL}^+_\perp$ comme implication possible, c.-à-d. que nous étudions si $O_\pi \models (A \sqsubseteq B, \alpha)$, où $A, B \in N_C \cup \{\top, \perp\}$. Noter que $O_\pi \models (A \sqsubseteq B, \alpha)$ ssi $O_\pi \cup \{(C \sqsubseteq A, 1), (B \sqsubseteq D, 1)\} \models (C \sqsubseteq D, \alpha)$, où C, D sont des nouveaux concepts atomiques. De la même manière qu'une ontologie standard $\mathcal{EL}^+_\perp$, nous fournissons d'abord dans le tableau 2 les règles de normalisation nécessaires pour transformer une ontologie π- $\mathcal{EL}^+_\perp$ en forme normale, puis nous donnons dans le tableau 3 les règles d'inférences nécessaires pour calculer l'implication.

Pour obtenir ces règles, nous introduisons d'abord le lemme suivant.

Lemme 1 *Soit O_π une ontologie $\mathcal{EL}^+_\perp$ possibiliste qui contient deux axiomes (ϕ, α_1) et (ϕ, α_2) alors O_π et $O'_\pi = \{O_\pi \backslash (\phi, \alpha_1), (\phi, \alpha_2)\} \cup \{(\phi, max(\alpha_1, \alpha_2))\}$ sont équivalents dans le sens que $\forall \mathcal{I} \in \Omega, \pi_O(\mathcal{I}) = \pi_{O'}(\mathcal{I})$. Ce qui signifie que, O_π et O'_π induisent la même distribution de possibilités.*

$$(\text{PNR}_0) \quad \frac{(C_1 \sqcap \top \sqcap C_2 \sqsubseteq D, \alpha)}{(C_1 \sqcap C_2 \sqsubseteq D, \alpha)}$$

$$(\text{PNR}_1) \quad \frac{(C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D, \alpha)}{(\perp \sqsubseteq D, \alpha)} :$$

$$(\text{PNR}_2) \quad \frac{(C \sqsubseteq D_1 \sqcap D_2, \alpha)}{(C \sqsubseteq D_1, \alpha) \quad (C \sqsubseteq D_2, \alpha)}$$

$$(\text{PNR}_3) \quad \frac{(\exists r.C \sqsubseteq D, \alpha)}{(C \sqsubseteq A, 1) \quad (\exists r.A \sqsubseteq D, \alpha)} :$$
$C \notin N_c, A$ est un nouveau concept atomique

$$(\text{PNR}_4) \quad \frac{(C \sqsubseteq D, \alpha)}{(C \sqsubseteq A, 1) \quad (A \sqsubseteq D, \alpha)} :$$
$C, D \notin N_c \cup \{\perp, \top\}, A$ est un nouveau concept atomique

$$(\text{PNR}_5) \quad \frac{(B \sqsubseteq \exists r.C, \alpha)}{(B \sqsubseteq \exists r.A, 1) \quad (A \sqsubseteq C, \alpha)} :$$
$C \notin N_c, A$ est un nouveau concept atomique

$$(\text{PNR}_6) \quad \frac{(C_1 \sqcap C \sqcap C_2 \sqsubseteq D, \alpha)}{(C \sqsubseteq A, 1) \quad (C_1 \sqcap A \sqcap C_2 \sqsubseteq D, \alpha)} : C \notin N_c \cup \{\perp, \top\}$$

Table 2 – Règles de normalisation possibiliste.

$$(\text{PIR}_0) \quad \frac{}{(A \sqsubseteq A, 1)} : A \in N_c \cup \{\perp, \top\}$$

$$(\text{PIR}_1) \quad \frac{}{(C \sqsubseteq \top, 1)} : C \sqsubseteq \top$$

$$(\text{PIR}_2) \quad \frac{}{(r, 1)} : r \in O$$

$$(\text{PIR}_3) \quad \frac{(C \sqsubseteq D', \alpha_1) \quad (D' \sqsubseteq D, \alpha_2)}{(C \sqsubseteq D, \min(\alpha_1, \alpha_2))}$$

$$(\text{PIR}_4) \quad \frac{(A \sqsubseteq B_1 ... A \sqsubseteq B_n, \alpha_1) \quad (B_1 \sqcap ... \sqcap B_n \sqsubseteq B, \alpha_2)}{(A \sqsubseteq B, \min(\alpha_1, \alpha_2))} :$$
$A, B, B_i \in N_c \cup \{\perp, \top\}$

$$(\text{PIR}_5) \quad \frac{(A \sqsubseteq \exists r.B, \alpha_1) \quad (B \sqsubseteq C, \alpha_2)}{(A \sqsubseteq \exists r.C, \min(\alpha_1, \alpha_2))} : A, B, C \in N_c \cup \{\perp, \top\}, C \neq \perp$$

$$(\text{PIR}_6) \quad \frac{(A \sqsubseteq \exists r.B, \alpha) \quad (B \sqsubseteq \perp, 1)}{(A \sqsubseteq \perp, \alpha)} : A, B \in N_c \cup \{\perp, \top\}$$

$$(\text{PIR}_7) \quad \frac{(A \sqsubseteq \exists r.B, \alpha_1) \quad (r \sqsubseteq s, \alpha_2)}{(A \sqsubseteq \exists s.B, \min(\alpha_1, \alpha_2))}$$

$$(\text{PIR}_8) \quad \frac{(r_1 \sqsubseteq r_2, \alpha_1) \quad (r_2 \sqsubseteq r_3, \alpha_2)}{(r_1 \sqsubseteq r_3, \min(\alpha_1, \alpha_2))}$$

$$(\text{PIR}_9) \quad \frac{(A \sqsubseteq \exists r_1.B, \alpha_1) \quad (B \sqsubseteq \exists r_2.A_2, \alpha_2) \quad (r_1 \circ r_2 \sqsubseteq s, \alpha_3)}{(A_1 \sqsubseteq \exists s.A_2, \min(\alpha_1, \alpha_2, \alpha_3))}$$
$: A_1, A_2 \in N_c \cup \{\perp, \top\}$

Table 3 – Règles d'inférences possibiliste.

Preuve 1 *La preuve découle immédiatement de la Définition 1 sur la distribution de possibilités.*

En utilisant ce lemme, nous avons les propositions suivantes.

Proposition 1 *Soient $\mathcal{O}_\pi$ une ontologie possibiliste exprimée en $\mathcal{EL}^+_\perp$ et $\mathcal{O}^N_\pi$ une ontologie obtenue à partir de $\mathcal{O}_\pi$ en appliquant les règles de normalisation données dans le tableau 2. Alors $\mathcal{O}_\pi$ et $\mathcal{O}^N_\pi$ induisent la même distribution de possibilités.*

Preuve 2 *Nous donnons la preuve pour certaines règles de normalisation, les autres sont de la même façon*

(PNR_0). Soient $\mathcal{O}_\pi = \{C_1 \sqcap \top \sqcap C_2 \sqsubseteq D, \alpha\}$ et $\mathcal{O}'_\pi = \{C_1 \sqcap C_2 \sqsubseteq D, \alpha\}$ deux ontologies $\mathcal{EL}^+_\perp$, donc $\mathcal{O}_\pi$ et $\mathcal{O}'_\pi$ induisent la même distribution de possibilités. Soit $\pi \models (C_1 \sqcap \top \sqcap C_2 \sqsubseteq D, \alpha)$, ce qui signifie que $N(C_1 \sqcap \top \sqcap C_2 \sqsubseteq D) \geqslant \alpha$. Plus précisément $(C_1^M \cap \Delta^M \cap C_2^M \subseteq D^M)$ un modèle de $(C_1 \sqcap \top \sqcap C_2 \sqsubseteq D)$. Alors :
$(C_1^M \cap \Delta^M \cap C_2^M \subseteq D^M) = (C_1^M \cap C_2^M \subseteq D^M)$.
$N(C_1^M \cap \Delta^M \cap C_2^M \subseteq D^M) = N(C_1^M \cap C_2^M \subseteq D^M)$.
$N(C_1 \sqcap \top \sqcap C_2 \sqsubseteq D) = N(C_1 \sqcap C_2 \sqsubseteq D)$, ce qui signifie que $\mathcal{O}_\pi$ et $\mathcal{O}'_\pi$ induisent la même distribution de possibilités. Pour le deuxième cas $\pi \not\models (C_1 \sqcap \top \sqcap C_2 \sqsubseteq D, \alpha)$. Nous avons : $C_1 \sqcap \top \sqcap C_2 \sqsubseteq D \equiv C_1 \sqcap C_2 \sqsubseteq D$, alors $N(C_1 \sqcap \top \sqcap C_2 \sqsubseteq D) = N(C_1 \sqcap C_2 \sqsubseteq D)$, ce qui signifie que $\mathcal{O}_\pi$ et $\mathcal{O}'_\pi$ induisent la même distribution de possibilité.

(PNR_1). Soient $\mathcal{O}_\pi = \{C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D, \alpha\}$ et $\mathcal{O}'_\pi = \{\perp \sqsubseteq D, \alpha\}$ deux ontologies $\mathcal{EL}^+_\perp$ donc $\mathcal{O}_\pi$ et $\mathcal{O}'_\pi$ induisent la même distribution de possibilités. Soit $\pi \models (C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D, \alpha)$, ce qui signifie que $N(C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D) \geqslant \alpha$. Plus précisément $(C_1^M \cap \emptyset \cap C_2^M \subseteq D^M)$ un modèle de $(C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D)$. Alors :
$(C_1^M \cap \emptyset \cap C_2^M \subseteq D^M) = (\emptyset \subseteq D^M)$.
$N(C_1^M \cap \emptyset \cap C_2^M \subseteq D^M) = N(\emptyset \subseteq D^M)$.
$N(C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D) = N(\perp \sqsubseteq D)$, ce qui signifie que $\mathcal{O}_\pi$ and $\mathcal{O}'_\pi$ induisent la même distribution de possibilités. Maintenant, l'autre cas $\pi \not\models (C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D, \alpha)$, nous avons : $C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D \equiv \perp \sqsubseteq D$, alors $N(C_1 \sqcap \perp \sqcap C_2 \sqsubseteq D) = N(\perp \sqsubseteq D)$, ce qui signifie que $\mathcal{O}_\pi$ and $\mathcal{O}'_\pi$ induisent la même distribution de possibilités.

(PNR_3) : Soient $\mathcal{O}_\pi = \{(C \sqsubseteq D_1 \sqcap D_2, \alpha)\}$ et $\mathcal{O}'_\pi = \{(C \sqsubseteq D_1, \alpha_1)$ et $(C \sqsubseteq D_2, \alpha_2)\}$ deux ontologies $\mathcal{EL}^+_\perp$, donc $\mathcal{O}_\pi$ et $\mathcal{O}'_\pi$ induisent la même distribution de possibilités. soit $\pi \models (C \sqsubseteq D_1 \sqcap D_2, \alpha)$ donc C^M, D_1^M et D_2^M des modèles de C, D_1 et D_2, alors $(C)^M \subseteq (D_1^M \cap D_2^M)$ un modèle de $C \sqsubseteq D_1 \sqcap D_2$ alors :
$(C)^M \subseteq (D_1^M \cap D_2^M) = (C^M \subseteq D_1^M) \cap (C^M \subseteq D_2^M)$, par la suite $\pi \models (C \sqsubseteq D_1)$ et $\pi \models (C \sqsubseteq D_2)$, ce qui signifie que les deux ontologies induisent la même distribution de possibilités. Inversement, supposons que $\pi \models (C \sqsubseteq D_1, \alpha_1)$ et $\pi \models (C \sqsubseteq D_2, \alpha_2)$, soient C^M, D_1^M et D_2^M des modèles des C, D_1 et D_2, alors $C^M \subseteq D_1^M$ et $C^M \subseteq D_2^M$ alors $C^M \subseteq (D_1^M \cap D_2^M)$ par conséquent $\pi \models C \sqsubseteq D_1 \sqcap D_2$ et $\pi_O(\mathcal{I}) = \pi_{O'}(\mathcal{I})$. L'autre cas lorsque $\pi \not\models (C \sqsubseteq D_1 \sqcap D_2, \alpha)$, nous avons $C \sqsubseteq D_1 \sqcap D_2 \equiv (C \sqsubseteq D_1) \sqcap (C \sqsubseteq D_2)$ alors $\pi \not\models (C \sqsubseteq D_1)$ et $\pi \not\models (C \sqsubseteq D_2)$ par conséquent $\pi_{\mathcal{O}_\pi} = \pi_{\mathcal{O}'_\pi}$.

Étudions maintenant les règles d'inférences. Soit $cl(\mathcal{O}_\pi)$ la clôture de l'ontologie $\mathcal{O}_\pi$ obtenue par l'application des règles données dans le tableau 2 et le tableau 3, alors les deux ontologies induisent la même distribution de possibilités.

Proposition 2 *Soient $\mathcal{O}_\pi$ une ontologie $\mathcal{EL}^+_\perp$ possibiliste et $cl(\mathcal{O}_\pi)$ sa clôture obtenue en appliquant les règles décrites dans le tableau 2 et le tableau 3. Alors $\mathcal{O}_\pi$ et $cl(\mathcal{O}_\pi)$ induisent la même distribution de possibilités.*

Preuve 3 *Nous donnons la preuve d'une règle d'inférences, les autres sont de la même façon, la preuve revient à déduire que l'application des règles de normalisation et des règles d'inférences ne change pas la distribution des possibilités, donc il suffit de répéter l'application des règles et de voir la distribution des possibilités. supposons que $(C_1 \sqsubseteq C_2, \alpha_1) \in \mathcal{O}$ et $(C_2 \sqsubseteq C_3, \alpha_2) \in \mathcal{O}'$. Montrons qu'en appliquant la règle (PIR_3) ne modifie pas la distribution des possibilités, c'est-à-dire $\mathcal{O}' = \{\mathcal{O} \sqcup C_1 \sqsubseteq C_3, \min(\alpha_1, \alpha_2)\}$. Soit $\mathcal{I} = (\Delta^\mathcal{I}, .^\mathcal{I})$ une interprétation. Nous avons quatre cas :*

- *$\mathcal{I} \models C_1 \sqsubseteq C_2$ et $\mathcal{I} \models C_2 \sqsubseteq C_3$ par définition de la relation de satisfaction, nous avons $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$ et $C_2^\mathcal{I} \subseteq C_3^\mathcal{I}$ donc $C_1^\mathcal{I} \subseteq C_3^\mathcal{I}$ ce qui signifie que $\mathcal{I} \models C_1 \sqsubseteq C_3$. Par conséquent $\pi_\mathcal{O}(\mathcal{I}) = \pi_{\mathcal{O}'}(\mathcal{I})$*

- *$\mathcal{I} \models C_1 \sqsubseteq C_2$ et $\mathcal{I} \not\models C_2 \sqsubseteq C_3$ soit $\mathcal{O}'' = \mathcal{O} \backslash \{(C_1 \sqsubseteq C_2, \alpha_1)$ et $(C_2 \sqsubseteq C_3, \alpha_2)\}$ alors*
  $\pi_\mathcal{O}(\mathcal{I}) = \min(\pi_{\mathcal{O}''}, 1 - \alpha_2)$
  $= \min(\pi_{\mathcal{O}''}, 1 - \alpha_2, 1 - \min(\alpha_1, \alpha_2))$
  $= \pi_{\mathcal{O}'}(\mathcal{I})$

- *$\mathcal{I} \not\models C_1 \sqsubseteq C_2$ et $\mathcal{I} \models C_2 \sqsubseteq C_3$: est montrée de la même façon*

- *$\mathcal{I} \not\models C_1 \sqsubseteq C_2$ et $\mathcal{I} \not\models C_2 \sqsubseteq C_3$ soit $\mathcal{O}'' = \mathcal{O} \backslash \{(C_1 \sqsubseteq C_2, \alpha_1), (C_2 \sqsubseteq C_3, \alpha_2)\}$ nous avons*
  $\pi_\mathcal{O}(\mathcal{I}) = \min(\pi_{\mathcal{O}''}(\mathcal{I}), 1 - \alpha_1, 1 - \alpha_2)$
  $= \min(\pi_{\mathcal{O}''}(I), 1 - \alpha_1, 1 - \alpha_2, 1 - \min(\alpha_1, \alpha_2))$
  $= \pi_{\mathcal{O}'}(\mathcal{I})$

La proposition suivante formalise l'implication possible (Définition 2). Elle est donnée pour la relation de subsomption de concept, comme étant la tâche principale dans $\mathcal{EL}$. Cependant, elle peut être généralisé à n'importe quel axiome ϕ.

Proposition 3 *Soit $cl(\mathcal{O}_\pi)$ la clôture de l'ontologie $\mathcal{O}_\pi$ obtenue par l'application des règles de normalisation et d'inférences à l'ontologie $\mathcal{O}_\pi$. Soient A,B $\in N_c$ deux concepts dans l'ontologie. Donc, $\mathcal{O}_\pi \models (A \sqsubseteq B, \alpha)$ si*

— $(A \sqsubseteq B, \beta) \in cl(\mathcal{O}_\pi)$ *avec* $\beta \geqslant \alpha$,
— $(A \sqsubseteq \bot, \beta) \in cl(\mathcal{O}_\pi)$ *avec* $\beta \geqslant \alpha$.

Preuve 4 *La preuve est immédiate. Elle découle de la définition des règles d'inférences donnée dans le tableau 3.*

Proposition 4 *Soit* $cl(\mathcal{O}_\pi)$ *la clôture de l'ontologie* $\mathcal{O}_\pi$ *obtenue par l'application des règles de normalisation et d'inférences présentées respectivement dans tableau 2 et tableau 3. Soient* $A,B \in N_c$ *deux concepts dans l'ontologie, donc* $cl(\mathcal{O}_\pi)$ *est calculée en temps polynomial par rapport à la taille de l'ontologie* $|\mathcal{O}_\pi|$.

Preuve 5 *La preuve est immédiate. Si nous affectons un degré égal 1 à tous les axiomes de l'ontologie, donc nous obtenons alors les règles de normalisation et celles d'inférences standard, donc la complexité du calcul de* $cl(\mathcal{O}_\pi)$ *est similaire à celle de la clôture de l'ontologie standard* $cl(\mathcal{O})$*, qui est polynomial par rapport à* $|\mathcal{O}|$

4 Conclusion

Dans cet article, nous avons étudié une extension possibiliste de la logique de description $\mathcal{EL}^+_\bot$. Nous avons introduit la syntaxe et la sémantique de cette extension. Pour traiter l'incertitude qualitative attachée aux axiomes de l'ontologie, nous avons utiliser l'opérateur minimum. Un résultat important montré dans cet article est que la complexité de calcul de subsomption reste polynomiale.

Dans le cadre des travaux futurs, nous élargirons ce travail en ajoutant le domaine nominal et concret afin de proposer un fragment possibiliste *OWL2-EL*. Nous envisagerons également d'étudier une extension quantitative de $\mathcal{EL}$.

5 Remerciement

Ce travail a bénéficié de l'aide de l'Agence Nationale de la Recherche, projet ANR CHAIRE IA BE4musIA.

Références

[1] Baader, Franz: *The description logic handbook : Theory, implementation and applications*. Cambridge university press, 2003.

[2] Baader, Franz, Sebastian Brandt et Carsten Lutz: *Pushing the EL envelope*. Dans *IJCAI*, tome 5, pages 364–369, 2005.

[3] Baader, Franz, Sebastian Brandt et Carsten Lutz: *Pushing the EL envelope further*. 2008.

[4] Baader, Franz, Carsten Lutz et Boontawee Suntisrivaraporn: *Is tractable reasoning in extensions of the description logic EL useful in practice*. Dans *Proceedings of the 2005 International Workshop on Methods for Modalities (M4M-05)*, tome 450. Citeseer, 2005.

[5] Bateman, John A, Joana Hois, Robert Ross et Thora Tenbrink: *A linguistic ontology of space for natural language processing*. Artificial Intelligence, 174(14) :1027–1071, 2010.

[6] Benferhat, Salem et Zied Bouraoui: *Min-based possibilistic DL-Lite*. Journal of Logic and Computation, 27(1) :261–297, 2017.

[7] Borgwardt, Stefan et Rafael Peñaloza: *Fuzzy description logics–a survey*. Dans *International Conference on Scalable Uncertainty Management*, pages 31–45. Springer, 2017.

[8] Boutouhami, Khaoula, Salem Benferhat, Faiza Khellaf et Farid Nouioua: *Uncertain lightweight ontologies in a product-based possibility theory framework*. International Journal of Approximate Reasoning, 88 :237–258, 2017.

[9] Castells, Pablo, Miriam Fernandez et David Vallet: *An adaptation of the vector-space model for ontology-based information retrieval*. IEEE transactions on knowledge and data engineering, 19(2) :261–272, 2006.

[10] Dubois, Didier, Jérôme Mengin et Henri Prade: *Possibilistic uncertainty and fuzzy features in description logic. A preliminary discussion*. Dans *Capturing Intelligence*, tome 1, pages 101–113. Elsevier, 2006.

[11] Dubois, Didier et Henri Prade: *Possibility theory*. Dans *Computational complexity*, pages 2240–2252. Springer, 2012.

[12] Ison, Jon, Matúš Kalaš, Inge Jonassen, Dan Bolser, Mahmut Uludag, Hamish McWilliam, James Malone, Rodrigo Lopez, Steve Pettifer et Peter Rice: *EDAM : an ontology of bioinformatics operations, types of data and identifiers, topics and formats*. Bioinformatics, 29(10) :1325–1332, 2013.

[13] Liu, Kaihong, William R Hogan et Rebecca S Crowley: *Natural language processing methods and systems for biomedical ontology learning*. Journal of biomedical informatics, 44(1) :163–179, 2011.

[14] Lukasiewicz, Thomas: *Expressive probabilistic description logics*. Artificial Intelligence, 172(6-7) :852–883, 2008.

[15] Müller, Hans Michael, Eimear E Kenny et Paul W Sternberg: *Textpresso : an ontology-based information retrieval and extraction system for biological literature*. PLoS biology, 2(11) :e309, 2004.

[16] Smith, Barry, Anand Kumar et Thomas Bittner: *Basic formal ontology for bioinformatics*. 2005.

Actes JIAF 2020

An Application for Merging Open-Domain Ontologies

Zied Bouraoui, Sébastien Konieczny, Truong-Thanh Ma, Ivan Varzinczak
CRIL, Univ. Artois & CNRS, France

{bouraoui,konieczny,ma,varzinczak}@cril.fr

Résumé

Les connaissances conceptuelles, représentées dans des ontologies, jouent un rôle important dans de nombreux domaines, notamment le Web sémantique, la recherche d'informations et le traitement du langage naturel. Une attention considérable a récemment été consacrée au problème de l'unification et de la liaison des ontologies disponibles. Alors que la grande majorité des travaux existants se concentrent sur l'appariement ou l'alignement des ressources, dans cet article, nous étudions l'application de la théorie de la fusion des croyances à la fusion d'ontologies afin d'obtenir un point de vue unique. Nous considérons le contexte où différentes ontologies partagent la même terminologie (c'est-à-dire en supposant qu'elles sont déjà alignées les unes aux autres), mais elles expriment les connaissances de manière différente et potentiellement conflictuelle. Afin d'obtenir une vue unifiée des connaissances véhiculées par les différentes ontologies, nous commençons par fournir un modèle de fusion basé sur la sémantique, qui travaille à partir des interprétations correspondant aux ontologies. Nous proposons une caractérisation formelle du modèle et décrivons le cadre de la fusion d'ontologies. Nous montrons l'efficacité de la méthode par une évaluation expérimentale sur des ontologies existantes en domaine ouvert.

Abstract

Conceptual knowledge, encoded in ontologies or knowledge graphs, are playing an important role in many areas including Semantic Web, Information Retrieval and Natural Language Processing. Considerable attention has recently been devoted to the problem of unifying and linking available ontologies. While the vast majority of existing work focuses on matching or aligning resources, in this paper, we investigate the application of belief merging theory to ontology merging in order to obtain a unique point of view. We consider the setting where different ontologies share the same terminology (i.e. assuming that they are already mapped to each other), but they express knowledge in different and potentially conflicting ways. In order to get a unified view of the knowledge conveyed by the different ontologies, we start by providing a semantic-based merging model that retrieves all the interpretations in which the outcome can be found. We propose a formal characterisation of the model and describe the whole framework for ontology merging. We support the demonstration of the effectiveness of the method by an experimental evaluation of the method on existing open-domain ontologies.

1 Introduction

Structured knowledge about concepts and properties are commonly used in fields such as machine learning (ML) [17], natural language processing (NLP) [14], information retrieval (IR) [4] and Semantic Web [9]. They are typically encoded using ontologies or knowledge graphs. The key difference between these two frameworks is that knowledge graphs are considerably less expressive than ontologies, and therefore can be seen as a simpler form of ontologies [1]. An important point, however, is that many ontologies sharing the same knowledge are available. Several open domain ontologies, such as SUMO, OpenCyc, Wikidata, Babelnet, and others, are available on the Web. Therein, they often express knowledge of a particular domain such as food, sport, dance, etc., using different terminology. Consider for example the concept of a "*Box*", for which SUMO ontology uses the concept name "*Box*", BabelNet uses the concept name "*Carton*" *(bn :00016327n)*, and WikiData uses "*Parcel*" *(Q13107365)* as concept name. This observation has led to a number of methods aiming at unifying and link ontologies to each other. Therefore, ontology integration approaches, such as *ontology matching, mapping and alignment*, have emerged [7, 19].

Ontology mapping aims to find semantic correspondences between similar elements of different ontologies [13], e.g. the "*Puppy*" concept in an ontology corresponds

1. In this paper, we shall use ontologies to refer to knowledge graphs as well.

to the "*Dog*" concept in another ontology. Some studies on the matter are those by Pan et al. [15] and Dragoni [6]. Alignment methods offer the discovery of correspondences between ontologies, in other words, a set of mappings is an alignment. In particular, the process of ontology alignment takes two or more input ontologies and produces a set of relationships between concepts that map them semantically with each other [20, 22] e.g. the concepts of "*Mare*", "*Stallion*" and "*Zebra*" in different ontologies are equivalent (or subsumed by) to "*Horse*" concept in another ontology. Ontology matching is the process of automatically generating correspondences between terms of different ontologies, or said differently is the task of finding relationships between entities expressed in different ontologies. Some recent studies, including [5, 7, 16], focus on these aspects.

Naturally, when expressing knowledge of a given domain, and assuming that the same terminology (i.e. individual, concept and role names are the same), many points of view are possible, and therefore many types of conflicts may arise. By conflict, we do not only refer to logical inconsistency, but also semantic conflicts that might appear the knowledge are structured in different ways. Consider for instance the concept "*Process*" as defined by the following three ontologies : SUMO, WikiData and BabelNet.

— **Wikidata :** Procedure **is a** Technique and the Technique **is a** Process.

— **SUMO :** Technique **is a** Procedure and the Technique **is a** Process.

— **Babelnet :** Technique and Procedure **are a** Process.

In this example, the three ontologies are already matched to each other[2]. While there is no logical inconsistency between ontologies, an emerging problem is the fact that the aforementioned ontologies structure the knowledge about the domain in different and potentially conflicting ways (i.e. Wikidata says that "*Procedure*" is a *"Technique"* while SUMO states that "*Technique*" is a "*Procedure*". Then, one issue that occurs is whether Wikidata or SUMO statements will be selected in terms of merging). Ontology merging is, hence, the process of combining two (or more) ontologies sources into a target ontology while solving conflicts (semantic and logical) between them. In this paper, we focus on the conflicts that arise when the ontologies sources express, using the same terminology, the knowledge about a particular domain in different incompatible ways. In the example, mentioned above, one possible solution to deal with semantic conflicts would be to consider all the concepts as equivalent (e.g. *"Procedure"* ≡ *"Technique"*). However, we would prefer to avoid this solution as it leads to flatten the hierarchy of the output ontology (i.e. the result of merging). Recall that the main purpose of ontology is to structure conceptual knowledge in a hierarchical way.

2. Details about the matching is described in section V

In order to solve those problems, we propose an ontology merging method that relies on the solid theoretical foundations of propositional belief merging (i.e. [18, 10]), which has extensively been studied for beliefs encoded in propositional languages. We introduce ontology merging operators that rely on semantics to retrieve a possible world, in which a solution that solve conflicts between ontology sources can be found. Notice that, we focus crucially on handling the semantic conflicts between sources, and assuming these sources are already matched before the merging process is conducted.

2 Background

To introduce our method, we choose description logics, as they provide the formal foundations of ontologies and ontology languages such as OWL. For simplicity, we will consider $\mathcal{EL}$ [1], which is one of the most basic description logics.

The basic ingredients of DLs are individuals, concepts and roles, which respectively correspond at the semantic level to objects, sets of objects, and binary relations between objects. The ontological knowledge is expressed using a set of concept inclusion axioms of the form $C \sqsubseteq D$, which intuitively encodes that every instance of C is also an instance of the concept D. More formally, let N_C, N_R, N_I be three pairwise disjoint sets where N_C denotes a set of atomic concepts, N_R denotes a set of atomic relations (roles) and N_I denotes a set of individuals. The $\mathcal{EL}$ concept expressions are built according to the following grammar :

$$C ::= \top \mid N_C \mid C \sqcap C \mid \exists r.C$$

where $r \in N_R$.

An $\mathcal{EL}$ ontology (a.k.a. knowledge base) consists of a set of general concept inclusion (GCI) axioms of the form $C \sqsubseteq D$, meaning that C is more specific than D or simply C is subsumed by D. Furthermore, a set of equivalence axioms of the form $C \equiv D$ is abbreviation for two general concept inclusions $C \sqsubseteq D$ and $D \sqsubseteq C$. There is also a set of concept assertions of the form $C(a)$, and a set of role assertions of the form $r(a, b)$. Note that in this paper, we consider assertion free ontologies. Moreover, given $A, B, A_1, A_2 \in N_C$, an $\mathcal{EL}$ TBox $\mathcal{T}$ is in **normal form** [2, 21] if it consists of inclusions of the form : $A \sqsubseteq B, A_1 \sqcap A_2 \sqsubseteq B, A \sqsubseteq \exists r.B, \exists r.A \sqsubseteq B$. Then, we assume that all ontologies in this paper are in normal form.

The semantics is in terms of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ which consist of a non-empty interpretation domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps each individual $a^{\mathcal{I}} \in N_I$ into an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each concept $A \in N_C$ into a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role $r \in N_R$ into a subset $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Table 1 summarizes the syntax and semantics of $\mathcal{EL}$. An interpretation $\mathcal{I}$ is said to be a model of (or satisfies) a GCI axiom, denoted by $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Syntax	Semantics
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
a	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\top$	$\Delta^{\mathcal{I}}$
$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} s.t.(x,y) \in r^{\mathcal{I}}, y \in C^{\mathcal{I}}\}$

TABLE 1 – Syntax and semantics of description logic $\mathcal{EL}$

Similarly, $\mathcal{I}$ satisfies a concept (resp. role) assertion, denoted $\mathcal{I} \models C(a)$ (resp. $\mathcal{I} \models r(a,b)$), if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$). An interpretation $\mathcal{I}$ is a model of an ontology $\mathcal{O}$ if it satisfies all the axioms in $\mathcal{O}$. Assuming every ontology is consistent. An ontology is said to be consistent if it has a model. Otherwise, it is inconsistent. An axiom Φ is entailed by an ontology, denoted by $\mathcal{O} \models \Phi$, if Φ is satisfied by every model of $\mathcal{O}$. We say that C is subsumed by D w.r.t an ontology $\mathcal{O}$ iff $\mathcal{O} \models C \sqsubseteq D$. Similarly, we say that a is an instance of C w.r.t. $\mathcal{O}$ iff $\mathcal{O} \models C(a)$.

3 Semantic-Based Ontology Merging

In this section, we present our approach for merging open-domain ontologies. We first provide an overview of our method and then give details about each step.

3.1 Model-Based Framework for Merging Ontologies

Given a set of ontologies describing knowledge of a particular domain and using the same terminology (same concept and role names), one can distinguish two possible cases : (1) the case where the ontologies **agree** on the same claims (e.g. all ontologies claim that a concept A is subsumed by B) and (2) the other case where there is **a disagreement** (or conflict) on how to express knowledge.

In the first case, there is no conflict and the result of merging should contain the statements that are common to the involved ontologies. In the second case, a solution, appropriately solves the conflicts, needs to find all possible statements and select the most plausible one, in which the different sources agree. This case could be in form of semantic disagreement (e.g. "*Technique*" is subsumed by "*Procedure*" in SUMO ontology whereas Wikidata state that "*Procedure*" is subsumed by "*Technique*") or logical disagreement (e.g. "*University*" and "*High school*" are separate (disjointed) in an ontology while another ontology state that "*High school*" is subsumed by "*University*")

Definition 1 (Semantic Conflict) *Let $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ be a set of ontologies that share the same signature, i.e. the same concept/role/individual names. Let $A \sqsubseteq B$ be an axiom of $\mathcal{O}_i$ ($\mathcal{O}_i \models A \sqsubseteq B$). We say that $A \sqsubseteq B$ is in a semantic conflict if there is an ontology $\mathcal{O}_j$ s.t. $\mathcal{O}_j \models B \sqsubseteq A$.*

In this paper, while our work goes beyond the approach of checking for logical inconsistencies, we follow a semantic (model-based) approach for merging, by first computing the set of statements which are the closest to the input ontologies. When this result is obtained, we compute from this set of selected statements an ontology that gives us the (global) result of the merging. Intuitively, given the signature of the different ontologies, we generate all possible statements to express knowledge. These statements will be encoded using a set of interpretations. Once these interpretations are computed, we select the most plausible interpretation that agree with the sources. We propose an ontology merging framework to represent the merging process as well as to deal with the existing conflicts. Our merging framework is summarized in Figure 1.

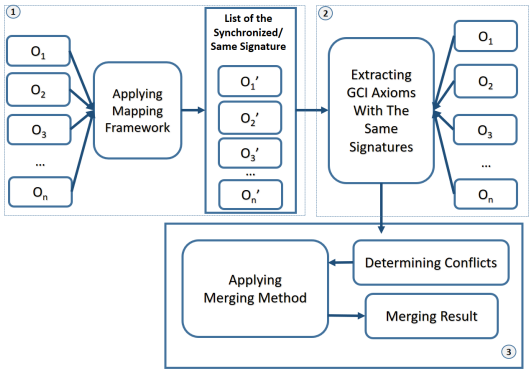


FIGURE 1 – A general framework of the merging process

The framework is split into three main parts : (1) applying an existing mapping process to generate a list of synchronized names ; (2) extracting the axioms with synchronized names *(i.e. rewrite all ontologies with the same signatures)* from the initial sources ; (3) applying the ontology merging method proposed in this paper. Namely, the steps of the third part are following :

1. Given the signature of the different ontologies, we enumerate all possible ways to express knowledge intuitively *(therein, one possible way will be the result of merging)*.
2. We express these possible ways semantically in terms of the interpretation.
3. We apply merging theory to measure the satisfaction of each solution w.r.t. ontology profile.

In the following, we will describe the merging steps in detail.

3.2 Generating Possible Merging Solutions

Given the set of axioms, we first rewrite them in normal form [1]. These rules are re-written by a combination bet-

ween the relations and the set of concepts names, called *pattern* [3]. We define the patterns as follows :

Definition 2 (Pattern) *Let the symbols $\star, \diamond$ be the representative placeholders of the concepts names. Say that we use π to denote a pattern. The patterns are as follows : subsumption ($\star \sqsubseteq \diamond$), intersection ($\star \sqcap \diamond$), existence ($\star \sqsubseteq \exists r.\diamond$, $\exists r.\star \sqsubseteq \diamond$). The set of patterns is denoted as ξ.*

In order to have an axiom, we collect concept pairs from the set of concepts defined as follows :

Definition 3 (Concept Pair) *Let N_C be the set of concept names. A pair of concept names is an element of $N_C \times N_C$. For brevity, we shall often write a pair (A, A') as AA'. The set of concept name pairs is defined as N_C^2.*

We utilize Definition 3 to generate the concept names for the combination per each pattern. In other words, let $(A, B) \in N_C^2$ and π be a pattern. The instantiation of π with AB, denoted π_{AB}, is the result of replacing $\star$ with A and $\diamond$ with B in π. For example, π has a pattern as $(\star \sqsubseteq \exists r.\diamond)$ and AB is a concept name pair, we have the outcome of replacing as $A \sqsubseteq \exists r.B$. Replacement of the concept names with the placeholders in the patterns is called "a concept pattern". The set of concept patterns is defined as follows :

Definition 4 (Concept Pattern) *Let ξ be the set of patterns and let N_C^2 be the set of concept name pairs. Let $\times$ be a combination between the patterns and the concept name pairs. The set of concept patterns, denoted as α, is defined as follows :*

$$\alpha = N_C^2 \times \xi$$

Here, we have a function which takes as input a pair of atomic concepts and a pattern to map them aimed finally at collecting the actual axioms. Next, we combine between the concept patterns to each other re-called a *combination*. In this paper, we concentrate on *"combinations"* to generate all possible ways to express knowledge. The set of combinations is defined as follows :

Definition 5 (Combination) *Let α be the set of concept patterns. The set of combinations, denoted as $\mathcal{H}$, is defined as follows :*

$$\mathcal{H} = \bigcup_{2 \leq n \leq |\alpha|} \{X \mid X \subseteq \alpha \text{ and } |X| = n\}$$

We use Definition 4 and 5 to generate all the possible knowledge that can be expressed the same terminology. To illustrate the generation of the combinations, considering three concepts $N_C = \{A, B, C\}$ we apply Definition 3 to have N_C^2=$\{AB, AC, BC, \ldots\}$. After that, we apply Definition 2 and Definition 4 to combine N_C^2 with the patterns ξ to have the concept patterns $\alpha = (N_C^2 \times \xi) = \{A \sqsubseteq B, B \sqsubseteq C, A \sqsubseteq \exists r.C \ldots\}$. Finally, the result of the set of combinations *(apply Definition 5)* is as follows : $\mathcal{H}$=$\{\{A \sqsubseteq B, A \sqsubseteq C\}, \{A \sqsubseteq B, B \sqsubseteq C\}, \ldots\}$. After generating all possible knowledge (combinations) where an agreement between the merging sources can be found, we need to generate the corresponding interpretations.

Next, we discuss the constraints and how to generate all possible interpretations. Regarding the constraints, there are three constraint cases as follows : (1) the interpretation doesn't not contain the empty set, (2) each concept must exist at least one representative individual, (3) the interpretation is non duplicated. Explaining to the first case, the interpretation is normally able to contain the empty set (e.g. $\mathcal{W}$=$\{A^{\mathcal{I}}$=$\{\}$, $B^{\mathcal{I}}$=$\{b, c\}$, $C^{\mathcal{I}}$=$\{c\}\}$ where the set of concept A is empty). However, we need to eliminate these cases since a concept interpreted as an empty set will be subsumed by all the concepts of the ontology including those which are semantically different. Therefore, we focus crucially on generating the interpretations in which each concept will store completely individuals, e.g. $\mathcal{W}$=$\{A^{\mathcal{I}}$=$\{a\}$,$B^{\mathcal{I}}$=$\{a, b, c\}$,$C^{\mathcal{I}}$=$\{a, c\}\}$. More formally, the set of all possible interpretations is generated and satisfied the constraints defined as follows :

Definition 6 *Let $\mathcal{I}$ be an interpretation that will be used for the merging process. The constraints for generating the possible interpretations are as follows : one interpretation $\mathcal{I}$ is **removed** from the set of possible interpretations if :*

- *There is $A \in N_C$ s.t. $A^{\mathcal{I}} = \emptyset$; or*
- *There is $a \in N_I$ s.t. for every $A \in N_C, a^{\mathcal{I}} \notin A^{\mathcal{I}}$; or*
- *There is $\mathcal{I}' = \langle \Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'} \rangle$ s.t. $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ and $\cdot^{\mathcal{I}} = \cdot^{\mathcal{I}'}$.*

From the constraints of Definition 6, the collected interpretations are able to be a representative of the common knowledge expressing in ontologies sources. Next, we introduce a closure definition. This definition is also the foundation of collecting all possible interpretations.

Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ with $\mathcal{A}_{\mathcal{I}} = \{A(a) \mid a \in N_I \text{ and } \mathcal{I} \models A(a)\} \cup \{r(a, b) \mid a, b \in N_I \text{ and } \mathcal{I} \models r(a, b)\}$. We denote the pseudo-ABox induced by $\mathcal{I}$. The closure definition is :

Definition 7 (Closure) *Let Φ be an axiom, B be a concept, r be a role and $\{a, b\}$ be individuals. The closure of Φ is defined as follows : $Cl_{\Phi}(\mathcal{A}_{\mathcal{I}})$=$\{B(a) \mid \langle \Phi, \mathcal{A}_{\mathcal{I}} \rangle \models B(a)\} \cup \{r(a, b) \mid \langle \Phi, \mathcal{A}_{\mathcal{I}} \rangle \models r(a, b)\}$.*

An illustration to explain the closure as follows : assuming we have Φ=$\{A \sqsubseteq B, B \sqsubseteq C\}$ and $\mathcal{S}$= $\{A^{\mathcal{I}}$= $\{a\}$, $B^{\mathcal{I}}$= $\{b\}$, $C^{\mathcal{I}}$= $\{c\}\}$, then the set of closures with Φ is $Cl_{\Phi}(\mathcal{S})$= $\{A^{\mathcal{I}}$=$\{a\}$, $B^{\mathcal{I}}$= $\{a, b\}$, $C^{\mathcal{I}}$= $\{a, b, c\}\}$. The reason why we need this definition is to find/generate the interpretations based on the considering axioms. Now, we generate the interpretations using the closure and the set of combinations (axioms). The set of interpretations is defined as follows :

3. We can see these patterns as second-order is-a relations, whose instances are the concepts from the ontologies.

Definition 8 *Let $\mathcal{H}$ be a set of combinations and $\mathcal{A}_\mathcal{I}$ be the pseudo-ABox. The closure of $\Phi_i \in \mathcal{H}$ is denoted as $Cl_{\Phi_i}(\mathcal{A}_\mathcal{I})$. The set of interpretations, denoted as $\mathcal{W}$, is defined as follows :*

$$\mathcal{W} = \bigcup_{1 \leq i \leq |\mathcal{H}|} Cl_{\Phi_i}(\mathcal{A}_\mathcal{I})$$

Furthermore, assuming the case of $\Phi_2 = \{C \sqsubseteq B \sqcap A\}$, the normalization rules of normal form $\mathcal{EL}$ [21] will be applied to have $\Phi_2 = \{C \sqsubseteq B, C \sqsubseteq A\}$, then the set of closures with Φ_2 is $Cl_{\Phi_2}(\mathcal{A}_\mathcal{I})$=$\{A^\mathcal{I}$=$\{a, c\}$,$B^\mathcal{I}$=$\{b, c\}$,$C^\mathcal{I}$=$\{c\}\}$. Finally, we aggregate them (apply Definition 6, 7 8) to have the set of interpretations. The result of the set of all interpretations is as follows : $\mathcal{W}$=$\{\{A^\mathcal{I}$=$\{a\}$,$B^\mathcal{I}$=$\{b, c\}$,$C^\mathcal{I}$=$\{a, b, c\}\}$,$\{A^\mathcal{I}$=$\{a\}$, $B^\mathcal{I}$=$\{a, b, c\}$, $C^\mathcal{I}$=$\{a, b, c\}\}$, ...$\}$.

In order to illustrate particularly for generating the combinations and all the set of possible interpretations, the examples will be presented in detail as follows :

Example 1 *Let $\odot$ be a merging ontology. We consider two examples $\odot_1, \odot_2$ of the three following ontologies $\odot = \{O_1, O_2, O_3\}$ in which three concepts A, B, C have the same names ($A, B, C \in O_{1,2,3}$)*

$$\odot_1 := \{\, O_1 : A \sqsubseteq B, B \sqsubseteq C, O_2 : C \sqcap A \sqsubseteq B, O_3 : B \sqsubseteq A, A \sqsubseteq \exists r.C \,\}$$

$$\odot_2 := \{\, O_1 : C \sqsubseteq B, A \sqsubseteq B, O_2 : A \sqsubseteq B, B \equiv C, O_3 : A \sqsubseteq C \sqcap B \,\}$$

First of all, the combination between the concept names $N_C^2 = \{A, B, C\}$ and six forms (corresponding to the concept patterns) will generate $6 \times |N_C^2| = 6 \times 3 = 18$ patterns (column 1 of Figure 2). Next step, we apply Definition 4 to collect the set of combinations. The number of the set of combinations is (2-permutations of 18 (C_n^2) - duplication cases = C_{18}^2 - duplication cases) = 153 - 49 = 104 combinations (duplication cases are the combinations of same concepts (e.g. $\{A \sqsubseteq B, A \sqcap B\}$, $\{B \sqsubseteq C, \exists r.C \sqsubseteq B\}$)). The set of combinations is presented in column 2 of Figure 2.

Next, all interpretations are also generated by the initial interpretation ($A^\mathcal{I}$=$\{a\}$, $B^\mathcal{I}$=$\{b\}$, $C^\mathcal{I}$=$\{c\}$, $r^\mathcal{I}$=(a, b)) and the set of combinations $\mathcal{H}$. Applying the Definition 8, the total number of collected interpretations are ($\Sigma_{k \in \mathcal{H}} |(N_{I_k})|$)=104 interpretations. The set of combinations is presented in column 3 of Figure 2. Notice that, each interpretation must satisfy Definition 6. Particularly, each interpretation $\mathcal{I}$ must contain the individuals a, b, c. The purpose of this work is to avoid losing any pieces of information.

After collecting the set of all possible interpretations, we introduce an ontology merging method based on the belief merging theory in the next section.

3.3 Selection of The Best Interpretation

In this section, we provide ontology merging operators and explain how to compute the distance between $\mathcal{I}$ and O. The reason to consider this distance is to seek the interpretations to be the representative of ontologies sources that satisfy their axioms. First of all, let O be an ontology and let $\Phi \in O$. The set of all interpretations is noted $\mathcal{W}$. $Mod(\Phi)$ denotes the set of models of Φ, i.e., $Mod(\Phi) = \{\mathcal{I} \in \mathcal{W} \mid \mathcal{I} \models \Phi\}$. Additionally, let an ontology O be a finite set of axioms Φ, i.e, $O = \{\Phi_1, ..., \Phi_n\}$ with the interpretation $\mathcal{I}$. Let $\odot$ be a set of ontologies sources ($\odot = \{O_1, ..., O_n\}$). Let $\oplus$ be an ontology merging operator which assigns each ontology set $\odot$ to a set of selected interpretations, denoted by $\oplus(\odot)$. A model-based operator is defined by selecting the interpretations that are the closest from the ontologies sources. First, we compute the distance between the interpretation $\mathcal{I}$ and the ontology O through axioms $\Phi_i \in O$ as :

Concept Patterns (Alpha):	Set of Combinations ($\mathcal{H}$):	All of Interpretations ($\mathcal{W}$):
1. ('A', '⊑', 'B')	1. (('A', '⊑', 'B'), ('A', '⊑', 'C'))	1. [['a'], ['a', 'b'], ['a', 'c']]
2. ('B', '⊑', 'A')	2. (('A', '⊑', 'B'), ('C', '⊑', 'A'))	2. [['a', 'c'], ['a', 'b', 'c'], ['c']]
3. ('A', '≡', 'B')	3. (('A', '⊑', 'B'), ('A', '≡', 'C'))	3. [['a', 'c'], ['a', 'b', 'c'], ['a', 'c']]
4. ('A', '⊓', 'B')	4. (('A', '⊑', 'B'), ('A', '⊓', 'C'))	4. [['a'], ['a', 'b'], ['a', 'b', 'c']]
5. ('A', '⊑ ∃R', 'B')	5. (('A', '⊑', 'B'), ('A', '⊑ ∃R', 'C'))	5. [['a'], ['a', 'b', 'c'], ['c']]
6. ('A', '∃R ⊑', 'B')	6. (('A', '⊑', 'B'), ('A', '∃R ⊑', 'C'))	6. [['a'], ['a', 'b', 'c'], ['a', 'b', 'c']]
7. ('A', '⊑', 'C')	7. (('A', '⊑', 'B'), ('B', '⊑', 'C'))	7. [['a', 'b'], ['b'], ['a', 'b', 'c']]
8. ('C', '⊑', 'A')	8. (('A', '⊑', 'B'), ('C', '⊑', 'B'))	8. [['a', 'b', 'c'], ['b'], ['c']]
9. ('A', '≡', 'C')	9. (('A', '⊑', 'B'), ('B', '≡', 'C'))	9. [['a', 'b', 'c'], ['b'], ['a', 'b', 'c']]
10. ('A', '⊓', 'C')	10. (('A', '⊑', 'B'), ('B', '⊓', 'C'))	10. [['a', 'b'], ['b'], ['b', 'c']]
11. ('A', '⊑ ∃R', 'C')	11. (('A', '⊑', 'B'), ('B', '⊑ ∃R', 'C'))	11. [['a', 'b', 'c'], ['b', 'c'], ['c']]
12. ('A', '∃R ⊑', 'C')	12. (('A', '⊑', 'B'), ('B', '∃R ⊑', 'C'))	12. [['a', 'b', 'c'], ['b', 'c'], ['b', 'c']]
13. ('B', '⊑', 'C')	13. (('B', '⊑', 'A'), ('A', '⊑', 'C'))	13. [['a', 'b'], ['a', 'b'], ['a', 'b', 'c']]
14. ('C', '⊑', 'B')	14. (('B', '⊑', 'A'), ('C', '⊑', 'A'))	14. [['a', 'b', 'c'], ['a', 'b', 'c'], ['c']]
15. ('B', '≡', 'C')	15. (('B', '⊑', 'A'), ('A', '≡', 'C'))	15. [['a'], ['b'], ['a', 'b', 'c']]
16. ('B', '⊓', 'C')	16. (('B', '⊑', 'A'), ('A', '⊓', 'C'))	16. [['a'], ['a', 'b', 'c'], ['a', 'c']]
17. ('B', '⊑ ∃R', 'C')	17. (('B', '⊑', 'A'), ('A', '⊑ ∃R', 'C'))	17. [['a', 'b', 'c'], ['b'], ['b', 'c']]
18. ('B', '∃R ⊑', 'C')	18. (('B', '⊑', 'A'), ('A', '∃R ⊑', 'C'))	18. [['a', 'c'], ['b', 'c'], ['c']]

FIGURE 2 – Column 1 : patterns of concepts A, B, C (line 1 : $A \sqsubseteq B$, line 6 : $\exists r.A \sqsubseteq B$) ; Column 2 : Set of Combinations with n=2 ; Column 3 : All interpretations ($A^\mathcal{I}$=$\{a\}$, $B^\mathcal{I}$=$\{a, b\}$, $C^\mathcal{I}$=$\{a, c\}$)

Definition 9 *Let $\mathcal{I}$ be an $\mathcal{EL}$ interpretation and let O be an ontology. The distance between $\mathcal{I}$ and O is defined :*

$$dist(\mathcal{I}, O) = |\{\Phi_i \in O \mid \mathcal{I} \not\models \Phi_i\}|$$

So the distance from $\mathcal{I}$ to O is the number of axioms in O that are not satisfied by $\mathcal{I}$. Considering the unsatisfaction is also known as the disagreement of $\mathcal{I}$ with O, in this case the disagreement will be computed particularly at axiom level. Notice that, when computing the distance of an interpretation to the ontology O, we consider both explicit and implicit knowledge. $\mathcal{I} = \{A^\mathcal{I} = \{a\}, \{B^\mathcal{I} = \{b\}, \{C^\mathcal{I} = \{a, b, c\}\}$, then $dist(\mathcal{I}, O_1) = 2$ is equal to $dist(\mathcal{I}, O_2) = 2$ when considering the closure, but it is not the case if we only consider explicit knowledge.

Next, ontology merging operators are based on the aggregation of these distances. The idea is to find the closest information to the overall knowledge set. So, suppose that we have an aggregation function g [8], then we define :

Definition 10 ($\oplus_g$) *Let $\mathcal{I}$ be an interpretation, let $\odot$ be a set of merged ontologies, and g be an aggregation function. We define the distance $dist_g$ as follows :*

$$dist_g(\mathcal{I}, \odot) = g_{O \in \odot}(dist(\mathcal{I}, O))$$

Interpretations	O_1	O_2	O_3	d_{max}	d_Σ	d_{Gmax}
A(a), B(a,b,c), C(a,b,c)	0	0	0	0	0	(0,0,0)
A(a,b), B(a,b,c), C(a,b,c)	0	0	0	0	0	(0,0,0)
A(a,c), B(a,b,c), C(a,b,c)	0	0	0	0	0	(0,0,0)
A(a,c), B(a,b,c), C(a,c)	0	1	0	1	1	(1,0,0)
...						
A(a,b), B(a,b,c), C(c)	0	1	1	1	2	(1,1,0)
A(a,b,c), B(a,b,c), C(a,c)	0	1	1	1	2	(1,1,0)
...						
A(a), B(a,b), C(b,c)	1	1	1	1	3	(1,1,1)
A(a), B(b,c), C(b,c)	1	1	1	1	3	(1,1,1)
A(a,b,c), B(b,c), C(c)	1	2	1	2	4	(2,1,1)
...						

TABLE 2 – The distances between the interpretations $\mathcal{I}$ and the ontologies of $\odot_2$

In this paper we will focus on the aggregation functions $g \in \{\max, \Sigma, leximax\}$ [4], that echoes to the most usual propositional merging operators [10, 18, 12, 11]. We have an assignment that maps each knowledge base set $\odot$ to a pre-order $\leq_\odot$ over interpretations (on $\mathcal{W}$), refering to [11]. And finally we define our ontology merging operators $\oplus_g$ as

Definition 11 *Let $\odot$ be a set of ontologies, g be an aggregation function, and $\mathcal{I}, \mathcal{J} \in \mathcal{W}$. Let $\leq^g_\odot$ be a pre-order over interpretations using g. We define the ontology merging operator $\oplus_g$ as*

$$\mathcal{I} \leq^g_\odot \mathcal{J} \; iif \; dist_g(\mathcal{I}, \odot) \leq dist_g(\mathcal{J}, \odot)$$

$$Mod(\oplus_g(\odot)) = \min(\leq^g_\odot)$$

Example 2 *Let us continue with example 1 to illustrate the Definitions. Considering $\odot_1$, the calculation of the distance between the interpretation $\mathcal{I}$ and each ontology O is as follows : assuming we have the interpretation $\mathcal{I}_1 = \{A^{\mathcal{I}_1} = \{a, b, c\}, B^{\mathcal{I}_1} = \{b\}, C^{\mathcal{I}_1} = \{a, b, c\}, r^{\mathcal{I}_1} = (a, b)\}$*

Computing the distance from $\mathcal{I}$ to $\odot_1$ as follows : $dist(\mathcal{I}_1, O_1) = d^{\mathcal{I}_1}_{O_1} = 1$ because $\mathcal{I}_1$ is unsatisfied an axiom of O_1 (namely $\mathcal{I}_1 \not\models A \sqsubseteq B$ and $\mathcal{I}_1 \models B \sqsubseteq C$); similarly, $d^{\mathcal{I}_1}_{O_2} = 1$, an explanation with O_2 : $(C \sqcap A)^{\mathcal{I}_1} = \{a, b, c\}$ and $\{a, b, c\} \notin B^{\mathcal{I}} = \{b\}$; Finally, the distance of $\mathcal{I}$ with O_3 is $d^{\mathcal{I}_1}_{O_3} = 1$ due to $\mathcal{I}_1 \models B \sqsubseteq A$ and $\mathcal{I}_1 \not\models A \sqsubseteq \exists r.C$. Accordingly, we have the distance between $\mathcal{I}$ and $\odot_1$ with applying the Definition 10, 11 as follows : $dist_{\max}(\mathcal{I}_1, \odot_1) = max(d^{\mathcal{I}_1}_{O_1}, d^{\mathcal{I}_1}_{O_2}, d^{\mathcal{I}_1}_{O_3}) = max(1, 1, 1) = 1$, $dist_\Sigma(\mathcal{I}_1, \odot_1) = \sum_{i=1}^{3}(d^{\mathcal{I}_1}_{O_i}) = (1+1+1) = 3$, $dist_{Gmax}(\mathcal{I}_1, \odot_1) = (1, 1, 1)$.

From the outcome of $\odot_2$, there are three interpretations selected : $Mod(\oplus_g(\odot_2)) = \{\{A^{\mathcal{I}} = \{a\}, B^{\mathcal{I}} = \{a, b, c\}, C^{\mathcal{I}} = \{a, b, c\}\}; \{A^{\mathcal{I}} = \{a, b\}, B^{\mathcal{I}} = \{a, b, c\}, C^{\mathcal{I}} = \{a, b, c\}\}; \{A^{\mathcal{I}} = \{a, c\}, B^{\mathcal{I}} = \{a, b, c\}, C^{\mathcal{I}} = \{a, b, c\}\}$.

4. We will write *Gmax* instead of *leximax* like in [10].

Algorithm 1 SIF algorithm

Input in IList=[] : Interpretations List (the closest distance)
Output in SIF={}

```
Let OneI ← ∅
for each concept, indivList : IList do
    N ← getConceptName(concept)
    OneI[N] ← OneI[N] ∪ getIndividuals(indivList)
end for
for each nameConcept, valueList : OneI do
    F ← Frequency(valueList) //F_A={a :3, b :1, c :1}
    T ← len(nameConcept)/2   /*T=3/2=1.5*/
    for each i, v : F do
        if v > T and v = len(IList) then
            SIF[nameConcept] ← SIF[nameConcept] ∪ i
        end if
    end for
end for
return SIF
```

Once the set of closest interpretations from the profile is selected (using the operators described in this section), we have to build an ontology that correspond to these interpretations. This is the aim of the SIF algorithm that we present in the next section.

3.4 Expressing Result of Merging

In the previous section, we computed the set of interpretations that best represent the information contained in the profile with less disagreement. But we need to translate this result into an ontology. This is the translation that we propose here. We need only one interpretation to represent the outcome of merging such that avoid the use of the logical disjunction ($\vee$). Hence, we built a *Selected Interpretation Frequency* (SIF) algorithm based on the frequency of individuals of each concept to decide the selection. The algorithm's idea is to collect/put all individuals *(of the selected interpretations)* into one interpretation. Next, we compute the frequency of individuals using majority technique to select the final result.

Dataset	Type	Number of Concept	Number of Axiom	Number of is-a relation
SUMO	(1)	4558	587842	5330
	(2)	3432	2138	2138
Wikidata	(1)	69188843	2941036	2941036
	(2)	119152	16876	16876
Babelnet	(1)	6113467	277036611	15831054
	(2)	119957	165121	165121

TABLE 3 – The number of Concepts, Axioms, i-a Relations of three ontologies sources. (1) is the number of the original ontology, (2) is the number after the mapping process.

Notice that we only consider assertions free open-domain ontologies and we only focus on merging terminological knowledge. A pseudo-ABox is then assumed in this paper. Therefore, when there is a real ABox and a semantic conflict problem arise, our merging method is able to be applied.

Let us now define the full ontology merging process as :

Definition 12 *$f_\oplus$ is called a full ontology merging operator if $\oplus$ is an ontology merging operator and*

$$f_\oplus(\odot) = SIF(Mod(\oplus_g(\odot)))$$

Accordingly, the SIF algorithm will be described in detail as follows : the input of SIF algorithm is the list of closest interpretations and their output is one most general interpretation. Explaining explicitly as follows : assume that there are the individuals in each concept including $a \in A$, $b \in B$, $c \in C$ [5]. Firstly, *(continue with Example 2)* collect all the individuals of interpretations *(line 2 to 5)*, the outcome is $\{A : \{a, a, a, b, c\}$, $B : \{a, a, a, b, b, b, c, c, c\}$, $C : \{a, a, a, b, b, b, c, c, c\}\}$. Secondly, we computed the frequency of the individuals in each concept, the result includes : $\{A^I = \{a : 3, b : 1, c : 1\}$, $B^I = \{a : 3, b : 3, c : 3\}$, $C^I = \{a : 3, b : 3, c : 3\}\}$ *(line 7)*. Next, the purpose is to select one most plausible interpretation, we utilize the idea of [3] majority vote algorithm to find a threshold *($T = n/2$) (line 8)* and to filter out the individuals in each concept. Furthermore, we use the threshold and the unanimity rule *(v=len(IList) - line 10)* to select one most plausible interpretation *(line 9 to 13)*. In the case of the example $\odot_2$, the selected result with using the SIF algorithm is as follows : $f_\oplus(\odot_2) = \{A^I = \{a\}, B^I = \{a, b, c\}, C^I = \{a, b, c\}\}$. Finally, due to the output of SIF algorithm is the one interpretation, it has also derived from the set of combinations generated before. Therefore, we pick up the one combination corresponding (satisfying) to the selected interpretation upshot *(one interpretation corresponds to one combination (syntactic))*. In the following, we show how to translate the selected interpretation into an ontology.

Definition 13 *Let $\mathcal{H}$ be the set of combinations (axioms) and $f_\oplus$ be the full ontology merging operator. The syntactic ontology merging result, denoted as $\mathcal{Y}$, is defined as*

$$\mathcal{Y} = \{M \mid M \in \mathcal{H} \text{ and } f_\oplus(\odot) \models M\}$$

Applying Definition 13, we are looking for an axiom in the combinations $\mathcal{H}$ that $f_\oplus(\odot)$ satisfies $M \in \mathcal{H}$ *(one interpretation $f_\oplus(\odot)$ corresponds to one axiom M)*. The syntactic ontology merging outcome of the example $\odot_2$ is : $\mathcal{Y} = \{A \sqsubseteq C, B \sqsubseteq C\}$.

4 Experimental Evaluation

This section shows the effectiveness of our approach.

5. All interpretations contain these individuals.

```
Brandy,bn:03463167n,Q676745,A14
Brandy,bn:00874259n,Q301905,A15
Brandy,bn:00005720n,Q503252,A16
Brandy,bn:00041425n,Q693078,A17
Centimeter,bn:00017163n,Q174728,A18
Fish,bn:00004288n,Q2711102,A19
Fish,bn:02768902n,Q3840698,A20
Fish,bn:00052347n,Q168422,A21
Fish,bn:00015903n,Q625161,A22
Fish,bn:00014270n,Q640454,A23
Fish,bn:00023566n,Q1063438,A24
```

FIGURE 3 – Example of mappings and concept name synchronization (The four elements including (1) is SUMO, (2) is Babelnet, (3) is Wikidata, (4) Synchronized Concept Names)

4.1 Methodology

We consider the following three ontologies : SUMO [6], Wikidata [7], Babelnet [8], which are large open domain ontologies. In our experiment, we consider axioms of the form $A \sqsubseteq B$, which are the most common axioms used to express knowledge, where A and B are concept names, for which we have a mapping between the different sources. First of all, we collected the concepts from the three knowledge sources. The number of concepts gathered is as follows : 4558 concepts for SUMO, 69188843 concepts for WikiData, and 6113467 for Babelnet *(including the total number of Babel synsets : 15780364 and the total number of Babel senses : 808974108)*. The number of the considered axioms (is-a relation) of the original ontologies is as follows : 5330 axioms for SUMO, 2941036 for Wikidata, 15831054 for Babelnet. We also provide statistics of the ontologies before and after the mapping process at Table 3. The application input is then the three open-domain ontologies and the outcome is the merged knowledge that agree with the input ontologies. Additionally, our main approach focuses crucially on the same concepts from ontologies sources. However, the concept names of the set of considered practical ontologies is different about vocabulary/lexical. Therefore, we utilize the mapping process between the three sources to seek the correspondences between them. For collecting and utilizing the same concept names, we create a synchronization of the new names of concepts based on those mappings. The mapping process is represented in the next subsection.

4.2 Mapping

The first step we determine the mapping between the three ontologies. From SUMO, we use the WordNet ID associated to each concept, which is given by the annotations associated to the concepts, to link it to its corresponding WikiData concept. For BabelNet, we use then Babelnet's

6. http ://www.adampease.org/OP
7. https ://www.wikidata.org
8. https ://babelnet.org

synset to link the WikiData concepts to BabelNet concepts, and thus to the concepts from SUMO. For SUMO concepts that do not have WordNet ID, we use BabelNet API [9] to find the corresponding BabelNet ID using the name of the concept. This work is to ensure that the concept names are the same [10]. The mapping process includes the steps as follows : (1) because the SUMO's concept number is the smallest, hence, we collect the SUMO's concept names firstly. At each concept, we extract WordNet ID in the annotation of concepts ; (2) we implement two methods to collect Wikidata ID : (i) First, we use WordNet ID *(extracted at step 1)* to determine the synset of Babelnet. After that, we utilize BabelNet API to retrieve Wikidata ID for each BabelSense in a BabelSynset. *(The correspondences between Wikidata and Babelnet are available in the BabelNet's dataset).* (ii) The second one uses the SUMO's concept name to seek Wikidata ID through BabelSynset. Finally, the dataset is the combination between (i) and (ii). We also remove the duplication cases after combining them. (3) In the Wikidata, we check and collect the label/name of Wikidata ID through the qwikidata library [11]. Based on the mapping data collected from the three above steps, a synchronization process of the concept names based on each mapping (correspondence) conducted to generate the synchronized names. Namely, we assign a new name at the end position of each mapping *(column 4 of Figure 3)*. Then, a list of synchronized concept names collected as in Figure 3. This list is also the first part of the proposed framework. The reason why the synchronization of sources is essential because the concept names are not exactly the same in terms of lexical/character. Moreover, the considering problem in this paper is the ontologies sharing the same terminology. Therefore, the mapping process expected to synchronize the names. The number of links between the three sources ontologies is 156152 mappings. After the mapping process completed, we consider the concepts of the is-a relation with the new names.

4.3 Evaluation Task

For the evaluation of our merging method, we consider the top-7-levels of the three ontologies. The purpose of splitting the considered levels is to be able to evaluate the computation time needed to generate the combinations/interpretations as well as determine the number of conflicts at each level. For each level, we output the number of axioms of each ontology, the number of conflicts between sources, and the number of computed interpretations. The number of is-a relations *(with synchronized names)* for each level is given in Table 4. The experimental results are then provided in Table 5 where for each level, we show the number of generated interpretations and the time needed to compute the result of merging. We evaluate the merging process from generating all possible interpretations to the final outcome of the most plausible interpretation. The reason why we considered different levels (where level $n + 1$ include level n) is that the extension of each level increases the number of considered axioms *(is-a relation)*, the number of interpretations and the number of conflicts and then allow us to properly evaluate their impact on the time needed to compute the result of merging.

Level	SUMO	Wikidata	Babelnet
1	2336	2672	1284
2	2473	4007	11119
3	4904	5533	20969
4	5631	8238	64081
5	8098	17982	113971
6	11350	37841	204709
7	17996	69441	374260
8	30489	111640	600766
9	48631	157447	892876
10	60921	201538	1299495

TABLE 4 – The number of synchronizing the concept names with is-a relation.

4.4 Quantitative Results

Overall, from Table 5, we can see that our model of merging is fully efficient in computing the result and solving conflicts. We consider the common concepts shared between the three ontology sources to seek the conflicts (the intersection part of sources). Namely, we focus on the axioms that express knowledge in different ways.

Next, we compute the interpretations needed to determine the result of merging. Note that, all concepts are handled based on the synchronized names between the three sources. An experiment tested our proposal based on the number of conflicts and the processing time. The conflicts start to appear in level 2 with 783859 concepts. Subsequently, the number of contradiction increases in proportion to the expansive levels. The time of processing, in fact, is fast and effective. A piece of evidence is at level 10 with 15802 conflicts, the processing time is around 180,54 minutes (10832,55 seconds). In addition, the result of merging is quite expected because it solves the problem of heterogeneous and ambiguous information well. The most plausible interpretation satisfied all the axioms and obtained fully the information from the three ontologies.

4.5 Qualitative Result

In this section we will show how our full ontology merging operators solve semantic conflicts between the three

9. available at https://babelnet.org/guide

10. The datasets and the implementation of the merging operator is available at https://github.com/ontologymerging/beliefmerging

11. https ://qwikidata.readthedocs.io

Level	Concepts	Axioms (is-a)	Conflicts	Interpretations	Merging Time (s)
1	29748	S : 32 W : 10772 B : 18944	0	0	0
2	783859	S : 66 W : 38342 B : 745451	1	104	0,674
3	1717363	S : 120 W : 549710 B : 1167533	7	728	4,277
4	2972086	S : 251 W : 1420062 B : 1551773	53	5512	33,642
5	3968698	S : 615 W : 1682041 B : 2286042	326	33904	216,183
6	5503343	S : 1452 W : 2276445 B : 3225446	783	81432	527,298
7	8601261	S : 2698 W : 3872494 B : 4726069	1628	169312	1143,482
8	12907446	S : 4252 W : 6439584 B : 6463610	4086	425256	2827,383
9	19598261	S : 5656 W : 8520703 B : 11071902	9313	968552	6386,971
10	28189097	S : 6692 W : 9786145 B : 18396260	15802	1643408	10832,548

TABLE 5 – The experimental result with SUMO (S), Wikidata (W) and BabelNet(B). The result reports the number of concepts, axioms, interpretations and time to compute the result merging.

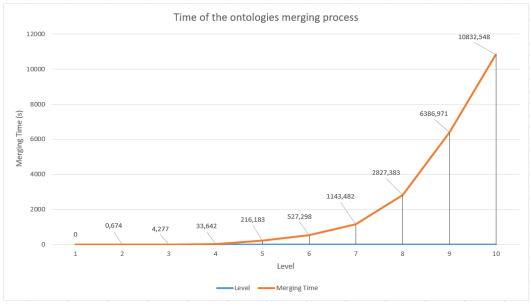


FIGURE 4 – Considering the computation time of ontology merging process in seven levels (time is expressed in seconds).

ontologies on illustrative simple (yet instructive) cases. In each example, we provide the input pieces of knowledge and the result of merging.

Example 3 *Considering the three concepts (Procedure, Technique, Process) represented as in Figure 5. This is the example in the introduction section :*

The merging upshot is : $Mod(\oplus(\odot_{Ex4}))= \{Procedure^{\mathcal{I}} = \{proced, tech\}, Techniques^{\mathcal{I}} = \{tech, proced\}, Process^{\mathcal{I}} = \{tech, proced, procce\}\}$. *The "Technique" and "Procedure" concept is opposite to each other, so the result can be an equivalent relation. However, we avoid this outcome.*

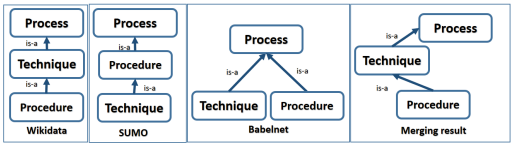


FIGURE 5 – Structure of concepts in Example 4

Example 4 *Considering the three concepts (Currency, Money, Top) represented as in Figure 6. We select to illustrate because both concepts belongs to Top concept.*

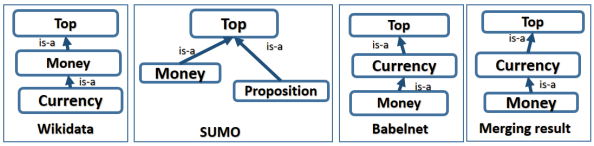


FIGURE 6 – Structure of concepts in Example 5

The merging upshot is : $Mod(\oplus(\odot_{Ex5}))= \{Top^{\mathcal{I}} = \{mon, cur\}, Money^{\mathcal{I}} = \{mon, cur\}, Currency^{\mathcal{I}} = \{cur, mon\}\}$. *The syntactic result is the same as Babelnet because they covered all how to express all the knowledge.*

5 Concluding remarks

This novel work is an initial part of a bigger research agenda to develop efficient algorithms and implementations for exploiting open-domain ontologies. Our main contribution is the use in practical case of belief merging theory, largely developed in the field of Knowledge Representation and Reasoning within propositional logic, to deal with merging problems existing when conflicts arise between knowledge coming from multiple open-domain ontology sources. To this end, we provided a formal definition of a semantic-based merging operator. We then demonstrated our approach using real examples of open-domain ontologies. A framework of ontology merging and how mapping between different sources is developed and made available online. Finally, experimental results as well as several practical examples are provided.

As future work, we plan to study how to ontology merging with the more expressive ontologies to cover most of the information from the open-domain ontologies. Furthermore, we implement an universal ontology merging methodology/tool to exploit open-domain ontologies.

Acknowledgments

This work was supported by the ANR CHAIRE IA BE4musIA : BElief change FOR better MUlti-Source Information Analysis.

Références

[1] Baader, Franz, Sebastian Brandt et Carsten Lutz: *Pushing the EL Envelope*. Dans *IJCAI'05*, page 364–369, San Francisco, CA, USA, 2005.

[2] Baader, Franz, Rafael Peñaloza et Boontawee Suntisrivaraporn: *Pinpointing in the Description Logic $\mathcal{EL}$*. Dans *KI'2007*, pages 52–67. Springer Berlin Heidelberg, 2007, ISBN 978-3-540-74565-5.

[3] Boyer, Robert S. et J. Strother Moore: *MJRTY—A Fast Majority Vote Algorithm*, pages 105–117. Springer Netherlands, Dordrecht, 1991, ISBN 978-94-011-3488-0.

[4] Chen, Zheng, Sun Yu, Wan Shengxian et Yu Dianhai: *RLTM : An Efficient Neural IR Framework for Long Documents*. Dans *IJCAI'19*, pages 5457–5463, 2019.

[5] Chocron, Paula et Marco Schorlemmer: *Attuning Ontology Alignments to Semantically Heterogeneous Multi-Agent Interactions*. Dans *ECAI'16*, page 871–879, NLD, 2016. ISBN 9781614996712.

[6] Dragoni, Mauro: *Multilingual Ontology Mapping in Practice : A Support System for Domain Experts*. Dans *ISWC'15*, tome 9367, pages 169–185, octobre 2015, ISBN 978-3-319-25009-0.

[7] Euzenat, Jérôme: *Interaction-based ontology alignment repair with expansion and relaxation*. Dans *IJCAI'17*, pages 185–191, Melbourne, Australia, août 2017. AAAI Press. euzenat2017a.

[8] Everaere, P., Sébastien Konieczny et Pierre Marquis: *On Egalitarian Belief Merging*. Dans *AAAI*, 2014.

[9] Gonçalves, Rafael, Matthew Horridge, Rui Li, Yu Liu, Mark Musen, Csongor Nyulas, Evelyn Obamos, Dhananjay Shrouty et David Temple: *Use of OWL and Semantic Web Technologies at Pinterest*, pages 418–435. ISWC'2019, octobre 2019, ISBN 978-3-030-30795-0.

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

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

[12] Lin, Jinxin et Alberto O. Mendelzon: *Knowledge Base Merging by Majority*, pages 195–218. Springer, Dordrecht, 1999, ISBN 978-94-017-1317-7.

[13] Mao, Ming: *Ontology Mapping : An Information Retrieval and Interactive Activation Network Based Approach*. Dans *The Semantic Web*, pages 931–935, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[14] Navigli, Roberto: *Natural Language Understanding : Instructions for (Present and Future) Use*. Dans *IJCAI'18*, pages 5697–5702, juillet 2018.

[15] Pan, Rong, Zhongli Ding, Yang Yu et Yun Peng: *A Bayesian Network Approach to Ontology Mapping*. Dans *The Semantic Web – ISWC 2005*, pages 563–577. Springer Berlin Heidelberg, novembre 2005.

[16] Pavão, Madalena et Catia Pesquita: *Complex matching for multiple ontologies : an exploratory study*. Dans *OM@ISWC*, 2018.

[17] Rakotoarison, Herilalaina, Marc Schoenauer et Michèle Sebag: *Automated Machine Learning with Monte-Carlo Tree Search*. Dans *IJCAI'19*, pages 3296–3303, août 2019.

[18] Revesz, Peter: *On the Semantics of Arbitration*. Intl. J. of Algebra and Computation, 07, mai 1997.

[19] Silva, Jomar da, Kate Revoredo, Fernanda Araujo Baião et Jérôme Euzenat: *Interactive ontology matching : using expert feedback to select attribute mappings*. Dans *OM - ISWC'18*, pages 25–36, octobre 2018.

[20] Solimando, Alessandro, Ernesto Jiménez-Ruiz et Christoph Pinkel: *Evaluating Ontology Alignment Systems in Query Answering Tasks*. Dans *ISWC'14*, 2014.

[21] Suntisrivaraporn, Boontawee: *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. Ph.D. thesis. Technical University of Dresden, juin 2009.

[22] Tulasi, R.Lakshmi et Maram Srinivasa Rao: *Survey on Techniques for Ontology Interoperability in Semantic Web*. Global. Journal of Computer Science and Technology,, 2014, ISSN 0975-4172.

Actes JIAF 2020

On the Decidability of a Fragment of Preferential LTL

Anasse Chafik **Fahima Cheikh-Alili**
Jean-François Condotta **Ivan Varzinczak**

CRIL, Univ Artois & CNRS, France
{chafik,cheikh,condotta,varzinczak}@cril.fr

Abstract

Linear Temporal Logic (LTL) has found extensive applications in Computer Science and Artificial Intelligence, notably as a formal framework for representing and verifying computer systems that vary over time. Non-monotonic reasoning, on the other hand, allows us to formalize and reason with exceptions and the dynamics of information. The goal of this paper is therefore to enrich temporal formalisms with non-monotonic reasoning features. We do so by investigating a preferential semantics for defeasible LTL along the lines of that extensively studied by Kraus et al. in the propositional case and recently extended to modal and description logics. The main contribution of the paper is a decidability result for a meaningful fragment of preferential LTL that can serve as the basis for further exploration of defeasibility in temporal formalisms.

1 Introduction

Specification and verification of dynamic computer systems is an important task, given the increasing number of new computer technologies being developed. Recent examples include blockchain technology and the various existing tools for home automation or the different production chains provided by Industry 4.0. Therefore, it is fundamental to ensure that systems based on them have the desired behavior but, above all, satisfy ever more demanding safety standards. This becomes even more critical with the increasing deployment of artificial intelligence techniques as well as the need to explain their behaviors.

Several approaches for qualitative analysis of computer systems have been developed. Among the most fruitful are the different families of temporal logic. The success of these is due mainly to their simplified syntax compared to that of first-order logic, their intuitive syntax, semantics and their good computational properties. One of the members of this family is Linear Temporal Logic [12, 16], known as LTL, is wildly used in formal verification and specification of computer programs.

Despite the success and wide use of linear temporal logic, it remains limited for modeling and reasoning about the real aspects of computer systems or those that depend on them. In fact, computer systems are not either 100% secure or 100% defective, and the properties we wish to check may have innocuous and tolerable exceptions, or conversely, exceptions that must be carefully addressed in order to guarantee the overall reliability of the system. Similarly, the expected behavior of a system may be correct not for all possible execution, but rather for its most "normal" or expected executions.

It turns out that LTL, because it is a logical formalism of the so-called classical type, whose underlying reasoning is that of mathematics and not that of common sense, does not allow at all to formalize the different nuances of the exceptions and even less to treat them. First of all, at the level of the object language (that of the logical symbols), it has operators behaving monotonically, and at the level of reasoning, posses a notion of logical consequence which is monotonic too, and consequently, it is not adapted to the evolution of defeasible facts.

Non-monotonic reasoning, on the other hand, allows to formalize and reason with exceptions, it has been widely studied by the AI community for over 40 years now. Such is the case of Kraus et al. [8] , known as the KLM approach.

However, the major contributions in this area are limited to the propositional framework. It is only recently that some approaches to non-monotonic reasoning, such as belief revision, default rules and preferential approaches, have been studied for more expressive logics than propo-

sitional logic, including modal [2, 4] and description logics [3]. The objective of our study is to establish a bridge between temporal formalisms for the specification and verification of computer systems and approaches to non-monotonic reasoning, in particular the preferential one, which satisfactorily solves the limitations raised above.

In this paper, we define a logical framework for reasoning about defeasible properties of program executions, we investigate the integration of preferential semantics in the case of LTL, hereby introducing preferential linear temporal logic $LTL^{\sim}$. The remainder of the paper is structured as follows: In Section 3 we set up the notation and appropriate semantics of our language. In Sections 4, 5 and 6, we investigate the satisfiability problem of this formalism.

2 Preliminaries

2.1 Linear Temporal Logic

The purpose of this section is to highlight the terminology and notation associated to the underlying linear temporal logic we shall assume in the remainder of the paper.

Let $\mathcal{P}$ be a finite set of *propositional atoms*. The set of operators in the *Linear Temporal Logic* can be split into two parts: the set of *Boolean connectives* ($\neg, \wedge$), and that of *temporal operators* ($\Box, \Diamond, \bigcirc, \mathcal{U}$), where $\Box$ reads as *always*, $\Diamond$ as *eventually*, $\bigcirc$ as *next* and $\mathcal{U}$ as *until*. The set of well-formed sentences expressed in LTL is denoted by $\mathcal{L}$. They are built up according to the following grammar:

$$\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Box\alpha \mid \Diamond\alpha \mid \bigcirc\alpha \mid \alpha\mathcal{U}\alpha$$

The temporal structure is a chronological linear succession of time points, we use the set of natural numbers in order to label each of these time points i.e., $(\mathbb{N}, <)$. Hence, a temporal interpretation associates each time point t with a truth assignment of all propositional atoms. A temporal interpretation is defined as follows:

Definition 1 (Temporal interpretation) *A* temporal interpretation *I is a mapping function $V : \mathbb{N} \longrightarrow 2^{\mathcal{P}}$ which associates each time point $t \in \mathbb{N}$ with a set of propositional atoms $V(t)$ corresponding to the set of propositions that are true in t. (Propositions not belonging to $V(t)$ are assumed to be false at the given time point.)*

We define the truth of a sentence in an LTL interpretation $I = V$ at a time point $t \in \mathbb{N}$, denoted by $I, t \models \alpha$ recursively as follows:

- $I, t \models p$ if $p \in V(t)$;
- $I, t \models \neg\alpha$ if $I, t \not\models \alpha$;
- $I, t \models \alpha \wedge \alpha'$ if $I, t \models \alpha$ and $I, t \models \alpha'$;
- $I, t \models \alpha \vee \alpha'$ if $I, t \models \alpha$ or $I, t \models \alpha'$;
- $I, t \models \Box\alpha$ if $I, t' \models \alpha$ for all $t' \in \mathbb{N}$ s.t. $t' \geq t$;
- $I, t \models \Diamond\alpha$ if $I, t' \models \alpha$ for some $t' \in \mathbb{N}$ s.t. $t' \geq t$;
- $I, t \models \bigcirc\alpha$ if $I, t + 1 \models \alpha$;
- $I, t \models \alpha\mathcal{U}\alpha'$ if $I, t' \models \alpha'$ for some $t' \geq t$ and for all $t \leq t'' < t'$ we have $I, t'' \models \alpha$.

A well-formed sentence $\alpha \in \mathcal{L}$ is *satisfiable* if there is a temporal interpretation I and a time point $t \in \mathbb{N}$ such that $I, t \models \alpha$.

2.2 KLM approach to non-monotonic reasoning

We give a brief outline to Kraus et al.'s [8] approach to non-monotonic reasoning. A propositional defeasible consequence relation $\mid\sim$ is defined as a binary relation on sentences of an underlying propositional logic. The semantics of preferential consequence relation is in terms of *preferential models*; these are partially-ordered structures with states labeled by propositional valuations. The semantics essentially allows for a strict partial order on states, with the states that are lower down being more plausible, normal or in a general case preferred, than those that are higher up.

Definition 2 (Preferential Models) *A preferential model on a set of atomic propositions $\mathcal{P}$ is is a tuple $\mathscr{P} \stackrel{\text{def}}{=} (S, l, \prec)$ where S is a set of elements called states, $l : S \longrightarrow 2^{\mathcal{P}}$ is a mapping which assigns to each state s a single world $m \in 2^{\mathcal{P}}$ and $\prec$ is a* strict partial *order on S satisfying smoothness condition.*

A preferential model satisfies a statement $\alpha \mid\sim \beta$ iff the minimal α-states are also β-states.

3 Preferential LTL

In this paper, we introduce a new formalism of reasoning about time that is able to distinguish between normal and exceptional points of time. We do so by investigating a defeasible extension of LTL with a preferential semantics.

The following example introduces a case scenario we shall be using in the remainder of this section, with the purpose of giving a motivation for this formalism and better illustrating the definitions in what follows.

Example 1. We have a computer program in which the values of its variables change with time. In particular, the agent wants to check two parameters, say x and y. These two variables take one and only one value between 1 and 3 on each iteration of the program. We represent the set of atomic propositions by $\mathcal{P} = \{x_1, x_2, x_3, y_1, y_2, y_3\}$ where x_i (resp. y_i) for all $i \in \{1, 2, 3\}$ is true iff the variable x (resp. y) has the value i in a current iteration. Figure 1 depicts a temporal interpretation corresponding to a possible behavior of such a program:

Under normal circumstances, the program assigns the value 3 to y whenever $x = 2$. We can express this fact using

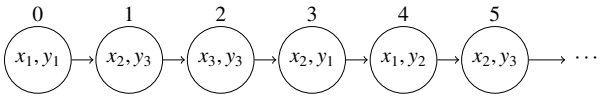


Figure 1: LTL interpretation V (for $t > 5$, $V(t) = V(5) = \{x_2, y_3\}$)

classical LTL as follows: $\Box(x_2 \rightarrow y_3)$, with $x_2 \rightarrow y_3$ is defined by $\neg x_2 \vee y_3$. Nevertheless, the agent notices that there is one exceptional iteration (Iteration 3) where the program assigns the value 1 to y when $x = 2$.

Some might consider that the current program is defective at some points of time. In LTL, the statement $\Box(x_2 \rightarrow y_3) \wedge \Diamond(x_2 \wedge y_1)$ will always be false, since y cannot have two different values in an iteration where $x = 2$. Nonetheless we want to propose a logical framework that is exception tolerant for reasoning about a system's behavior. In order to express this general tendency ($x_2 \rightarrow y_3$) while taking into account that there might be some exceptional iterations which do not crash the program.

3.1 Introducing defeasible temporal operators

Britz & Varzinczak [4] introduced new modal operators called defeasible modalities. In their setting, defeasible operators, unlike their classical counterparts, are able to single out normal worlds from those that are less normal or exceptional in the reasoner's mind.

We extend the vocabulary of classical LTL by adding defeasible modalities operators to its language. We add a new set of operators denoted as the set of *defeasible operators* ($\boxminus$, $\diamondminus$) to the existing vocabulary of *LTL*. They are built up according to the following grammar:

$$\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Box\alpha \mid \Diamond\alpha \mid \bigcirc\alpha \mid \alpha \mathcal{U} \alpha \mid \boxminus\alpha \mid \diamondminus\alpha$$

The intuition behind these new operators is the following: $\boxminus$ reads as *defeasible always* and $\diamondminus$ reads as *defeasible eventually*.

Example 2. Going back to our example 1, we can describe the normal behavior of the program using the statement $\boxminus(x_2 \rightarrow y_3) \wedge \Diamond(x_2 \wedge y_1)$. In all normal future time points, the program assigns the value 3 to y when $x = 2$. Although unlikely, there are some exceptional time points in the future where $x = 2$ and $y = 1$. But those are 'ignored' by the defeasible always operator.

The set of all well-formed $LTL^{\sim}$ sentences is denoted by $\mathcal{L}^{\sim}$. It is worth to mention that any well-formed sentence $\alpha \in \mathcal{L}$ is a sentence of $\mathcal{L}^{\sim}$.

We denote a subset of our language that contains only Boolean connectives, the two defeasible operators $\boxminus$, $\diamondminus$ and their classical counterparts by $\mathcal{L}^{\star}$.

Next we shall discuss how to interpret statements that have this defeasible aspect and how to determine the truth values of each well-formed sentence in $\mathcal{L}^{\sim}$.

3.2 Preferential semantics

First of all, in order to interpret the sentences of $\mathcal{L}^{\sim}$ we consider, as stated on the preliminaries, $(\mathbb{N}, <)$ to be a temporal structure. Hence, a temporal interpretation that associates each time point t with a truth assignment of all propositional atoms.

The preferential component of the interpretation of our language is directly inspired by the preferential semantics proposed by Shoham [14] and used in the KLM approach [8]. The preference relation $\prec$ is a strict partial order on our points of time. Following Kraus et al. [8], $t \prec t'$ means that t is more preferred than t'. The reasoner has now the tools to express the preference between points of time by comparing them w.r.t. each other, with time points lower down the order being more preferred than those higher up.

Definition 3 (Minimality w.r.t. $\prec$) *Let $\prec$ be a strict partial order on a set $\mathbb{N}$ and $N \subseteq \mathbb{N}$. The set of the minimal elements of N w.r.t. $\prec$, denoted by $min_{\prec}(N)$, is defined by $min_{\prec}(N) \stackrel{\text{def}}{=} \{t \in N \mid$ there is no $t' \in N$ such that $t' \prec t\}$.*

Definition 4 (Well-founded set) *Let $\prec$ be a strict partial order on a set $\mathbb{N}$. We say $\mathbb{N}$ is* well-founded w.r.t. $\prec$ *iff $min_{\prec}(N) \neq \emptyset$ for every $\emptyset \neq N \subseteq \mathbb{N}$.*

Following the line of reasoning of Kraus et al. [8], we define the preferential temporal interpretation as such:

Definition 5 (Preferential temporal interpretation) *An $LTL^{\sim}$ interpretation on a set of propositional atoms $\mathcal{P}$, also called preferential temporal interpretation on $\mathcal{P}$, is a pair $I \stackrel{\text{def}}{=} (V, \prec)$ where V is a temporal interpretation on $\mathcal{P}$, and $\prec \subseteq \mathbb{N} \times \mathbb{N}$ is a strict partial order on $\mathbb{N}$ such that $\mathbb{N}$ is well-founded w.r.t. $\prec$.*

We denote the set of preferential temporal interpretations by $\mathfrak{I}$.

In what follows, given a preference relation $\prec$ and a time point $t \in \mathbb{N}$, the set of *preferred time points relative to t* is the set $min_{\prec}([t, +\infty[)$ which is denoted in short by $min_{\prec}(t)$.

It is also worth to point out that given a preferential interpretation $I = (V, \prec)$ and $\mathbb{N}$, the set $min_{\prec}(t)$ is always a non-empty subset of $[t, +\infty[$ at any time point $t \in \mathbb{N}$.

Preferential temporal interpretations provide us with an intuitive way of interpreting sentences of $\mathcal{L}^{\sim}$. Let $\alpha \in \mathcal{L}^{\sim}$, let $I = (V, \prec)$ be a preferential interpretation, and let t be a time point in I in $\mathbb{N}$. Satisfaction of α at t in I, denoted $I, t \models \alpha$, is defined as follows:

- $I, t \models \boxminus\alpha$ if $I, t' \models \alpha$ for *all* $t' \in min_{\prec}(t)$;
- $I, t \models \diamondminus\alpha$ if $I, t' \models \alpha$ for *some* $t' \in min_{\prec}(t)$.

The truth values of Boolean connectives and classical modalities are defined the same way as in LTL. The intuition behind a sentence like $\boxminus\alpha$ is that α holds in *all* preferred time points that come after t. $\diamondminus\alpha$ intuitively means that α holds on at least one preferred time point relative in the future of t.

A well-formed sentence $\alpha \in \mathcal{L}^{\sim}$ is *preferentially satisfiable* if there is an $I \in \Im$ and a time point $t \in \mathbb{N}$ such that $I, t \models \alpha$. We can show that $\alpha \in \mathcal{L}^{\sim}$ is *preferentially satisfiable* iff there exists a preferential temporal interpretation I s.t. $I, 0 \models \alpha$. A sentence $\alpha \in \mathcal{L}^{\sim}$ is *valid* iff for all $I \in \Im$ and time points t in $\mathbb{N}$, we have $I, t \models \alpha$. A valid sentence is denoted by $\models \alpha$.

Example 3. Going back to Example 1, we can see that the time points 5 and 1 are more "normal" than iteration 3. By adding preferential preference $\prec := \{(5, 3), (1, 3)\}$, we denote the preferential temporal interpretation by $I = (V, \prec)$. We have the following:

- $I, 0 \not\models \Box(x_2 \rightarrow y_3) \wedge \Diamond(x_2 \wedge y_1)$;
- $I, 0 \models \boxminus(x_2 \rightarrow y_3) \wedge \Diamond(x_2 \wedge y_1)$.

We can see that the addition of $\prec$ relation preserves the truth values of all classical temporal sentences. Moreover, for every $\alpha \in \mathcal{L}$, we have that α is satisfiable in LTL if and only if α is preferentially satisfiable in $LTL^{\sim}$.

We discuss some properties of these defeasible modalities next. In what follows, let α, β be well-formed sentences in $\mathcal{L}^{\sim}$. We have:

$$\models \boxminus\alpha \leftrightarrow \neg\, \diamondminus\neg\alpha$$

Similarly as the case of $\Box$, we have the duality between our defeasible operators $\boxminus$ and $\diamondminus$.

$$\models \Box\alpha \rightarrow \boxminus\alpha\,, \models \diamondminus\alpha \rightarrow \Diamond\alpha$$

Intuitively, This property states that if a statement holds in all of future time points of any given point of time t, it holds on all our *future preferred* time points. As intended, this property establishes the defeasible always as "weaker" than the classical always. It can commonly be accepted since the set of all preferred future states are in the future. This is why we named $\boxminus$ *defeasible always*. On the other hand, we see that $\diamondminus$ is "stronger" than classical eventually, the statement within $\diamondminus$ holds at a preferable future.

The axiom of distributivity (K) can be stated in terms of our defeasible operators. We can also verify the validity of these two statements $\models \boxminus(\alpha \wedge \beta) \leftrightarrow (\boxminus\alpha \wedge \boxminus\beta)$ and $\models (\boxminus\alpha \vee \boxminus\beta) \rightarrow \boxminus(\alpha \vee \beta)$, the converse of the second statement is not always true.

The reflexivity axiom (T) for the classical operators does not hold in the case of defeasible modalities. We can easily find an interpretation $I = (V, \prec)$ where $I, t \not\models \boxminus\alpha \rightarrow \alpha$. Indeed, since we can have $t \notin min_{\prec}(t)$ for a temporal point t, we can have $I, t \models \boxminus\alpha$ and $I, t \models \neg\alpha$.

One thing worth pointing out is the set of future preferred time points changes dynamically as we move forward in time. Given three time points $t_1 \leq t_2 \leq t_3$, $t_3 \notin min_{\prec}(t_1)$ whilst $t_3 \in min_{\prec}(t_2)$ could be true in some cases. Hence, if $I, t \models \boxminus\boxminus\alpha$ does not imply that for all $t' \in min_{\prec}(t)$, $I, t' \models \boxminus\alpha$. Therefore, the transitivity axiom (4) does not hold also in our defeasible modalities. On the other hand, given those three time points, $t_3 \notin min_{\prec}(t_1)$ implies that $t_3 \notin min_{\prec}(t_2)$.

Since we do not have a version of the axioms (T) and (4) for our defeasible operators, we do not have the collapsing property on the case $\boxminus$, $\diamondminus$. Redundant sentences such as $\Box\Box\ldots\Box\alpha$ can be reduced to $\Box\alpha$. It is not the case for our preferential operators $\boxminus$ and $\diamondminus$.

3.3 State-dependent preferential interpretations

We shall now define a class of well-behaved $LTL^{\sim}$ interpretations that will be useful in the remainder of this paper.

Definition 6 (State-dependent preferential interpretations) *Let $I = (V, \prec) \in \Im$. I is state-dependent preferential interpretation iff for every $i, j, i', j' \in \mathbb{N}$, if $V(i') = V(i)$ and $V(j') = V(j)$, then $(i, j) \in \prec$ iff $(i', j') \in \prec$.*

In what follows, $\Im^{sd}$ denotes the set of all state-dependent interpretations. The intuition behind setting up this restriction is to have a more compact form of expressing preference over time points. In a way, time points with similar valuations are considered to be identical with regards to $\prec$, they express the same preferences towards other time points. Moreover, we have some interesting properties that do not in the general case. In particular, we have the following property :

Proposition 1 *Let $I = (V, \prec) \in \Im^{sd}$ and let $i, i', j, j' \in \mathbb{N}$ s.t. $i \leq i'$, $i' \leq j'$ and $j \in min_{\prec}(i)$. If $V(j) = V(j')$, then $j' \in min_{\prec}(i')$.*

This property is specific to the class of state-dependent interpretations. However, the following proposition is true for every $I \in \Im$.

Proposition 2 *Let $I = (V, \prec) \in \Im$ and let $i, j \in \mathbb{N}$ s.t. $j \in min_{\prec}(i)$. For all $i \leq i' \leq j$, we have $j \in min_{\prec}(i')$.*

4 A useful representation of preferential structures

One of the objectives of this paper is to establish some computational properties about the satisfiability problem. In order to do this, we introduce into the sequel different structures inspired by the approach followed by Sistla and Clarke in [15]. They observe that in every LTL interpretation, there is a time point t after which every t-successor's

valuation occurs infinitely many times. This is an obvious consequence of having an infinite set of time points and a finite number of possible valuations. That is the case also for $LTL^{\sim}$ interpretations.

Lemma 1 *Let $I = (V, \prec) \in \mathfrak{I}$. There exists a $t \in \mathbb{N}$ s.t. for all $l \in [t, +\infty[$, there is a $k > l$ where $V(l) = V(k)$.*

For an interpretation $I \in \mathfrak{I}$, we denote the first time point where the condition set in Lemma 1 is satisfied by t_I. We can split each temporal structure into two intervals, namely, an initial and a final part.

Definition 7 *Let $I = (V, \prec) \in \mathfrak{I}$. We define:*

$$\begin{array}{lcl} \mathit{init}(I) & \stackrel{\text{def}}{=} & [0, \mathrm{t}_I[; \\ \mathit{final}(I) & \stackrel{\text{def}}{=} & [\mathrm{t}_I, +\infty[; \\ \mathit{range}(I) & \stackrel{\text{def}}{=} & \{V(i) \mid i \in \mathit{final}(I)\}; \\ \mathit{size}(I) & \stackrel{\text{def}}{=} & \mathit{length}(\mathit{init}(I)) + \mathit{card}(\mathit{range}(I)). \end{array}$$

With $\mathit{length}(\cdot)$ we denote the length of a sequence and $\mathit{card}(\cdot)$ denotes the cardinality of a set.

In the size of I we count the number of time points in the initial part and the number of valuations contained in the final part. In what follows, we discuss some properties concerning this notions and the state dependent interpretations.

Proposition 3 *Let $I = (V, \prec) \in \mathfrak{I}^{sd}$ and let $i \leq j \leq i' \leq j'$ be time points in $\mathit{final}(I)$ s.t. $V(j) = V(j')$. Then we have $j \in \mathit{min}_{\prec}(i)$ iff $j' \in \mathit{min}_{\prec}(i')$.*

Lemma 2 *Let $I = (V, \prec) \in \mathfrak{I}^{sd}$ and $i \leq i'$ be time points of $\mathit{final}(I)$ where $V(i) = V(i')$. Then for every $\alpha \in \mathcal{L}^{\star}$, we have $I, i \models \alpha$ iff $I, i' \models \alpha$.*

What we have in Lemma 2 is that given an interpretation $I \in \mathfrak{I}^{sd}$, points of time in $\mathit{final}(I)$ that have the same valuations satisfy exactly the same sentences.

Definition 8 *[Faithful Interpretation] Let $I = (V, \prec) \in \mathfrak{I}^{sd}$, $I' = (V', \prec') \in \mathfrak{I}^{sd}$ be two interpretations over the same of atomic propositions $\mathcal{P}$. We say that I, I' are faithful interpretations iff:*

- $\{V(i) \mid i \in \mathbb{N}\} = \{V'(i) \mid i \in \mathbb{N}\}$,
- *for all $i, j, i', j' \in \mathbb{N}$ s.t. $V'(i') = V(i)$ and $V'(j') = V(j)$, we have $(i, j) \in \prec$ iff $(i', j') \in \prec'$.*

In the sequel, we write $\mathit{init}(I) \equiv \mathit{init}(I')$ as shorthand for the condition that states: $\mathit{length}(\mathit{init}(I)) = \mathit{length}(\mathit{init}(I'))$ and for each $i \in \mathit{init}(I)$ we have $V(i) = V'(i)$.

Lemma 3 *Let $I = (V, \prec) \in \mathfrak{I}^{sd}$, $I' = (V', \prec') \in \mathfrak{I}^{sd}$ be two faithful interpretations over $\mathcal{P}$ such that $V'(0) = V(0)$ (in case $\mathit{init}(I)$ is empty), $\mathit{init}(I) \equiv \mathit{init}(I')$, and $\mathit{range}(I) = \mathit{range}(I')$. Then for all $\alpha \in \mathcal{L}^{\star}$, we have the following:*

$$I, 0 \models \alpha \text{ iff } I', 0 \models \alpha.$$

Lemma 3 implies that the ordering of time points in $\mathit{final}(\cdot)$ does not matter, and what matters is the $\mathit{range}(\cdot)$ of valuations contained within it. It is worth to mention that Lemma 2 and 3 hold only in the case interpretations in $\mathfrak{I}^{sd}$ and they are not always true in the general case.

Sistla & Clarke [15] introduced the notion of acceptable sequences. We adapt this notion for our preferential temporal structures. We introduce then the notion of pseudo-interpretations that will come in handy in showing decidability of the satisfiability problem in $\mathcal{L}^{\star}$ in the upcoming section.

In the sequel, the term temporal sequence or sequence in short, will denote a sequence of ordered integer numbers. A sequence allows to represent a set of time points. Sometimes, we will consider integer intervals as sequences. Moreover, given two sequences N_1, N_2, the union of N_1 and N_2, denoted by $N_1 \cup N_2$, is the sequence containing only elements of N_1 and N_2. An acceptable sequence is a temporal sequence that is built relatively to a preferential temporal interpretation I as follows:

Definition 9 (Acceptable sequence w.r.t. I) *Let $I = (V, \prec) \in \mathfrak{I}$ and N be a sequence of temporal time points. N is an acceptable sequence w.r.t. I iff for all $i \in N \cap \mathit{final}(I)$ and for all $j \in \mathit{final}(I)$ s.t. $V(i) = V(j)$, we have $j \in N$.*

The intuition behind the notion of acceptable sequence is to construct new interpretations from a given interpretation. The particularity we are looking for is that any picked time point in $\mathit{init}(\cdot)$ ($\mathit{final}(\cdot)$ resp.) will remain in the initial part (final part resp.) of the new interpretation. It is worth to point out that an acceptable sequence w.r.t. a preferential temporal interpretation can either be finite or infinite. Also, we can see that $\mathbb{N}$ is an acceptable sequence w.r.t. any interpretation $I \in \mathfrak{I}$.

Given N an acceptable sequence w.r.t. I, if N has a time point t in $\mathit{final}(I)$, then all time points t' that have the same valuation as t must be in N. Thus, we have an infinite sequence of time points. As such, we can define an initial part and a final part, in a similar way as $LTL^{\sim}$ interpretations. We let $\mathit{init}(I, N)$ to be the largest subsequence of N that is a subsequence of $\mathit{init}(I)$. Note that in the case of having an acceptable sequence N w.r.t. I that does not contain any time point of $\mathit{final}(I)$, N is finite. Formally, we define the notions $\mathit{init}(\cdot)$, $\mathit{final}(\cdot)$, $\mathit{range}(\cdot)$, and $\mathit{size}(\cdot)$ for acceptable sequences as follows:

Definition 10 *Let $I = (V, \prec) \in \mathfrak{I}$, and let N be an acceptable sequence w.r.t. I. We have the following:*

$$\begin{array}{lcl} \mathit{init}(I, N) & \stackrel{\text{def}}{=} & N \cap \mathit{init}(I); \\ \mathit{final}(I, N) & \stackrel{\text{def}}{=} & N \setminus \mathit{init}(I, N); \\ \mathit{range}(I, N) & \stackrel{\text{def}}{=} & \{V(t) \mid t \in \mathit{final}(I, N)\}; \\ \mathit{size}(I, N) & \stackrel{\text{def}}{=} & \mathit{length}(\mathit{init}(I, N)) + \mathit{card}(\mathit{range}(I, N)). \end{array}$$

It is worth to mention, thanks to Definition 9, that given an acceptable sequence w.r.t. I, we have $size(I, N) \leq size(I)$.

We define the notion of pseudo-interpretations next.

Definition 11 (Pseudo-interpretation over N) *Let $I = (V, \prec) \in \Im$ and N be an acceptable sequence w.r.t. I. The pseudo-interpretation over N is the tuple $I^N \stackrel{\text{def}}{=} (N, V^N, \prec^N)$ where:*

- *$V^N : N \longrightarrow 2^{\mathcal{P}}$ is a valuation function over N, where for all $i \in N$, we have $V^N(i) = V(i)$,*
- *$\prec^N \subseteq N \times N$ is a preference relation over N, where for all $(i, j) \in N^2$, we have $(i, j) \in \prec^N$ iff $(i, j) \in \prec$.*

The truth values of $\mathcal{L}^\star$ sentences in pseudo-interpretations are defined in a similar fashion as for preferential temporal interpretations. With $\models_{\mathscr{P}}$ we denote the truth values of sentences in a pseudo-interpretation. We highlight truth values for classical and defeasible modalities.

- $I^N, t \models_{\mathscr{P}} \Box\alpha$ if $I^N, t' \models_{\mathscr{P}} \alpha$ for all $t' \in N$ s.t. $t' \geq t$;
- $I^N, t \models_{\mathscr{P}} \Diamond\alpha$ if $I^N, t' \models_{\mathscr{P}} \alpha$ for some $t' \in N$ s.t. $t' \geq t$;
- $I^N, t \models_{\mathscr{P}} \boxminus\alpha$ if for all $t' \in N$ s.t. $t' \in min_{\prec^N}(t)$, we have $I^N, t \models_{\mathscr{P}} \alpha$;
- $I^N, t \models_{\mathscr{P}} \diamondminus\alpha$ if $I^N, t' \models_{\mathscr{P}} \alpha$ for some $t' \in N$ s.t. $t' \in min_{\prec^N}(t)$.

Here are some interesting properties displayed in the case of acceptable sequences.

Proposition 4 *Let $I = (V, \prec) \in \Im$, N_1, N_2 be two acceptable sequences w.r.t. I. Then $N_1 \cup N_2$ is an* acceptable sequence *w.r.t. I s.t. $size(I, N_1 \cup N_2) \leq size(I, N_1) + size(I, N_2)$.*

Proposition 5 *Let $I = (V, \prec) \in \Im$ and N be an acceptable sequence w.r.t. I. If for all distinct $t, t' \in N$, we have $V(t') = V(t)$* only *when both $t, t' \in final(I, N)$, then $size(I, N) \leq 2^{|\mathcal{P}|}$.*

5 Bounded-model property

The main contribution of this paper is to establish certain computational properties regarding the satisfiability problem in $\mathcal{L}^\star$. Our algorithmic problem is as follows: Given an input sentence $\alpha \in \mathcal{L}^\star$, decide whether α is preferentially satisfiable. In this paper, we show that this problem is decidable.

Our proof is based on the proof of complexity in the case of propositional linear temporal logic by Sistla & Clarke [15]. Let $\mathcal{L}^\star$ be the fragment of $\mathcal{L}^{\sim}$ that contains only Boolean connectives and temporal operators ($\Box$, $\boxminus$, $\Diamond$, $\diamondminus$). With $|\alpha|$ we denote the number of symbols within the sentence α.

The main result of the present paper is summarized in the following theorem:

Theorem 1 (Bounded-model property) *If $\alpha \in \mathcal{L}^\star$ is $\Im^{sd}$-satisfiable, we can find an interpretation $I \in \Im^{sd}$ such that $size(I) \leq |\alpha| \times 2^{|\mathcal{P}|}$ and $I, 0 \models \alpha$.*

The proof will be given in the remainder of the section.

This theorem states that given a satisfiable sentence $\alpha \in \mathcal{L}^\star$, there exists an interpretation satisfying α of which the size is bounded. Briefly speaking, since $\alpha \in \mathcal{L}^\star$ is $\Im^{sd}$-satisfiable, there exists an interpretation where $I, 0 \models \alpha$. We can then, based on I, construct an interpretation I' , that satisfies also the sentence i.e., $I', 0 \models \alpha$ and it is bounded on its size by $|\alpha| \times 2^{|\mathcal{P}|}$.

The goal of this section is to show how to build such bounded interpretation $size(\cdot)$ wise. Let $\alpha \in \mathcal{L}^\star$ and let $I \in \Im^{sd}$ be s.t. $I, 0 \models \alpha$. The first step is to characterize an acceptable sequence N w.r.t. I such that N is bounded first of all, and "keeps" the satisfiability of the sub-sentences α_1 contained in α i.e., if $I, t \models \alpha_1$, then $I^N, t \models_{\mathscr{P}} \alpha_1$ (see Definition 11). We do so by building inductively a bounded pseudo-interpretation step by step by selecting what to take from the initial interpretation I for each sub-sentence α_1 contained in α to be satisfied. In what follows, we introduce the notion of $Anchors(\cdot)$ as a strategy for picking out the desired time points from I. Definitions 13–15 tell us how to pick said time points.

Definition 12 (Acceptable sequence transformation AS) *Let $I = (V, \prec) \in \Im$ and let N be a sequence of time points. Let N' be the sequence of all time points t' in $final(I)$ for which there is $t \in N \cap final(I)$ with $V(t) = V(t')$. With $AS(I, N) \stackrel{\text{def}}{=} N \cup N'$ we denote the* acceptable sequence transformation *of N w.r.t. I.*

The sequence $AS(I, N)$ is the acceptable sequence version of N w.r.t. I. In the previous definition, N' is the sequence of all time points t' having the same valuation as any given time point $t \in N$ that is in $final(I)$. It is also worth to point out that N' can be empty in the case of there being no time point $t \in N$ that is in $final(I)$. In this case, the sequence N is a "finite" acceptable sequence w.r.t. I where $AS(I, N) = N$. This notation is mainly used to ensure that we are using the acceptable version of any sequence.

Definition 13 (Chosen occurrence w.r.t. α) *Let $I = (V, \prec) \in \Im$, $\alpha \in \mathcal{L}^{\sim}$ and N be an acceptable sequence w.r.t. I s.t. there exists a time point t in N with $I, t \models \alpha$. The* chosen occurrence *satisfying α in N, denoted by $\mathfrak{t}_\alpha^{I,N}$, is defined as follows:*

$$\mathfrak{t}_\alpha^{I,N} \stackrel{\text{def}}{=} \begin{cases} \min_<\{t \in final(I, N) \mid I, t \models \alpha\}, \\ \quad if\ \{t \in final(I, N) \mid I, t \models \alpha\} \neq \emptyset \\ \max_<\{t \in init(I, N) \mid I, t \models \alpha\},\ otherwise \end{cases}$$

It is imperative to distinguish between the natural ordering of the temporal structure $<$ and the preference relation

$\prec$. This is the strategy to pick out the time point that satisfies a given sentence α in N. If said sentence is in the final part, we pick the first time point that satisfies it, since we have the guarantee to find infinitely many time points having the same valuations as $t_\alpha^{I,N}$ that also satisfy α (see Lemma 2). If not, we pick the last occurrence in the initial part that satisfies α. Thanks to Definition 13, we can limit the number of time points taken that satisfy the same sentence.

Definition 14 (Selected time points *ST*) *Let $I = (V, \prec) \in \Im$, N be an acceptable sequence w.r.t. I and $\alpha \in \mathcal{L}^{\sim}$ s.t. there exists a time point t in N with $I, t \models \alpha$. With $ST(I, N, \alpha) \stackrel{\text{def}}{=} AS(I, (t_\alpha^{I,N}))$ we denote the* selected time points *of N and α. It is worth pointing out $(t_\alpha^{I,N})$ is a sequence of only one element.*

Given a sentence $\alpha \in \mathcal{L}^{\sim}$ and an acceptable sequence N w.r.t. I s.t. there is at least one time point t where $I, t \models \alpha$, the sequence $ST(I, N, \alpha)$ is the acceptable sequence transformation of the sequence $(t_\alpha^{I,N})$. If $t_\alpha^{I,N} \in init(I)$, the sequence $ST(I, N, \alpha)$ is the sequence $(t_\alpha^{I,N})$. Otherwise, the sequence $ST(I, N, \alpha)$ is the sequence of all time points t in $final(I)$ that have the same valuation as $t_\alpha^{I,N}$. In both cases, we can see that $size(I, ST(I, N, \alpha)) = 1$.

Given an interpretation $I = (V, \prec)$ and N an acceptable sequence w.r.t I, we denote the set of all valuations within N w.r.t. I by $val(I, N) \stackrel{\text{def}}{=} \{V(t) \mid t \in N\}$. The *representative sentence* of a valuation v is formally defined as $\alpha_v \stackrel{\text{def}}{=} \bigwedge\{p \mid p \in v\} \wedge \bigwedge\{\neg p \mid p \notin v\}$.

Definition 15 (Distinctive reduction *DR*) *Let $I = (V, \prec) \in \Im$ and let N be an acceptable sequence w.r.t. I. With $DR(I, N) \stackrel{\text{def}}{=} \bigcup_{v \in val(I,N)} ST(I, N, \alpha_v)$ we denote the distinctive reduction of N.*

Given an acceptable sequence N w.r.t. I, $DR(I, N)$ is the sequence containing the chosen occurrence $t_{\alpha_v}^{I,N}$ that satisfies the representative α_v in N for each $v \in val(I, N)$. In other words, we pick the selected time points for each possible valuation in N. There are two interesting results with regard to $DR(I, N)$. The first one is that $DR(I, N)$ is an acceptable sequence w.r.t. I. This can easily be proven since $ST(I, N, \alpha_v)$ is also an acceptable sequence w.r.t. I, and the union of all $ST(I, N, \alpha_v)$ is an acceptable sequence w.r.t. I (see proposition 4). The second result is that $size(I, DR(I, N)) \leq 2^{|\mathcal{P}|}$. Indeed, thanks to Proposition 4, we can see that $size(I, DR(I, N)) \leq \sum_{v \in val(I,N)} ST(I, N, \alpha_v)$. Moreover, we have $size(I, ST(I, N, \alpha_v)) = 1$ for each $v \in val(I, N)$. On the other hand, since we assume a finite set of atomic propositions $\mathcal{P}$, there are at most $2^{|\mathcal{P}|}$ possible valuations. Thus, we can assert that $\sum_{v \in val(I,N)} size(I, ST(I, N, \alpha_v)) \leq 2^{|\mathcal{P}|}$. Therefore, we conclude that $size(I, DR(I, N)) \leq 2^{|\mathcal{P}|}$.

We introduce now the notion of *Anchors*$(\cdot)$.

Definition 16 (Anchors) *Let a sentence $\alpha \in \mathcal{L}^\star$ starting with a temporal operator , let $I = (V, \prec) \in \Im^{sd}$, and let T be a non-empty acceptable sequence w.r.t. I s.t. for all $t_i \in T$ we have $I, t_i \models \alpha$. The sequence $Anchors(I, T, \alpha)$ is defined as:*

$$
\begin{array}{lll}
Anchors(I, T, \Diamond\alpha_1) & \stackrel{\text{def}}{=} & ST(I, \mathbb{N}, \alpha_1); \\
Anchors(I, T, \Box\alpha_1) & \stackrel{\text{def}}{=} & \emptyset; \\
Anchors(I, T, \Diamond\!\!\!\!-\alpha_1) & \stackrel{\text{def}}{=} & \bigcup_{t_i \in T} ST(I, AS(I, min_\prec(t_i)), \alpha_1); \\
Anchors(I, T, \boxminus\alpha_1) & \stackrel{\text{def}}{=} & DR(I, \bigcup_{t_i \in T} AS(I, min_\prec(t_i))).
\end{array}
$$

Given an acceptable sequence T w.r.t. $I \in \Im^{sd}$ where all of its time points satisfy α (α is a sentence starting with a classical or defeasible modality), the function $Anchors(I, T, \alpha)$ is an acceptable sequence w.r.t. I that contains the selected time points satisfying the sub-sentence α_1 in α (except for $\Box\alpha_1$). Our goal is to have the selected time points that satisfy α_1 for each $t_i \in T$.

It is worth to point out that the choice of $Anchors(I, T, \Box\alpha_1) = \emptyset$ is due to the fact α_1 is implicitly satisfied starting from the first time $t_0 \in T$. So no matter what time point t we pick after t_0, we have $I, t \models \alpha_1$. On the other hand, by the nature of the semantics of $\boxminus\alpha_1$, all $t \in \bigcup_{t_i \in T} AS(I, min_\prec(t_i))$ satisfy α_1. The acceptable sequence $Anchors(I, T, \boxminus\alpha_1)$ contains only the selected time points for each distinct valuation in $\bigcup_{t_i \in T} AS(I, min_\prec(t_i))$.

The following are some properties of *Anchors*$(\cdot)$ that are worth mentioning:

Lemma 4 *Let $\alpha_1 \in \mathcal{L}^\star$ be a sentence starting with a temporal operator, $I = (V, \prec) \in \Im^{sd}$ and let T be a non-empty acceptable sequence w.r.t. I where for all $t \in T$ we have $I, t \models \Diamond\!\!\!\!-\alpha_1$. Then for all $t, t' \in Anchors(I, T, \Diamond\!\!\!\!-\alpha_1)$ s.t. $V(t) = V(t')$ and $t \neq t'$, we have $t, t' \in final(I, Anchors(I, T, \Diamond\!\!\!\!-\alpha_1))$.*

Proposition 6 *Let $\alpha \in \mathcal{L}^\star$ be a sentence starting with a temporal operator, $I = (V, \prec) \in \Im^{sd}$. Let T be a non-empty acceptable sequence w.r.t. I where for all $t \in T$ we have $I, t \models \alpha$. Then, we have $size(I, Anchors(I, T, \alpha)) \leq 2^{|\mathcal{P}|}$.*

Proposition 7 *Let $\alpha_1 \in \mathcal{L}^\star$, $I = (V, \prec) \in \Im^{sd}$, let T be a non-empty acceptable sequence w.r.t. I s.t. for all $t \in T$ we have $I, t \models \boxminus\alpha_1$. For all acceptable sequences N w.r.t. I s.t. $Anchors(I, T, \boxminus\alpha_1) \subseteq N$ and for all $t_i \in N \cap T$, we have the following: Let $I^N = (V^N, \prec_N)$ be the pseudo-interpretation over N, for all $t' \in N$, if $t' \notin min_\prec(t_i)$, then $t' \notin min_{\prec^N}(t_i)$.*

Proposition 7 helps us mitigate the dynamic nature of $min_\prec(t_i)$. The selected time points help us circumvent adding time points that were not originally "preferred" w.r.t. t_i in I, and becoming preferred in the reduced structure I^N that we want to build. The strategy of building

Anchors$(\cdot)$ is mainly due to the fact that we want to preserve the truth values of defeasible sub-sentences of α in the bounded interpretation.

With *Anchors*$(\cdot)$ defined, we introduce the notion of *Keep*$(\cdot)$ that will help us compute recursively starting from the initial satisfiable sentence α down to its literals, the selected time points to pick in order to build our pseudo-interpretation.

Definition 17 (Keep) *Let $\alpha \in \mathcal{L}^\star$ be in NNF, $I = (V, \prec) \in \mathfrak{I}^{sd}$, and let T be a non-empty acceptable sequence w.r.t. I s.t. for all $t \in T$ we have $I, t \models \alpha$. The sequence $Keep(I, T, \alpha)$ is defined recursively as follows:*

- $Keep(I, T, \ell) = \emptyset$, *where ℓ is a literal;*
- $Keep(I, T, \alpha_1 \wedge \alpha_2) = Keep(I, T, \alpha_1) \cup Keep(I, T, \alpha_2)$;
- $Keep(I, T, \alpha_1 \vee \alpha_2) = Keep(I, T_1, \alpha_1) \cup Keep(I, T_2, \alpha_2)$; *where $T_1 \subseteq T$ (resp. $T_2 \subseteq T$) is the sequence of all $t_1 \in T$ (resp. $t_2 \in T$) s.t. $I, t_1 \models \alpha_1$ (resp. $I, t_2 \models \alpha_2$)*
- $Keep(I, T, \Diamond\alpha_1) = Anchors(I, T, \Diamond\alpha_1) \cup Keep(I, Anchors(I, T, \Diamond\alpha_1), \alpha_1)$;
- $Keep(I, T, \Box\alpha_1) = Keep(I, T, \alpha_1)$;
- $Keep(I, T, \Diamond\!\!\!\!-\,\alpha_1) = Anchors(I, T, \Diamond\!\!\!\!-\,\alpha_1) \cup Keep(I, Anchors(I, T, \Diamond\!\!\!\!-\,\alpha_1), \alpha_1)$;
- $Keep(I, T, \boxminus\alpha_1) = Anchors(I, T, \boxminus\alpha_1) \cup Keep(I, T', \alpha_1)$, *where* $T' = \bigcup_{t_i \in T} AS(I, min_{\prec}(t_i))$.

Given an acceptable sequence N w.r.t. I, we need to make sure if we have $t \in N$ in our acceptable sequence s.t. $I, t \models \alpha$, then $I^N, t \models_{\mathscr{P}} \alpha$. The function $Keep(I, T, \alpha)$ returns the acceptable sequence of time s.t. if $Keep(I, T, \alpha) \subseteq N$ and $t \in T$, then said condition is met. We prove this in Lemma 5.

With $\mu(\alpha)$ we denote the number of classical and non-monotonic modalities contained in the sentence α.

Proposition 8 *Let $\alpha \in \mathcal{L}^\star$ be in NNF, $I = (V, \prec) \in \mathfrak{I}^{sd}$, and let T be a non-empty acceptable sequence w.r.t. I s.t. for all $t \in T$ we have $I, t \models \alpha$. Then, we have $size(I, Keep(I, T, \alpha)) \leq \mu(\alpha) \times 2^{|\mathcal{P}|}$.*

Lemma 5 *Let $\alpha \in \mathcal{L}^\star$ be in NNF, $I = (V, \prec) \in \mathfrak{I}^{sd}$, and let T be a non-empty acceptable sequence w.r.t. I s.t. for all $t \in T$ we have $I, t \models \alpha$. For all acceptable sequences N w.r.t I, if $Keep(I, T, \alpha) \subseteq N$, then for every $t \in N \cap T$, we have $I^N, t \models_{\mathscr{P}} \alpha$.*

Since we build our pseudo-interpretation I^N by adding selected time points for each sub-sentence α_1 contained within the initial sentence α, we need to make sure that said sub-sentence remains satisfied in I^N. Lemma 5 ensures that.

Definition 18 (Pseudo-interpretation transformation) *Let $I = (V, \prec) \in \mathfrak{I}^{sd}$ and let N be an infinite acceptable sequence w.r.t. I. The pseudo-interpretation $I^N = (V^N, \prec^N)$ can be transformed to $I' = (V', \prec') \in \mathfrak{I}^{sd}$ as follows:*

- *For all $i \geq 0$, we have $V'(i) = V^N(t_i)$.*
- *And for all $(t_i, t_j) \in N^2$, we have $(t_i, t_j) \in \prec^N$ iff $(i, j) \in \prec'$.*

We can now prove our bounded-model theorem.

Proof: [Proof of Theorem 1] We assume that $\alpha \in \mathcal{L}^\star$ is $\mathfrak{I}^{sd}$-satisfiable. The first thing we notice is that $|\alpha| \geq \mu(\alpha) + 1$. Let α' be the NNF of the sentence α. As a consequence of the duality rules of $\mathcal{L}^\star$, we can deduce that $\mu(\alpha') = \mu(\alpha)$. Let $I = (V, \prec) \in \mathfrak{I}^{sd}$ s.t. $I, 0 \models \alpha'$. Let T_0 be the acceptable sequence w.r.t. I where $T_0 = AS(I, (0))$. We can see that $size(I, T_0) = 1$. Since for all $t \in T_0$ we have $I, t \models \alpha'$ (see Lemma 2) we can compute recursively $U = Keep(I, T_0, \alpha')$. Thanks to Proposition 8, we conclude that U is an acceptable sequence w.r.t. I s.t. $size(I, U) \leq \mu(\alpha') \times 2^{|\mathcal{P}|}$. Let $N = T_0 \cup U$ be the union of T_0 and U and let $I^N = (N, V^N, \prec^N)$ be its pseudo-interpretation over N. Thanks to Proposition 4, we have $size(I, N) \leq 1 + \mu(\alpha') \times 2^{|\mathcal{P}|}$. Not only that, but thanks to Lemma 5, since $0 \in N \cap T_0$ and $Keep(I, T_0, \alpha') \subseteq N$, we have $I^N, 0 \models_{\mathscr{P}} \alpha'$. In case that N is finite, we replicate the last time point t_n infinitely many times. Notice that the size does not change if we replicate the last element. We can transform the pseudo interpretation I^N to $I' \in \mathfrak{I}^{sd}$ by changing the labels of N into a sequence of natural numbers minding the order of time points in N (see Definition 18). We can see that truth values are kept in I'. Moreover, we have $size(I') = size(I, N)$ and $I', 0 \models \alpha$. Consequently, we have $size(I') \leq 1 + \mu(\alpha') \times 2^{|\mathcal{P}|}$.

We conclude that from a given interpretation I s.t. $I, 0 \models \alpha$ we can build an interpretation I' s.t. $I', 0 \models \alpha$ and $size(I') \leq 1 + \mu(\alpha') \times 2^{|\mathcal{P}|}$. Without loss of generality, we conclude that $size(I') \leq |\alpha| \times 2^{|\mathcal{P}|}$. □

6 The satisfiability problem in $\mathcal{L}^\star$

The main goal in this section is to provide an algorithm allowing to decide whether a sentence $\alpha \in \mathcal{L}^\star$ is $\mathfrak{I}^{sd}$-satisfiable or not. For this purpose, first we focus on particular interpretations of the class $\mathfrak{I}^{sd}$, namely the ultimately periodic interpretations (UPI in short), and a finite representation of these interpretations, called ultimately periodic pseudo-interpretation (UPPI in short). As we will see in the second part of this section, to decide the $\mathfrak{I}^{sd}$-satisfiability of a sentence $\alpha \in \mathcal{L}^\star$, the proposed algorithm guess a bounded UPPI in a first step. Then, it checks the satisfiability of α by the UPI behind the guessed UPPI.

First, we define particular state dependent interpretations.

Definition 19 (Ultimately periodic interpretation) *Let $I \in \Im$ and $P = card(range(I))$. The interpretation I is an* ultimately periodic interpretation *iff $I \in \Im^{sd}$ where:*

- *For each distinct $t, t' \in [t_I, t_I + P[$ (see Definition 1), we have $V(t) \neq V(t')$;*
- *and for all $t \in [t_I, +\infty[$ we have $V(t) = V(t_I + (t - t_I) \bmod P)$.*

An ultimately periodic interpretation I, or UPI for short, is a state dependent interpretation s.t. each time point's valuation in $final(I)$ is replicated periodically. Given a UPI, $P = card(range(I))$ denotes the length of the period and the interval $[t_I, t_I + P[$ is the first period which is replicated periodically throughout the final part. It is worth pointing out that for every $t \in final(I)$, we have $V(t) \in \{V(t') \mid t' \in [t_I, t_I + P[\}$, which is one of the consequences of the definition above. Thanks to Lemma 3, we can prove the following proposition.

Proposition 9 *Let $\mathcal{P}$ be a set of atomic propositions, $I \in \Im^{sd}$, $i = length(init(I))$ and $P = card(range(I))$. There exists an ultimately periodic interpretation $I' = (V', \prec') \in \Im^{sd}$ s.t. I, I' are faithful interpretations over $\mathcal{P}$, $init(I') \equiv init(I)$, $range(I') = range(I)$ and $V'(0) = V(0)$. Moreover, for all $\alpha \in \mathcal{L}^\star$, we have $I, 0 \models \alpha$ iff $I', 0 \models \alpha$.*

It is worth to point out that the size of an interpretation and its UPI counterparts are equal. It can easily be seen that these interpretations have the same initial part and the same range of valuations in the final part.

With UPI defined, we introduce the notion of Ultimately Periodic Pseudo-Interpretation or UPPI for short. UPPI is a finite representation of UPI that helps us verify the satisfiability of sentences in $\mathcal{L}^\star$. Formally, it is defined as such:

Definition 20 (Ultimately Periodic Pseudo-Interpretation) *An ultimately periodic pseudo-interpretation is a tuple $M = (i, P, V_M, \prec_M)$ where:*

- *i, P are two integers such that $i \geq 0$ and $P > 0$ (where i is intended to be the starting point of the period, P is the length of the period);*
- *$V_M : [0, i + P[\longrightarrow 2^{\mathcal{P}}$ is a mapping function that associates each $t \in [0, i + P[$ with a set of atoms;*
- *the preference relation $\prec_M \subseteq 2^{\mathcal{P}} \times 2^{\mathcal{P}}$ is a strict partial order.*

The interval $[0, i[$ corresponds to the initial temporal part of the underlying interpretation and $[i, i + P[$ represents a temporal period that is infinitely replicated in order to determine the final temporal part of the interpretation.

For $M = (i, P, V_M, \prec_M)$ a UPPI, with $size(M) = i + P$ we denote the *size of* M. From an ultimately periodic pseudo-interpretation we define an ultimately periodic interpretation in the following way :

Definition 21 *Given a UPPI $M = (i, P, V_M, \prec_M)$, we define the interpretation $\mathfrak{l}(M) = (V, \prec)$ by:*

- *For every $t \geq 0$,*

$$V(t) \stackrel{\text{def}}{=} \begin{cases} V_M(t), \text{ if } t < i; \\ V_M(i + (t - i) \bmod P), \text{ otherwise.} \end{cases}$$

- $\prec \stackrel{\text{def}}{=} \{(t, t') \mid (V(t), V(t')) \in \prec_M\}$.

It is worth to point out that given a UPPI M, $\mathfrak{l}(M) = (V, \prec)$ is a state dependent interpretation i.e., $\mathfrak{l}(M) \in \Im^{sd}$. Moreover, we have $size(\mathfrak{l}(M)) = size(M)$.

From an ultimately periodic interpretation we define an ultimately periodic pseudo-interpretation in the following way:

Definition 22 *Given a UPI $I = (V, \prec)$, we define the UPPI $M(I) = (i, P, V_M, \prec_M)$ by:*

- *$i = length(init(I))$, $P = card(range(I))$;*
- *$V_M(t) = V(t)$ for all $t \in [0, i + P[$;*
- *for all $t, t' \in [0, i + P[$, $(V(t), V(t')) \in \prec_M$ iff $(t, t') \in \prec$*

Same as the previous transformation (Definition 21), given a UPI I, we have $size(M(I)) = size(I)$.

Now we extend the notion of preferred time points w.r.t a time point for a UPPI :

Definition 23 (Set of preferred time points) *Let $M = (i, P, V_M, \prec_M)$ be a UPPI and a time point $t \in [0, i + P[$. The set of preferred time points of t w.r.t. M, denoted by $min_{\prec_M}(t)$, is defined as follows: $min_{\prec_M}(t) = \{t' \in [min_<\{t, i\}, i + P[\mid$ there is no $t'' \in [min_<\{t, i\}, i + P[$ with $(V_M(t''), V_M(t')) \in \prec_M\}$.*

We have the following property.

Proposition 10 *Let $M = (i, \pi, V_M, \prec_M)$ be a UPPI, $\mathfrak{l}(M) = (V, \prec)$ and $t, t', t_M, t'_M \in \mathbb{N}$ s.t.:*

$$t_M = \begin{cases} t, \text{ if } t < i; \\ i + (t - i) \bmod \pi, \text{ otherwise.} \end{cases}$$

$$t'_M = \begin{cases} t', \text{ if } t' < i; \\ i + (t' - i) \bmod \pi, \text{ otherwise.} \end{cases}$$

We have the following: $t' \in min_\prec(t)$ iff $t'_M \in min_{\prec_M}(t_M)$.

Now that UPPI is defined, we can move on how to check the satisfiability of a well-formed sentence α. We define for a UPPI $M = (i, P, V_M, \prec_M)$ and a sentence $\alpha \in \mathcal{L}^\star$, a labelling function $lab^M_\alpha(\cdot)$ which associates a set of subsentences of α for all $t \in [0, i + P[$.

Definition 24 (Labelling function) *Let* $M = (i, P, V_M, \prec_M)$ *be a UPPI,* $\alpha \in \mathcal{L}^\star$. *The set of subsentences of* α *for all* $t \in [0, i + P[$, *denoted by* $lab^M_\alpha(t)$, *is defined as follows:*

- $p \in lab^M_\alpha(t)$ *iff* $p \in V_M(t)$;
- $\neg\alpha_1 \in lab^M_\alpha(t)$ *iff* $\alpha_1 \notin lab^M_\alpha(t)$;
- $\alpha_1 \wedge \alpha_2 \in lab^M_\alpha(t)$ *iff* $\alpha_1, \alpha_2 \in lab^M_\alpha(t)$;
- $\alpha_1 \vee \alpha_2 \in lab^M_\alpha(t)$ *iff* $\alpha_1 \in lab^M_\alpha(t)$ *or* $\alpha_2 \in lab^M_\alpha(t)$;
- $\Diamond\alpha_1 \in lab^M_\alpha(t)$ *iff* $\alpha_1 \in lab^M_\alpha(t')$ *for some* $t' \in [min_<\{t, i\}, i + P[$;
- $\Box\alpha_1 \in lab^M_\alpha(t)$ *iff* $\alpha_1 \in lab^M_\alpha(t')$ *for all* $t' \in [min_<\{t, i\}, i + P[$;
- $\Diamond\!\!\!\!-\alpha_1 \in lab^M_\alpha(t)$ *iff* $\alpha_1 \in lab^M_\alpha(t')$ *for some* $t' \in min_{\prec_M}(t)$;
- $\Box\!\!\!\!-\alpha_1 \in lab^M_\alpha(t)$ *iff* $\alpha_1 \in lab^M_\alpha(t')$ *for all* $t' \in min_{\prec_M}(t)$.

Lemma 6 *Let a UPPI* $M = (i, P, V_M, \prec_M)$, $\alpha \in \mathcal{L}^\star$ *and* $t \in \mathbb{N}$, $\mathsf{I}(M), 0 \models \alpha$ *iff* $\alpha \in lab^M_\alpha(0)$.

We accept a UPPI M as a model for $\alpha \in \mathcal{L}^\star$ iff $\alpha \in lab^M_\alpha(0)$. Otherwise, M is rejected.

Proposition 11 *Let* $\alpha \in \mathcal{L}^\star$. *We have* α *is* $\Im^{sd}$*-satisfiable iff there exists a UPPI* M *such that* $\mathsf{I}(M), 0 \models \alpha$ *and* $size(\mathsf{I}(M)) \leq |\alpha| \times 2^{|\mathcal{P}|}$.

Theorem 2 $\Im^{sd}$*-satisfiability problem for* $\mathcal{L}^\star$ *sentences is* decidable.

7 Concluding remarks

The contributions of this paper are as follows: we introduced $LTL^\sim$ formalism with its more expressive syntax and semantics. We defined also the class of state-dependent interpretations $\Im^{sd}$ and the fragment $\mathcal{L}^\star$. We then showed that $\Im^{sd}$-satisfiability in $\mathcal{L}^\star$ is a decidable problem.

It is worth pointing out that it is hard to define a tableaux method for our logic similar to Wolper's [16]. The main reason is that we do not have defeasible versions of the axioms of reflexivity (T) and transitivity (4), and therefore nested defeasible modalities cannot be reduced as in the classical case. Furthermore, at present we have $\not\models \Box\!\!\!\!-\alpha \leftrightarrow \alpha \wedge \bigcirc \Box\!\!\!\!-\alpha$ and $\not\models \Diamond\!\!\!\!-\alpha \leftrightarrow \alpha \vee \bigcirc \Diamond\!\!\!\!-\alpha$. That is why we decided to tackle the satisfiability problem of our logic before establishing a semantic tableaux for $LTL^\sim$.

Going forward, we will explore a far richer $LTL^\sim$ formalism with the introduction of defeasible counterparts to $\bigcirc$ and $\mathcal{U}$. We will investigate also the implementation of defeasible consequence $\mid\sim$ à la KLM in our logic.

We would like to thank the anonymous reviewers for reviewing this paper. Their helpful comment improved the quality of this work, and gave us some insight for future endeavors.

References

[1] Ben-Ari, M.: *Mathematical Logic for Computer Science, third edition*. Springer, 2012, ISBN 9781447141297.

[2] Britz, K., T. Meyer, and I. Varzinczak: *Preferential reasoning for modal logics*. Electronic Notes in Theoretical Computer Science, 278:55 – 69, 2011, ISSN 1571-0661. Proc. of the 7th Workshop on Methods for Modalities and the 4th Workshop on Logical Aspects of Multi-Agent Systems.

[3] Britz, K., T. Meyer, and I. Varzinczak: *Semantic foundation for preferential description logics*. In Wang, Dianhui and Mark Reynolds (editors): *AI 2011: Advances in Artificial Intelligence*, pages 491–500, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg, ISBN 978-3-642-25832-9.

[4] Britz, K. and I. Varzinczak: *From KLM-style conditionals to defeasible modalities, and back*. Journal of Applied Non-Classical Logics, 28(1):92–121, 2018.

[5] Gabbay, D. M.: *The declarative past and imperative future: Executable temporal logic for interactive systems*.

[6] Gabbay, D. M.: *Theoretical foundations for non-monotonic reasoning in expert systems*. In Apt, Krzysztof R. (editor): *Logics and Models of Concurrent Systems*, pages 439–457, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg, ISBN 978-3-642-82453-1.

[7] Goré, R.: *Tableau methods for modal and temporal logics*. In D'Agostino, M., D.M. Gabbay, R. Hähnle, and J. Posegga (editors): *Handbook of Tableau Methods*. Kluwer Academic Publishers, 1999.

[8] Kraus, S., D. Lehmann, and M. Magidor: *Nonmonotonic reasoning, preferential models and cumulative logics*. Artificial Intelligence, 44:167–207, 1990.

[9] Laverny, N. and J. Lang: *From knowledge-based programs to graded belief-based programs, part i: On-line reasoning**. Synthese, 147(2):277–321, Nov 2005, ISSN 1573-0964.

[10] Makinson, D.: *How to Go Nonmonotonic*, pages 175–278. Springer Netherlands, Dordrecht, 2005, ISBN 978-1-4020-3092-5.

[11] O., Arieli and A. Avron: *General patterns for nonmonotonic reasoning: From basic entailments to plausible relations*. Logic Journal of the IGPL, 8:119–148, 2000.

[12] Pnueli, A.: *The temporal logic of programs*. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57, Oct 1977.

[13] Shoham, Y.: *A semantical approach to nonmonotic logics*. pages 275–279, January 1987.

[14] Shoham, Y.: *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, 1988.

[15] Sistla, A. P. and E. M. Clarke: *The complexity of propositional linear temporal logics*. J. ACM, 32(3):733–749, July 1985, ISSN 0004-5411.

[16] Wolper, P.: *Temporal logic can be more expressive*. Information and Control, 56(1):72 – 99, 1983, ISSN 0019-9958.

Actes JIAF 2020

A Distributed and Clustering-based Algorithm for the Enumeration Problem in Abstract Argumentation*

Sylvie Doutre Mickaël Lafages Marie-Christine Lagasquie-Schiex

IRIT, Université de Toulouse, France

doutre@irit.fr mickael.lafages@irit.fr lagasq@irit.fr

Résumé

Le calcul de l'acceptabilité dans les systèmes d'argumentation reçoit une attention croissante. Dans les systèmes de grande envergure, avec une structure en clusters, ce calcul se montre particulièrement difficile. Cet article présente un algorithme distribué, *AFDivider*, qui énumère les ensembles acceptables sous plusieurs sémantiques, en commençant par découper le système d'argumentation en clusters grâce à une méthode de partitionnement spectral, avant de calculer simultanément dans chaque partition des parties des ensembles acceptables. Cet algorithme est prouvé correct et complet pour les sémantiques stable, complète et préférée, et des résultats empiriques sont présentés.

Abstract

Computing acceptability semantics of abstract argumentation frameworks is receiving increasing attention. Large-scale instances, with a clustered structure, have shown particularly difficult to compute. This paper presents a distributed algorithm, *AFDivider*, that enumerates the acceptable sets under several labelling-based semantics. This algorithm starts with cutting the argumentation framework into clusters thanks to a spectral clustering method, before computing simultaneously in each cluster parts of the labellings. This algorithm is proven to be sound and complete for the stable, complete and preferred semantics, and empirical results are presented.

1 Introduction

Argumentation is a reasoning model which has been of application in multi-agent systems for years (see [16] for an overview). The development of argumentation techniques and of their computation drives such applications.

Among the various argumentation models, the one that is considered in this paper has been defined by Dung [24] : an abstract argumentation framework (AF) considers arguments as abstract entities, and focuses on their attack relationships, hence representing arguments and their underlying conflicts by a directed graph. Which arguments can be accepted is defined by [24] as a collective notion, by a semantics : a set of arguments is collectively acceptable under the semantics. Four semantics (*grounded*, *stable*, *complete* and *preferred*) were defined by Dung, and a variety of other semantics have followed (see [7] for an overview). Several enrichments of the argumentation framework have also been proposed (*e.g.* [8, 17]).

The enumeration of all the acceptable sets of an AF under a given semantics is a problem that has received a lot of attention (see [21] for an overview). This problem has been shown to be computationally intractable for some of the above-mentioned semantics [26]. A competition, ICCMA, that compares argumentation solvers on their ability to solve this problem (and other decision problems) was created a few years ago. [1] The last editions of this competition have been analyzed : [12, 36] highlight that some AF instances have been particularly hard to solve, and that others were not solved at all, considering the *preferred* semantics notably. Many of these instances are of Barabási–Albert (BA) type [1], which is a structure found in several large-scale natural and human-made systems, such as the World Wide Web and some social networks [4]. More generally, these hard graphs are non-dense, but contain parts which are dense : [2] such graphs have a *clustered structure*.

Recent algorithms, proposed for an efficient enumeration of the acceptable sets, are based on a cutting of the AF [18, 25, 28], along with, for some of them, the use of distributed, parallel computation in each part, to construct

*This work was accepted and published in the proceedings of the International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2019) [23].

1. International Competition on Computational Models of Argumentation (ICCMA) http://argumentationcompetition.org/.

2. The density in an argumentation graph is the ratio "number of existing attacks" over "number of potential attacks" (this last number is equal to n^2 with n being the number of arguments).

the acceptable sets [19]. In this research line, our paper presents a new "cutting and distributed computing" algorithm, called *AFDivider*, for the enumeration of the acceptable sets of an AF, under the *stable*, *preferred* and *complete* labelling semantics. The cutting of the AF is done in a new way, using spectral clustering methods. Compared to the existing approaches, the added value of *AFDivider* is its way to split the AF and thus to distribute the solving hardness of the whole AF into smaller parts, the reunifying process requiring less checks than the construction of the labellings over the whole AF. *AFDivider* is shown to be sound and complete. The algorithm has been empirically tested, and the results have been compared to those of two solvers of the ICCMA 2017 edition, *Pyglaf* [3] and *ArgSemSAT* [20].

The paper starts with presenting the background of this work (Section 2), before describing the algorithm (Section 3). Soundness and completeness of the algorithm are proven in Section 4. A preliminary empirical analysis is conducted (Section 5). Related works are presented in Section 6. Perspectives for future work are then opened.

2 Background

2.1 Abstract Argumentation

According to [24], an abstract argumentation framework consists of a set of arguments and of a binary attack relation between them.

Definition 1 (AF) *An* argumentation framework *(AF) is a pair* $\Gamma = \langle A, R \rangle$ *where A is a finite*[3] *set of abstract arguments and* $R \subseteq A \times A$ *is a binary relation on A, called the attack relation :* $(a,b) \in R$ *means that a attacks b.*

Hence, an argumentation framework can be represented by a directed graph with arguments as vertices and attacks as edges. Figure 1 shows an example of an AF.

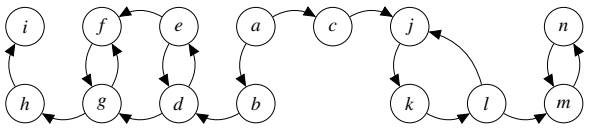


FIGURE 1 – Example of an argumentation framework *AF*.

Acceptability semantics can be defined in terms of labellings [7, 15].

Definition 2 (Labelling) *Let* $\Gamma = \langle A, R \rangle$ *be an AF, and* $S \subseteq A$. *A* labelling *of S is a total function* $\ell : S \to \{\texttt{in}, \texttt{out}, \texttt{und}\}$. *The set of all labellings of S is denoted as* $\mathscr{L}(S)$. *A labelling of* Γ *is a labelling of A. The set of all labellings of* Γ *is denoted as* $\mathscr{L}(\Gamma)$.

3. According to [24], the set of arguments is not necessarily finite. Nevertheless, in this paper, it is reasonable to assume that it is finite.

We write $\texttt{in}(\ell)$ *for* $\{a | \ell(a) = \texttt{in}\}$, $\texttt{out}(\ell)$ *for* $\{a | \ell(a) = \texttt{out}\}$ *and* $\texttt{und}(\ell)$ *for* $\{a | \ell(a) = \texttt{und}\}$.

Definition 3 (Legally labelled arguments, valid labelling) *An* `in`*-labelled argument is said to be* legally `in` *iff all its attackers are labelled* `out`. *An* `out`*-labelled argument is said to be* legally `out` *iff at least one of its attackers is labelled* `in`. *An* `und`*-labelled argument is said to be* legally `und` *iff it does not have any attacker that is labelled* `in` *and one of its attackers is not labelled* `out`.
A valid labelling *is a labelling in which all arguments are legally labelled.*

Let $\Gamma = \langle A, R \rangle$ be an AF, and $\ell \in \mathscr{L}(\Gamma)$ be a labelling. Different kinds of labelling can be defined :

Definition 4 Admissible, complete, grounded, preferred, stable ℓ *is an* admissible labelling *of* Γ *iff for any argument* $a \in A$ *such that* $\ell(a) = \texttt{in}$ *or* $\ell(a) = \texttt{out}$, *a is legally labelled.* ℓ *is a* complete labelling *of* Γ *iff for any argument* $a \in A$, *a is legally labelled.* ℓ *is the* grounded labelling *of* Γ *iff it is the complete labelling of* Γ *that minimizes (w.r.t* $\subseteq$*) the set of* `in`*-labelled arguments.* ℓ *is a* preferred labelling *of* Γ *iff it is a complete labelling of* Γ *that maximizes (w.r.t* $\subseteq$*) the set of* `in`*-labelled arguments.* ℓ *is a* stable labelling *of* Γ *iff it is a complete labelling of* Γ *which has no* `und`*-labelled argument.*

Note that each complete labelling includes the grounded labelling. This property will be used by the algorithm presented in Section 3 in order to compute the AF labellings in a distributed way. Let $\Gamma = \langle A, R \rangle$ be an AF, and $\mathscr{L}(\Gamma)$ be its set of labellings, semantics can be defined.

Definition 5 (Semantics) *A* semantics σ *is a total function* σ *that associates to* Γ *a subset of* $\mathscr{L}(\Gamma)$. *The set of labellings under semantics* σ, *with* σ *being either the complete (co), the grounded (gr), the stable (st) or the preferred (pr) semantics, is denoted by* $\mathscr{L}_\sigma(\Gamma)$. *A labelling* ℓ *is a* σ*-labelling iff* $\ell \in \mathscr{L}_\sigma(\Gamma)$.

Example 1 *Let us consider the AF given in Figure 1. Table 1 shows the labellings corresponding to the different semantics (the other possible labellings are not given). Note that this AF has no stable labelling.*

2.2 Clustering Methods

A *cluster* in a graph can be defined as a connected subgraph. Finding clusters is a subject that has been widely studied (see [33, 37]). The clustering approach implemented in our algorithm is based on a spectral analysis of a defined similarity matrix of the graph. We chose this clustering method as it is well suited for a non-dense graph (see Sections 3.1 and 3.2 for more explanation). We give here a succinct description of this approach (for details, see [38]) :

- Computation of a similarity matrix of the graph. In this squared matrix, the values represent how much two nodes are similar according to a given similarity criterion [4], and the rows may be seen as the coordinates of the graph nodes in a similarity space.
- Computation of the Laplacian matrix of this similarity matrix. The rows of this Laplacian matrix represent how much a node is similar to the others and how much each of its neighbours contributes to its global similarity with its neighbourhood.
- Computation of the eigenvectors (see [35]) of the Laplacian matrix with their associated eigenvalues.
- These eigenvalues are sorted by increasing order. A number n of them is kept with their associated eigenvectors. [5]
- A matrix whose columns are the remaining eigenvectors is built. Its rows represent the new node coordinates in a space that maximizes the proximity between similar nodes. In that space, the euclidean distance between two nodes shows how much a node is similar to another.
- Then a simple algorithm of clustering such as *KMeans* is applied to that new data set, seeking for a partition into n parts, based on the coordinates of the nodes (see [31] for more information about *KMeans* algorithm).

An illustration of this method on the running example is given in Section 3.2 while the similarity criterion used is explicited in Section 3.1.

3 The Algorithm

This section presents the *AFDivider* algorithm designed for the *complete*, *stable* and *preferred* semantics (denoted by σ). It computes the semantics labellings of an AF by first removing trivial parts of the AF (the *grounded* labelling, as done in [18]), then cutting the AF into clusters and computing simultaneously in each cluster labelling parts, before finally reunifying compatible parts to get the σ-labellings of the whole AF. Each of these steps will be presented and then illustrated on the running example.

3.1 Description

Given an argumentation framework $\Gamma = \langle A, R \rangle$, the *AFDivider* algorithm (Alg. 1) starts with computing the grounded labelling of Γ (line 1). Indeed in each of the semantics σ we are interested in, the arguments labelled `in` or `out` in the grounded labelling are labelled in the same way in all the σ-labellings. It is a fixed part. Note that the function *ComputeGroundedLabelling*(Γ) returns a partial labelling of Γ in which the arguments are labelled `in` or `out`. The `und`-labelled arguments according to the grounded semantics do not belong to ℓ_{gr}.

Algorithm 1: *AFDivider* algorithm.

Data: Let $\Gamma = \langle A, R \rangle$ be an AF and σ be a semantics
Result: $\mathscr{L}_\sigma \in 2^{\mathscr{L}(\Gamma)}$: the set of the σ-labellings of Γ

1 $\ell_{gr} \leftarrow ComputeGroundedLabelling(\Gamma)$
2 $CCSet \leftarrow SplitConnectedComponents(\Gamma, \ell_{gr})$
3 **for all** $\gamma_i \in CCSet$ **do in parallel**
4 $\quad ClustSet \leftarrow ComputeClusters(\gamma_i)$
5 $\quad \mathscr{L}_\sigma^{\gamma_i} \leftarrow ComputeCompLabs(\sigma, ClustSet)$
6 $\mathscr{L}_\sigma \leftarrow \varnothing$
7 **if** $\nexists \gamma_i \in CCSet$ *s.t.* $\mathscr{L}_\sigma^{\gamma_i} = \varnothing$ **then** $\mathscr{L}_\sigma \leftarrow \{\ell_{gr}\} \times \prod_{\gamma_i \in CCSet} \mathscr{L}_\sigma^{\gamma_i}$
8 **return** $\mathscr{L}_\sigma$

Γ is then split into disjoint sub-AFs obtained after removing the arguments labelled `in` or `out` in the grounded labelling (line 2). The *CCSet* variable is the set of connected components computed.

Given that there is no relation between them, the labelling computation of those connected components can be made in a simultaneous way (line 3) according to the chosen semantics.

For each of these connected components, a clustering is made (line 4) using the spectral clustering method presented in Section 2.2. The similarity matrix on which the spectral analysis relies is a kind of adjacency matrix where the directionality of edges is omitted and where the matrix values are the number of edges between two arguments. Basically, the more an argument will be related to another, the more similar the two arguments will be considered.

This similarity criterion is particularly relevant for non-dense graphs with a clustered structure. Indeed, it produces sparse matrices and as a consequence the eigenvector equation system to solve will be simplified as there will be many zero values. This is what motivated our choice for the spectral clustering method.

After this clustering process, *ComputeCompLabs* (Alg. 2) is called to compute in a distributed way all the labellings of the connected component according to σ (line 5).

Finally, given that ℓ_{gr} is a fixed part of all σ-labellings of Γ and that all the connected components are completely independent, to construct the σ-labellings of the whole AF, a simple Cartesian product is made (line 7) between the labellings of all the components and the grounded one.

4. Similarity here reflects how much connected two arguments are ; it does not compare anyhow the arguments themselves. The more connected two arguments are, the more we want them to not be in different clusters.

5. Sorted in ascending order, the eigenvalue sequence represents how the similarity within clusters increases as the number of clusters grows. Obviously, the more clusters, the more homogeneous they will get, but also, the more cases to compute. A compromise between the number of clusters and homogeneity is needed. A heuristic (called "elbow heuristic") to find the appropriate number of dimensions to keep, consists in detecting the jump in the eigenvalues sequence.

If one of the components has no labelling then the whole AF has no labelling (so $\mathscr{L}_\sigma = \varnothing$).

Consider now Alg. 2 that computes the component labellings in a distributed way, relying on the clustering made. The σ-labellings of each cluster are computed simultaneously (line 1). Unlike the case of connected components used in Alg. 1, there exist attacks between clusters. In order to compute all the possible σ-labellings of a given cluster, every case concerning its inward attacks (attacks whose target is in the current cluster but the source is from another cluster) have to be considered. Given that the sources of an inward attack could be labelled `in`, `out` or `und` in their own cluster, the σ-labellings of the current cluster have to be computed for all the labelling combinations of inward attack sources.

Algorithm 2: *ComputeCompLabs* algorithm.

Data: Let *ClustSet* be a set of cluster structures for a component γ, σ be a semantics
Result: $\mathscr{L}_\sigma \in 2^{\mathscr{L}(\gamma)}$: the set of the σ-labellings of γ

1 **for all** $\kappa_j \in ClustSet$ **do in parallel**
  $\mathscr{L}_\sigma^{\kappa_j} \leftarrow ComputeClustLabs(\sigma, \kappa_j)$
2 $\mathscr{L}_\sigma \leftarrow$
  $ReunifyCompLabs(\bigcup_{\kappa_j \in ClustSet} \mathscr{L}_\sigma^{\kappa_j}, ClustSet)$
3 **if** $\sigma = pr$ **then** $\mathscr{L}_\sigma \leftarrow \{\ell \,|\, \ell \in \mathscr{L}_\sigma$ s.t. $\nexists \ell' \in \mathscr{L}_\sigma$ s.t. $\texttt{in}(\ell) \subset \texttt{in}(\ell')\}$
4 **return** $\mathscr{L}_\sigma$

Note that having "well shaped" clusters (*i.e.* clusters with few inter cluster attacks) reduces considerably the number of cases to compute, as there are few edges cut. Thus this algorithm is well suited for clustered non-dense graphs.

Once that, for all clusters, the *ComputeClustLabs* function has computed the σ-labellings for all the possible cases (this is done by calling any sound and complete procedure computing the semantics labellings), the *ReunifyCompLabs* function is called in order to reunify compatible labelling parts. Labelling parts are said to be compatible together when all the targets of the inter cluster attacks are legally labelled in the resulting reunified labelling.

A special step has to be done for the *preferred* semantics as this reunifying process does not ensure the maximality (w.r.t $\subseteq$) of the set of `in`-labelled arguments (so not all of the labellings produced in line 2 are *preferred* ones). A maximality check is done (line 3) in order to keep only the wanted labellings.

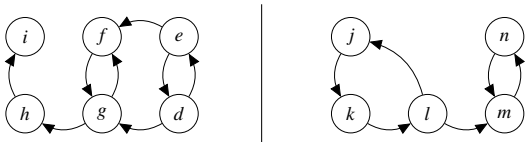
Note that, when computing the *stable* semantics, the set of labellings $\mathscr{L}_\sigma$ returned by the function *ReunifyCompLabs* may be empty. It happens when one of the component clusters has no *stable* labelling.

3.2 An Illustrating Example

In this section, the behaviour of our algorithms is illustrated on the AF given in Figure 1 for the *preferred* semantics, as it is the most complex semantics of the three targeted ones.

The first step consists in computing the grounded labelling in order to eventually split the AF into sub-AFs. The grounded labelling of the AF restricted only to the `in`-labelled and `out`-labelled arguments is : $\ell_{gr} = \{(a, \texttt{in}), (b, \texttt{out}), (c, \texttt{out})\}$.

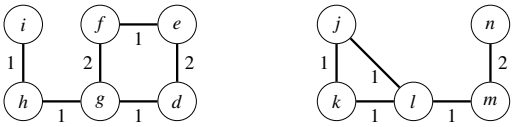
Removing arguments a, b and c from the AF produces two connected components, as illustrated in Figure 2.



(a) Component 1 : γ_1. (b) Component 2 : γ_2.

FIGURE 2 – Connected components resulting from the grounded removal pre-processing.

Then simultaneously γ_1 and γ_2 are clustered using the spectral clustering method This is done by several steps. First, we consider the similarity matrices of γ_1 and γ_2 according to our criterion, *i.e.* the number of attacks between arguments. They may also be seen as the adjacency matrices of the weighted non-directed graphs obtained from γ_1 and γ_2 (see Figure 3). Given that the AF relation density is low, the matrices are rather sparse.



(a) Component 1 : γ_1. (b) Component 2 : γ_2.

$$M_a^{\gamma_1} = \begin{array}{c} \\ d \\ e \\ f \\ g \\ h \\ i \end{array} \begin{array}{c} \begin{array}{cccccc} d & e & f & g & h & i \end{array} \\ \begin{bmatrix} \mathbf{0} & 2 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ 2 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} & 2 & \mathbf{0} & \mathbf{0} \\ 1 & \mathbf{0} & 2 & \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} \end{bmatrix} \end{array}$$

(c) Similarity matrix of γ_1.

$$M_a^{\gamma_2} = \begin{array}{c} \\ j \\ k \\ l \\ m \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} j & k & l & m & n \end{array} \\ \begin{bmatrix} \mathbf{0} & 1 & 1 & \mathbf{0} & \mathbf{0} \\ 1 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ 1 & 1 & \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & 2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 2 & \mathbf{0} \end{bmatrix} \end{array}$$

(d) Similarity matrix of γ_2.

FIGURE 3 – Step 1 of the spectral clustering.

$$M_d^{\gamma_2} = \begin{array}{c} \\ j \\ k \\ l \\ m \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} j & k & l & m & n \end{array} \\ \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \end{array}$$

(a) Degree matrix of γ_2.

$$M_d^{\gamma_2} - M_a^{\gamma_2} = M_l^{\gamma_2} = \begin{array}{c} \\ j \\ k \\ l \\ m \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} j & k & l & m & n \end{array} \\ \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 3 & -2 \\ 0 & 0 & 0 & -2 & 2 \end{bmatrix} \end{array}$$

(b) Laplacian matrix of γ_2.

FIGURE 4 – Step 2 of the spectral clustering for γ_2.

Once the AF similarity matrix is constructed, data are projected in a new space in which similarity is maximised. If a certain structure exists in the data set, we will see in that space some agglomerates appear, corresponding to the node clusters. To do this projection, we compute the n smallest eigenvalues [6] of the Laplacian matrix obtained from the similarity matrix and the vectors associated with them (this n is an arbitrary parameter; in this example we have chosen to keep all the vectors, *i.e.* $n = 5$). Indeed, the eigenvectors found will correspond to the basis of that similarity space and the eigenvalues to the variance on the corresponding axes. Given that we are looking for homogeneous groups, we will consider only the axis on which the variance is low, and so the eigenvectors that have small eigenvalues. The space whose basis is the n selected eigenvectors (corresponding to the n smallest eigenvalues) is then a compression of similarity space (*i.e.* we keep only the dimension useful for a clustering).

Let us take as an example the case of γ_2. Its degree matrix $M_d^{\gamma_2}$ and its Laplacian matrix $M_l^{\gamma_2}$ are given in Figure 4.

The eigenvalues of $M_l^{\gamma_2}$ sorted in ascending order are :

$$\begin{array}{ccccc} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 \end{array}$$
$$\begin{bmatrix} 2.476651 \times 10^{-16} & 5.857864 \times 10^{-1} & 3.000000 & 3.414214 & 5.000000 \end{bmatrix}$$

and their associated eigenvectors are :

$$\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array}$$
$$\begin{bmatrix} -0.4472136 & 0.4397326 & 7.071068 \times 10^{-1} & 0.3038906 & 0.1195229 \\ -0.4472136 & 0.4397326 & -7.071068 \times 10^{-1} & 0.3038906 & 0.1195229 \\ -0.4472136 & 0.1821432 & -5.551115 \times 10^{-17} & -0.7336569 & -0.4780914 \\ -0.4472136 & -0.4397326 & -2.775558 \times 10^{-16} & -0.3038906 & 0.7171372 \\ -0.4472136 & -0.6218758 & -1.665335 \times 10^{-16} & 0.4297663 & -0.4780914 \end{bmatrix}$$

Now that the similarity space is found, the following step is to find how many groups we have in that space. This number can be founded using the eigenvalue sequence sorted in ascending order and identifying in this sequence the "best elbow" (*i.e.* the point that corresponds to a quick growth of the variance, see Figure 5). To compute that "best elbow", in Figure 5, we consider the second derivative (green line with triangles) of the ascending order sequence. As the second derivative represents the concavity of the eigenvalue sequence, we can take the first value of the second derivative above a certain threshold (red line without symbol) determined experimentally (*i.e.* the first position where the eigenvalue sequence is enough convex). The first point of the second derivative, corresponding to the concavity formed by the first three eigenvalues, is the first value above the threshold; so we determine that the "best elbow" is in position 2.

In our example, the number of clusters determined by that heuristic is thus 2.

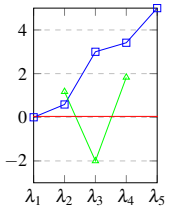


FIGURE 5 – Step 3 of the spectral clustering.

Once the number of clusters is chosen, we must to find the partition of the set of arguments. This is done using a *KMeans* type algorithm [32] [7] applied on the kept eigenvectors following the chosen number of clusters.

The matrix composed by the kept eigenvectors (the *two* first eigenvectors, 2 being the number of clusters) is :

$$\begin{array}{c} \\ j \\ k \\ l \\ m \\ n \end{array} \begin{array}{c} \begin{array}{cc} v_1 & v_2 \end{array} \\ \begin{bmatrix} -0.4472136 & 0.4397326 \\ -0.4472136 & 0.4397326 \\ -0.4472136 & 0.1821432 \\ -0.4472136 & -0.4397326 \\ -0.4472136 & -0.6218758 \end{bmatrix} \end{array}$$

The lines of this matrix correspond to the coordinates of the nodes in the compressed similarity space. With a *KMeans* algorithm we can find groups of datapoint in that space and so have the partition of arguments we wanted (here $\{j,k,l\}$ and $\{m,n\}$) as indicated on Figure 6.

The complete result given by the spectral clustering is shown in Figure 7. κ_1 and κ_2 are the clusters determined from γ_1, and κ_3 and κ_4 are the ones from γ_2.

After the clustering, the next step of our algorithm is the computation of *preferred* labellings. This computation is made simultaneously in the different clusters using an external solver (one of the best solvers identified in the ICCMA competition, see [36]). Recall that, for each cluster, every case concerning its inward attacks (attacks whose target is in the current cluster but the source is from another

6. There exist algorithms, such as *Krylov-Schur* method, able to compute eigenvectors from smallest to greatest eigenvalue and to stop at any wanted step (*e.g.* the number of vectors found). With such an algorithm it is not necessary to find all the solutions as we are interested only in the small eigenvalues.

7. Given n observations, a *KMeans* algorithm aims to partition the n observations into k subsets such that the distance between the elements inside each subset is minimized. Here we have $n = 5$ and $k = 2$.

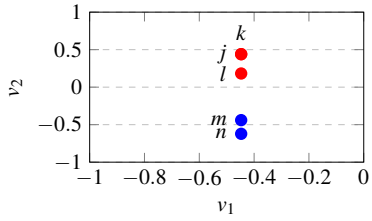
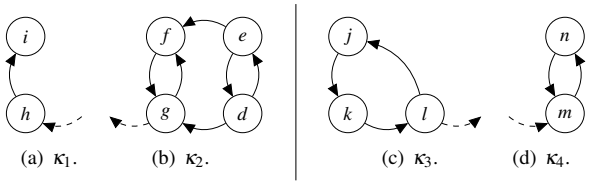


FIGURE 6 – Step 4 of the spectral clustering.



(a) κ_1. (b) κ_2. (c) κ_3. (d) κ_4.

FIGURE 7 – Identified clusters.

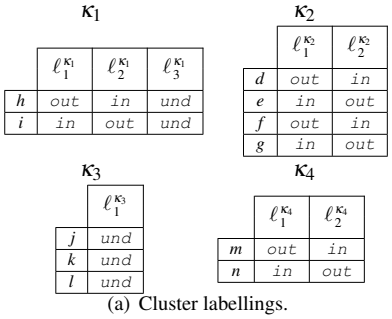
cluster) have to be considered. Given that the sources of an inward attack could be labelled `in`, `out` or `und` in their own cluster, the σ-labellings of the current cluster have to be computed for all the labelling combinations of inward attack sources. For instance, for κ_1 (resp. κ_4), three cases for h and so for i (resp. for m and so for n) must be considered. Whereas for κ_2 and κ_3, there is no inward attack, the computed labellings only depend on the content of the cluster. The tables in Figure 8(a) show the computed labelling parts for each cluster. Notice that although three cases are computed for κ_4, only two labellings are obtained. This is due to the maximality of the *preferred* semantics. Indeed, even though m is attacked by an `und`-labelled argument, n may be labelled `in` as it defends itself against m. As a consequence, m would be labelled `out`.

The last step of our algorithm is the reunifying phase (line 2, Alg. 2). In this step, the constructed labellings are those in which all the target arguments are legally labelled. As an example, $\ell_1^{\kappa_1}$ cannot be reunified with $\ell_2^{\kappa_2}$ as h would be illegally `out`-labelled. Figure 8(b) shows the valid reunified labellings for each component.

In that particular example all the reunified labellings are maximal w.r.t $\subseteq$ of the set of `in`-labelled arguments, so the maximality check (line 3, Alg. 2) does not change the set of labellings. [8]

Finally, the *preferred* labellings of the whole AF are constructed by performing a Cartesian product of the component labellings and of the grounded one. See the final computed *preferred* labellings in Table 1, Section 2.1 (labellings ℓ_1 and ℓ_2).

8. To highlight the necessity of the maximality check, let us take as minimal example the AF defined by $\langle\{a,b\},\{(a,b),(b,a)\}\rangle$ and a partition of it in which each argument is in a different cluster. For each cluster, we will have three possible labellings as the inward attack source may be labelled `in`, `out` or `und` in the other cluster. The reunifying phase will thus admit the labelling $\{(a,\texttt{und}),(b,\texttt{und})\}$ which is not a *preferred* labelling.



κ_1

	$\ell_1^{\kappa_1}$	$\ell_2^{\kappa_1}$	$\ell_3^{\kappa_1}$
h	out	in	und
i	in	out	und

κ_2

	$\ell_1^{\kappa_2}$	$\ell_2^{\kappa_2}$
d	out	in
e	in	out
f	out	in
g	in	out

κ_3

	$\ell_1^{\kappa_3}$
j	und
k	und
l	und

κ_4

	$\ell_1^{\kappa_4}$	$\ell_2^{\kappa_4}$
m	out	in
n	in	out

(a) Cluster labellings.

γ_1

	$\ell_1^{\gamma_1}$	$\ell_2^{\gamma_1}$
d	out	in
e	in	out
f	out	in
g	in	out
h	out	in
i	in	out

γ_2

	$\ell_1^{\gamma_2}$
j	und
k	und
l	und
m	out
n	in

(b) Component labellings.

FIGURE 8 – Labellings computed using our algorithm.

4 Soundness and Completeness

This section presents formal properties of *AFDivider* : soundness and completeness for the *complete*, the *stable* and the *preferred* semantics. Let σ be one of these three semantics. To be *sound* for σ means that the algorithm produces only σ-labellings. To be *complete* for σ means that the algorithm produces all the σ-labellings. In other words, given σ, *AFDivider* produces only and all the σ-labellings.

In order to prove these properties, we rely on the notions of *top-down* and *bottom-up* semantics decomposability introduced in [5] and then developed in [6]. In a few words, a semantics σ is said to be *top-down* decomposable if, for all AF Γ and for all its partitions into sub-AFs, the set of σ-labellings of Γ is included in the set of valid labellings obtained by reunifying the σ-labellings of its sub-AFs. A semantics σ is said to be *bottom-up* decomposable if, for all AF Γ and for all its partitions into sub-AFs, the set of valid labellings obtained by reunifying the σ-labellings of its sub-AFs is included in the set of σ-labellings of Γ. A semantics is said to be *fully* decomposable if it is *top-down* and *bottom-up* decomposable. These notions of *top-down* and *bottom-up* semantics decomposability can also be defined w.r.t. a specific type of partition. For instance, the partition selector denoted by $\mathscr{S}_{USCC}$ only produces partitions in which SCCs (Strongly Connected Components) are not split into different parts. In [6] it has been proven that :

- The *stable* and *complete* semantics are *fully* decomposable.
- The *preferred* semantics is *top-down* decomposable.
- The *preferred* semantics is *fully* decomposable w.r.t.

$\mathscr{S}_{USCC}$.

Proposition 1 AFDivider *is sound and complete for the complete and the stable semantics.*

SKETCH OF PROOF. *Let σ be a fully decomposable semantics. Let $\Gamma = \langle A, R\rangle$ be an AF. Let ℓ_{gr} be the grounded labelling of Γ restricted to the* in*-labelled and* out*-labelled arguments. Let $\Omega = \{\omega_{gr}, \omega_1^1, \ldots, \omega_{n_1}^1, \ldots, \omega_1^k, \ldots, \omega_{n_k}^k\}$ be a partition of A such that ω_{gr} is the set of arguments labelled in ℓ_{gr} and such that for all i and j, ω_i^j is the set of arguments corresponding to the cluster j of the component i determined by the component clustering performed by* AFDivider.

Given that for all clusters, the labellings are computed for all possible labellings of the cluster inward attack sources, and given that σ is fully decomposable, the set of valid reunified labellings produced by AFDivider *is equal to $\mathscr{L}_\sigma(\Gamma)$.*

And so AFDivider *is sound and complete for the* complete *and the* stable *semantics.* ■

Proposition 2 AFDivider *is sound and complete for the preferred semantics.*

SKETCH OF PROOF. *Let σ be the* preferred *semantics. Let $\Gamma = \langle A, R\rangle$ be an AF. Let ℓ_{gr} be the grounded labelling of Γ restricted to the* in*-labelled and* out*-labelled arguments. Let $\{\gamma_1, \ldots, \gamma_k\}$ be the set of all connected components obtained by* AFDivider *after removing ℓ_{gr}. Let $\Omega = \{\omega_{gr}, \omega_1^1, \ldots, \omega_{n_1}^1, \ldots, \omega_1^k, \ldots, \omega_{n_k}^k\}$ be a partition of A such that ω_{gr} is the set of arguments labelled in ℓ_{gr} and such that for all i and j, ω_i^j is the set of arguments corresponding to the cluster j of the component γ_i determined by the component clustering performed by* AFDivider.

Given that the preferred *semantics is* top-down *decomposable, and given that for all clusters, the labellings are computed for all possible labellings of the cluster inward attack sources, then for each component γ_i, $\mathscr{L}_{pr}(\gamma_i)$ is included in the set of valid reunified labellings produced by the function ReunifyCompLabs (Alg. 2, line 2). The maximality check (line 3) makes Alg. 2 sound and complete for the* preferred *semantics.*

Let $\Omega' = \{\omega_{gr}, \omega^1, \ldots, \omega^k\}$ be a partition of A such that ω_{gr} is the set of arguments labelled in ℓ_{gr} and such that for all i : $\omega^i = \bigcup_{j=1}^{n_i}(\omega_j^i)$. Let $S = \{(a,b) | \exists i \text{ s.t. } (a,b) \in (\omega_{gr} \times \omega^i) \cap R\}$ be the set of all attacks going from an argument labelled in ℓ_{gr} to an argument non present in ℓ_{gr}. Note that all the sources of these attacks are out*-labelled in ℓ_{gr}. Let $\Gamma' = \langle A, R'\rangle$ with $R' = R \setminus S$, be the AF obtained from Γ when removing the attacks in S. Given that the sources of attacks removed to obtained Γ' from Γ are all* out*-labelled arguments, we have $\mathscr{L}_\sigma(\Gamma') = \mathscr{L}_\sigma(\Gamma)$. Note that $\Omega' \in \mathscr{S}_{USCC}(\Gamma')$. Indeed, for all i, $(\omega_{gr} \times \omega^i) \cap R' = \varnothing$ and for all $j \neq i$, $(\omega^j \times \omega^i) \cap R' = \varnothing$.*

Given that the preferred *semantics is* fully *decomposable w.r.t. $\mathscr{S}_{USCC}$ then the set of valid labellings obtained by reunifying the σ-labellings of the sub-AFs corresponding to Ω' equal to $\mathscr{L}_\sigma(\Gamma')$. Given that Alg. 2 is sound and complete for the* preferred *semantics, the Cartesian product made in Alg. 1 (line 7) computes exactly $\mathscr{L}_\sigma(\Gamma')$. As a consequence, Alg. 1 computes exactly $\mathscr{L}_\sigma(\Gamma)$.* AFDivider *is thus sound and complete for the* preferred *semantics.* ■

5 Experimental Results

In this section we present some experimental results conducted with the *AFDivider* algorithm. The experiments have been made on some hard instances of the ICCMA competition, which are mostly of Barabási–Albert (BA) type. They all are non-dense and have a clustered structure.

To compute the labellings of a cluster given a particular labelling of its inward attack sources, we have used an already existing solver called "*Pyglaf*", one of the best solvers at the ICCMA 2017 session, which transforms the AF labelling problem into a SAT problem [3]. In this paper, we compare our algorithm (using *Pyglaf*) with *Pyglaf* itself, and with *ArgSemSAT* [20], for the preferred, the complete and the stable semantics.

For each experiment, we used 6 cores of a Intel Xeon Gold 6136 processor, each core having a frequency of 3 GHz. The RAM size was 45GB. As at least two of the three used solvers are multithreaded (*Pyglaf* and *AFDivider*), we have chosen to compare them using both CPU and real time (the CPU time includes the user and the system times). Note that, for our algorithm, the durations cover both the clustering time and the computation of labellings time. The timeout has been set to 1 hour for the real time.

Table 2 gives the obtained results on 8 significant instances :[9] i_1 to i_8 for respectively BA_120_70_1.apx, BA_100_60_2.apx, BA_120_80_2.apx, BA_180_60_4.apx, basin-or-us.gml.20.apx, BA_100_80_3.apx, amador-transit_20151216_1706.gml.80.apx and BA_-200_70_4.apx. Note that these instances have a number of labellings under the *preferred* and *stable* semantics that is particularly large (more than a hundred thousand), and even larger for the complete semantics.

In Table 2, it is worth noting that, first, none of the chosen instances is solved by *ArgSemSAT*; second, that none of the three solvers can provide results for the complete semantics; third, that our algorithm is far better than *Pyglaf* on those instances for the preferred semantics.[10] Actually, we can observe a real order of magnitude change which increases with the hardness of the instances : from 39 seconds to 5 seconds for i_1 and from almost one hour to 31 seconds for i_5 (i_6 to i_8 being unsolved by *Pyglaf* in less than one hour). The last chosen instance (i_8), with its more than ten billion *preferred* labellings, presents a memory representation challenge; a compressed representation of the labellings is to be found to tackle such instances. This is also the case for the complete semantics. Finally, concer-

9. *amador-transit_20151216_1706.gml.80.apx* and *basin-or-us.gml.20.apx* are instances which come from real data of the traffic domain.

10. Note that *Pyglaf* is also multi-core. Moreover, when we compare *Pyglaf* and *AFDivider*, we use a computer with the same number of cores. So the fact that there is a more important parallelization in *AFDivider* (so more threads) is not what explains the difference in runtime for the preferred semantics.

ning the stable semantics, *Pyglaf* and *AFDivider* give similar results : in term of real time, *Pyglaf* is slightly better except on i_7. Nevertheless, it is worth noting that, in term of CPU time, *AFDivider* is generally better than *Pyglaf*; this last point needs further studies.

Overall, these preliminary experimental results show that the AF clustering approach brings a real added value in terms of resolution time in the case of the preferred semantics, and that an additional analysis will be necessary for identifying how to improve the results for the other semantics.

Moreover, the algorithm is being tested on other instances of the ICCMA competition, with a structure which may be dense or non clustered. The use of other kinds of clustering methods is also under study.

6 Related Work

There exist many approaches for enumerating semantics labellings, but most of them are non-direct, in the sense that they reduce the semantics computation to other problems (most of the time to the SAT problem). Such non-direct approaches may use some kind of cutting process and even distributed computation (it is the case of *Pyglaf* [3]). Direct approaches, such as *AFDivider*, are less common. It is with the existing direct approaches that we compare in this section the *AFDivider* algorithm.

Here are some direct approach algorithms which use some kind of cutting techniques : [11] [25], that presents an algorithm based on a dynamic analysis of an argumentation framework ; [28], where the algorithm computes the labellings of an AF following its SCC decomposition ; [18], where the *R-PREF* algorithm is based on [28]'s approach, with the addition of applying the decomposition process recursively when the labellings under construction break the SCCs ; [19], where the *P-SCC-REC* algorithm, inspired by notions introduced in [5, 9, 29, 30], is the parallelized version of *R-PREF* ; [11], where the algorithm splits the AF in two parts (without breaking SCCs), and computes their labelling before reunifying them. Let us compare *AFDivider* with these approaches in two respects.

- First, on their ability to break SCCs : [28] and [11] do not do so ; [18] and [19] can do so, given a current SCC and an ancestor labelling, but only when the ancestor labelling has some particular effects on the current SCC ; [25] always breaks SCCs as at each step at most one argument is added or removed from the considered sub-AF. Nevertheless, this way of updating argument after argument in [25] generates a lot of computations and uses a lot of memory. *AFDivider*, and this is one of its advantages, breaks SCCs whenever it is well suited to have well shaped clusters.
- Second, on their ability to compute the labellings in a distributed way : [11, 18, 25, 28] are fully sequential. *AFDivider* and [19] use distributed computation, but in [19], the computation of one labelling is mainly sequential (it is very unlikely that the greedy phase suffices to generate a labelling). Furthermore parallelizing following labellings could overload the CPUs as the number of solutions in hard AF problems may be huge.

To conclude, what distinguishes best *AFDivider* from the other ones is that cutting the AF into clusters limits the combinatorial effect due to the number of labellings, to the cluster. The other approaches propagate this effect to the whole AF. This property makes *AFDivider* well suited for non-dense AF with a clustered structure. Indeed, in such a structure, the reunifying phase will be less expensive than exploring the whole AF to construct each of the labellings.

An incremental algorithm that computes labellings has been proposed in [2] but it does not concern the enumeration problem. Other works such as [14, 22, 39] might be related to our approach as they analyze some kind of AF matrices ; however, it is not done in order to cluster the AF.

7 Conclusion

AFDivider is the first algorithm that uses spectral clustering methods to compute semantics labellings. After removing the trivial part of the AF (grounded labelling), the algorithm cuts the AF into small pieces (the identified clusters), then it computes simultaneously (in each cluster) labelling parts of the AF, before reunifying compatible parts to get the whole AF labellings. Soundness and completeness of this algorithm are proven for the *stable*, the *complete* and the *preferred* semantics.

We compared the behaviour of our algorithm with other ones that also use some kind of clustering. Among the various advantages of our method (its ability to break SCCs and to compute the labellings in a distributed way), we highlighted the fact that cutting the AF into clusters has the great advantage of limiting the solving hardness to the clusters. This algorithm is particularly well suited for non-dense AFs with a clustered structure, such as the ones which are among the hardest instances of the ICCMA competition.

An empirical analysis of *AFDivider* on the benchmarks of the competition is underway and some preliminary results are presented in this paper. Nevertheless, more exhaustive experiments are planned, in particular :

- an analysis of the impact of the partition on the solving time, from a random one to a clustered one ; different clustering methods may also be compared ;
- a complete comparison with the other existing solvers used in ICCMA competition including the 2019 edition (for instance, CoquiAAS [27], or μ-toksia [34],

11. For an overview on the different AF splitting possibilities see [10].

which is the winner of the 2019 edition) ;
- and finally the use of *AFDivider* for the other tasks, on the other semantics, of the competition (see [13]).

Another interesting question to answer is how to know in a reasonable time if an AF is well suited for the *AFDivider* algorithm. In fact, this is a double question : "what is a theoretical characterization of such an AF ?" and "given an AF, what is the computational cost for checking whether it respects this characterization ?".

Moreover, among future works, this approach may be extended to enriched argumentation frameworks (*e.g.* with a support relation or with higher-order interactions), and to other acceptability semantics.

8 Acknowledgements

This work was supported by the ANR-11-LABEX-0040-CIMI project of the CIMI International Centre for Mathematics and Computer Science in Toulouse.

Références

[1] Albert, R. et A. L. Barabási: *Statistical mechanics of complex networks*. Reviews of modern physics, 74(1) :47, 2002.

[2] Alfano, G., S. Greco et F. Parisi: *Efficient Computation of Extensions for Dynamic Abstract Argumentation Frameworks : An Incremental Approach.* Dans *IJCAI*, pages 49–55, 2017.

[3] Alviano, M.: *The pyglaf argumentation reasoner*. Dans *OASIcs-OpenAccess Series in Informatics*, tome 58, 2018.

[4] Barabási, A.-L *et al.*: *Network science*. Cambridge university press, 2016.

[5] Baroni, P., G. Boella, F. Cerutti, M. Giacomin, L. W. N. van der Torre et S. Villata: *On Input/Output Argumentation Frameworks*. Dans *COMMA*, pages 358–365, 2012.

[6] Baroni, P., G. Boella, F. Cerutti, M. Giacomin, L. Van Der Torre et S. Villata: *On the input/output behavior of argumentation frameworks*. Artificial Intelligence, 217 :144–197, 2014.

[7] Baroni, P., M. Caminada et M. Giacomin: *An introduction to argumentation semantics*. Knowledge Eng. Review, 26(4) :365–410, 2011.

[8] Baroni, P., F. Cerutti, M. Giacomin et G. Guida: *AFRA : Argumentation framework with recursive attacks*. International Journal of Approximate Reasoning, 52(1) :19–37, 2011.

[9] Baroni, P., M. Giacomin et B. Liao: *On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems : A division-based method*. Artificial Intelligence, 212 :104–115, 2014.

[10] Baroni, P., M. Giacomin et B. Liao: *Locality and modularity in abstract argumentation*. Dans *Handbook of formal argumentation*, pages 937–979. College Publication, 2018.

[11] Baumann, R., G. Brewka et R. Wong: *Splitting argumentation frameworks : An empirical evaluation*. Dans *TAFA workshop*, pages 17–31. Springer, 2011.

[12] Bistarelli, S., F. Rossi et F. Santini: *Not only size, but also shape counts : abstract argumentation solvers are benchmark-sensitive*. J. Log. Comput., 28(1) :85–117, 2018.

[13] Bistarelli, S., F. Santini, L. Kotthoff, Th. Mantadelis et C. Taticchi: *Int. Competition on Computational Models of Argumentation*, 2019. https://www.iccma2019.dmi.unipg.it/.

[14] Butterworth, J. et P.E. Dunne: *Spectral Techniques in Argumentation Framework Analysis*. COMMA, 287 :167, 2016.

[15] Caminada, M.: *On the Issue of Reinstatement in Argumentation*. Dans *JELIA*, pages 111–123, 2006.

[16] Carrera, Álvaro et Carlos A Iglesias: *A systematic review of argumentation techniques for multi-agent systems research*. Artificial Intelligence Review, 44(4) :509–535, 2015.

[17] Cayrol, C. et M C. Lagasquie-Schiex: *On the acceptability of arguments in bipolar argumentation frameworks*. Dans Godo, L. (rédacteur) : *ECSQARU*, pages 378–389, 2005.

[18] Cerutti, F., M. Giacomin, M. Vallati et M. Zanella: *An SCC recursive meta-algorithm for computing preferred labellings in abstract argumentation.* Dans *KR*, 2014.

[19] Cerutti, F., I. Tachmazidis, M. Vallati, S. Batsakis, M. Giacomin et G. Antoniou: *Exploiting Parallelism for Hard Problems in Abstract Argumentation*. Dans *AAAI*, pages 1475–1481, 2015.

[20] Cerutti, F., M. Vallati, M. Giacomin et T. Zanetti: *ArgSemSAT-2017*, 2017.

[21] Charwat, G., W. Dvořák, S. A. Gaggl, J. P. Wallner et S. Woltran: *Methods for solving reasoning problems in abstract argumentation–a survey*. Artificial intelligence, 220 :28–63, 2015.

[22] Corea, Carl et Matthias Thimm: *Using Matrix Exponentials for Abstract Argumentation*. Dans *SAFA workshop*, pages 10–21, 2016.

[23] Doutre, S., M. Lafages et M.-Ch. Lagasquie-Schiex: *A Distributed and Clustering-Based Algorithm for the Enumeration Problem in Abstract Argumentation*. Dans *PRIMA 2019*, tome 11873 de *LNCS*, pages 87–105. Springer, 2019.

	arguments														σ			
	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*gr*	*co*	*pr*	*st*
ℓ_1	in	out	out	out	in	out	in	out	in	und	und	und	out	in		×	×	
ℓ_2	in	out	out	in	out	in	out	in	out	und	und	und	out	in		×	×	
ℓ_3	in	out	out	out	in	out	in	out	in	und	und	und	und	und		×		
ℓ_4	in	out	out	in	out	in	out	in	out	und	und	und	und	und		×		
ℓ_5	in	out	out	und	und	und	und	und	und	und	und	und	und	und	×	×		
ℓ_6	in	out	out	und	und	und	und	und	und	und	und	und	out	in		×		

TABLE 1 – Labellings of the *AF* of Fig. 1 under the grounded, complete, preferred and stable semantics.

			i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
PR	Nb lab. (≈)		0.28×10^6	1.07×10^6	1.28×10^6	1.37×10^6	1.96×10^6	4.47×10^6	11.75×10^6	10.74×10^9
	AFDivider	end state	✓	✓	✓	✓	✓	✓	✓	MO
		CPU time	0 :07.35	0 :14.05	0 :19.89	0 :31.39	0 :28.01	0 :46.27	12 :15.16	
		real time	0 :05.84	0 :27.98	0 :20.42	0 :35.05	0 :31.31	1 :09.10	12 :39.21	
	Pyglaf	end state	✓	✓	✓	✓	✓	TO	TO	TO
		CPU time	0 :45.33	6 :18.60	11 :06.51	15 :21.07	54 :49.31			
		real time	0 :39.00	6 :04.37	10 :12.22	14 :51.09	54 :20.72			
	ArgSemSAT	end state	TO	TO	TO	TO	TO	TO	TO	TO
ST	Nb lab. (≈)		Idem preferred case							
	AFDivider	end state	✓	✓	✓	✓	✓	✓	✓	MO
		CPU time	0 :06.52	0 :13.29	0 :18.01	0 :28.50	0 :26.66	0 :45.56	1 :39.14	
		real time	0 :06.26	0 :13.20	0 :18.78	0 :31.02	0 :29.46	0 :50.79	1 :48.30	
	Pyglaf	end state	✓	✓	✓	✓	✓	✓	✓	TO
		CPU time	0 :05.43	0 :17.31	0 :24.78	0 :31,50	0 :41.69	1 :13.10	3 :35.76	
		real time	0 :03.02	0 :09.22	0 :14.76	0 :18.43	0 :21.15	0 :42.57	1 :53.95	
	ArgSemSAT	end state	TO	TO	TO	TO	TO	TO	TO	TO
CO	Nb lab. (≈)		0.80×10^9	5.22×10^9	9.31×10^9	11.93×10^9	16.18×10^9	49.58×10^9	-	22×10^{15}
	AFDivider	end state	MO	MO	MO	MO	MO	MO	MO	MO
	Pyglaf	end state	TO	TO	TO	TO	TO	TO	TO	TO
	ArgSemSAT	end state	TO	TO	TO	TO	TO	TO	TO	TO

TABLE 2 – Experimental results (PR : preferred, CO : complete, ST : stable, MO : "Memory Overflow", TO : "stop with TimeOut", "-" : "missing data"). The time result format is "minutes :seconds.centiseconds".

[24] Dung, P. M.: *On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games*. Artificial Intelligence, 77 :321–357, 1995.

[25] Dvořák, W., R. Pichler et S. Woltran: *Towards fixed-parameter tractable algorithms for abstract argumentation*. Artificial Intelligence, 186 :1–37, 2012.

[26] Kröll, M., R. Pichler et S. Woltran: *On the complexity of enumerating the extensions of abstract argumentation frameworks*. Dans *IJCAI*, pages 1145–1152, 2017.

[27] Lagniez, J. M., E. Lonca, et J. G. Mailly: *CoQuiAAS v3.0. ICCMA 2019 Solver Description*, 2019.

[28] Liao, B.: *Toward incremental computation of argumentation semantics : A decomposition-based approach*. Annals of Mathematics and Artificial Intelligence, 67(3-4) :319–358, 2013.

[29] Liao, B. et H. Huang: *Partial semantics of argumentation : basic properties and empirical*. Journal of Logic and Computation, 23(3) :541–562, 2013.

[30] Liao, B., L. Jin et R. C. Koons: *Dynamics of argumentation systems : A division-based method*. Artificial Intelligence, 175(11) :1790–1814, 2011.

[31] Lloyd, S.: *Least squares quantization in PCM*. IEEE transactions on information theory, 28(2) :129–137, 1982.

[32] Lloyd, S.: *Least squares quantization in PCM*. IEEE transactions on information theory, 28(2) :129–137, 1982.

[33] Malliaros, F. D et M. Vazirgiannis: *Clustering and community detection in directed networks : A survey*. Physics Reports, 533(4) :95–142, 2013.

[34] Niskanen, A. et M. Järvisal: *μ-toksia. Participating in ICCMA 2019*, 2019.

[35] Robert, M. K.: *Elementary linear algebra*. University of Queenland, 2013.

[36] Rodrigues, O., E. Black, M. Luck et J. Murphy: *On Structural Properties of Argumentation Frameworks : Lessons from ICCMA*. Dans *SAFA workshop*, pages 22–35, 2018.

[37] Schaeffer, S. E.: *Graph clustering*. Computer science review, 1(1) :27–64, 2007.

[38] Von Luxburg, U.: *A tutorial on spectral clustering*. Statistics and computing, 17(4) :395–416, 2007.

[39] Xu, Y. et C. Cayrol: *Initial sets in abstract argumentation frameworks*. Journal of Applied Non-Classical Logics, 28(2-3) :260–279, 2018.