MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université de Paris

Master 1 Informatique – IAD

Rapport de projet TER

RATIONALISATION DES SYSTÈMES D'ARGUMENTATION

Assia BOURAÏ – Félix DE FRAMOND

Année universitaire : 2019 – 2020

Encadrant : Elise BONZON

REMERCIEMENTS

Nos premiers remerciements s'adressent à Mme Elise BONZON qui a bien voulu nous accorder sa confiance en acceptant d'encadrer ce travail. Elle a su nous guider et enrichir ce dernier par ses remarques constructives. Nous lui témoignons notre estime et notre gratitude pour sa disponibilité et pour avoir assuré le suivi de ce projet tout au long du semestre.

Nos remerciements s'adressent également à tous nos enseignants de l'UFR Math-Info pour leurs précieux enseignements et surtout pour avoir assuré une continuité pédagogique malgré les difficultés rencontrées dues à la crise sanitaire que nous vivons depuis maintenant deux mois.

RÉSUMÉ

La littérature sur l'intelligence artificielle modélise la divergence des points de vue et les conflits d'opinions dans les systèmes multi-agents par des cadres d'argumentation abstraits. Chaque agent dispose d'un cadre d'argumentation propre à lui-même, composé d'arguments qu'il avance et d'une relation d'attaque binaire entre eux.

Dès lors qu'on dispose d'un certain nombre de cadres d'argumentation, on peut se poser la question de savoir si cet ensemble de cadres qui forment un système peut être représenté sous forme d'un seul cadre d'argumentation général, auquel s'ajoute une attribution de valeur à chaque argument ainsi qu'une relation de préférence entre arguments suivant les dites valeurs pour chaque agent.

Notre travail porte donc sur l'étude des aspects algorithmiques de la rationalisation des systèmes d'argumentation, et de proposer des expérimentations permettant cela.

TABLE DES MATIÈRES

INTRODUCTION GÉNÉRALE

1. Introduction générale

1.1 Contexte et motivations

L'argumentation est un mécanisme utilisé dans notre vie quotidienne pour prendre des décisions. En effet, lorsque les avis divergent sur quelque chose, celle-ci intervient pour expliquer un point de vue ou pour convaincre sur la prise d'une décision donnée.

Dans notre projet, on s'intéresse à l'argumentation et pour cela nous nous sommes essentiellement basés sur l'étude de deux articles traitants la problématique de rationalisation des profils de cadres d'argumentation abstraite [1, 2].

1.2 Contributions et organisation du rapport

Notre rapport est structuré en trois chapitres comme suit :

Dans le premier chapitre, nous définissons ce qu'est l'argumentation abstraite, nous présentons la terminologie ainsi que les différentes notations utilisées tout au long du rapport.

Dans le deuxième chapitre, nous abordons la problématique de rationalisation des systèmes d'argumentation en détaillant les concepts principaux et en présentant des algorithmes qui feront l'objet du dernier chapitre.

TERMINOLOGIE ET NOTATIONS

2. Terminologie et notations

2.1 Introduction

Dung (1995) [4], a défini le cadre d'argumentation **AF (Argumentation Framework)** comme étant une relation binaire représentant un système d'argumentation sous forme de graphe où les nœuds et les arcs correspondent respectivement aux arguments et aux attaques entre arguments.

Dans ce chapitre, nous définissons la terminologie utilisée dans notre rapport et nous introduisons les notations avec lesquelles nous travaillons.

2.2 Cadre d'argumentation abstraite

Un cadre d'argumentation est une paire $\mathrm{AF} = \langle \mathrm{Arg}, \rightharpoonup \rangle$, où Arg est un ensemble fini d'arguments et $\rightharpoonup$ la relation d'attaque entre les arguments.

Exemple 1 : On reprend l'exemple des articles étudiés [1, 2].

La pollution devient un problème de santé majeur dans les grandes villes. Les conseils municipaux se posent la question de l'interdiction éventuelle des véhicules polluants, en particulier des voitures diesel. Un conseil municipal pourrait avoir les arguments suivants :

(A) Les voitures diesel devraient être interdites dans les centres villes afin de réduire la pollution

(B) Les artisans, qui bénéficient d'une aide spéciale de la part du conseil municipal, ne peuvent pas changer leurs véhicules, car cela leur coûterait trop cher.

(C) La ville peut offrir une aide financière aux artisans.

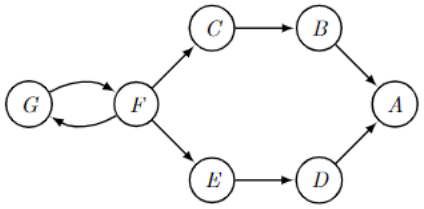
(D) Il n'y a que très peu d'alternatives à l'utilisation de voitures diesel. Plus précisément, l'autonomie des voitures électriques est médiocre, car il n'y a pas assez de bornes de recharge autour.

(E) La ville peut mettre en place plus de bornes de recharges.

(F) En période de crise financière, la ville ne doit pas s'engager à dépenser de l'argent supplémentaire.

(G) Les questions de santé et de changements climatiques sont importantes, la ville doit donc dépenser ce qui est nécessaire pour lutter contre la pollution.

Le graphe qui suit montre l'AF généré par ces arguments.



On remarque que dans cet AF, il est ambiguë d'accepter ou de rejeter l'argument A, car {A, C, E, G} et {B, D, F} sont tous les deux des ensembles admissibles, comme nous le définirons dans ce qui suit.

D'après Dung [4], on définit les notions suivantes :

- Un ensemble S d'arguments est dit sans conflits s'il n'existe pas d'arguments A,B dans S tels que A attaque B ou B attaque A.
- Étant donné un ensemble S, un argument A est dit acceptable si et seulement si pour chaque argument B : si B attaque A alors B est attaqué par un autre argument appartenant à S. On dit alors que A est défendu de l'attaque de l'argument B.
- Un ensemble S d'arguments est dit admissible s'il est sans conflit et que chaque argument de S est acceptable, i.e, S défend chacun de ses arguments.

2.3 Audience-specific Value-based Argumentation Framework

D'après Bench-Capon (2003) [3], on définit un cadre d'argumentation basé sur les valeurs spécifiques à l'audience **AVAF (Audience-specific Value-based Argumentation Framework)** comme étant un AF doté d'une fonction qui associe chaque argument à une valeur (sociale ou morale) combinée à un ordre de préférence déclaré sur ces valeurs.

Il est a noté que le mappage des arguments aux valeurs est fixe et le même pour tout le monde mais les préférences sur les valeurs sont propres à un agent particulier, ce qu'on appelle « audience ».

La relation d'ordre de préférence est une relation binaire réflexive et transitive. La partie stricte d'une relation d'ordre de préférence $\succcurlyeq$ est notée $\succ$ et la partie indifférente (peut exprimer que les valeurs sont équivalentes) est notée $\sim$. On note alors, $x \succ y$ si $x \succcurlyeq y$ et $non\ y \succcurlyeq x$ et $x \sim y$ si $x \succcurlyeq y$ et $y \succcurlyeq x$.

Un AVAF est donc défini comme étant un 5-uplet $\langle Arg, \rightharpoonup, Val, val, \succcurlyeq \rangle$, avec :

- $\langle Arg, \rightharpoonup \rangle$, cadre d'argumentation AF.
- Val, ensemble fini de valeurs.
- $val: Arg \rightarrow Val$, fonction qui associe à chaque argument sa valeur.
- $\succcurlyeq$, la relation d'ordre de préférence de l'agent.

Soit $=_{val}$ la relation d'équivalence sur les arguments telle que $A =_{val} B$ si et seulement si $val\ (A) = val\ (B)$.

2.4 Relation de défaite

Étant donné un $AVAF\ \langle Arg, \rightharpoonup, Val, val, \succcurlyeq \rangle$, on dit qu'un argument $A \in Arg$ bat un argument $B \in Arg$, noté $A \rightleftharpoons B$, si et seulement si $A \rightharpoonup B$ et $non\ val\ (B) \succ val(A)$.

La notation $\rightleftharpoons$ désigne la relation de défaite induite par l'AVAF. Cette relation peut être vue comme étant un nouveau cadre d'argumentation tel que $AF = \langle Arg, \rightleftharpoons \rangle$.

Exemple 1 :

Reprenons l'exemple vu précédemment concernant les arguments avancés par le conseil municipal.

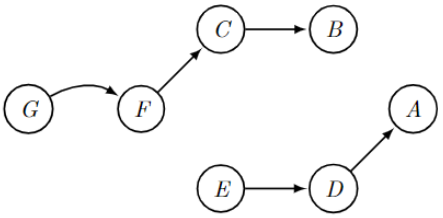
On peut associer à cet exemple quatre types de valeurs : $Val = \{env,\ soc,\ eco,\ infra\}$

- Les arguments A et G portent sur la responsabilité environnementale, $val(A) = val(G) = env$
- Les arguments B et C concernent l'équité sociale, $val(B) = val(C) = soc$.
- L'argument F concerne l'économie, $val(F) = eco$
- Les arguments D et E portent sur l'efficacité des infrastructures $val(D) = val(E) = infra$.

Imaginons qu'un conseiller particulier veuille promouvoir les valeurs de responsabilité environnementale et d'efficacité des infrastructures par rapport aux deux autres valeurs.

On obtient alors la relation de d'ordre de préférence qui suit : $env \sim infra \succ soc \sim eco$

La relation de défaite correspondante est représentée par le graphe qui suit :



On voit bien que trois attaques ont été supprimées par rapport à l'AF précédent.

Dans ce nouvel AF, il est clair qu'on peut accepter l'argument A car il est attaqué par l'argument D qui est lui-même attaqué par E qui n'est attaqué par aucun autre argument.

Dans ce cas-là, on dit que E défend A.

2.5 Conclusion

Nous avons introduit dans ce premier chapitre les bases de l'argumentation abstraite en définissant ce qu'est un AF, un AVAF et une relation de défaite entre arguments. Ces notions seront utilisées pour l'étude des différents algorithmes que nous allons introduire dans le chapitre qui suit.

RATIONALISATION DES SYSTÈMES D'ARGUMENTATION

3. Rationalisation des systèmes d'argumentation

3.1 Introduction

Comme nous l'avons vu dans le chapitre précédent, un système d'argumentation abstrait est un modèle des connaissances d'un agent, qui consiste à mettre en relation, sous forme d'un graphe orienté des arguments. Dans sa version valuée, chaque argument est associé à une valeur représentant le sujet dont il parle. Un agent peut avoir un ordre de préférences sur ces valeurs.

Chaque agent peut donc avoir, à partir d'un même système d'argumentation commun initial, des systèmes d'argumentation personnels différents. Un profil de systèmes d'argumentation est alors un ensemble de cadres d'argumentation propres à chaque agent.

Dans ce chapitre, nous allons définir de manière concrète le problème de rationalisation des cadres d'argumentation, c'est-à-dire, passer d'un profil de systèmes d'argumentation à un système d'argumentation commun doté d'une assignation de valeurs et d'une relation de préférence pour chaque argument.

3.2 Définition du problème de rationalisation

Soit $N = \{1, \dots, n\}$ un ensemble fini d'agents. Chaque agent fournit un AF avec un nombre donné d'arguments et d'attaques. On nomme cet ensemble d'AF profile d'AFs ou profile de systèmes d'argumentation.

Chaque AF d'un tel profil peut être vu comme étant le résultat d'avoir imposé la relation de préférences sur les valeurs de l'agent correspondant à un AF maître. On peut alors noter les $\langle Arg\ i, \succeq i \rangle$ tel que $Arg\ i$ est l'ensemble d'arguments de l'agent i et $\succeq i$ est la relation de défaite

sur $Arg\ i$ adoptée par i . On note alors le profil d'AF's comme suit : $AF = (\langle Arg\ 1\ , \rightleftharpoons 1\ \rangle, \ldots, \langle Arg\ n\ , \rightleftharpoons n\ \rangle)$ avec $\mathrm{Arg}: = Arg\ 1 \cup \ldots \cup Arg\ n$.

Un profil d'AFs $AF = (\langle Arg\ 1\ , \rightleftharpoons 1\ \rangle, \ldots, \langle Arg\ n\ , \rightleftharpoons n\ \rangle)$ est dit rationalisable, étant donné un ensemble de contraintes, s'il existe une relation d'attaque $\rightharpoonup$ sur $\mathrm{Arg}: = Arg\ 1 \cup \ldots \cup Arg\ n$, un ensemble de valeurs Val avec une fonction qui associe chaque argument à une valeur $\mathrm{val} : \mathrm{Arg} \rightarrow Val$ et une relation de préférence sur Val $(\succcurlyeq 1, \ldots, \succcurlyeq n)$, chacun répondant aux dites contraintes, de telle sorte que, pour chaque agent $i \in N$ et pour tout argument $A, B \in Arg$,

$A \rightleftharpoons i\ B$ si et seulement si $A \rightharpoonup B$ mais $val(B) i \succ val(A)$. On note $\langle Arg, \rightharpoonup \rangle$ comme étant l'AF maître et $\rightharpoonup$ la relation d'attaque de ce dernier.

3.3 Rationalisation dans le cas mono-agent

Le problème de rationalisation peut être appliqué sur différentes types de contraintes:

- La relation d'attaque de l'AF maître $\rightharpoonup$ est fixée,
- Une assignation de valeurs aux arguments $\langle Val, val\ \rangle$ est fixée,
- Le nombre de valeurs $|Val|$ peut-être borné par une certaine valeur k,
- La relation d'ordre des préférences $\succcurlyeq i$ doit être complète.

Dans notre projet, nous nous sommes basé sur le cas mono-agent et nous avons étudié la rationalisation dans le cas d'absence de contraintes, le cas où la relation d'attaque de l'AF maître est fixée et enfin le cas où une assignation de valeurs aux arguments est fixée.

Dans le cas mono-agent on dispose d'un AF doté d'une relation de défaite $\langle Arg, \rightleftharpoons \rangle$. Le but de la rationalisation est de trouver un AVAF $\langle Arg, \rightharpoonup, Val, val, \succcurlyeq \rangle$ qui possède le même ensemble d'arguments et qui induit la relation $\rightleftharpoons$.

3.3.1 Absence de contraintes

Les articles [1, 2] sur lesquels nous avons travaillé prouvent qu'en l'absence de contraintes un AF unique peut facilement être rationalisable. En effet, cela peut être fait de manière triviale en : considérant tout simplement la relation d'attaque de l'AF comme étant la relation de défaite de l'agent $(\rightharpoonup) := (\rightleftharpoons)$, en choisissant l'assignation de valeurs $\langle Val, val\ \rangle$ de manière arbitraire et en prenant en compte une relation d'ordre de préférences $\succcurlyeq$ de sorte que l'agent soit indifférent entre deux valeurs.

3.3.2 Relation d'attaque de l'AF maître fixée

Les articles [1, 2] étudiés présentent la rationalisation d'un cadre d'argumentation dans le cas où la relation d'attaque de l'AF maître est fixée comme suit.

Proposition 1 : Un AF$\langle Arg, \rightleftharpoons \rangle$ est rationalisable par un AVAF avec une relation d'attaque de l'AF maître fixée $\rightharpoonup$ si et seulement si toutes les conditions suivantes sont vérifiées :

i. $(\rightleftharpoons) \subseteq (\rightharpoonup)$, les attaques de l'AF de l'agent sont bien présentes dans l'AF maître,

ii. $(\rightharpoonup \setminus \rightleftharpoons)$ est acyclique, les attaques supprimées de l'AF maître forment un graphe acyclique.

iii. $(\rightleftharpoons) \cap (\rightharpoonup \setminus \rightleftharpoons)^{+} = \emptyset$.

Les conditions (ii) et (iii) permettent de vérifier que la relation d'ordre des préférences de l'agent est bien transitive et irréflexive et qu'on ne supprime pas d'attaques qui doivent en fait rester.

3.3.3 Assignation des valeurs fixée

Les articles [1, 2] étudiés présentent la rationalisation d'un cadre d'argumentation dans le cas où l'assignation des valeurs aux arguments est fixée, comme suit.

Proposition 2 : Un AF $\langle Arg, \rightleftharpoons \rangle$ est rationalisable par un AVAF avec une relation d'attaque de l'AF maître $\rightharpoonup$ et une assignation de valeurs aux arguments fixées si et seulement si les trois conditions suivantes sont vérifiées :

i. $(\rightleftharpoons) \subseteq (\rightharpoonup)$,

ii. La relation $\bigcup_{A(\rightharpoonup \setminus \rightleftharpoons)B} \{(val(A), val(B))\}$ est acyclique,

iii. $(\rightleftharpoons) \cap (\rightharpoonup \setminus \rightleftharpoons)^{+}_{val} = \emptyset$

La condition (i) est requise pour que l'agent n'ajoute pas d'attaques en plus de celles présentes dans l'AF maître.

Les conditions (ii) et (iii) permettent de vérifier que :

- La relation d'ordre des préférences est bien définie, c'est-à-dire que la relation est transitive et irréflexive,
- les attaques entre arguments portant la même valeur ne peuvent pas être supprimées,
- On ne supprime pas d'attaques qui doivent en fait rester

3.4 Conclusion

Nous avons présenté dans ce chapitre la rationalisation des profils de systèmes d'argumentation et nous avons détaillé deux propositions fondamentales qui concernent des contraintes jugées importantes dans la résolution du problème et que nous utiliserons dans la partie implémentation des algorithmes.

IMPLÉMENTATION DES ALGORITHMES

4. Implémentation des algorithmes

4.1 Introduction

Dans ce chapitre, on s'intéresse à l'implémentation des propositions énoncées dans le chapitre précédent.

Notre objectif est d'obtenir une preuve constructiviste des propositions 1. et 2., c'est-à-dire :

- Pour la proposition 1., obtenir un algorithme permettant, étant donné un AF $\langle Arg, \rightleftharpoons \rangle$ et une relation d'attaque $\rightharpoonup$ vérifiant les conditions (i), (ii) et (iii) de la proposition 1, d'exposer une relation d'ordre partiel entre les différents arguments permettant de construire un AVAF qui rationalise l'AF.
- Pour la proposition 2., obtenir un algorithme permettant, étant donné un AF $\langle Arg, \rightleftharpoons \rangle$.; une relation d'attaque $\rightharpoonup$, et une assignation de valeurs $\langle Val, val \rangle$., vérifiant les conditions (i), (ii) et (iii) de la proposition 2, d'exposer une relation d'ordre partiel $\succcurlyeq$ sur Val, permettant de construire un AVAF qui rationalise l'AF.

4.2 Rationalisation dans le cas où ou la relation d'attaque de l'AF maître fixée

Pour cet algorithme, il nous est apparu intéressant de considérer que les données du problème ne sont plus l'AF de l'agent ($\rightleftharpoons$) et l'AF maître ($\rightharpoonup$), mais l'AF de l'agent ($\rightleftharpoons$) et un nouvel AF ($\Rightarrow$) :=($\rightharpoonup \setminus \rightleftharpoons$) qui représente les attaques présentes dans l'AF maître mais pas dans l'AF de l'agent. Cela permet de simplifier la résolution dans le cas de la proposition 2.

Si on traduit les conditions pour le nouveau problème, cela donne :

i. $(\rightleftharpoons) \subset (\rightleftharpoons \cup \Rightarrow)$.

ii. $(\Rightarrow)$ est acyclique

iii. $(\rightleftharpoons) \cap (\Rightarrow)^{+} = \emptyset$

On ajoute une condition supplémentaire :

iv. $(\rightleftharpoons) \cap (\Rightarrow) = \emptyset$

Il est nécessaire d'ajouter la condition (iv) pour avoir une parfaite équivalence entre les deux problèmes : avant le changement de variable, il n'était pas évident que $(\rightleftharpoons) \cap (\rightharpoonup \backslash \rightleftharpoons) = \emptyset$, mais ce n'est plus le cas après le changement de variable. Pour passer du nouveau problème à l'ancien problème, il suffit de définir l'AF maître $(\rightharpoonup)$ comme $(\rightleftharpoons) \cup (\Rightarrow)$.

Le nouveau problème devient donc d'obtenir un algorithme qui, étant donné un AF $\langle Arg, \rightleftharpoons \rangle$, une relation $(\Rightarrow)$ vérifiant les quatre conditions citées précédemment et une relation d'attaque égale à $(\rightleftharpoons \cup \Rightarrow)$, permet d'exposer une relation d'ordre des préférences $\succcurlyeq$ entre les différents arguments et donc de construire un AVAF qui rationalise l'AF.

La solution à ce problème est facile, il suffit de définir la relation d'ordre des préférences comme étant égale à $(\Rightarrow)$, dans le sens où $(A,B) \in (\Rightarrow) \Leftrightarrow val(B) \succ val(A)$. Si A et B ne sont pas reliés dans $\Rightarrow$, val(B) et val(A) ne sont pas comparables.

- **<u>Preuve que le la rationalisation donne bien un AVAF :</u>**

Tout d'abord, $\succcurlyeq$ est bien une relation d'ordre car $(\Rightarrow)$ est acyclique, d'après la condition (ii).

On souhaite montrer que toutes les attaques de $(\rightleftharpoons)$ sont présentes dans l'AVAF obtenu, et que toutes les attaques de $(\Rightarrow)$ n'y sont pas.

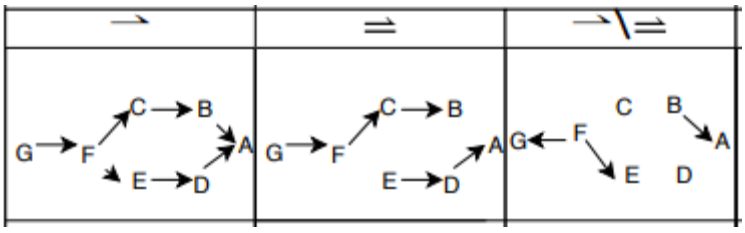
Premièrement, toutes les attaques (A,B) de $(\Rightarrow)$ ne sont effectivement pas dans l'AVAF obtenu car on a : $val(B) \succ val(A)$.

De plus, toutes les attaques (A,B) de $(\rightleftharpoons)$ sont effectivement dans l'AVAF obtenu car, si par l'absurde on avait $val(B) \succ val(A)$, alors cela voudrait dire que $(\Rightarrow)$ possède un chemin de A vers B par construction de la relation d'ordre, ce qui contredit (iii).

Exemple:

On applique l'algorithme sur les AF de l'agent $(\rightleftharpoons)$ et l'AF maître $(\rightharpoonup)$ représentés ci-dessous.

La relation d'ordre des préférences $(\succcurlyeq)$ est égale à $(\rightharpoonup \backslash \rightleftharpoons)$.



Le graphe ($\rightleftharpoons$) est obtenu à partir de ($\rightharpoonup$) en retirant tous les arcs (A,B) telles que $val(B) \succ val(A)$. C'est évident, car ($\rightleftharpoons$) est effectivement le graphe obtenu à partir de ($\rightharpoonup$) et en retirant tous les arcs de ($\rightharpoonup \backslash \rightleftharpoons$).

Le code python permettant d'obtenir la relation d'ordre est le suivant :

```
def prop2(avaf, af) :
  return (np.subtract(af, avaf))
```

4.3 Rationalisation dans le cas où l'assignation des valeurs est fixée

On souhaite ici créer un algorithme pour rationaliser un AF dans le cas de la proposition 2.

On fait le même changement de variable que dans la partie précédente, ($\Rightarrow$) :=($\rightharpoonup \backslash \rightleftharpoons$).

Si on traduit les conditions de la proposition 2, cela donne :

i. ($\rightleftharpoons$) $\subset$ ($\rightleftharpoons \cup \Rightarrow$) (évident),

ii. La relation $\cup_{A(\rightharpoonup \backslash \rightleftharpoons)B} \{(val(A), val(B))\}$ est acyclique,

iii. ($\rightleftharpoons$) $\cap$ $(\Rightarrow)^{+}_{val} = \emptyset$.

On ajoute une condition supplémentaire, pour les mêmes raisons que celles énoncées dans la section précédente :

iv. ($\rightleftharpoons$) $\cap$ ($\Rightarrow$) = $\emptyset$.

L'objectif est d'obtenir un algorithme permettant, étant donné un AF $\langle Arg, \rightleftharpoons \rangle$, une relation ($\Rightarrow$), une assignation de valeurs $\langle Val, val \rangle$ vérifiant les conditions (i), (ii) et (iii), (iv), et une relation d'attaque valant ($\rightleftharpoons \cup \Rightarrow$), d'exposer une relation d'ordre des préférences ($\succcurlyeq$) entre les différents arguments permettant de construire un AVAF qui rationalise l'AF.

Pour toute relation $\mathcal{R}$, on note $\mathcal{R}_{val}$ la relation telle que x $\mathcal{R}_{val}$ y si et seulement si il existe A et B, tels que x = $val(A)$, y = $val(B)$ et A $\mathcal{R}$ B.

Alors il suffit de prendre ($\rightleftharpoons_{val}$) et ($\Rightarrow_{val}$), d'utiliser l'algorithme de rationalisation de la proposition 1 sur ces deux nouvelles relations, ce qui nous donne immédiatement l'ordre souhaité : ($\Rightarrow_{val}$), dans le sens où (A,B) $\in \Rightarrow_{val}$ <=> $val(B) \succ val(A)$. Si A et B ne sont pas reliés dans $\Rightarrow_{val}$, val(B) et val(A) ne sont pas comparables.

- **<u>Démonstration de l'algorithme :</u>**

Il faut démontrer deux choses : premièrement, que les relations ($\rightleftharpoons_{val}$) et ($\Rightarrow_{val}$) remplissent les conditions de la proposition 1 (sous sa nouvelle forme), et deuxièmement, que le la rationalisation nous donne bien un AF de la forme $\langle Arg, \rightleftharpoons \rangle$.

On définit les notions suivantes :

- Chemin de valeurs : suite de valeurs <v_1, …, v_k, …, v_n>, telle que pour toutes valeurs consécutives v_i et v_{i+1}, il existe un arc (A, B) telle que $val(A)$= v_i et $val(B)$= v_{i+1}.
- Cycle sur les valeurs : même chose, mais $v_n = v_1$.

Commençons par prouver que si la proposition 2 est applicable sur les relations (⇌) et (⇏), alors la proposition 1 est applicable sur les relations ($\rightleftharpoons_{val}$) et ($\not\Rightarrow_{val}$):

(i) est triviale.

(ii) : ($\not\Rightarrow_{val}$) est bien acyclique si et seulement si (⇏) n'a pas de cycle sur les valeurs.

(iii) : si on a un chemin sur les valeurs (A, B, C, …, Z) dans (⇏) et un arc reliant les valeurs A et Z dans ⇌, alors on a un chemin (A, B, C, …, Z) dans ($\not\Rightarrow_{val}$) et un arc (A,Z) dans ($\rightleftharpoons_{val}$), et inversement.

(iv) : ($\rightleftharpoons_{val}$) et ($\not\Rightarrow_{val}$) n'ont pas de sommets en commun car sinon, on aurait deux valeurs distinctes reliées par un arc de (⇏) et un arc de (⇌), ce qui contredit (iii).

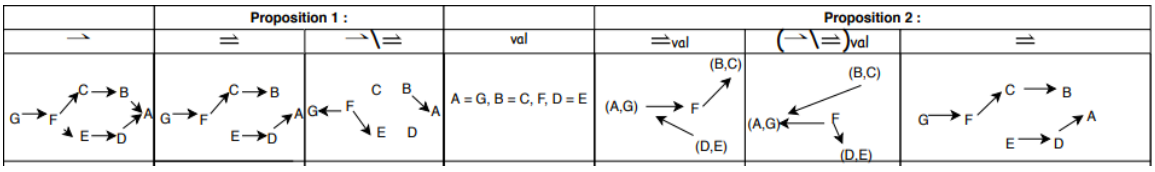
- **<u>Preuve que le la rationalisation donne bien un AVAF :</u>**

On souhaite montrer que toutes les attaques de (⇌) sont présentes dans l'AVAF obtenu, et que toutes les attaques de (⇏) n'y sont pas.

- Toutes les attaques (A,B) de (⇏) ne sont effectivement pas dans l'AVAF obtenu car l'attaque $(val(A), val(B))$ est présente dans ($\not\Rightarrow_{val}$), donc on a $val(B) \succ val(A)$.
- Toutes les attaques (A,B) de (⇌) sont effectivement dans l'AVAF obtenu car, si par l'absurde on avait $val(B) \succ val(A)$, alors cela voudrait dire que ($\not\Rightarrow_{val}$) possède un chemin de $val(A)$ vers $val(B)$ par construction, donc que (⇏) possède un chemin de valeurs entre A et B, et sachant que (⇌) a une attaque de A vers B, cela contredit (iii).

Exemple :

Dans cet exemple, les graphes (⇀) et (⇌), et les valeurs val sont donnés. A partir de ces données, on peut calculer ($\rightleftharpoons_{val}$) et également $(⇀ \setminus ⇌)_{val}$, qui est notre nouvelle relation d'ordre. Puis on retire toutes les arêtes (A,B) de (⇀) telles que $val(B) \succ val(A)$, et on obtient bien (⇌).



Le code python permettant d'obtenir la relation d'ordre est donc :

```
def prop3(avaf, af, val) :
  return(div_mat_class(np.subtract(af, avaf), val))
```

La fonction div_mat_class divise une matrice par une relation d'équivalence, donnée sous forme de liste de listes, chaque liste regroupant des arguments équivalents.

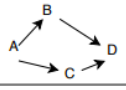
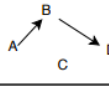
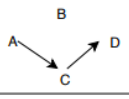
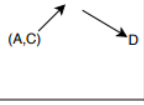
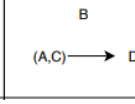
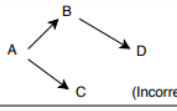
4.4 Exemples

Dans ce qui suit nous illustrons quelques exemples sur lesquels nous avons testé nos fonctions.

Pour chaque exemple, la relation d'attaque maître ($\rightharpoonup$) et l'AF de l'agent ($\rightleftharpoons$) sont déterminées arbitrairement, ainsi que la fonction val, et on peut en déduire ($\rightleftharpoons_{val}$) et ($\rightharpoonup \setminus \rightleftharpoons)_{val}$. Ensuite l'AVAF est calculé grâce à la méthode de la proposition 2, et comme il a été prouvé précédemment, le ($\rightleftharpoons$) obtenu par notre algorithme de rationalisation est systématiquement le même que celui que le ($\rightleftharpoons$) défini au départ.

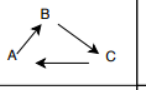
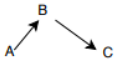
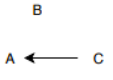
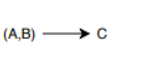
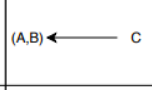
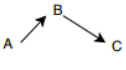
a) Cas pour lequel la rationalisation n'est pas assurée :

Dans cet exemple, le choix de val ne permet pas de rationaliser, car il y a un cycle valeurs entre A et C. Il est donc normal que le ($\rightleftharpoons$) obtenu ne soit pas le même que le ($\rightleftharpoons$) défini au départ.

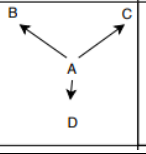
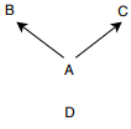
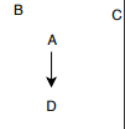
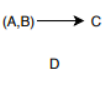
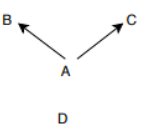
	Proposition 1 :			Proposition 2 :		
$\rightharpoonup$	$\rightleftharpoons$	$\rightharpoonup \setminus \rightleftharpoons$	val	$\rightleftharpoons_{val}$	$(\rightharpoonup \setminus \rightleftharpoons)_{val}$	$\rightleftharpoons$
			A = C, B, D (Incorrect car cycle sur les valeurs)			 (Incorrect)

c) Cycle :

Nous avons essayé des cycles de longueur paire et impaire, le résultat est le même.

	Proposition 1 :			Proposition 2 :		
$\rightharpoonup$	$\rightleftharpoons$	$\rightharpoonup \setminus \rightleftharpoons$	val	$\rightleftharpoons_{val}$	$(\rightharpoonup \setminus \rightleftharpoons)_{val}$	$\rightleftharpoons$
			A = B, C			

d) Etoile :

	Proposition 1 :			Proposition 2 :		
$\rightharpoonup$	$\rightleftharpoons$	$\rightharpoonup \setminus \rightleftharpoons$	val	$\rightleftharpoons_{val}$	$(\rightharpoonup \setminus \rightleftharpoons)_{val}$	$\rightleftharpoons$
			A = B, C, D			

Nous avons également étudié des graphes disjoints (cela revient à appliquer les transformations sur chaque composante connexe séparément).

Les exemples ont été vérifiés informatiquement, en codant les conditions des propositions 1 et 2.

Par exemple, les 3 conditions de la proposition 1 ont été codées ainsi :

La fonction setone : met tous les coefficients non nuls d'une matrice à 0,

La fonction égal : test d'égalité entre matrices,

La fonction closure : retourne la fermeture transitive d'un graphe,

La fonction istherecycle : teste la présence d'un cycle,

La fonction intersect : teste si deux matrices ont un coefficient en commun.

```
def condition1(a,b):
  for i in range(len(a)) :
    for j in range(len(b)):
      if((b[i,j]==1) & (a[i,j]!=1)):
        return 1
  return 0 #retourne 0 quand tous les arcs de b sont inclus dans a



def condition2(a,b):
  """ vérifie que (=/-) est acyclique retourne 1 s'il n'y a pas de
cycle """
  c= np.subtract(a,b)

  return istherecycle(c)

def condition3(a,b):
  n = closure (np.subtract(a,b))

  return (intersect(n,b))  #retourne 0 quand l'intersection = un
ensemble vide
```

Pour la proposition 2 : (ma désigne L'AF maître $\rightharpoonup$ et af désigne la relation de défaite de l'agent $\Rightharpoonup$)

```
def associe_ma_val(ma,af,val): # divise la matrice (ma-af) par la
relation d'équivalence val
  mat= np.subtract(ma,af)
  new_mat = np.matlib.zeros((len(val),len(val)))
  for i  in range(len(mat)):
    for j in range(len(mat)):
```

```
      if(mat[i,j]==1):
        new_mat[int(val[i]),int(val[j])]=1
  return new_mat

def condition1(ma,af):
  for i in range(len(ma)) :
    for j in range(len(ma)):
      if((af[i,j]==1) & (ma[i,j]!=1)):
        return 0
  return 1

def condition2(ma,af,val):
  return not(istherecycle(associe_ma_val(ma,af,val)))

def condition3(ma,af,val):
  mat=closure (associe_ma_val(ma,af,val))
  return not(does_intersect(af,mat))
```

Nous avons également généré des graphes générés aléatoirement puis nous avons testé les conditions (ii) et (iii) de la proposition 1. sur les matrices de ces graphes.

La méthode de génération est la suivante : On crée la matrice de la relation $\rightleftharpoons$ en mettant des coefficients à 1 aléatoirement jusqu'à atteindre size coefficients, puis on crée la matrice de la relation $\rightharpoonup$ en mettant des coefficients à 1 aléatoirement jusqu'à atteindre size2 coefficients.

Voici le code :

```
def add_aleat(m) : #met a 1 un coefficient aleatoire de la matrice
m
  h = random.randint(0,len(m) - 1)
  l = random.randint(0,len(m) - 1)
  if m[l,h] != 1 and l != h :
    m[h,l] = 1
    return(1)
  return(0)

def genere_af(size) :
  j = 0
  af = np.matlib.zeros((size,size))
  while(j < size) :
    j += add_aleat(af)
  return(af)

def complete_af_avaf(af, size2) :
  j = 0
  avaf = np.matrix.copy(af)
  while(j < size2) :
    j += add_aleat(avaf)
  return(avaf)
```

4.5 Conclusion

Nous avons donc exposé des algorithmes permettant de rationaliser un AF en un AVAF, dans le cas où la seule contrainte est la relation d'attaque de L'AF maître qui est fixée, ainsi que dans le cas où la relation d'attaque de l'AF maître $\rightleftharpoons$ est fixée mais également l'assignation de valeurs aux arguments $\langle Val, val \rangle$. Dans les deux cas, la solution se fait en temps et en espace polynomial.

CONCLUSION GÉNÉRALE

5. Conclusion Générale

Nous avons décrit dans ce rapport le concept de rationalisation de profil de cadres d'argumentation abstraite. Nous nous sommes essentiellement basés sur le cadre argumentatif à un seul agent et nous avons montré qu'il pouvait être rationalisé, c'est-à-dire, attribuer des préférences à cet agent permettant de justifier la relation d'attaque entre différents arguments de son point de vue et ce en prenant en compte différentes contraintes. De plus, cette rationalisation se fait en temps polynomial.

RÉFÉRENCES

6. Références

[1] S.Airiau, E.Bonzon, U.Endriss, N.Maudet, J.Rossit, (2016), Rationalisation of Profiles of Abstract Argumentation Frameworks, AAMAS 350-357.

[2] S.Airiau, E.Bonzon, U.Endriss, N.Maudet, J.Rossit, (2017), Rationalisation of Profiles of Abstract Argumentation Frameworks: Characterisation and Complexity. J. Artif. Intell. Res. 60: 149-177.

[3] Bench-Capon, T. J. M, (2003)

, Persuasion in practical argument using value-based argumentation frameworks. Journal of Logic and Computation, 13(3), 429–448.

[4] Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. Artificial Intelligence, 77(2), 321–358.

GLOSSAIRE

7. Glossaire

AF	Argumentation Framework
AVAF	Audience-specific Value-based Argumentation Framework