Évaluation graduelle d'arguments

C. Cayrol
M.C. Lagasquie-Schiex

Décembre 2001

Rapport IRIT 2001-18-R

Révision numéro 1 de Mars 2002

Résumé

L'argumentation est basée sur l'échange et l'évaluation d'arguments interagissant. Dans ce document, et en utilisant le cadre de travail proposé par [Dun95], nous nous sommes consacrées à l'analyse des relations d'attaque entre arguments, et à leur impact dans la définition d'une évaluation graduelle d'arguments (ainsi, cette évaluation prendra en compte les attaquants, les attaquants des attaquants et ainsi de suite). Cette étude est menée au travers d'un ensemble de cas et débouche alors sur diverses approches.

Tout d'abord une première approche totalement intuitive (l'approche 0) dont l'analyse comparative avec une approche existante (celle de [BH01]) nous conduit à préciser les divers principes qui sous-tendent l'évaluation d'arguments basée sur l'attaque entre arguments.

On définit alors deux nouvelles approches reposant sur ces principes (l'approche 1 dite "locale" et l'approche 2 dite "globale"). Puis on effectue une comparaison entre ces approches et les approches existantes ([BH01], [JV99]).

Remerciements

Nous tenons à remercier Mr Philippe BESNARD pour sa relecture extrêmement approfondie et constructive de la version précédente de ce rapport.

Grâce à lui, nous avons pu corriger et enrichir ce rapport sur de nombreux points.

Il nous a aussi suggéré de nombreuses pistes pour poursuivre ce travail qui feront l'objet d'autres rapports.

Table des matières

Chapitre 1

Introduction

[Dun95] a montré que le cadre de l'argumentation constitue un outil puissant permettant aussi bien l'étude de nombreux systèmes formels de raisonnement de sens commun que la définition d'une sémantique pour les programmes logiques.

L'argumentation est basée sur l'échange et l'évaluation d'arguments concernant une assertion donnée. On trouve des applications notamment dans le domaine juridique, dans les systèmes d'aide à la prise de décision collective ou d'aide à la négociation. La caractéristique fondamentale d'un système d'argumentation est la présence d'interactions et notamment de relations de contrariété entre les arguments avancés. Si l'argument prend par exemple la forme d'une preuve logique, on peut avancer des arguments pour une proposition et des arguments contre cette proposition, *i.e.* pour la proposition contraire.

Le processus d'argumentation comporte donc une étape d'évaluation de la force relative des arguments en présence, l'objectif étant de sélectionner les arguments les plus acceptables. On distingue :

– une évaluation dite *intrinsèque* qui évalue un argument indépendamment des interactions avec les autres arguments. On peut ainsi exprimer à quel point l'argument augmente la confiance en l'assertion qu'il supporte. Cette évaluation peut prendre différentes formes [KAEF95], en particulier une valeur numérique à interprétation probabiliste comme dans [Par97], ou être simplement représentée par une relation de préférence sur l'ensemble des arguments, comme dans [PS97] et [AC98].

– une évaluation *des interactions* selon laquelle un argument est évalué en fonction de ses "contrariants", des contrariants de ses contrariants (ses "défenseurs"), ... [1]

Évaluation intrinsèque et prise en compte des interactions ont très souvent été utilisées séparément, selon les applications envisagées. On trouve cependant quelques travaux qui proposent une combinaison de ces deux critères (voir par exemple [AC98]).

La prise en compte des interactions a le plus souvent conduit à des évaluations binaires : l'argument est accepté ou non. Ces évaluations peuvent être définies de manière individuelle, par exemple par des procédures d'étiquetage [JV99], ou de manière globale en donnant des ensembles d'arguments conjointement acceptés [Dun95]. Récemment, [BH01] a proposé une évaluation graduelle des interactions dans le cadre particulier d'arguments déductifs. L'ensemble des contrariants d'un argument est représenté par un arbre. La valeur calculée pour cet arbre représente la force relative de l'argument racine de l'arbre.

Notre objectif est de proposer et de discuter des principes généraux qui régissent la prise en compte des contrariants et des défenseurs pour une évaluation graduelle des arguments, puis de définir plusieurs modèles d'évaluation associés, dans un système d'argumentation abstrait.

Nous nous plaçons dans le cadre défini par [Dun95] : celui d'un système d'argumentation constitué d'un ensemble d'arguments et d'une relation binaire sur cet ensemble. On utilisera une représentation graphique des systèmes d'argumentation.

Nous cherchons dans ce document à comparer des arguments entre eux, essentiellement en fonction de leur position dans le graphe des contrariétés, mais aussi de leur valeur intrinsèque. Pour cela, nous calculons la *valeur d'un argument* issue du graphe des contrariétés et complétée/mise à jour éventuellement par la valeur intrinsèque de l'argument ; puis, nous comparons les arguments au travers de ces valeurs.

Nous mettons ici l'accent sur la gradualité de la fonction d'évaluation qui devrait nous permettre d'obtenir une relation d'ordre entre arguments plus riche que celles proposées en général dans la littérature (le plus

[1] Nous n'étudierons ici que les interactions dues à la notion de contrariété entre arguments ! Il peut exister d'autres types d'interaction (par exemple des arguments qui se renforceraient au lieu de s'attaquer).

souvent ces relations utilisent 3 valeurs – accepté, rejeté, non décidé).
La section 2 page ci-contre présente les notations et définitions relatives à un système d'argumentation abstrait et sa représentation sous forme de graphe des contrariétés.
Ensuite, dans un premier temps (voir section 3 page 7), nous proposons une approche simple (l'approche 0) qui définit *intuitivement* une relation d'ordre entre les valeurs en fonction de ce qui "nous semble correct et judicieux" par rapport à la relation de contrariété liant les arguments. Et nous cherchons à intégrer à cette relation la prise en compte d'une valeur intrinsèque. Cette approche est tout à fait informelle mais présente deux intérêts majeurs :

- proposer une évaluation basée sur les contrariétés entre arguments à partir de laquelle on peut essayer de rajouter la prise en compte d'une évaluation intrinsèque des arguments (ce travail d'agrégation de deux évaluations distinctes étant ensuite valable quelle que soit l'évaluation basée sur les contrariétés utilisée !) ;
- énumérer quelques exemples significatifs d'interactions dues à la contrariété entre arguments.

Cette approche intuitive sera analysée par comparaison avec une approche existante (celle de [BH01]). Un certain nombre de défauts est alors mis à jour, aussi bien pour l'approche 0 (essentiellement dus à l'aspect "intuitif" de l'approche - l'intuition pouvant ne pas être la même suivant les lecteurs !) que pour l'approche de [BH01].
Cette comparaison débouche alors sur 2 nouvelles approches cherchant à définir l'évaluation des arguments de manière axiomatique par la mise en œuvre de certains principes :

- l'une (l'approche 1 – voir section 4 page 21) s'appuie sur des principes ne prenant en compte que l'aspect "local" de la relation de contrariété (donc uniquement les attaquants directs de l'argument dont on cherche la valeur, les autres arguments n'intervenant qu'au travers des valeurs des arguments qu'ils attaquent) ;
- l'autre (l'approche 2 – voir section 5 page 25) à base de valeurs tuplées s'appuyant sur une prise en compte "globale" de la relation de contrariété (tous les arguments liés directement ou indirectement à l'argument étudié sont pris en compte).

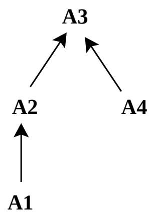
Puis, en section 6 page 45, on comparera ces 2 approches entre elles et avec d'autres approches de la littérature.

Chapitre 2

Notations et définitions

Nous nous plaçons dans le cadre abstrait défini par [Dun95]. Soit le *système d'argumentation* $<\mathcal{A}, \mathcal{R}>$, $\mathcal{A}$ étant un ensemble d'arguments et $\mathcal{R}$ une relation binaire sur $\mathcal{A}$ appelée *relation de contrariété* : soient A_i et $A_j \in \mathcal{A}$, $A_i \mathcal{R} A_j$ signifiera que A_j est contrarié par A_i, ou que A_i contrarie A_j (aussi noté $(A_i, A_j) \in \mathcal{R}$). Nous ne préciserons pas davantage le format des arguments, ni la relation de contrariété.

Notations : Soit $A \in \mathcal{A}$, l'ensemble $\{A_i \in \mathcal{A} | A_i \mathcal{R} A\}$ est noté $\mathcal{R}^-(A)$ et l'ensemble $\{A_i \in \mathcal{A} | A \mathcal{R} A_i\}$ est noté $\mathcal{R}^+(A)$.
$<\mathcal{A}, \mathcal{R}>$ définit un graphe orienté $\mathcal{G}$ (dit *graphe des contrariétés*).
Par exemple, le système $<\mathcal{A} = \{A_1, A_2, A_3, A_4\}, \mathcal{R} = \{(A_2, A_3), (A_4, A_3), (A_1, A_2)\}>$ définit le graphe $\mathcal{G}$ suivant :



Définition 1 *Un argument $A \in \mathcal{A}$ tel que $\mathcal{R}^-(A) = \emptyset$ sera une* feuille du graphe des contrariétés *défini par $<\mathcal{A}, \mathcal{R}>$.*

Nous utilisons ensuite la notion classique de "chemin dans un graphe" appliquée à $\mathcal{G}$:

Définition 2 *Dans le graphe des contrariétés $\mathcal{G}$, un* chemin de A vers B *est une suite d'arguments $A_1 - \ldots - A_n$ telle que :*
- $A = A_1$,
- $A_1 \mathcal{R} A_2$,
- ...
- $A_{n-1} \mathcal{R} A_n$,
- $A_n = B$.

Ce chemin sera noté $\mathcal{C}(A, B)$[1]*. La* longueur de ce chemin *est alors $n - 1$ (le nombre d'arêtes constituant ce chemin) et sera notée $l_{\mathcal{C}(A,B)}$*[2]*. On dira aussi que $l_{\mathcal{C}(A,B)}$ est la distance entre A et B.*

Définition 3 *Soient 2 chemins $\mathcal{C}(A_1, A_n) = A_1 - \ldots - A_n$ et $\mathcal{C}(B_1, B_m) = B_1 - \ldots - B_m$ notés respectivement $\mathcal{C}_A$ et $\mathcal{C}_B$.*
Ces deux chemins seront dits dépendants *ssi $\exists A_i \in \mathcal{C}_A$, $\exists B_j \in \mathcal{C}_B$ tel que $A_i = B_j$.* Indépendants *sinon.*
Ces deux chemins seront dits racine-dépendants *ssi $A_n = B_m$ et $\forall A_i \neq A_n \in \mathcal{C}_A$, $\nexists B_j \in \mathcal{C}_B$ tel que $A_i = B_j$.*

Nous allons utiliser les notions d'attaque et de défense directe ou indirecte inspirées par [Dun95] mais légèrement modifiées par nos soins[3]. On a donc :

[1]S'il existe plusieurs chemins entre A et B, on les notera en utilisant un indice supplémentaire : $\mathcal{C}(A,B)_1 \ldots \mathcal{C}(A,B)_n$.

[2]Ici aussi, s'il y a plusieurs chemins entre A et B, il peut y avoir plusieurs longueurs repérées par un indice supplémentaire : $l_{\mathcal{C}(A,B)_1} \ldots l_{\mathcal{C}(A,B)_n}$.

[3]Les notions définies dans [Dun95] sont celles d'attaque et de défense indirecte, sachant que pour [Dun95] un attaquant (resp. défenseur) direct est aussi un attaquant (resp. défenseur) indirect, ce qui n'est pas le cas ici.

Définition 4 *Soit $A \in \mathcal{A}$, les* attaquants directs *de A sont les éléments de $\mathcal{R}^-(A)$.*

Définition 5 *Soit $A \in \mathcal{A}$, les* défenseurs directs *de A sont les attaquants directs des éléments de $\mathcal{R}^-(A)$.*

Définition 6 *Soit $A \in \mathcal{A}$, les* attaquants indirects *de A sont les éléments A_i définis par :*
$\exists \mathcal{C}(A_i, A)$ tel que $l_{\mathcal{C}(A_i,A)} = 2k+1$, avec $k \geq 1$.

Définition 7 *Soit $A \in \mathcal{A}$, les* défenseurs indirects *de A sont les éléments A_i définis par :*
$\exists \mathcal{C}(A_i, A)$ tel que $l_{\mathcal{C}(A_i,A)} = 2k$, avec $k \geq 2$.

On dira plus généralement que, si l'argument A est un attaquant (direct ou indirect) de l'argument B, alors *A attaque B* (ou *B est attaqué par A*). De même, si l'argument A est un défenseur (direct ou indirect) de l'argument B, alors *A défend B* (ou *B est défendu par A*).

Définition 8 *Soit $A \in \mathcal{A}$, une* branche d'attaque *(resp.* de défense*) pour A est un chemin dans $\mathcal{G}$ d'une feuille vers A de longueur impaire (resp. paire). On dira alors que A est* racine d'une branche d'attaque *(resp.* de défense*).*

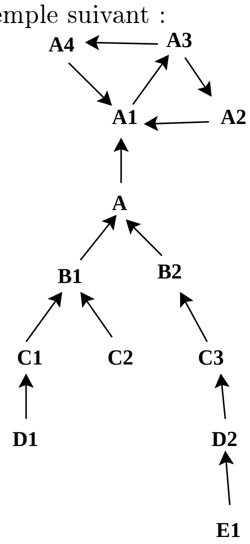
Remarquons que cette définition concerne uniquement les arguments du graphe qui sont "atteignables" par une des feuilles du graphe (chaque feuille représentant un "point d'entrée" dans le graphe). En fait, on l'utilisera aussi par abus de langage pour des arguments appartenant à des circuits et pas forcément reliés à des feuilles du graphe (voir la section 5.2 page 26).

Définition 9 *Un* circuit *est un chemin $\mathcal{C} = A_1 - \ldots - A_n$ tel que $A_n \mathcal{R} A_1$ et $\nexists A_i, A_j \in \mathcal{C}, i \neq j$ tel $A_i = A_j$. La longueur de ce circuit est alors de n*[4]*.*

Cette définition d'un circuit correspond à la définition d'un circuit élémentaire en théorie des graphes (ne contient pas 2 arcs ayant la même origine, ou le même but).

Définition 10 *Deux circuits $\mathcal{C}_A = A_1 - \ldots - A_n$ et $\mathcal{C}_B = B_1 - \ldots - B_m$ sont* interconnectés *si et seulement si $\exists i \in [1, n], \exists j \in [1, m]$ tels que $A_i = B_j$.*

Toutes ces notions sont illustrées sur l'exemple suivant :



Sur ce graphe $\mathcal{G}$, on a donc (entre autres) :
- un chemin de C_2 vers A de longueur 2 ($C_2 - B_1 - A$),
- 2 circuits $A_1 - A_3 - A_2$ et $A_1 - A_3 - A_4$, chacun de longueur 3 (remarquons que $A_1 - A_3 - A_2 - A_1 - A_3 - A_4$ n'est pas un circuit d'après notre définition),
- les chemins $D_1 - C_1 - B_1$ et $C_3 - B_2 - A$ sont indépendants, alors que $D_1 - C_1 - B_1 - A$ et $C_3 - B_2 - A$ sont racine-dépendants et que $D_1 - C_1 - B_1 - A$ et $C_2 - B_1 - A$ sont dépendants,
- les deux circuits cités précédemment sont interconnectés (en A_1 et A_3),
- D_1, C_2, E_1 sont les feuilles de $\mathcal{G}$,
- $D_1 - C_1 - B_1 - A$ est une branche d'attaque pour A, alors que $C_2 - B_1 - A$ est une branche de défense pour A,
- B_1 et B_2 sont les deux attaquants directs de A,
- C_1, C_2 et C_3 sont les trois défenseurs directs de A,

[4]Alors que la longueur du chemin est de $n - 1$!

- D_1 et D_2 sont les deux attaquants indirects de A,
- E_1 est le seul défenseur indirect de A.

Notations Étant donné que nous cherchons à comparer des arguments, nous posons donc les notations suivantes qui seront utilisées tout le long de ce document : soient A et B deux arguments
- A meilleur que B sera noté $A \succeq B$,
- puis, nous définissons A strictement meilleur que B, noté $A \succ B$, par : $A \succ B$ ssi $A \succeq B$ et NON $B \succeq A$ (remarquons que l'on a alors $A \succ B \Rightarrow A \succeq B$),
- et enfin nous avons A et B sont aussi bons l'un que l'autre, noté $A \sim B$, défini par : $A \sim B$ ssi $A \succeq B$ et $B \succeq A$.

La relation $\sim$ donnera lieu à plusieurs interprétations (voir section 5 page 25).

Chapitre 3

Approche 0 : un ordre intuitif entre arguments

Dans cette approche, on part d'une hypothèse forte :

Règle 1 (Équivalence structurelle) *2 arguments résultant de la même série d'attaque et de défense ont la même valeur (ils sont "racines" de la même* structure d'arbre/graphe *de contrariétés)*

et on cherche intuitivement à comparer 2 valeurs d'arguments étant données la relation de contrariété qui les lie aux autres arguments et la valeur intrinsèque des arguments.

Après les exemples, on énumérera les autres règles utilisées.

On va aussi procéder en 2 étapes :
- d'abord la prise en compte uniquement du graphe des contrariétés,
- puis rajout de la prise en compte des valeurs intrinsèques à chaque argument.

D'autre part, nous considèrerons ici que nous avons la définition suivante :

Définition 11 *Soit $(\mathcal{A}, \mathcal{R})$ un système d'argumentation. Soient V un ensemble ordonné de valeurs et v une application de $\mathcal{A} \to V$. Soient A et B deux arguments ayant pour valeurs respectives $v(A)$ et $v(B)$, la relation d'ordre entre les valeurs induit une relation d'ordre entre les arguments : si $v(A) \geq v(B)$, on dira que A est meilleur que B (noté $A \succeq B$ d'après la section 2 page 3).*

3.1 Les exemples à traiter

Nous avons tout d'abord cherché à répertorier divers exemples significatifs d'interactions entre arguments (l'interaction étant ici due à la relation de contrariété entre arguments).

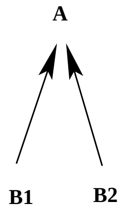
Nous avons ainsi identifié au moins 18 cas possibles répartis en trois classes distinctes :
- les cas correspondant à un graphe des contrariétés sans circuit :

0. un seul argument non contrarié.

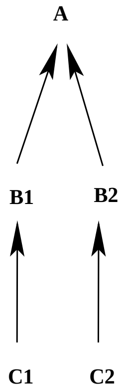
1. deux arguments, l'un contrariant l'autre :

2. trois arguments formant un unique chemin de contrariété de longueur 2 :

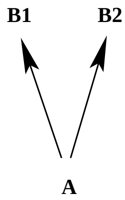
3. trois arguments, les deux premiers contrariant le troisième :



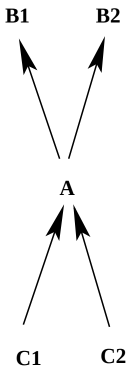
4. cinq arguments formant 2 chemins de contrariété racine-dépendants :



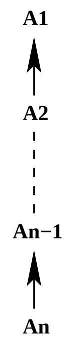
5. trois arguments, le premier contrariant les deux autres :



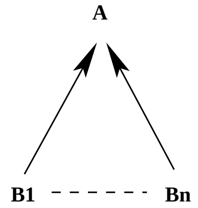
6. cinq arguments formant 2 chemins de contrariété ayant un point “d’intersection” autre que l’extrémité des chemins :



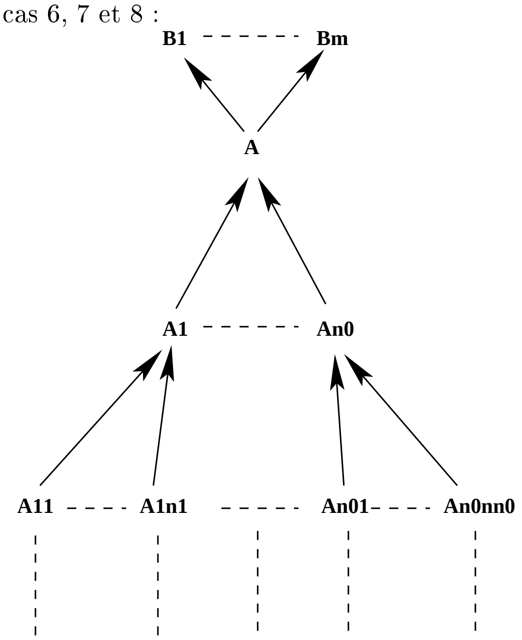
7. n arguments formant un chemin de contrariété de longueur $n-1$:



8. n arguments attaquant un même argument :

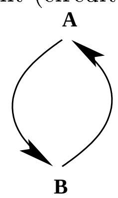


9. une généralisation des cas 6, 7 et 8 :

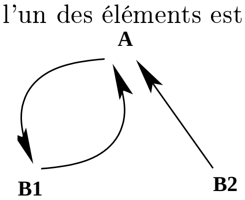


– les cas correspondant à un graphe des contrariétés avec circuit pair :

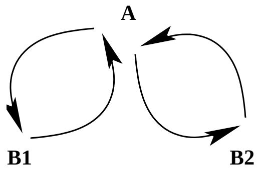
10. deux arguments s'attaquant mutuellement (circuit pair de longueur 2) :



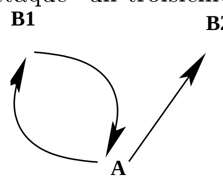
11. un circuit pair de longueur 2 dont l'un des éléments est attaqué par un troisième argument :



12. deux circuits pairs de longueur 2 s'attaquant par l'intermédiaire d'un élément commun :

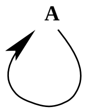


13. un circuit pair de longueur 2 qui attaque[1] un troisième argument :

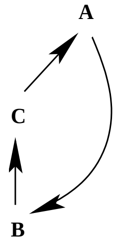


– les cas correspondant à un graphe des contrariétés avec circuit impair :

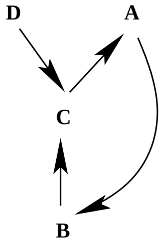
14. un argument auto-contrarié (qui s'attaque lui-même) formant ainsi un circuit impair de longueur 1 :



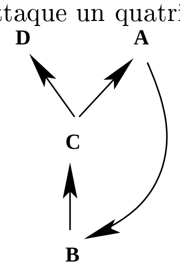
15. un circuit impair de longueur 3 :



16. un circuit impair de longueur 3 dont un des éléments est attaqué par un quatrième argument :



17. un circuit impair de longueur 3 qui attaque un quatrième argument :



Certains exemples seront rajoutés au cours du document en fonction des circonstances.

Remarque : il est tout à fait évident que certains de ces exemples n'ont pas forcément de signification dans le cadre pur de l'argumentation (par exemple, le cas 14 peut sembler surprenant si on se place dans le contexte d'un dialogue entre deux agents qui argumentent : un des agents se contredit lui-même !). Toutefois, ce travail se voulant assez général, nous ne nous sommes pas restreints du point de vue des exemples traités et nous avons plutôt cherché à énumérer le maximum de structures de graphes possibles.

[1] Un cycle *attaque un argument A* si et seulement si un des éléments du cycle attaque A.

3.2 Prise en compte uniquement du graphe des contrariétés

La valeur associée à un argument A sera notée $v(A)$. La valeur M représente la valeur maximale admise.

On va présenter les divers cas suivant leur nature (sans circuit, avec circuit pair, avec circuit impair).

3.2.1 Les cas sans circuit

Numéro Cas	Contraintes sur les valeurs
0	$v(A) = M$
1	$v(B) = M$, $v(A) \leq v(B)$
2	$v(C) = M$, $v(B) \leq v(A) \leq v(C)$
3	$v(A) \leq v(B_1) = v(B_2) = M$
4	$v(B_1) = v(B_2) \leq v(A) \leq v(C_1) = v(C_2) = M$
5	$v(B_1) = v(B_2) \leq v(A) = M$
6	$v(A) \leq v(B_1) = v(B_2) \leq v(C_1) = v(C_2) = M$
7	Généralise les cas 1 et 2 : Si n est pair alors l'argument A_1 est au bout d'une unique branche d'attaque ; on a alors : $M = v(A_n) \geq v(A_{n-2}) \geq \ldots \geq v(A_2) \geq v(A_{n-1}) \geq \ldots \geq v(A_1)$ et sinon (n impair), l'argument A_1 est au bout d'une unique branche de défense ; on a alors : $M = v(A_n) \geq v(A_{n-2}) \geq \ldots \geq v(A_1) \geq v(A_{n-1}) \geq \ldots \geq v(A_2)$
8	$v(A) \leq v(B_i) = M, \forall i = 1 \ldots n$ Remarquons que si un des B_i n'est pas une feuille alors on risque de perdre la propriété ci-dessus (par exemple quand un des B_i est attaqué et non défendu – voir le cas 2).
9	Il s'agit d'une généralisation des cas 6, 7 et 8 et on ne sait pas quoi conclure : $v(B_j)?v(A)?v(A_i) \leq M, \forall i = 1 \ldots n_0$ La relation entre la valeur de A et celle de A_i dépend de la nature de la branche qui les relie (branche d'attaque ou de défense), de même, pour la relation entre A et les B_j. Donc, il y a un problème quand 2 branches de nature différente se rejoignent (voir l'exemple 18 en section 3.2.4 page suivante).

La principale remarque ici est l'importance de la nature des branches (attaque ou défense) aboutissant à l'argument pour le calcul des contraintes d'ordre sur les valeurs.

3.2.2 Les cas avec circuits pairs

Numéro Cas	Contraintes sur les valeurs
10	$v(B) = v(A) \leq M$
11	$v(A) \leq v(B_1) \leq v(B_2) = M$
12	$v(A) = v(B_1) = v(B_2) \leq M$
13	$v(B_2) \leq v(A) = v(B_1) \leq M$

Ici, on a des résultats qui peuvent paraître surprenants :
- dans le cas 10, le fait que $v(A) = v(B)$ peut paraître restrictif par rapport à d'autres approches comme celle de [JV99] (qui favorise un des deux arguments), mais n'oublions pas notre hypothèse de départ (voir règle 1 page 7) !
- dans le cas 12, bien que A ait 2 contrariants directs et que B_1 et B_2 n'en aient qu'un, on a quand même l'égalité entre les 3 valeurs ; cela vient du fait que A, B_1, B_2 sont tout à la fois des attaquants et des attaqués (donc application du cas 3 ET du cas 5, avec les contradictions que cela implique et qui sont résolues par un "forçage à l'égalité").

3.2.3 Les cas avec circuits impairs

Numéro Cas	Contraintes sur les valeurs
14	$v(A) \leq M$
15	$v(A) = v(B) = v(C) \leq M$ C'est une application du cas 7 (avec une chaîne attaque/défense dont le "début"[a] est à la fois un attaquant et un défenseur !) [a]C'est une image ! En fait, il n'y a pas de début à un circuit.
16	$v(A) = v(B) = v(C) \leq v(D) = M$ Même remarque que pour le cas 15. Seul D y échappe puisqu'il est situé "avant" le circuit.
17	$v(A) = v(B) = v(C) = v(D) \leq M$ Même remarque que pour le cas 15 mais qui s'étend cette fois à D puisqu'il est situé "après" le circuit.

3.2.4 Analyse des exemples

De cette analyse ressortent les règles suivantes :

Règle 2 (Pas d'attaque) *Tout argument non attaqué a pour valeur M.*

Règle 3 (Attaque sans défense) *Tout argument A attaqué et non défendu a pour valeur $v(A) \leq$ à la valeur de son attaquant (et donc $\leq M$).*

Règle 4 (Attaque avec défense) *Tout argument A attaqué et défendu (son défenseur n'étant pas attaqué) a pour valeur $v(A) \leq$ à la valeur de son défenseur (et donc $\leq M$) mais $\geq$ à la valeur de son attaquant.*

Règle 5 (Défense) *Tout argument A attaqué et défendu a pour valeur $v(A) \leq$ à la valeur de ses défenseurs.*

Règle 6 (Unique branche d'attaque) *Tout argument A ayant une unique branche d'attaque a pour valeur $v(A) \leq$ aux valeurs de chacun de ses défenseurs, elles-mêmes étant toutes $\leq$ aux valeurs de chacun des attaquants.*

Règle 7 (Unique branche de défense) *Tout argument A ayant une unique branche de défense a pour valeur $v(A) \leq$ aux valeurs de chacun de ses défenseurs et $\geq$ aux valeurs de chacun de ses attaquants.*

Propriété 1 (Existence d'une fonction d'évaluation issue du graphe des contrariétés) *Il existe toujours une fonction d'évaluation v vérifiant les schémas et les règles données ci-dessus.*

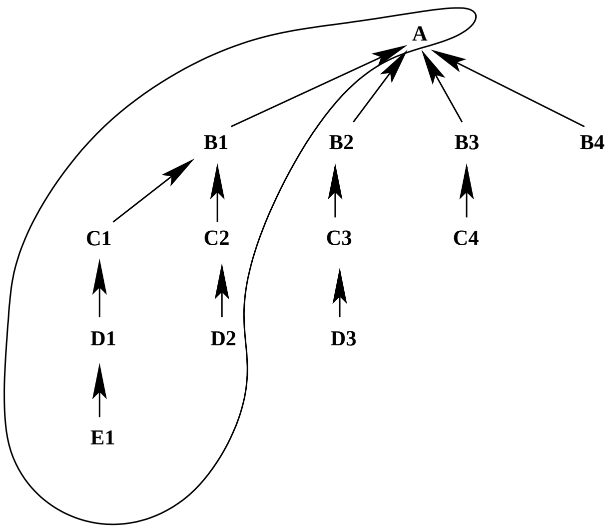
Preuve : Il suffit en effet de poser : $v(A) = M, \forall A$. □

Remarque : Les cas de circuit impair n'ont pas forcément un sens (si on considère des positions dans des arbres de jeu, on aurait ainsi un joueur qui se contredirait lui-même, qui chercherait à se faire perdre !).

Remarque : On constate rapidement (voir de nombreux cas) que cette approche présente au moins un inconvénient majeur qui est de "forcer à l'égalité" un certain nombre de valeurs pour respecter les contraintes d'ordre données précédemment.

Voyons cela sur un exemple supplémentaire numéroté 18 qui est un cas particulier du cas 9 déjà cité :

18. Cas d'un graphe des contrariétés mêlant branches d'attaque et de défense



Sur la partie entourée de ce graphe, appliquons nos contraintes d'ordre :
- $M = v(E_1) \geq v(C_1) \geq v(A) \geq v(D_1) \geq v(B_1)$,
- $M = v(D_2) \geq v(B_1) \geq v(C_2) \geq v(A)$ (ce qui est en contradiction avec la contrainte précédente à moins de poser $v(B_1) = v(C_2) = v(A) = v(D_1)$).

C'est justement ce phénomène qui gêne l'analyse plus complète du cas 9.

Attention, les règles 2 page ci-contre à 7 page précédente ne tiennent pas compte de la valeur intrinsèque des arguments.
Nous allons donc reprendre les différents cas cités précédemment en étudiant les sous-cas liés aux diverses valeurs intrinsèques possibles.

3.3 Prise en compte de la valeur intrinsèque des arguments

Nous noterons :
- $v_i(A)$ la valeur intrinsèque de l'argument A, c'est-à-dire donnée a priori à A,
- $v(A)$ la valeur de l'argument A issue du graphe des contrariétés,
- $v_c(A)$ la valeur combinée de l'argument A, c'est-à-dire celle qui tient compte de la valeur intrinsèque de A et de la valeur issue du graphe des contrariétés.

On considérera que :

$$v_c(A) = \Phi(v_i(A), v(A))$$

avec la fonction Φ restant à déterminer.
Passons en revue les différents cas possibles. En général, le nombre de sous-cas de chaque cas dépend du nombre n d'arguments apparaissant dans le dit-cas, puisqu'il s'agit de trouver tous les ordres possibles entre les v_i de ces n arguments (on a alors $n!$ possibilités). Toutefois, il arrive que ces cas puissent être vus comme la concaténation de 2 autres cas indépendants (c'est-à-dire qu'il n'y a pas de raison d'ordonner certains arguments par rapport à d'autres), cela réduit donc le nombre de sous-cas traités.
Ici aussi, notre analyse se fera suivant la nature des cas. Toutefois, cette analyse sera particulièrement appauvrie par le fait qu'en permanence, on se trouve devoir faire des choix qui complexifient énormément les résultats attendus. Dans la section 3.3.4 page 16, nous ne pourrons alors présenter que très peu de règles.

3.3.1 Les cas sans circuit

Numéro Cas	Contraintes sur les valeurs

0	$\boxed{1}$ seul sous-cas : $v_c(A) = g(v_i(A), v(A))$	
1	avec $\boxed{2}$ sous-cas possibles en entrée (2!) :	
	sous-cas 1 : $v_i(A) \leq v_i(B)$	$v_c(A) \leq v_c(B)$
	sous-cas 2 : $v_i(A) > v_i(B)$	2 contraintes possibles : soit $v_c(A) \leq v_c(B)$, soit $v_c(A) \geq v_c(B)$[a] [a]Suivant l'importance accordée à la valeur intrinsèque
2	avec $\boxed{6}$ sous-cas possibles en entrée (3!) :	
	sous-cas 1 : $v_i(B) \leq v_i(A) \leq v_i(C)$	$v_c(B) \leq v_c(A) \leq v_c(C)$
	sous-cas 2 : $v_i(B) \leq v_i(C) \leq v_i(A)$	2 contraintes possibles : soit $v_c(B) \leq v_c(A) \leq v_c(C)$, soit $v_c(B) \leq v_c(C) \leq v_c(A)$
	sous-cas 3 : $v_i(A) \leq v_i(B) \leq v_i(C)$	2 contraintes possibles : soit $v_c(B) \leq v_c(A) \leq v_c(C)$, soit $v_c(A) \leq v_c(B) \leq v_c(C)$
	sous-cas 4 : $v_i(A) \leq v_i(C) \leq v_i(B)$	3 contraintes possibles : soit $v_c(B) \leq v_c(A) \leq v_c(C)$, soit $v_c(A) \geq v_c(B) \geq v_c(C)$, soit $v_c(A) \geq v_c(C) \geq v_c(B)$
	sous-cas 5 : $v_i(C) \leq v_i(B) \leq v_i(A)$	2 contraintes possibles : soit $v_c(C) \geq v_c(B) \geq v_c(A)$, soit $v_c(B) \geq v_c(C) \geq v_c(A)$
	sous-cas 6 : $v_i(C) \leq v_i(A) \leq v_i(B)$	3 contraintes possibles : soit $v_c(B) \geq v_c(C) \geq v_c(A)$, soit $v_c(C) \geq v_c(B) \geq v_c(A)$, soit $v_c(C) \geq v_c(A) \geq v_c(B)$
3	avec $\boxed{4}$ sous-cas possibles en entrée et non pas 6 (en effet le cas 3 peut être vu comme la concaténation de 2 cas numéro 1) $\Rightarrow 2 \times 2 = 4$ (il n'y a pas de raison d'ordonner B_1 par rapport à B_2)	
	sous-cas 1 : $v_i(A) \leq v_i(B_1)$ et $v_i(A) \leq v_i(B_2)$	$v_c(A) \leq v_c(B_1)$ et $v_c(A) \leq v_c(B_2)$
	sous-cas 2 : $v_i(A) \geq v_i(B_1)$ et $v_i(A) \geq v_i(B_2)$	4 contraintes possibles : soit $v_c(A) \leq v_c(B_1)$ et $v_c(A) \leq v_c(B_2)$, soit $v_c(A) \geq v_c(B_1)$ et $v_c(A) \leq v_c(B_2)$, soit $v_c(A) \leq v_c(B_1)$ et $v_c(A) \geq v_c(B_2)$, soit $v_c(A) \geq v_c(B_1)$ et $v_c(A) \geq v_c(B_2)$
	sous-cas 3 : $v_i(A) \leq v_i(B_1)$ et $v_i(A) \geq v_i(B_2)$	$v_c(A) \leq v_c(B_1)$ puis 2 contraintes possibles : soit $v_c(A) \leq v_c(B_2)$, soit $v_c(A) \geq v_c(B_2)$
	sous-cas 4 : $v_i(A) \geq v_i(B_1)$ et $v_i(A) \leq v_i(B_2)$	$v_c(A) \leq v_c(B_2)$ puis 2 contraintes possibles : soit $v_c(A) \leq v_c(B_1)$, soit $v_c(A) \geq v_c(B_1)$
4	avec $\boxed{36}$ sous-cas possibles (et pas $120 = 5!$) puisqu'il s'agit de la concaténation de 2 cas numéro 2 (pas de raison d'ordonner le couple (B_1, C_1) par rapport au couple (B_2, C_2)) $\Rightarrow 6 \times 6 = 36$; nous ne rentrons pas dans le détail de ces sous-cas ; seul le sous-cas 1 est présenté ci-après :	
	sous-cas 1 : $v_i(B_1) \leq v_i(A) \leq v_i(C_1)$ et $v_i(B_2) \leq v_i(A) \leq v_i(C_2)$	$v_c(B_1) \leq v_c(A) \leq v_c(C_1)$ et $v_c(B_2) \leq v_c(A) \leq v_c(C_2)$
5	avec $\boxed{4}$ sous-cas possibles en entrée et non pas 6 (en effet le cas 5 peut être vu comme la concaténation de 2 cas numéro 1) $\Rightarrow 2 \times 2 = 4$ (il n'y a pas de raison d'ordonner B_1 par rapport à B_2)	
	sous-cas 1 : $v_i(A) \geq v_i(B_1)$ et $v_i(A) \geq v_i(B_2)$	$v_c(A) \geq v_c(B_1)$ et $v_c(A) \geq v_c(B_2)$

<table>
<tr><td rowspan="3"></td><td>sous-cas 2 : $v_i(A) \leq v_i(B_1)$ et $v_i(A) \leq v_i(B_2)$</td><td>4 contraintes possibles :
soit $v_c(A) \leq v_c(B_1)$ et $v_c(A) \leq v_c(B_2)$,
soit $v_c(A) \geq v_c(B_1)$ et $v_c(A) \leq v_c(B_2)$,
soit $v_c(A) \leq v_c(B_1)$ et $v_c(A) \geq v_c(B_2)$,
soit $v_c(A) \geq v_c(B_1)$ et $v_c(A) \geq v_c(B_2)$</td></tr>
<tr><td>sous-cas 3 : $v_i(A) \leq v_i(B_1)$ et $v_i(A) \geq v_i(B_2)$</td><td>$v_c(A) \geq v_c(B_2)$ puis 2 contraintes possibles :
soit $v_c(A) \leq v_c(B_1)$,
soit $v_c(A) \geq v_c(B_1)$</td></tr>
<tr><td>sous-cas 4 : $v_i(A) \geq v_i(B_1)$ et $v_i(A) \leq v_i(B_2)$</td><td>$v_c(A) \geq v_c(B_1)$ puis 2 contraintes possibles :
soit $v_c(A) \leq v_c(B_2)$,
soit $v_c(A) \geq v_c(B_2)$</td></tr>
<tr><td rowspan="2">6</td><td colspan="2">Il y a ici $\boxed{48}$ sous-cas à traiter. Détaillons le calcul de ce nombre de sous-cas.
il s'agit là d'ordonner 5 variables mais sachant qu'il n'y a pas de raison d'ordonner B_1 par rapport à B_2, ni C_1 par rapport à C_2 ;
il faut donc compter le nombre de cas où B_1 et B_2 (resp. C_1 et C_2) sont ordonnés sans rajouter d'information sur l'ordre des autres variables (ce sont les cas où ces 2 variables sont groupées dans la suite représentant l'ordre entre les 5 variables) ;
il y a 6 cas possibles permettant de grouper ces variables pour 1 ordre donné sur les variables restantes et comme il y a 6 façons possibles d'ordonner les 3 variables restantes, on a donc $6 \times 6 \times 2$ cas à ne pas prendre en compte $\Rightarrow$ il y a $5! - 72 = 48$[a] sous-cas à traiter ici !
Nous présenterons ci-après uniquement le sous-cas 1.

[a]On peut retrouver ce résultat en décomposant le problème sous la forme suivante : énumérons tous les cas possibles pour lesquels ni B_1 B_2, ni C_1 C_2 ne sont groupés ; on a alors 4 cas possibles (soit $C_1 < C_2$ et $B_1 < B_2$, soit $C_1 < C_2$ et $B_1 > B_2$, soit $C_1 > C_2$ et $B_1 < B_2$, soit $C_1 > C_2$ et $B_1 > B_2$), et pour chacun de ces cas, on a encore 2 sous-cas possibles (soit $C_1 < B_1$, soit $C_1 > B_1$) ; ici, il y a alors 6 possibilités ($A < C_1 < B_1 < C_2 < B_2$, $C_1 < A < B_1 < C_2 < B_2$, $C_1 < B_1 < A < C_2 < B_2$, $C_1 < B_1 < A < B_2 < C_2$, $C_1 < B_1 < C_2 < A < B_2$, $C_1 < B_1 < C_2 < B_2 < A$) ; ainsi, on a $6 \times 2 \times 4 = 48$ cas.</td></tr>
<tr><td>sous-cas 1 : $\{v_i(B_1), v_i(B_2)\} \leq v_i(A) \leq \{v_i(C_1), v_i(C_1)\}$</td><td>$v_c(A) \leq \{v_c(B_1), v_c(B_2)\} \leq \{v_c(C_1), v_c(C_1)\}$</td></tr>
<tr><td rowspan="2">7</td><td colspan="2">avec $\boxed{n!}$ sous-cas possibles ; seul le sous-cas 1 est présenté ici :</td></tr>
<tr><td>sous-cas 1 :
$v_i(A''_1) \leq \ldots \leq v_i(A''_p) \leq v_i(A'_1) \leq \ldots \leq v_i(A'_m)$
avec $m = \frac{n+1}{2}$,
$p = \frac{n}{2}$,
$A'_i = A_{2i}$ si n pair, A_{2i-1} sinon,
et $A''_i = A_{2i-1}$ si n pair, A_{2i} sinon.</td><td>$v_c(A''_1) \leq \ldots \leq v_c(A''_p) \leq v_c(A'_1) \leq \ldots \leq v_c(A'_m)$</td></tr>
<tr><td>8</td><td colspan="2">avec $\boxed{2^n}$ sous-cas possibles ; en effet, chaque B_i a 2 possibilités : soit $v_i(B_i) \leq v_i(A)$, soit $v_i(B_i) \geq v_i(A)$;
remarquons que l'on peut aussi calculer le nombre de sous-cas possibles de la façon suivante : nombre de parties différentes de p éléments tirés parmi n éléments avec p variant de 0 à $n = \Sigma_{p=0}^{n} C_n^p$ (et il se trouve, c'est heureux, que $\Sigma_{p=0}^{n} C_n^p = 2^n$ – la démonstration se fait par récurrence en utilisant le résultat intermédiaire $C_{n+1}^p = C_n^p + C_n^{p-1}$ pour $1 \leq p \leq n$).</td></tr>
<tr><td>9</td><td colspan="2">Non traité puisqu'on ne sait même pas comment ordonner les valeurs de base !</td></tr>
</table>

3.3.2 Les cas avec circuits pairs

<table>
<tr><th>Numéro Cas</th><th colspan="2">Contraintes sur les valeurs</th></tr>
<tr><td rowspan="4">10</td><td colspan="2">Ici, on présente $\boxed{3}$ sous-cas (au lieu de 2) car on veut mettre en évidence le cas particulier de l'égalité :</td></tr>
<tr><td>sous-cas 1 : $v_i(B) = v_i(A)$</td><td>$v_c(B) = v_c(A)$</td></tr>
<tr><td>sous-cas 2 : $v_i(B) < v_i(A)$</td><td>$v_c(B) < v_c(A)$</td></tr>
<tr><td>sous-cas 3 : $v_i(B) > v_i(A)$</td><td>$v_c(B) > v_c(A)$</td></tr>
</table>

11	Ici, on va avoir $\boxed{6}$ sous-cas possibles, car il s'agit de la concaténation de 2 sous-cas (les cas 3 et 2 – avec la position de A et de B_1 inversée –) mais qui ne sont pas indépendants (les mêmes variables apparaissent dans les 2 cas) $\Rightarrow$ nombre de sous-cas = le max des nombres de sous-cas de chaque cas ($6 = max(4,6)$) ; remarquons que l'on retrouve non seulement le même nombre de sous-cas mais les mêmes sous-cas (à l'échange des arguments A et B_1 près), car le cas 3 est moins restrictif que le cas 2 :	
	sous-cas 1 : $v_i(A) \leq v_i(B_1) \leq v_i(B_2)$	$v_c(A) \leq v_c(B_1) \leq v_c(B_2)$
	sous-cas 2 : $v_i(A) \leq v_i(B_2) \leq v_i(B_1)$	2 contraintes possibles : soit $v_c(A) \leq v_c(B_1) \leq v_c(B_2)$, soit $v_c(A) \geq v_c(B_2) \geq v_c(B_1)$
	sous-cas 3 : $v_i(B_1) \leq v_i(A) \leq v_i(B_2)$	2 contraintes possibles : soit $v_c(A) \leq v_c(B_1) \leq v_c(B_2)$, soit $v_c(B_1) \geq v_c(A) \geq v_c(B_2)$
	sous-cas 4 : $v_i(B_1) \leq v_i(B_2) \leq v_i(A)$	3 contraintes possibles : soit $v_c(A) \leq v_c(B_1) \leq v_c(B_2)$, soit $v_c(B_1) \geq v_c(A) \geq v_c(B_2)$, soit $v_c(B_1) \geq v_c(B_2) \geq v_c(A)$
	sous-cas 5 : $v_i(B_2) \leq v_i(A) \leq v_i(B_1)$	2 contraintes possibles : soit $v_c(B_2) \geq v_c(A) \geq v_c(B_1)$, soit $v_c(A) \geq v_c(B_2) \geq v_c(B_1)$
	sous-cas 6 : $v_i(B_2) \leq v_i(B_1) \leq v_i(A)$	3 contraintes possibles : soit $v_c(A) \geq v_c(B_2) \geq v_c(B_1)$, soit $v_c(B_2) \geq v_c(A) \geq v_c(B_1)$, soit $v_c(B_2) \geq v_c(B_1) \geq v_c(A)$
12	avec $\boxed{9}$ sous-cas possibles (et pas $6 = 3!$) puisqu'il s'agit de la concaténation de 2 cas numéro 10 dans lesquels on veut expliciter l'égalité entre les valeurs (pas de raison d'ordonner B_1 par rapport à B_2) $\Rightarrow 3 \times 3 = 9$; nous ne rentrons pas dans le détail de ces sous-cas ; seul le sous-cas 1 est présenté ci-après :	
	$v_i(A) = v_i(B_1) = v_i(B_2)$	$v_c(A) = v_c(B_1) = v_c(B_2)$
13	avec $\boxed{6}$ sous-cas possibles (3!) ; nous ne rentrons pas dans le détail de ces sous-cas ; seul le sous-cas 1 est présenté ci-après :	
	$v_i(A) \leq v_i(B_1) \leq v_i(B_2)$	$v_c(A) \leq v_c(B_1) \leq v_c(B_2)$

3.3.3 Les cas avec circuits impairs

Numéro Cas	Contraintes sur les valeurs	
14	$v_c(A) \leq M$	
15	avec $\boxed{6}$ sous-cas possibles (3!) ; seul le sous-cas 1 est présenté ici :	
	sous-cas 1 : $v_i(A) \leq v_i(B) \leq v_i(C)$	$v_c(A) \leq v_c(B) \leq v_c(C) \leq M$
16	avec $\boxed{24}$ sous-cas possibles (4!) ; seul le sous-cas 1 est présenté ici :	
	sous-cas 1 : $v_i(A) \leq v_i(B) \leq v_i(C) \leq v_i(D)$	$v_c(A) \leq v_c(B) \leq v_c(C) \leq v_c(D) \leq M$
17	avec $\boxed{24}$ sous-cas possibles (4!) ; seul le sous-cas 1 est présenté ici :	
	sous-cas 1 : $v_i(A) \leq v_i(B) \leq v_i(C) \leq v_i(D)$	$v_c(A) \leq v_c(B) \leq v_c(C) \leq v_c(D) \leq M$

3.3.4 Analyse des exemples

Les quelques règles utilisées dans ce tableau que l'on puisse expliciter :

Règle 8 (Combinaison valeur intrinsèque/valeur issue du graphes des contrariétés) *La valeur d'un argument est une fonction combinant la valeur intrinsèque de cet argument et la valeur issue du* *graphe des contrariétés.*

Règle 9 (Attaque sans défense) *La valeur combinée doit respecter la contrainte suivante :*

Si $v_i(A) \leq v_i(B)$ *et* B *attaque* A *sans être lui-même attaqué alors* $v_c(A) \leq v_c(B)$

Règle 10 (Attaque avec défense) *La valeur combinée doit respecter la contrainte suivante :*

Si A *est attaqué par* B *mais défendu par* C *et* $v_i(A) \leq v_i(C)$ *alors* $v_c(A) \leq v_c(C)$

Règle 11 (Accord entre valeur intrinsèque et valeur issue du graphes des contrariétés) *Si l'évaluation intrinsèque est, soit la même pour tous, soit respecte l'ordre imposé par l'évaluation issue du graphe des contrariétés, alors la valeur combinée respectera elle-aussi l'ordre imposé par l'évaluation issue du graphe des contrariétés.*

Propriété 2 (Existence d'une fonction évaluation combinée) $\forall v_i$ *une évaluation intrinsèque des arguments et* $\forall v$ *une évaluation issue du graphe des contrariétés (donc respectant les contraintes définies à la section 3.2 page 11), il existe toujours une fonction d'évaluation combinée* v_c *vérifiant les schémas et les règles données ci-dessus.*

Preuve : Il suffit en effet de poser : $v_c(A) = v(A), \forall A$ (on ignore $v_i(A)$). □

Il sera difficile de mettre à jour d'autres règles étant donnée la complexité des choix effectués.

3.4 Analyse de l'approche 0

Analysons l'approche 0 à la vue de la seule évaluation rencontrée dans la littérature qui soit véritablement graduelle[2] (celle de Besnard et Hunter [BH01]). Toutefois, notre analyse devra tenir compte des faits suivants :

- [BH01] ne prend pas en compte une valeur intrinsèque des arguments,
- [BH01] n'admet pas de circuit (pair ou impair) ; en fait, on peut l'appliquer à des circuits sachant alors que la valeur des éléments du circuit sera le point fixe de la fonction $\frac{1}{1+x}$ (l'inverse du nombre d'or – voir remarque dans la description du cas 7 dans la section 3.4.1).

3.4.1 Comparaison avec [BH01] sur les différents cas

Passons en revue les différents cas sans circuit[3] :

[2]Il existe d'autres évaluations mais jamais avec plus de 3 valeurs (accepté, rejeté, non décidé). Voir par exemple [JV99].
[3]Le cas 9 ne sera pas abordé puisqu'on ne sait pas comment le traiter à notre niveau !

Numéro Cas	Contraintes sur les valeurs
0	$v(A) = 1$
1	$v(A) = \frac{1}{2} \leq v(B) = 1$
2	$v(B) = \frac{1}{2} \leq v(A) = \frac{2}{3} \leq v(C) = 1$
3	$v(A) = \frac{1}{3} \leq v(B_1) = v(B_2) = 1$
4	$v(B_1) = v(B_2) = \frac{1}{2} \leq v(A) = \frac{1}{2} \leq v(C_1) = v(C_2) = 1$
5	$v(B_1) = v(B_2) = \frac{1}{2} \leq v(A) = 1$
6	$v(A) = \frac{1}{3} \leq v(B_1) = v(B_2) = \frac{3}{4} \leq v(C_1) = v(C_2) = 1$
7	On a les contraintes suivantes : $v(A_n) = 1$, $v(A_{n-i}) = \frac{1}{1+v(A_{n-i-1})}$, pour $i = 1 \ldots n-1$. Après quelques calculs, on constate rapidement que cette suite fonction de la variable i correspond à la formule suivante : $v(A_{n-i}) = \frac{Fib(i)}{Fib(i+1)}$ (Fib étant la suite de Fibonacci) (remarquons que $\frac{Fib(i)}{Fib(i+1)}$ converge par oscillation vers l'inverse du nombre Φ (le “ratio d'or” connu depuis l'antiquité) quand i tend vers l'infini ; cet inverse est noté ϕ et est égal à $0.61803\ldots$) Les valeurs obtenues sont ordonnées de la façon suivante : Si n est pair alors (l'argument A_1 est au bout d'une chaîne d'attaque et de défense dont le début est un attaquant) : $1 = v(A_n) \geq v(A_{n-2}) \geq \ldots \geq v(A_2)$ $\geq v(A_1) \geq \ldots \geq v(A_{n-1})$ et sinon (n impair – l'argument A_1 est au bout d'une chaîne d'attaque et de défense dont le début est un défenseur) : $1 = v(A_n) \geq v(A_{n-2}) \geq \ldots \geq v(A_1)$ $\geq v(A_2) \geq \ldots \geq v(A_{n-1})$
8	$v(A) = \frac{1}{1+\Sigma_{i=1}^{n} v(B_i)} \leq v(B_j) = 1, \forall j = 1 \ldots n$ (en effet, il est facile de démontrer par récurrence sur la structure de l'arbre menant à A que $0 \leq v(A) \leq 1$)

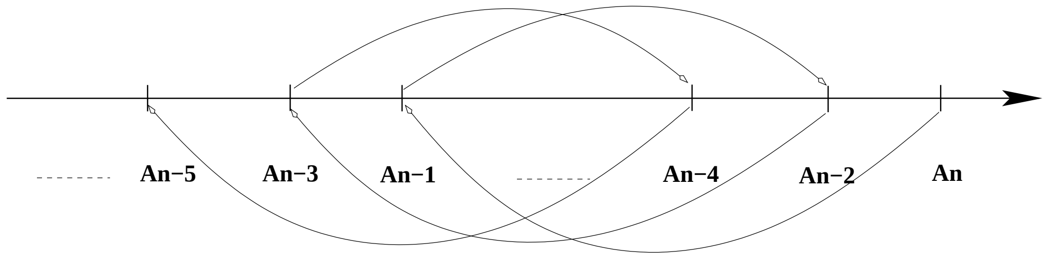
Les résultats de cette comparaison montrent que tant que l'on reste dans des cas simples, la fonction d'évaluation de [BH01] vérifie les contraintes d'ordre que nous souhaitons. Par contre, dans les cas complexes (le cas 7, par exemple), nous ne sommes plus en accord.

Notons que le résultat obtenu pour le cas 8 ($v(A) \leq v(B_j)$) n'est pas toujours valide quand les B_j ne sont plus des feuilles (voir par exemple le cas 2 qui est un cas particulier du cas 8 avec un B_j qui n'est pas une feuille ; on a alors $v(A) = \frac{2}{3} \geq v(B) = \frac{1}{2}$).

Ordre de [BH01] :

Ici, l'ordre et les distances sont significatifs !

L'ordre proposé par l'approche 0 :



Attention, les distances ne sont pas significatives, seul compte l'ordre entre les variables !

Il s'agit ici d'une question d'interprétation. Soit A un argument attaqué et non défendu, et soit B un argument attaqué et défendu par A, on a 2 choix possibles :
- soit $v(A) \leq v(B)$: on privilégie alors le fait que B soit au moins défendu alors que A ne l'est pas (même si cette défense est effectuée par un argument attaqué et non défendu !) ;
- soit $v(A) \geq v(B)$: on privilégie le fait que A soit situé, dans le graphe des contrariétés plus près des feuilles ; donc que B soit attaqué "plus souvent" que A (une fois en propre et x fois au travers de ses "ancêtres" défenseurs). Mais est-ce significatif ?

Un autre exemple comparatif : l'exemple 18 page 12

Sur la totalité de ce graphe, rappelons que, d'après l'approche 0, nos contraintes d'ordre entre arguments sont :

$$\begin{array}{c} E_1, D_2, D_3, C_4, B_4 \\ \succeq \\ C_1, B_2 \\ \succeq \\ C_3 \\ \succeq \\ A, B_1, C_2, D_1 \text{ (égalité forcée !)} \\ \succeq \\ B_3 \end{array}$$

Puis calculons les valeurs d'après [BH01] sur le graphe dans sa totalité :
- $v(E_1) = v(D_2) = v(D_3) = v(C_4) = v(B_4) = 1$,
- $v(D_1) = v(C_2) = v(C_3) = v(B_3) = \frac{1}{2}$,
- $v(C_1) = v(B_2) = \frac{2}{3}$,
- $v(B_1) = \frac{6}{13}$,
- $v(A) = \frac{78}{283}$.

Ce qui nous donne du point de vue de l'ordre entre les arguments :

$$\begin{array}{c} E_1, D_2, D_3, C_4, B_4 \\ \succeq \\ C_1, B_2 \\ \succeq \\ D_1, C_2, C_3, B_3 \\ \succeq \\ B_1 \\ \succeq \\ A \end{array}$$

Avec [BH01], nous n'avons pas de problème d'"égalité forcée". Par contre, remarquons que, si on se restreint à la partie entourée du graphe, alors la valeur de $A = \frac{13}{19}$, ce qui est supérieur à $v(B_1)$ et $v(D_1)$, mais aussi à $v(C_1)$ (A devient plus fort que son défenseur, alors qu'il y a une branche d'attaque menant à A et aucune menant à C_1 !).

3.4.2 Conclusion de l'analyse

L'approche 0 ne fournit pas les mêmes résultats que [BH01]. Pour comprendre pourquoi nous avons cherché à axiomatiser l'approche de [BH01]. Ainsi, nous avons identifié certains principes qui nous ont conduit

à proposer une nouvelle approche (approche 1 - section 4 page ci-contre) qui se présente comme une généralisation de [BH01] (voir la propriété 9 dans la section 6.4 page 47).
D'autre part, remarquons que [BH01] présente quand même des inconvénients (voir le traitement de l'exemple 18 page 12 en fin de section 3.4.1 page 17) qui vont être aussi ceux de l'approche 1. Nous serons alors amenés à modifier certains des principes posés pour proposer encore une troisième approche (approche 2 - section 5 page 25).

3.5 Conclusion sur l'approche 0

Cette approche étant uniquement intuitive, les résultats proposés sont bien-sûr sujets à caution (et d'ailleurs la section 3.4 page 17 en constitue une première critique). Par contre, elle a quand même permis :
- de proposer une gamme d'exemples significatifs qui seront repris dans la suite de ce document pour chacune des approches proposées ;
- d'étudier (superficiellement !) la prise en compte d'une valeur intrinsèque en plus des interactions entre arguments pour l'évaluation complète des arguments. L'application aux exemples montre de manière évidente la complexité croissante de ce dernier point si on ne pose aucune règle restrictive.
- de suggérer par comparaison avec [BH01] les principes fondateurs des approches 1 et 2 qui vont être présentées maintenant.

Chapitre 4

Approche 1 dite “locale” : prise en compte des interactions directes

4.1 Les principes retenus

L'étude de [BH01] nous conduit à sélectionner 4 principes essentiels pour définir une évaluation d'arguments basée sur l'interaction entre arguments (au sens de la contrariété).

P1 L'évaluation est maximale pour un argument sans attaquant et non maximale pour un argument attaqué et non défendu.

P2 L'évaluation d'un argument est fonction de l'évaluation de tous les attaquants directs (la contrariété directe).

P3 L'évaluation d'un argument est une fonction décroissante de l'évaluation de la contrariété directe.

P4 La contribution d'un attaquant direct d'un argument ne peut pas diminuer l'évaluation de la contrariété directe pour cet argument.

Étant donné qu'il s'agit ici de ne se préoccuper directement que du niveau de contrariété le plus proche, on parlera d'approche “locale”.

4.2 Formalisation

À partir de ces principes, on va formaliser une évaluation de la manière suivante.
On dispose d'un ensemble W totalement ordonné admettant un plus petit élément (V_{Min}) et d'une partie V de W, contenant V_{Min} et admettant un plus grand élément V_{Max}.

Définition 12 *Soit* $<\mathcal{A}, \mathcal{R}>$ *un système d'argumentation. Une évaluation est une application* $v : \mathcal{A} \to V$ *telle que :*

1. $\forall A \in \mathcal{A}$, $v(A) \geq V_{Min}$
2. $\forall A \in \mathcal{A}$, *si* $\mathcal{R}^-(A) = \emptyset$, *alors* $v(A) = V_{Max}$
3. $\forall A \in \mathcal{A}$, *si* $\mathcal{R}^-(A) = \{A_1, \ldots, A_n\} \neq \emptyset$, *alors* $v(A) = g(h(v(A_1), \ldots, v(A_n)))$

où $h : V^* \to W$ *telle que* (V^* *dénote l'ensemble des suites finies d'éléments de* V)
– $h(x) = x$
– $h() = V_{Min}$
– $h(x_1, \ldots, x_n, x_{n+1}) \geq h(x_1, \ldots, x_n)$
et $g : W \to V$ *telle que*
– $g(V_{Min}) = V_{Max}$
– $g(V_{Max}) < V_{Max}$
– g *est decroissante (si* $x \leq y$ *alors* $g(x) \geq g(y)$*)*

Remarque : $h(x_1, \ldots, x_n) \geq max(x_1, \ldots, x_n)$ est une conséquence des propriétés de la fonction h.

4.3 Propriétés

La formalisation proposée présente les propriétés suivantes.
Existence d'une évaluation v : voir en section 6 page 45 (propriété 8 page 47) un exemple d'évaluation défini dans le cadre du travail de [JV99].
P1 est satisfait car : $\forall A \in \mathcal{A}$ si A n'a pas d'attaquant ($\mathcal{R}^-(A)$ est vide), alors $v(A) = V_{\text{Max}}$ et $g(V_{\text{Max}}) < V_{\text{Max}}$.
P2 est satisfait car si $\mathcal{R}^-(A) = \{A_1, \dots, A_n\}$, $h(v(A_1), \dots, v(A_n))$ évalue la contrariété directe de A.
P3 est satisfait car la fonction g est supposée décroissante.
P4 est satisfait par les propriétés de la fonction h.

Proposition 1 *La fonction g satisfait pour tout $n \geq 1$:*

$$g(V_{Max}) \leq g^3(V_{Max}) \leq \dots \leq g^{2n+1}(V_{Max})$$

$$\leq g^{2n}(V_{Max}) \leq \dots \leq g^2(V_{Max}) \leq V_{Max}$$

Si de plus g est strictement décroissante et $g(V_{Max}) > V_{Min}$, les inégalités ci-dessus sont strictes.

Preuve : Par récurrence à partir de $V_{\text{Min}} \leq g(V_{\text{Max}}) < V_{\text{Max}}$ et par application de g deux fois consécutivement. □

4.4 Exemples

Reprenons quelques exemples.

4.4.1 Les cas sans circuit

Numéro Cas	Contraintes sur les valeurs
0	$v(A) = V_{\text{Max}}$
1	$v(B) = V_{\text{Max}}$ donc $v(A) = g(V_{\text{Max}})$
2	$v(C) = V_{\text{Max}}$, $v(B) = g(V_{\text{Max}})$ et $v(A) = g^2(V_{\text{Max}})$
3	$v(B_1) = v(B_2) = V_{\text{Max}} \geq v(A) = g(h(V_{\text{Max}}, V_{\text{Max}}))$
4	$v(C_1) = v(C_2) = V_{\text{Max}} \geq v(B_1) = v(B_2) = g(V_{\text{Max}})$ et $v(A) = g(h(g(V_{\text{Max}}), g(V_{\text{Max}})))$ LIEN ENTRE $v(A)$ et $v(C_1), v(C_2), v(B_1), v(B_2)$?
5	$v(A) = V_{\text{Max}} \geq v(B_1) = v(B_2) = g(V_{\text{Max}})$
6	$v(C_1) = v(C_2) = V_{\text{Max}} \geq v(A) = g(h(V_{\text{Max}}, V_{\text{Max}}))$ et $v(B_1) = v(B_2) = g(h(g(h(V_{\text{Max}} V_{\text{Max}})), g(h(V_{\text{Max}}, V_{\text{Max}}))))$ LIEN ENTRE $v(B_1), v(B_2)$ et $v(C_1), v(C_2), v(A)$?
7	$v(A) = g^{n-1}(V_{\text{Max}})$ Si n est pair $v(A_{n-1}) \leq \dots \leq v(A_3) \leq v(A_1) \leq v(A_2) \leq \dots \leq v(A_n) = V_{\text{Max}}$ Si n est impair $v(A_{n-1}) \leq \dots \leq v(A_2) \leq v(A_1) \leq v(A_3) \leq \dots \leq v(A_n) = V_{\text{Max}}$
8	$v(B_i) = V_{\text{Max}} \geq v(A) = g(h(\underbrace{V_{\text{Max}}, \dots, V_{\text{Max}}}_{n \text{ fois}}))$
9	Si $v(A_i)$ augmente (resp. diminue), $v(A)$ diminue (resp. augmente) en raison des propriétés de g et de h. $v(A) = g(h(v(A_1), \dots, v(A_{n_0}))) \leq g(h(v(A_1), \dots, v(A_{n_0 - 1})))$ Remarque : Si $h(x_1, \dots, x_n, x_{n+1}) > h(x_1, \dots, x_n)$ et g est une fonction strictement décroissante, alors l'inégalité ci-dessus est stricte. Cet exemple montre que si l'on veut tenir compte du nombre de contrariants, l'évaluation d'un argument doit pouvoir être strictement inférieure à $g(V_{\text{Max}})$. C'est le cas si la fonction g est strictement décroissante et si la fonction h vérifie $h(x_1, \dots, x_n, x_{n+1}) > h(x_1, \dots, x_n)$.

4.4.2 Les cas avec circuits pairs

Numéro Cas	Contraintes sur les valeurs
10	$v(A) = g(v(B))$ et $v(B) = g(v(A))$ donc $v(A)$ et $v(B)$ sont des points fixes de g^2. Par contre, les contraintes sur g ne permettent pas d'imposer $v(A) = v(B)$.
11	$v(B_2) = V_{\mathrm{Max}}$, $v(A) = g(h(v(B_1), V_{\mathrm{Max}}))$ et $v(B_1) = g(v(A))$ LIEN ENTRE $v(A), v(B_1)$?
12	$v(A) = g(h(v(B_1), v(B_2)))$, $v(B_1) = v(B_2) = g(v(A))$ LIEN ENTRE $v(B_2), v(A), v(B_1)$?
13	D'après le résultat de l'exemple 10, $v(A)$ et $v(B_1)$ sont des points fixes de g^2 et $v(B_2) = g(v(A))$ LIEN ENTRE $v(B_2)$ et $v(A), v(B_1)$?

4.4.3 Les cas avec circuits impairs

Numéro Cas	Contraintes sur les valeurs
14	$v(A)$ est un point fixe de g
15	$v(A), v(B), v(C)$ sont des points fixes de g^3 et sont égaux, puisque g^3 est décroissante. De plus, $v(A) = g(v(B))$ implique que $v(A) = v(B) = v(C) = x$, avec x un point fixe de g et $x < V_{\mathrm{Max}}$.
16	$v(D) = V_{\mathrm{Max}}$, $v(C) = g(h(v(B), V_{\mathrm{Max}}))$, $v(A) = g(v(C))$ et $v(B) = g(v(A))$ LIEN ENTRE $v(C), v(A), v(B)$?
17	D'après le résultat de l'exemple 15, $v(A) = v(B) = v(C)$ point fixe de g, et avec $v(D) = g(v(C))$ on a $v(D) = v(B) = v(A) = v(C)$.

L'étude de graphes avec circuits fait apparaître le rôle joué par les points fixes de g, g^2, ... On peut généraliser ces résultats dans le cas de circuits de longueur n : si n est impair, tous les arguments du circuit ont la même valeur qui est un point fixe de g, et si n est pair, tous les arguments du circuit ont pour valeur des points fixes de g^n.

Et sur l'exemple 18 page 12 On obtient les équations suivantes :
- $v(E_1) = v(D_2) = v(D_3) = v(C_4) = v(B_4) = V_{\mathrm{Max}}$
- $v(D_1) = v(C_2) = v(C_3) = v(B_3) = g(V_{\mathrm{Max}})$
- $v(C_1) = v(B_2) = g^2(V_{\mathrm{Max}})$
- $v(B_1) = g(h(g^2(V_{\mathrm{Max}}), g(V_{\mathrm{Max}})))$
- $v(A) = g(h(g(h(g^2(V_{\mathrm{Max}}), g(V_{\mathrm{Max}}))), g^2(V_{\mathrm{Max}}), g(V_{\mathrm{Max}}), V_{\mathrm{Max}}))$

On a donc :

$$\begin{array}{c} E_1, D_2, D_3, C_4, B_4 \\ \succeq \\ C_1, B_2 \\ \succeq \\ D_1, C_2, C_3, B_3 \end{array}$$

Par contre, les contraintes sur $v(A)$ et sur $v(B_1)$ sont insuffisantes.
Et c'est le même problème qui apparaît si l'on réduit cet exemple à la partie entourée du graphe : $E_1, D_2 \succeq C_1 \succeq D_1, C_2$ avec la difficulté de comparer $v(A)$ ($v(A) = g^2(h(g^2(V_{\mathrm{Max}}), g(V_{\mathrm{Max}}))))$ et $v(B_1)$ ($v(B_1) = g(h(g^2(V_{\mathrm{Max}}), g(V_{\mathrm{Max}}))))$ aux autres valeurs.

En résumé :

<table>
<tr><td>V_{Min}</td><td></td><td>$g(V_{\mathrm{Max}})$</td><td></td><td>V_{Max}</td></tr>
<tr><td>Une infinité d'attaquants directs</td><td>Un seul niveau de contrariété. Plus d'un attaquant direct</td><td>Un seul niveau et un seul attaquant direct</td><td>Plusieurs niveaux et un seul attaquant par niveau</td><td>Argument non contrarié</td></tr>
<tr><td></td><td colspan="3">Plusieurs niveaux et plusieurs attaquants par niveau</td><td></td></tr>
</table>

Chapitre 5

Approche 2 dite “globale” : des tuples pour prendre en compte toutes les interactions

5.1 Un exemple introductif

À la vue de l'exemple 18 page 12 (en particulier de sa partie entourée), on constate que les ordres envisagés jusque là posent un problème dès lors qu'un argument est racine à la fois de branches d'attaque et de branches de défense. En effet, soit on se retrouve à devoir “forcer” des égalités de valeurs (l'approche 0), soit on trouve des résultats qui paraissent surprenants (des arguments en bout de branche de défense ayant une valeur moindre que ceux qui sont en bout d'une branche de défense et d'une branche d'attaque ([BH01]), soit des arguments difficilement comparables aux autres (approche 1[1]).

Une solution consiste à développer un étiquetage permettant de mémoriser cette structure, en associant à chaque branche sa longueur (nombre d'arcs de la feuille jusqu'au nœud courant) dans le graphe des contrariétés, sachant que :

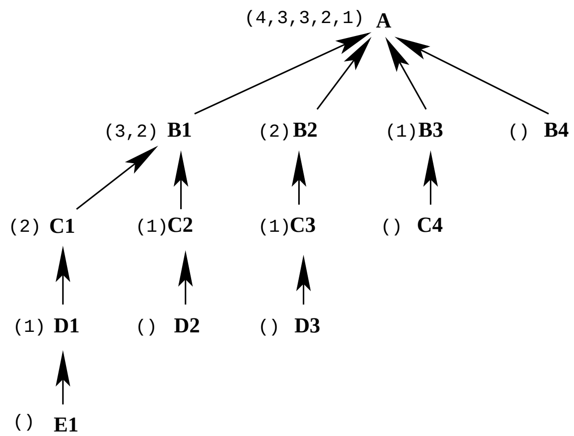
- on traite les circuits à part (soit en utilisant la longueur min du circuit, soit en réduisant le circuit à deux branches – 1 d'attaque et 1 de défense –, soit en supprimant carrément les circuits du graphe),
- une branche de longueur impaire est une branche d'attaque pour le nœud courant, alors qu'une branche de longueur paire est une branche de défense pour ce même nœud.

Remarquons que le fait, pour un argument, d'avoir une branche de défense signifie qu'il est défendu mais donc aussi qu'il a été attaqué. On ne considère donc pas une branche de défense comme un signe de victoire ! L'idée de branche de défense/d'attaque ne correspond donc pas exactement à l'idée d'un argument gagnant/perdant !

La comparaison de deux arguments se fera donc sur leur nombre respectif de branches d'attaque, de branches de défense et sur la longueur des dites branches. Il reste à définir une relation d'ordre sur ce type de tuples (de longueur variable !).

Ainsi, sur l'exemple 18 page 12, on a :

[1] Cela vient du fait que l'approche 1 n'impose pas nécessairement un ordre unique sur les arguments classés d'après leur valeur.



L'argument A a 3 branches d'attaque (2 de longueur 3 et 1 de longueur 1) et 2 branches de défense (1 de longueur 4 et l'autre de longueur 2).
Nous allons donc poser de nouveaux principes, tout en retrouvant quelques points communs avec ceux proposés pour l'approche 1 :

P1' L'évaluation est maximale pour un argument sans attaquant et non maximale pour un argument attaqué (qu'il soit défendu ou pas).

P2' L'évaluation d'un argument prend en compte toutes les branches dont cet argument est racine.

P3' L'amélioration de la défense ou la détérioration de l'attaque conduisent à augmenter la valeur d'un argument.

P4' L'amélioration de l'attaque ou la détérioration de la défense conduisent à diminuer la valeur d'un argument.

On va donc définir une évaluation qui prendra en compte les différentes branches menant à un argument (c'est donc une approche "globale"). Ce sera *l'évaluation par tuples.*

5.2 Définition d'une valeur tuplée

Nous conservons la règle 1 page 7 et nous posons plusieurs définitions constructives d'une valeur tuplée suivant les différents cas que l'on peut rencontrer dans un graphe des contrariétés. Attention, les valeurs tuplées que nous définissons sont des n-uplets non ordonnés.

Règle 12 (Valeur d'un argument n'appartenant pas à un circuit) *Soit l'argument A n'appartenant pas à un circuit, sa valeur sera :*
– si A n'est pas attaqué :

$$v(A) = () \text{ (aussi notée : } v(A) = (0)\text{)}$$

– si A a pour attaquants directs les arguments $B_1, \ldots, B_n$ dont les valeurs respectives sont les tuples $(b_1^1, \ldots, b_{m_1}^1), \ldots, (b_1^n, \ldots, b_{m_n}^n)$:

$$v(A) = (b_1^1 + 1, \ldots, b_{m_1}^1 + 1, \ldots, b_1^n + 1, \ldots, b_{m_n}^n + 1)$$

Le cas des circuits est assez particulier car le nombre de "branches"[2] est infini. On considérera le circuit comme un méta-argument. On va prendre la convention suivante :

> *On ne mémorisera que deux branches (celle d'attaque de longueur minimale et celle de défense de longueur minimale).*

D'autre part, le calcul s'effectuera en répertoriant pour chaque élément du circuit les attaques et les défenses internes au circuit, puis en rajoutant celles qui viennent de l'extérieur du circuit.

Règle 13 (Valeur d'un argument appartenant à un circuit) *Soit un argument A appartenant à $\mathtt{C}_1$ … $\mathtt{C}_m$ circuits. Soient $\mathtt{C}'_1$ … $\mathtt{C}'_n$ autres circuits interconnectés à l'un des $\mathtt{C}_i$ ou entre eux[3]. Soient X^1 …*

[2]L'utilisation de ce terme est un abus de langage puisque les éléments d'un circuit ne sont pas forcément liés à une feuille du graphe.
[3]A n'appartient donc à aucun des $\mathtt{C}'_i$.

X^p arguments n'appartenant pas aux circuits mais attaquants directs de l'un des circuits[4]*. Sachant que l'on note :*
– l_i : distance la plus courte entre le circuit $\mathtt{C}'_i$ et A ($l_i = Min_{C'_j \in \mathtt{C}'_i}(l_{\mathcal{C}(C'_j, A)})$),
– $l_{X^i} = l_{\mathcal{C}(X^i, A)}$ (distance[5] *entre l'argument X^i et A),*
– chaque argument X^i a pour valeur : $v(X^i) = (x^i_1, \ldots, x^i_{k^1})$.
on a alors :

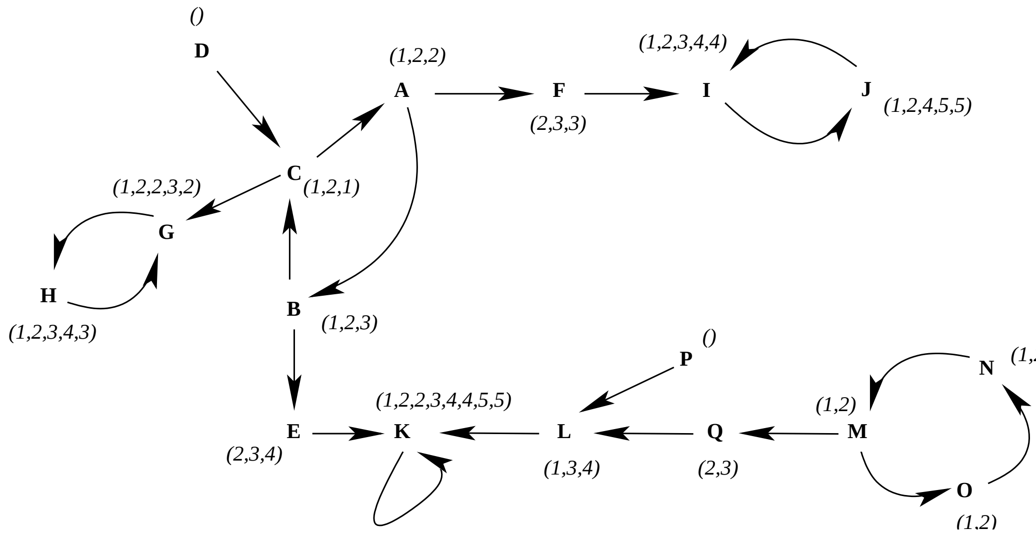
$$\begin{array}{rcl} v(A) & = & (\underset{m\ fois}{\overbrace{1, 2, \ldots, 1, 2,}} \\ & & 1 + l_1, 2 + l_1, \ldots, 1 + l_n, 2 + l_n, \\ & & x^1_1 + l_{X^1}, \ldots, x^1_{k^1} + l_{X^1}, \\ & & \ldots, \\ & & x^p_1 + l_{X^p}, \ldots, x^p_{k^p} + l_{X_p}) \end{array}$$

Remarque : pour calculer la valeur d'un argument A attaqué par un argument B ayant () pour valeur, il suffit d'utiliser la notation (0) pour B et d'appliquer soit la règle 12 page ci-contre, soit la règle 13 page précédente.

Propriété 3 (Existence d'une évaluation par tuples) *Soit V l'ensemble des multi-ensembles d'entiers, il existe toujours une fonction d'évaluation par tuples $v\ : \mathcal{A} \rightarrow V$ vérifiant les règles données ci-dessus.*

> **Preuve :** Il s'agit d'une application directe des règles 12 page ci-contre à 13 page précédente, puisque ces règles répertorient la totalité des cas pouvant se produire (avec ou sans circuit). □

Un exemple d'étiquetage complet : Prenons le graphe des contrariétés suivant et appliquons les règles précédentes :



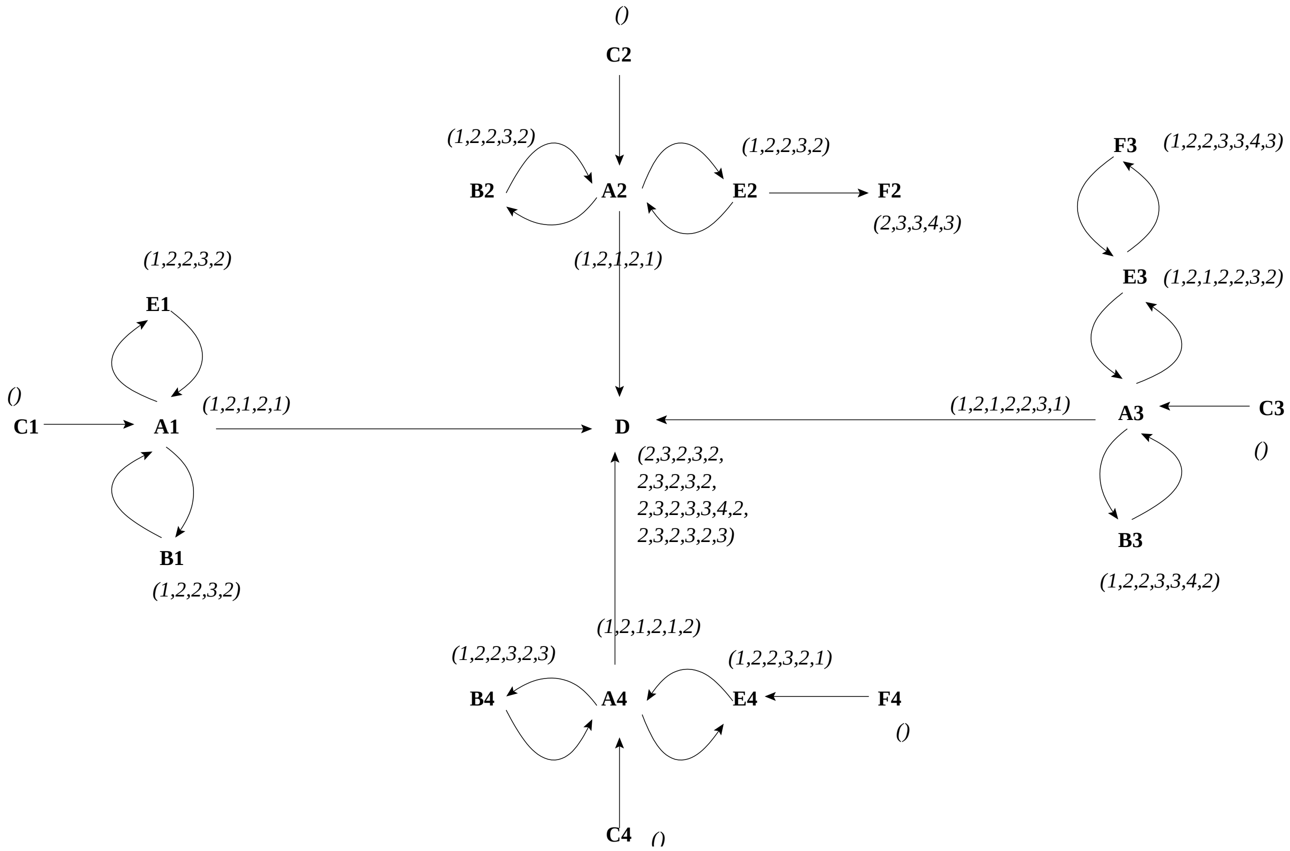
Pour D, E, F, L, Q et P, on utilise la règle 12 page ci-contre. Les valeurs de tous les autres arguments sont fournies par la règle 13 page précédente.

Remarquons au passage que le seul cas où un argument a la même valeur que son attaquant direct est le cas où l'argument et son attaquant appartiennent à un circuit sans attaquant !

Un autre exemple plus complet au niveau des circuits est le suivant :

[4]C'est-à-dire attaquants directs d'éléments des circuits : $\forall X^i$, $\exists \mathtt{C} \in \{\mathtt{C}_1, \ldots, \mathtt{C}_m, \mathtt{C}'_1, \ldots \mathtt{C}'_n\}$ tel que $\exists C \in \mathtt{C}$ et $X^i \mathcal{R} C$.

[5]S'il y a plusieurs chemins menant de X_i à A, on prendra en compte bien-sûr les $l^1_{X^i}$, $l^2_{X^i}$, … correspondantes.



Là, seuls C_1, C_2, C_3, C_4, F_2, F_4 et D sont calculés avec la règle 12 page 26. Tous les autres arguments sont pris en compte par la règle 13 page 26.

5.3 Application aux différents exemples

Reprenons les différents cas pour voir ce que cela pourrait donner.

5.3.1 Les cas sans circuit

Ici, le calcul s'effectue classiquement en partant des feuilles et en progressant vers la racine de l'arbre.

Numéro Cas	Contraintes sur les valeurs
0	$v(A) = ()$
1	$v(B) = ()$, $v(A) = (1)$
2	$v(C) = ()$, $v(B) = (1)$, $v(A) = (2)$
3	$v(B_1) = v(B_2) = ()$, $v(A) = (1, 1)$
4	$v(C_1) = v(C_2) = ()$, $v(B_1) = v(B_2) = (1)$, $v(A) = (2, 2)$
5	$v(A) = ()$, $v(B_1) = v(B_2) = (1)$
6	$v(C_1) = v(C_2) = ()$, $v(A) = (1, 1)$, $v(B_1) = v(B_2) = (2, 2)$

7	$v(A_n) = ()$, $v(A_{n-1}) = (1)$, $\ldots$, $v(A_2) = (n-2)$, $v(A_1) = (n-1)$
8	$v(B_i) = ()\ \forall i = 1 \ldots n$, $v(A) = \underbrace{(1, \ldots, 1)}_{n \text{ fois}}$
9	Il s'agit d'une généralisation des cas 6, 7 et 8 : $v(A) = (x_1, x_2, \ldots, x_p)$ avec p le nombre total de branches et x_i la longueur de chaque branche i et $v(B_j) = (x_1 + 1, x_2 + 1, \ldots, x_p + 1)$, $\forall B_j$.

On constate que l'on n'a aucun problème à calculer la valeur d'arguments appartenant à la fois à une branche de défense et à une branche d'attaque. Il reste maintenant à exploiter ces valeurs, ce qui est le sujet de la section 5.4.

5.3.2 Les cas avec circuits pairs

Numéro Cas	Contraintes sur les valeurs
10	$v(B) = v(A) = (1, 2)$
11	$v(A) = (1, 2, 1)$, $v(B_1) = (1, 2, 2)$, $v(B_2) = ()$
12	$v(A) = (1, 2, 1, 2)$, $v(B_1) = (1, 2, 2, 3)$, $v(B_2) = (1, 2, 2, 3)$
13	$v(A) = v(B_1) = (1, 2)$ et $v(B_2) = (2, 3)$

Quelques remarques sur ces résultats :
- dans le cas 10, le fait que $v(A) = v(B)$ est toujours du à l'hypothèse de départ (voir règle 1 page 7) ;
- dans le cas 12, on a désormais une différence entre d'un côté A et de l'autre B_1 et B_2 (ce qui reflète le fait que A appartient à deux circuits alors que B_1 et B_2 n'appartiennent qu'à un seul circuit).

5.3.3 Les cas avec circuits impairs

Numéro Cas	Contraintes sur les valeurs
14	$v(A) = (1, 2)$
15	$v(A) = v(B) = v(C) = (1, 2)$
16	$v(D) = ()$, $v(C) = (1, 2, 1)$, $v(A) = (1, 2, 2)$, $v(B) = (1, 2, 3)$
17	$v(A) = v(B) = v(C) = (1, 2)$ et $v(D) = (2, 3)$

5.4 Comparaison de valeurs "tuplées"

5.4.1 Introduction

Soient X et Y, 2 arguments ayant les valeurs tuplées respectives $v(X)$ et $v(Y)$, le plus simple pour comparer X et Y est d'établir une comparaison lexicographique de $v(X)$ et de $v(Y)$. Or, cette comparaison ne permet pas de conclure de manière constructive pour plusieurs raisons :
- Les branches de défense ont systématiquement une longueur supérieure aux branches d'attaque (par exemple, comment exploiter le fait que $v(X) = (1, 1, 1, 1, 1, 1) <_{lex} v(Y) = (2)$). Une solution serait alors

de ramener les longueurs des branches d'attaque et celles des branches de défense à une même "unité" (par exemple en diminuant de 1 la longueur de toutes les branches de défense).
  Mais alors, on se retrouve à ne plus pouvoir distinguer des cas pourtant bien dissemblables[6] : B, dont la valeur $v(B) = (2)$ devient $v(B) = (1)$, ne se distinguera plus de l'argument A dont la valeur d'origine $v(A) = (1)$ (A étant attaqué sans défense, alors que B était attaqué puis défendu).
- D'autre part, il faudrait pouvoir distinguer aussi un argument qui est attaqué (resp. défendu) sur beaucoup de niveaux et un argument qui est attaqué (resp. défendu) sur très peu de niveaux ($v(A) = (11)$ et $v(B) = (1)$). Or, le raisonnement n'est pas le même suivant qu'il s'agit d'une attaque ou d'une défense : plus une branche de défense est courte et plus elle est intéressante pour *valider un argument*[7], alors qu'à l'opposé, plus une branche d'attaque est courte et plus elle est intéressante *pour discréditer un argument.*
  La relation d'ordre entre arguments n'est donc pas obtenue de la même façon, suivant que l'on exploite les résultats obtenus sur l'aspect attaque ou sur l'aspect défense !

En suivant ces quelques indications, on peut déboucher sur plusieurs approches. Nous allons donc proposer ici deux sortes de comparaisons. Elles obéissent toutes les deux à la même idée de base : on fait d'abord une *comparaison quantitative* des arguments à partir des attaques et des défenses, puis, si on ne peut pas conclure parce que les "quantités sont égales", on passe à une *comparaison qualitative* des arguments toujours à partir des attaques et des défenses (en utilisant une comparaison lexicographique). La différence essentielle repose sur la manière dont on va réaliser la comparaison quantitative :
- soit en utilisant un rapport / taux entre défenses et attaques,
- soit en exploitant directement le nombre d'attaques et le nombre de défenses.

On remarquera quand même que l'utilisation d'une comparaison lexicographique lors de la phase de comparaison qualitative présente un inconvénient : c'est une méthode de comparaison locale au sens où seuls les plus petits éléments différents vont servir à trancher (en effet $(2, 10^n, \ldots, 10^n) <_{lex} (4, 4, \ldots, 4)$!).

5.4.2 Approche tuplée avec calcul d'un taux

Ici, on va travailler en partant d'un taux entre attaques et défenses et en privilégiant la minimalité du nombre global de branches.

Idées utilisées On pose :
- Plus il y a de branches d'attaque menant à un argument, et plus cet argument est "mauvais".
- Plus il y a de branches de défense menant à un argument, et plus cet argument est "bon" (avec un "bémol" : ce principe s'applique si l'argument est aussi attaqué, sinon on préfèrera qu'il y ait le moins possible de défenses – en effet, qui dit défense, dit aussi qu'il y a eu attaque –, l'idéal étant bien-sûr ni défense, ni attaque !).
- La valeur maximale sera celle d'un argument qui n'est ni attaqué, ni défendu.
- La valeur minimale sera celle d'un argument qui est attaqué par une infinité d'arguments, et jamais défendu. Cet argument n'existe bien-sûr pas ! Il sert juste de borne inférieure pour notre évaluation.
- Plus une branche d'attaque menant à un argument est longue, et moins elle a d'importance pour discréditer cet argument (donc plus elle a d'importance pour conforter cet argument).
- Plus une branche de défense menant à un argument est courte, et plus elle a d'importance pour conforter cet argument.
- La longueur des branches n'a d'importance que si le rapport entre nombre de branches d'attaque et nombre de branches de défense ne suffit pas pour comparer 2 arguments (les 2 rapports sont égaux !).

Il va donc falloir faire une "première passe" pour déterminer et *comparer d'abord le rapport entre nombre d'attaques et nombre de défenses* de chacun. Puis, si la première passe ne suffit pas pour déterminer le meilleur argument, on fera une "seconde passe" pour *rechercher la valeur maximale* (en effet, étant donné le calcul du rapport – voir ci-après –, il faut pouvoir trancher entre un argument jamais attaqué et un argument ayant une unique branche qui est de défense – ils ont le même rapport – !). Et enfin, une "troisième passe" sous la forme d'une comparaison lexicographique servira à *comparer la qualité respective des attaques et des défenses.* Ces deux dernières passes s'effectueront sur une version épurée des tuples (inutile en effet de conserver les éléments communs à chaque tuple : ils n'apporteront rien au résultat de la comparaison).

[6] Cette remarque est vraie aussi si on choisit d'augmenter de 1 la longueur de toutes les branches d'attaque.
[7] Dans le sens d'acceptation de cet argument : l'argument est meilleur.

Cette approche n'est pas prudente puisqu'elle cherche à conclure positivement quel que soit le cas (il n'y aura pas d'arguments incomparables !).

Notations Soit X un argument et $v(X)$ sa valeur, posons :
- $n_a(X)$ = nombre de branches d'attaque menant à X (c'est-à-dire nombre d'éléments impairs dans le tuple $v(X)$, et si X est une feuille alors $n_a(X) = 0$),
- $n_d(X)$ = nombre de branches de défense menant à X (c'est-à-dire nombre d'éléments pairs dans le tuple $v(X)$, et si X est une feuille alors $n_d(X) = 0$),
- $r(X) = \frac{n_d(X)-n_a(X)}{n_d(X)+n_a(X)}$ si $n_a(X) \neq 0$ ou $n_d(X) \neq 0$, sinon $r(X) = 1$ (quand $n_a(X) = 0$ ET $n_d(X) = 0$, donc quand $v(X) = ()$),
- $v'(X) = f_c(v(X))$, la fonction $f_c(t)$ consistant à ordonner le tuple t par valeurs croissantes,
- $v'_p(X) = f_p(v'(X))$, la fonction $f_p(t)$ consistant à ne garder du tuple t que les valeurs paires (celles représentant des longueurs de branches de défense),
- $v'_i(X) = f_i(v'(X))$, la fonction $f_i(t)$ consistant à ne garder du tuple t que les valeurs impaires (celles représentant des longueurs de branches d'attaque).

Justification des choix Les données $n_a(X)$, $n_d(X)$ servent à calculer le rapport $r(X)$ qui a les propriétés suivantes :
- $-1 \leq r(X) \leq 1$ (la valeur 1 correspond au cas "pas de branche d'attaque", alors que la valeur -1 correspond au cas "pas de branche de défense"),
- $r(X)$ augmente quand $n_d(X)$ augmente ou quand $n_a(X)$ diminue,
- $r(X)$ diminue quand $n_d(X)$ diminue ou quand $n_a(X)$ augmente,
- $r(X) = 0$ quand $n_a(X) = n_d(X)$.

On peut donc dire que plus $r(X)$ est grand et meilleur est l'argument X (moins d'attaque, plus de défense). Quand $r(X)$ est positif, alors il y a plus de défenses que d'attaques et à l'inverse, quand $r(X)$ est négatif alors il y a plus d'attaques que de défenses.
Quant à $v'(X)$, $v'_p(X)$, $v'_i(X)$, ils permettront la comparaison lexicographique du tuple. Remarquons que pour X dont la valeur est (), on a $v'(X) = v'_p(X) = v'_i(X) = ()$.

Définition La comparaison de 2 arguments A et B va se faire au travers de la comparaison des nombres $r(A)$ et $r(B)$ d'une part, et d'autre part au travers de la comparaison lexicographique des tuples $v(A)$ et $v(B)$ quand la première comparaison n'aura pas permis de trancher.
On utilisera la définition suivante :

Définition 13 *Soient deux tuples $t = \{x_1, \ldots, x_n\}$ et $t' = \{y_1, \ldots, y_p\}$. On définit $t - t'$ comme étant le tuple construit de la manière suivante :*

```
début
  t − t' = ()
  tant qu'il reste un xᵢ ∈ t faire
    n_t ← nombre d'occurrences de xᵢ dans t
    n_t' ← nombre d'occurrences de xᵢ dans t'
    n ← n_t − n_t'
    si n < 0 alors n ← 0
    mettre dans t − t' n occurrences de xᵢ
    enlever toutes les occurrences de xᵢ de t
fin
```

Puis on utilise l'algorithme 1 page suivante pour définir la relation entre les arguments en fonction de leurs valeurs tuplées.
Remarquons que le seul cas où l'on *force l'égalité* entre 2 arguments A et B est le suivant : A et B ont le même "rapport" entre attaques et défenses et l'un des deux arguments a *à la fois* de plus petites branches de défense et de plus petites branches d'attaque que l'autre. Par exemple, A a pour valeur $(1, 2, 51, 102)$, alors que B a pour valeur $(n, p, 31, 52)$ avec $1 < n < 31$ et $2 < p < 52$; ici, A est un argument avec sa plus petite défense forte mais aussi sa plus petite attaque forte, alors que B est un argument avec sa plus petite défense faible mais aussi sa plus petite attaque faible.
Attention, avec les notations choisies ici, $A \sim B$ peut signifier deux choses bien différentes : soit A et B ont la même valeur, soit leurs valeurs sont incomparables et on *impose* alors que A et B soient considérés comme "identiques". Ainsi, on obtient une relation totale entre arguments (tout argument est relié à tout

Algorithme 1: Comparaison de 2 tuples en utilisant les taux

% **Description des paramètres :** %
% $v(A)$, $v(B)$: 2 tuples %

début
- **si** $v(A) = v(B)$ **alors** $A \sim B$ % Cas 1 %
- **sinon**
  - **si** $r(A) > r(B)$ **alors** $A \succ B$ % Cas 2 %
  - **sinon**
    - **si** $r(A) < r(B)$ **alors** $A \prec B$ % Cas 3 %
    - **sinon**
      - % $r(A) = r(B)$ donc recherche de la valeur maximale () après épuration %
      - $v_e(A) \leftarrow v(A) - v(B)$ % on garde les éléments de $v(A)$ qui ne sont pas dans $v(B)$ %
      - $v_e(B) \leftarrow v(B) - v(A)$ % on garde les éléments de $v(B)$ qui ne sont pas dans $v(A)$ %
      - **si** $v_e(A) = ()$ **alors** $A \succ B$ % Cas 4 %
      - **sinon**
        - **si** $v_e(B) = ()$ **alors** $A \prec B$ % Cas 5 %
        - **sinon**
          - % comparaisons lexicographiques entre d'une part les tuples $v'_{e_p}(A)$ et $v'_{e_p}(B)$ %
          - % et d'autre part les tuples $v'_{e_i}(A)$ et $v'_{e_i}(B)$ avec les notations : %
          - % $v'_{e_p}(X)$ tuple ordonné par ordre croissant des valeurs paires de $v_e(X)$ %
          - % $v'_{e_i}(X)$ tuple ordonné par ordre croissant des valeurs impaires de $v_e(X)$ %
          - **si** $v'_{e_p}(A) \leq_{lex} v'_{e_p}(B)$ *et* $v'_{e_i}(A) \geq_{lex} v'_{e_i}(B)$ **alors** $A \succ B$ % Cas 6 %
          - **sinon**
            - **si** $v'_{e_p}(A) \geq_{lex} v'_{e_p}(B)$ *et* $v'_{e_i}(A) \leq_{lex} v'_{e_i}(B)$ **alors** $A \prec B$ % Cas 7 %
            - **sinon** $A \sim B$ % on décide de ne pas trancher ! – Cas 8 %

fin

autre argument). Par contre, on remarquera que cette évaluation n'est plus une mesure mais plutôt un ordonnancement.

> Par contre, on verra en section 5.4.5 page 40 que cette relation *n'est pas une relation d'ordre*, ce qui peut poser quelques problèmes !

5.4.3 Approche tuplée avec comparaison du nombre des branches

Dans cette approche beaucoup plus simple, mais moins “complète” sur certains points (voir section 5.4.4 page 34), on prend en compte *directement* le nombre d'attaques et de défenses, sans passer par l'intermédiaire d'un taux. De plus, on ne cherche pas à optimiser en fonction du nombre global de branches.

Idées utilisées On pose :
- Plus il y a de branches d'attaque menant à un argument, et plus cet argument est “mauvais”.
- Plus il y a de branches de défense menant à un argument, et plus cet argument est “bon”.
- La valeur maximale sera celle d'un argument qui n'est ni attaqué, ni défendu.
- La valeur minimale sera celle d'un argument qui est attaqué par une infinité d'arguments, et jamais défendu. Cet argument n'existe bien-sûr pas ! Il sert juste de borne inférieure pour notre évaluation.
- Plus une branche d'attaque menant à un argument est longue, et moins elle a d'importance pour discréditer cet argument (donc plus elle a d'importance pour conforter cet argument).
- Plus une branche de défense menant à un argument est courte, et plus elle a d'importance pour conforter cet argument.
- La longueur des branches n'a d'importance que si le nombre de branches de chaque nature ne suffit pas pour comparer 2 arguments.

Il va donc falloir faire une “première passe” pour déterminer et *comparer d'abord le nombre d'attaques et de défenses* de chacun. Si ces deux critères sont en accord ou en désaccord, alors on pourra conclure. Sinon, si la première passe ne suffit pas pour déterminer le meilleur argument (les deux critères sont identiques), on fera une “seconde passe” sous la forme d'une comparaison lexicographique qui servira à *comparer la qualité respective des attaques et des défenses* (ici, il n'y a plus de phase d'épuration). Nous adoptons ici une *attitude prudente*. Donc, nous allons avoir plusieurs cas où nous refuserons de conclure (contrairement au type de comparaison utilisant les taux et présenté en section 5.4.2 page 30 pour lequel nous "forcions" le résultat de la comparaison pour éviter les indécisions).

Notations Soit X un argument et $v(X)$ sa valeur, posons :
- $n_a(X)$ = nombre de branches d'attaque menant à X (c'est-à-dire nombre d'éléments impairs dans le tuple $v(X)$), en posant que $n_a(X) = 0$ si X est une feuille ;
- $n_d(X)$ = nombre de branches de défense menant à X (c'est-à-dire nombre d'éléments pairs dans le tuple $v(X)$), en posant que $n_d(X) = \infty$ si X est une feuille ;
- $v'(X) = f_c(v(X))$, la fonction $f_c(t)$ consistant à ordonner le tuple t par valeurs croissantes,
- $v'_p(X) = f_p(v'(X))$, la fonction $f_p(t)$ consistant à ne garder du tuple t que les valeurs paires (celles représentant des longueurs de branches de défense),
- $v'_i(X) = f_i(v'(X))$, la fonction $f_i(t)$ consistant à ne garder du tuple t que les valeurs impaires (celles représentant des longueurs de branches d'attaque).

Justification des choix Les données $n_a(X)$, $n_d(X)$ servent à effectuer la comparaison *quantitative* (pour savoir si X est meilleur que Y, il faut que X ait moins d'attaques ET plus de défenses que Y).
Quant à $v'(X)$, $v'_p(X)$, $v'_i(X)$, ils permettront la comparaison lexicographique des tuples dans le cas où X et Y auraient le même nombre d'attaques et de défenses. Remarquons que pour X dont la valeur est (), on a $v'(X) = v'_p(X) = v'_i(X) = ()$.

Définition La comparaison de 2 arguments A et B va se faire avec l'algorithme 2.

Algorithme 2: Comparaison de 2 tuples en utilisant les comparaisons des nombres de branches

% **Description des paramètres :** %
% $v(A)$, $v(B)$: 2 tuples %

début
- **si** $v(A) = v(B)$ **alors** $A \succeq B$ ET $B \succeq A$ % Cas 1 %
- **sinon**
  - **si** $n_a(A) = n_a(B)$ *et* $n_d(A) = n_d(B)$ **alors**
    - % comparaisons lexicographiques entre $v'_p(A)$ et $v'_p(B)$ et entre $v'_i(A)$ et $v'_i(B)$ %
    - **si** $v'_p(A) \leq_{lex} v'_p(B)$ *ET* $v'_i(A) \geq_{lex} v'_i(B)$ **alors** $A \succ B$ % cas 2 %
    - **sinon**
      - **si** $v'_p(A) \geq_{lex} v'_p(B)$ *ET* $v'_i(A) \leq_{lex} v'_i(B)$ **alors** $A \prec B$ % cas 3 %
      - **sinon** $A \not\succeq B$ et $A \not\preceq B$ % On décide de ne pas trancher ! cas 4 %
  - **sinon**
    - **si** $n_a(A) \geq n_a(B)$ *ET* $n_d(A) \leq n_d(B)$ **alors** $A \prec B$ % cas 5 %
    - **sinon**
      - **si** $n_a(A) \leq n_a(B)$ *ET* $n_d(A) \geq n_d(B)$ **alors** $A \succ B$ % cas 6 %
      - **sinon** $A \not\succeq B$ et $A \not\preceq B$ % On décide de ne pas trancher ! Cas 7 %

fin

Il s'agit de comparer A et B au travers de la comparaison des nombres $n_a(A)$ et $n_a(B)$ d'une part, et $n_d(A)$ et $n_d(B)$ d'autre part. Puis, si on ne peut pas conclure, au travers de la comparaison lexicographique des tuples $v(A)$ et $v(B)$.
Le cas où $A \succeq B$ ET $B \succeq A$, noté $A \sim B$, signifie alors que A et B sont "équivalents". Ici, le seul cas où A et B sont considérés "équivalents" est celui où leurs valeurs sont égales (équivalence structurelle).
Remarquons qu'il y a ici deux cas où l'on considère que les 2 arguments A et B sont incomparables (et non "identiques" comme dans l'approche des tuples avec taux) :
- soit A et B ont les mêmes nombres d'attaques et de défenses et l'un des deux arguments a *à la fois* de plus courtes branches de défense et de plus courtes branches d'attaque que l'autre[8]. Par exemple, A a pour valeur $(1, 2, 51, 102)$, alors que B a pour valeur $(n, p, 31, 52)$ avec $1 < n < 31$ et $2 < p < 52$; ici, A est un argument avec sa plus petite défense forte mais aussi sa plus petite attaque forte, alors que B est un argument avec sa plus petite défense faible mais aussi sa plus petite attaque faible.

[8] Ce cas traité par l'approche tuplée avec taux débouchera sur une conclusion différente : A et B sont "identiques", $A \sim B$.

– soit l'un des deux arguments a à la fois un plus grand nombre d'attaque ET un plus grand nombre de défense que l'autre.

Cela vient du fait que nous adoptons ici une vision *prudente* de l'agrégation des deux critères que sont les défenses et les attaques (tant du point de vue quantitatif, que qualitatif).

On verra en section 5.4.5 page 40 que cette relation *est une relation d'ordre*, ce qui est donc plus satisfaisant que celle donnée en section 5.4.2 page 30 !

5.4.4 Application aux différents exemples

Les deux approches par tuples choisies nous amènent toutes les deux aux mêmes résultats sur les exemples suivants. Il vaudra donc mieux privilégier celle donnée en section 5.4.3 page 32 puisqu'elle définit une relation d'ordre.

Les cas sans circuit

Numéro Cas	Contraintes sur les valeurs
0	$v(A) = ()$ Ici, soit $r(A) = 1$, soit $n_a(A) = 0$ et $n_d(A) = \infty$
1	$v(B) = ()$, $v(A) = (1)$ Ici, – soit parce que $r(B) = 1 > r(A) = -1$, – soit parce que $n_a(A) = 1$, $n_d(A) = 0$, $n_a(B) = 0$ et $n_d(B) = \infty$, on a : $B \succ A$
2	$v(C) = ()$, $v(B) = (1)$, $v(A) = (2)$ Ici, – soit parce que $r(C) = r(A) = 1 > r(B) = -1$, et que () est la valeur max, – soit parce que $n_a(A) = 0$, $n_d(A) = 1$, $n_a(B) = 1$, $n_d(B) = 0$, $n_a(C) = 0$ et $n_d(C) = \infty$, on a : $C \succ A \succ B$
3	$v(B_1) = v(B_2) = ()$, $v(A) = (1, 1)$ Ici, – soit parce que $r(B_1) = r(B_2) = 1 > r(A) = -1$, – soit parce que $n_a(A) = 2$, $n_d(A) = 0$, $n_a(B_1) = n_a(B_2) = 0$ et $n_d(B_1) = n_d(B_2) = \infty$, on a : $B_1 \sim B_2 \succ A$
4	$v(C_1) = v(C_2) = ()$, $v(B_1) = v(B_2) = (1)$, $v(A) = (2, 2)$ Ici, – soit parce que $r(C_1) = r(C_2) = r(A) = 1 > r(B_1) = r(B_2) = -1$ et que () est la valeur max, – soit parce que $n_a(A) = 0$, $n_d(A) = 2$, $n_a(B_1) = n_a(B_2) = 1$ et $n_d(B_1) = n_d(B_2) = 0$, $n_a(C_1) = n_a(C_2) = 0$ et $n_d(C_1) = n_d(C_2) = \infty$, on a : $C_1 \sim C_2 \succ A \succ B_1 \sim B_2$
5	$v(A) = ()$, $v(B_1) = v(B_2) = (1)$ Ici, – soit parce que $r(A) = 1 > r(B_1) = r(B_2) = -1$, – soit parce que $n_a(A) = 0$, $n_d(A) = \infty$, $n_a(B_1) = n_a(B_2) = 1$ et $n_d(B_1) = n_d(B_2) = 0$, on a : $A \succ B_1 \sim B_2$
6	$v(C_1) = v(C_2) = ()$, $v(A) = (1, 1)$, $v(B_1) = v(B_2) = (2, 2)$ Ici, – soit parce que $r(C_1) = r(C_2) = r(B_1) = r(B_2) = 1 > r(A) = -1$, et que () est la valeur max, – soit parce que $n_a(A) = 2$, $n_d(A) = 0$, $n_a(B_1) = n_a(B_2) = 0$ et $n_d(B_1) = n_d(B_2) = 2$, $n_a(C_1) = n_a(C_2) = 0$ et $n_d(C_1) = n_d(C_2) = \infty$, on a : $C_1 \sim C_2 \succ B_1 \sim B_2 \succ A$

7	$v(A_n) = ()$, $v(A_{n-1}) = (1)$, ... , $v(A_2) = (n-2)$, $v(A_1) = (n-1)$ Ici, – soit parce que $r(A_i) = (-1)^{n-i}$, – soit parce que $n_a(A_i) = 1$ et $n_d(A_i) = 0$ si A_i est un "attaquant" ou $n_a(A_i) = 0$ et $n_d(A_i) = 1$ si A_i est un "défenseur", et $(n-i-2) < (n-i)$ quel que soit i on a 2 résultats (un pour les "attaquants" et un pour les "défenseurs") : $A_n \succ A_{n-2} \succ \ldots \succ A_1$ (si n est impair, sinon A_2) et $A_{n-1} \prec A_{n-3} \prec \ldots \prec A_2$ (si n est impair, sinon A_1). Cela nous donne donc 2 suites que l'on peut mixer de la façon suivante : Si n impair, $A_n \succ A_{n-2} \succ \ldots \succ A_1 \succ A_2 \succ \ldots \succ A_{n-3} \succ A_{n-1}$ et si n pair, $A_n \succ A_{n-2} \succ \ldots \succ A_2 \succ A_1 \succ \ldots \succ A_{n-3} \succ A_{n-1}$
8	$v(B_i) = ()\ \forall i = 1 \ldots n$, $v(A) = \underbrace{(1, \ldots, 1)}_{n \text{ fois}}$ Ici, – soit parce que $r(B_i) = 1 > r(A) = -1$, – soit parce que $n_a(A) = n$, $n_d(A) = 0$, $n_a(B_i) = 0$ et $n_d(B_i) = \infty\ \forall i$, on a : $B_1 \sim \ldots \sim B_n \succ A$

Le cas 9 n'est pas traité ici car trop général.
Remarquons que le cas 7 permet d'obtenir un ordre semblable à celui de [BH01], mais sans point fixe (donc pas de convergence). L'ordre issu des tuples ne respectera pas donc pas la règle 5 page 12 proposée pour l'approche 0.

Les cas avec circuits pairs

Numéro Cas	Contraintes sur les valeurs
10	$v(B) = v(A) = (1, 2)$ Ici, les 2 valeurs sont égales donc inutile de calculer quoi que ce soit de plus : $A \sim B$.
11	$v(A) = (1, 2, 1)$, $v(B_1) = (1, 2, 2)$, $v(B_2) = ()$ Ici, – soit parce que $r(B_2) = 1 > r(B_1) = 1/3 > r(A) = -1/3$, – soit parce que $n_a(A) = 2$, $n_d(A) = 1$, $n_a(B_1) = n_a(B_2) = 1$ et $n_d(B_1) = 2$ et $n_d(B_2) = \infty$, on a : $B_2 \succ B_1 \succ A$
12	$v(A) = (1, 2, 1, 2)$, $v(B_1) = (1, 2, 2, 3)$, $v(B_2) = (1, 2, 2, 3)$ Ici, – soit parce que $r(B_1) = r(B_2) = r(A) = 0$, – soit parce que $n_a(A) = n_a(B_1) = n_a(B_2) = 2$ et $n_d(A) = n_d(B_1) = n_d(B_2) = 2$, et avec $(1, 1) \leq_{lex} (1, 3)$ donc $B_1 \sim B_2 \succ A$
13	$v(A) = v(B_1) = (1, 2)$ et $v(B_2) = (2, 3)$ Ici, – soit parce que $r(B_1) = r(B_2) = r(A) = 0$, – soit parce que $n_a(A) = n_a(B_1) = n_a(B_2) = 1$ et $n_d(A) = n_d(B_1) = n_d(B_2) = 1$, et avec $(1) \leq_{lex} (3)$ on a : $A \sim B_1 \prec B_2$.

Les cas avec circuits impairs

Numéro Cas	Contraintes sur les valeurs
14	$v(A) = (1, 2)$ Aucune comparaison à faire.

15	$v(A) = v(B) = v(C) = (1,2)$ Ici, les 3 valeurs sont égales donc inutile de calculer quoi que ce soit de plus : $A \sim B \sim C$.
16	$v(D) = ()$, $v(C) = (1,2,1)$, $v(A) = (1,2,2)$, $v(B) = (1,2,3)$ Ici, – soit parce que $r(D) = 1 > r(A) = 1/3 > r(B) = r(C) = -1/3$, – soit parce que $n_a(A) = 1$, $n_a(B) = n_a(C) = 2$, $n_a(D) = 0$ et $n_d(A) = 2$, $n_d(B) = n_d(C) = 1$, $n_d(D) = \infty$, et avec, pour départager B et C, $(1,1) <_{lex} (1,3)$ on a : $D \succ A \succ B \succ C$
17	$v(A) = v(B) = v(C) = (1,2)$ et $v(D) = (2,3)$ Ici, on a – soit $r(D) = r(A) = r(B) = r(C)$, – soit $n_a(A) = n_a(B) = n_a(C) = n_a(D) = 1$, $n_d(A) = n_d(B) = n_d(C) = n_d(D) = 1$, et avec $(1) \leq_{lex} (3)$ on a donc : $A \sim B \sim C \prec D$.

Remarque : Pour les exemples présentés ici, quelle que soit l'approche utilisée (avec taux ou avec comparaison des nombres), on obtient les mêmes résultats.

Comparaison avec l'approche 0 et l'ordre de [BH01] On constate que :

- sur les cas sans circuit :
  - on obtient avec les approches par tuples les mêmes résultats qu'avec [BH01], au "bémol" près qui concerne le cas numéro 7 pour lequel nous n'avons pas de point fixe avec les approches par tuples, alors qu'il y en a un avec [BH01] ;
  - de même, les différences essentielles entre l'approche 0 et les approches par tuples se situent :
    - sur le fait que l'approche 0 correspond à un ensemble d'évaluations parmi lesquelles on peut avoir des inégalités non strictes, ce qui n'est pas le cas des approches par tuples ;
    - sur le cas numéro 7 : là, les résultats obtenus sont en partie contradictoires avec ceux de l'approche 0.
- pour les cas avec circuits :
  - on ne fait pas de comparaison avec [BH01] puisque cette approche ne prend pas en compte les circuits ;
  - d'autre part, avec les approches par tuples (quel que soit le principe de comparaison – avec taux ou avec comparaison des nombres de branches), on obtient des résultats un peu différents de ceux obtenus par l'approche 0 ; en effet, dans certains cas, les approches par tuples fournissent des inégalités strictes alors que dans l'approche 0 fournit soit des inégalités non strictes, soit des égalités (ces égalités sont dues au fait que quand un argument est l'aboutissement d'une branche d'attaque et d'une branche de défense, on doit "forcer l'égalité" pour rester compatible avec les principes proposés – voir section 3 page 7).

Malgré cela, dans la plupart des cas, les résultats obtenus confirment "en gros", ou affinent les intuitions que nous avons concernant la comparaison des divers arguments.

Comparaison des deux approches par tuples Étudions encore quelques exemples :

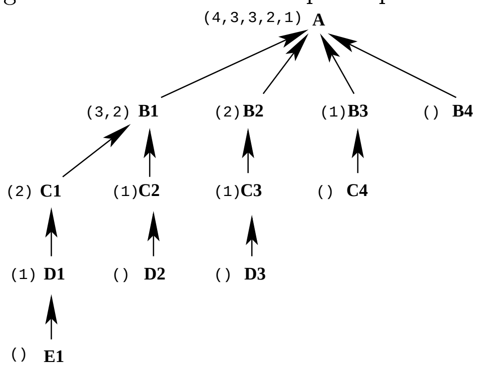
19. Prenons A_1 attaqué par B et non défendu et A_2 attaqué par B_1 et B_2 et non défendu. On a alors $v(A_1) = (1)$ et $v(A_2) = (1,1)$. En appliquant les règles précédentes, on a
    - avec l'approche des tuples avec taux, on obtient : $r(A_1) = r(A_2) = -1$ et on compare (1) avec $(1,1)$: or, après épuration on a $v_e(A_1) = ()$ et $v_e(A_2) = (1)$, donc $A_1 \succ A_2$.
    - avec l'approche des tuples avec comparaison directe des nombres, on obtient : $n_a(A_1) = 1$ et $n_a(A_2) = 2$ avec $n_d(A_1) = n_d(A_2) = 0$, donc $A_1 \succ A_2$.

    On a donc le même résultat.
20. Prenons A_1 attaqué par B et défendu par C et A_2 attaqué par B_1 et B_2 et défendu par C_1 contre B_1 et par C_2 contre B_2. On a alors $v(A_1) = (2)$ et $v(A_2) = (2,2)$. En appliquant les règles précédentes, on a
    - avec l'approche des tuples avec taux, on obtient : $r(A_1) = r(A_2) = 1$ et on compare (2) avec $(2,2)$: or, après épuration on a $v_e(A_1) = ()$ et $v_e(A_2) = (2)$, donc $A_1 \succ A_2$.

– avec l'approche des tuples avec comparaison directe des nombres, on obtient : $n_d(A_1) = 1$ et $n_d(A_2) = 2$ avec $n_a(A_1) = n_a(A_2) = 0$, donc $A_1 \prec A_2$.

Ici, on a un résultat totalement opposé. Cette différence de traitement provient du fait que l'approche par taux procède à une phase d'épuration qui consiste essentiellement *dans le cadre de l'exemple donné ici* à privilégier les arguments qui ont le moins de branches (car qui dit branche, même s'il s'agit d'une branche de défense, dit qu'il y a eu attaque !). Ce mécanisme s'applique aussi dans l'exemple 19 mais là il "va dans le même sens" que l'approche avec comparaison directe des nombres, puisque les branches sont des branches d'attaque.

21. Reprenons l'exemple 18 page 12 avec son évaluation par tuples :



Ici,

– avec l'approche des tuples avec taux, on a sur le graphe complet :

	E_1	D_1	C_1	B_1	D_2	C_2	D_3	C_3	B_2	C_4	B_3	B_4	A
r	1	-1	1	0	1	-1	1	-1	1	1	-1	1	$-1/5$

Et si on se restreint à la partie entourée (les branches $E_1 - \ldots - A$ et $D_2 - \ldots - A$) le seul changement porte sur $r(A) = 0$ (puisque sa valeur est $(4, 3)$).

Puis, sur le graphe complet, on lève les indécisions (entre E_1 et C_1, puis entre D_3 et B_2) en utilisant les épurations puis éventuellement les comparaisons lexicographiques et on obtient les relations suivantes :

$$E_1 \sim D_2 \sim D_3 \sim C_4 \sim B_4$$

$$\succ$$

$$C_1 \sim B_2$$

$$\succ$$

$$B_1$$

$$\succ$$

$$A$$

$$\succ$$

$$D_1 \sim C_2 \sim C_3 \sim B_3$$

De la même façon, mais cette fois sur la partie entourée du graphe, on lève les indécisions (entre E_1 et C_1, puis entre B_1 et A) en utilisant les comparaisons lexicographiques et on obtient les relations suivantes :

$$E_1 \sim D_2 \succ C_1 \succ B_1 \succ A \succ D_1 \sim C_2$$

Remarquons aussi que A est moins "bon" que son défenseur C_1, ce qui est plus satisfaisant que l'ordre issu de [BH01].

– avec l'approche des tuples avec comparaison directe des nombres, on a sur le graphe complet :

$$\begin{array}{ccc} E_1 \sim D_2 \sim D_3 \sim C_4 \sim B_4 & & E_1 \sim D_2 \sim D_3 \sim C_4 \sim B_4 \\ \succ & & \succ \\ C_1 \sim B_2 & & C_1 \sim B_2 \\ \succ & \text{mais aussi} & \succ \\ B_1 & & A \\ \succ & & \succ \\ D_1 \sim C_2 \sim C_3 \sim B_3 & & D_1 \sim C_2 \sim C_3 \sim B_3 \end{array}$$

A et B_1 sont ici incomparables (cas 7 de l'algorithme 2 page 33).
De la même façon, mais cette fois sur la partie entourée du graphe, on obtient les relations suivantes :

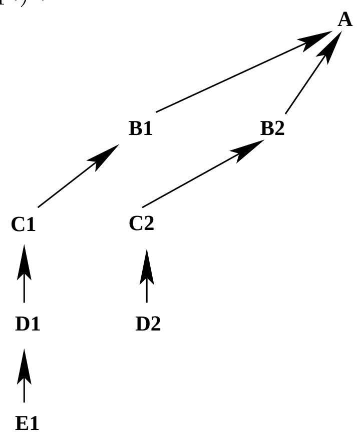
$$E_1 \sim D_2 \succ C_1 \succ B_1 \succ A \succ D_1 \sim C_2$$

A et B_1 sont ici comparables, ce qui n'est pas surprenant, par rapport au résultat obtenu sur la totalité du graphe, puisque A *a ici changé de valeur* !
Ici-aussi A est moins "bon" que son défenseur C_1, ce qui est plus satisfaisant que l'ordre issu de [BH01].

22. Dans l'exemple 18 page 12, si on se polarise sur la partie entourée, on trouve :

- $v(A) = (4, 3)$ avec la technique des tuples,
- $v(A) = \frac{13}{19}$ avec la technique de [BH01].

Or, modifions légérement le graphe de telle manière qu'il y ait désormais 2 branches menant à A et se "joignant en A" (et pas en B_1 !) :



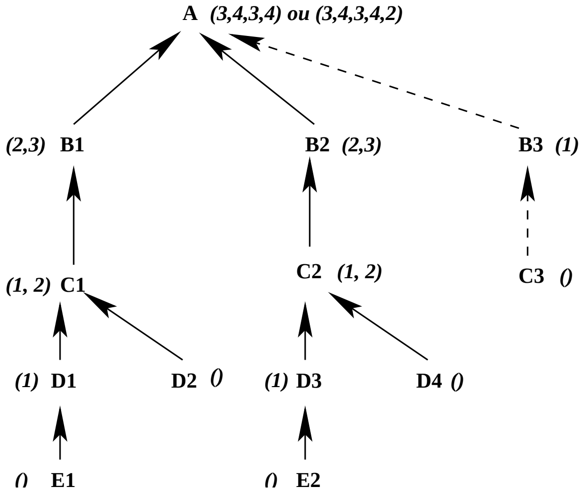
On trouve désormais les valeurs suivantes :

- $v(A) = (4, 3)$ avec la technique des tuples,
- $v(A) = \frac{15}{34}$ avec la technique de [BH01].

On constate donc, qu'avec les approches par tuples, $v(A)$ ne change pas, alors, qu'avec l'approche proposée par [BH01], $v(A)$ diminue.

Pourquoi une telle différence ? Les approches par tuples considèrent chaque branche entre A et une feuille du graphe comme indépendante des autres branches (dans l'exemple, le fait qu'il y ait une arête commune aux deux branches $A - B_1 - C_1 - D_1 - E_1$ et $A - B_1 - C_2 - D_2$ ne change pas le calcul de la valeur de A !). Effectivement, par définition (voir les règles 12 page 26 à 13 page 26), le fait que certains arcs soient communs à différentes branches (quelle que soit leur nature) n'entre absolument pas en ligne de compte pour le calcul de la valeur ! Seul compte le nombre de branches distinctes menant à l'argument dont on calcule la valeur ainsi que la longueur de ces branches. Cela sera confirmé par la propriété 4 page 40.

23. Soit le graphe des contrariétés suivant :



Si on utilise les règles de comparaison, on obtient :
- si la branche $C_3 - B_3 - A$ est présente :
  - pour l'approche tuplée avec taux : on obtient la relation entre arguments suivante :
  $$\begin{array}{c} E_1 \sim E_2 \sim D_2 \sim D_4 \sim C_3 \\ \succ \\ A \\ \succ \\ B_1 \sim B_2 \\ \succ \\ C_1 \sim C_2 \\ \succ \\ D_1 \sim D_3 \sim B_3 \end{array}$$
  - pour l'approche tuplée avec comparaison du nombre de branches : on obtient la relation entre arguments suivante :
  $$\begin{array}{c} E_1 \sim E_2 \sim D_2 \sim D_4 \sim C_3 \\ \succ \\ B_1 \sim B_2 \\ \succ \\ C_1 \sim C_2 \\ \succ \\ D_1 \sim D_3 \sim B_3 \end{array}$$
  Par contre A n'est comparable qu'avec E_1, E_2, D_2, D_4, C_3. On a donc $E_1 \sim E_2 \sim D_2 \sim D_4 \sim C_3 \succ A$
- si la branche $C_3 - B_3 - A$ est absente :
  - pour l'approche tuplée avec taux : on obtient la relation entre arguments suivante :
  $$\begin{array}{c} E_1 \sim E_2 \sim D_2 \sim D_4 \sim C_3 \\ \succ \\ B_1 \sim B_2 \sim A \\ \succ \\ C_1 \sim C_2 \sim A \\ \succ \\ D_1 \sim D_3 \sim B_3 \end{array}$$
  Ce qui est pour le moins suspect en ce qui concerne la place de A ! Cela est du au fait que la relation $\sim$ a deux significations bien distinctes et incompatibles (voir section 5.4.2 page 30).
  - pour l'approche tuplée avec comparaison du nombre de branches : on obtient la même chose que si la branche $C_3 - B_3 - A$ était présente :

$$E_1 \sim E_2 \sim D_2 \sim D_4 \sim C_3$$
$$\succ$$
$$B_1 \sim B_2$$
$$\succ$$
$$C_1 \sim C_2$$
$$\succ$$
$$D_1 \sim D_3 \sim B_3$$

Donc A est toujours incomparable avec les autres arguments, excepté avec E_1, E_2, D_2, D_4, C_3. Sur cet exemple, on constate que l'approche tuplée avec taux propose plus de relations entre arguments que l'approche tuplée avec comparaison. Par contre, certaines de ces relations sont surprenantes (voir dans l'exemple ci-dessus, le cas de A) et incompatibles avec la notion de relation d'ordre ! Ceci sera d'ailleurs confirmé par la propriété 5.

5.4.5 Propriétés des évaluations par tuples

Soit V l'ensemble des multi-ensembles d'entiers et soit v l'application de $\mathcal{A}$ dans V, on a quelques propriétés concernant les 2 approches par tuples :

Propriété 4 (Indépendance des branches) *Soit un argument A ayant pour attaquants directs A_1 de valeur $v(A_1) = (a_1^1, \ldots, a_{m_1}^1)$, ... , A_n de valeur $v(A_n) = (a_1^n, \ldots, a_{m_n}^n)$. Soit l'argument A' ayant pour attaquants directs A_1^1 de valeur $v(A_1^1) = (a_1^1)$, ... , $A_{m_1}^1$ de valeur $v(A_{m_1}^1) = (a_{m_1}^1)$, ... , A_1^n de valeur $v(A_1^n) = (a_1^n)$, ... , $A_{m_n}^n$ de valeur $v(A_{m_n}^n) = (a_{m_n}^n)$. Alors $v(A) = v(A')$.*

Preuve : C'est immédiat et cela découle directement des règles de calcul 12 page 26 et 13 page 26. □

Cette proposition montre bien l'aspect "indépendant" de la prise en compte de chaque branche même quand les dites-branches ne sont pas indépendantes graphiquement !

Propriété 5 *La fonction d'évaluation par tuples avec taux définie par l'algorithme 1 page 32 n'induit pas un pré-ordre sur l'ensemble des arguments.*

Preuve : Il suffit d'un contre-exemple montrant l'absence de transitivité : soient les trois arguments A, B et C ayant pour valeurs respectives $(3, 6)$, $(2, 3, 5, 6)$ et $(1, 4)$. On a bien $A \succ B$ et $B \succ C$ mais on n'a pas $A \succ C$ (on est ici dans le cas numéro 8 de l'algorithme 1 page 32 : $A \sim C$!). □

Propriété 6 *La fonction d'évaluation par tuples avec comparaison des nombres de branches définie par l'algorithme 2 page 33 induit un pré-ordre partiel $\succeq$ sur l'ensemble des arguments dont les éléments maximaux seront les feuilles ayant pour valeur $()$.*

Preuve : La relation $\succeq$ définie par l'algorithme 2 page 33 est :

- réflexive : $A \succeq A$ car $v(A) = v(A)$, donc on a $A \succeq A$ ET $A \succeq A$ (cas 1 de l'algorithme 2 page 33) ;
- transitive : Posons $A \succeq B$ et $B \succeq C$ et examinons tous les cas possibles :
  - soit $v(A) = v(B)$:
    - soit $v(B) = v(C)$: on a alors $v(A) = v(C)$ donc $A \succeq C$,
    - soit $n_a(B) \leq n_a(C)$ ET $n_d(B) > n_d(C)$: on a donc $n_a(B) = n_a(A) \leq n_a(C)$ ET $n_d(B) = n_d(A) > n_d(C)$, donc $A \succeq C$,
    - soit $n_a(B) < n_a(C)$ ET $n_d(B) \geq n_d(C)$: on a donc $n_a(B) = n_a(A) < n_a(C)$ ET $n_d(B) = n_d(A) \geq n_d(C)$, donc $A \succeq C$,
    - soit $n_a(B) = n_a(C)$ ET $n_d(B) = n_d(C)$ ET $v_p(B) \leq_{lex} v_p(C)$ ET $v_i(B) \geq_{lex} v_i(C)$: on a donc $n_a(B) = n_a(A) = n_a(C)$ ET $n_d(B) = n_d(A) = n_d(C)$ ET $v_p(B) = v_p(A) \leq_{lex} v_p(C)$ ET $v_i(B) = v_i(A) \geq_{lex} v_i(C)$, donc $A \succeq C$;
  - soit $n_a(A) \leq n_a(B)$ ET $n_d(A) > n_d(B)$:
    - soit $v(B) = v(C)$: on a alors $n_a(A) \leq n_a(B) = n_a(C)$ ET $n_d(A) > n_d(B) = n_d(C)$ donc $A \succeq C$,
    - soit $n_a(B) \leq n_a(C)$ ET $n_d(B) > n_d(C)$: on a donc $n_a(A) \leq n_a(B) \leq n_a(C)$ ET $n_d(A) > n_d(B) > n_d(C)$, donc $A \succeq C$,
    - soit $n_a(B) < n_a(C)$ ET $n_d(B) \geq n_d(C)$: on a donc $n_a(A) \leq n_a(B) < n_a(C)$ ET $n_d(A) > n_d(B) \geq n_d(C)$, donc $A \succeq C$,

- soit $n_a(B) = n_a(C)$ ET $n_d(B) = n_d(C)$: on a donc $n_a(A) \leq n_a(B) = n_a(C)$ ET $n_d(A) > n_d(B) = n_d(C)$, donc $A \succeq C$;
- soit $n_a(A) < n_a(B)$ ET $n_d(A) \geq n_d(B)$:
  - soit $v(B) = v(C)$: on a alors $n_a(A) < n_a(B) = n_a(C)$ ET $n_d(A) \geq n_d(B) = n_d(C)$ donc $A \succeq C$,
  - soit $n_a(B) \leq n_a(C)$ ET $n_d(B) > n_d(C)$: on a donc $n_a(A) < n_a(B) \leq n_a(C)$ ET $n_d(A) \geq n_d(B) > n_d(C)$, donc $A \succeq C$,
  - soit $n_a(B) < n_a(C)$ ET $n_d(B) \geq n_d(C)$: on a donc $n_a(A) < n_a(B) < n_a(C)$ ET $n_d(A) \geq n_d(B) \geq n_d(C)$, donc $A \succeq C$,
  - soit $n_a(B) = n_a(C)$ ET $n_d(B) = n_d(C)$: on a donc $n_a(A) < n_a(B) = n_a(C)$ ET $n_d(A) \geq n_d(B) = n_d(C)$, donc $A \succeq C$;
- soit $n_a(A) = n_a(B)$ ET $n_d(A) = n_d(B)$ ET $v_p(A) \leq_{lex} v_p(B)$ ET $v_i(A) \geq_{lex} v_i(B)$:
  - soit $v(B) = v(C)$: on a alors $n_a(A) = n_a(B) = n_a(C)$ ET $n_d(A) = n_d(B) = n_d(C)$ ET $v_p(A) \leq_{lex} v_p(B) = v_p(C)$ ET $v_i(A) \geq_{lex} v_i(B) = v_i(C)$ donc $A \succeq C$,
  - soit $n_a(B) \leq n_a(C)$ ET $n_d(B) > n_d(C)$: on a donc $n_a(A) = n_a(B) \leq n_a(C)$ ET $n_d(A) = n_d(B) > n_d(C)$, donc $A \succeq C$,
  - soit $n_a(B) < n_a(C)$ ET $n_d(B) \geq n_d(C)$: on a donc $n_a(A) = n_a(B) < n_a(C)$ ET $n_d(A) = n_d(B) \geq n_d(C)$, donc $A \succeq C$,
  - soit $n_a(B) = n_a(C)$ ET $n_d(B) = n_d(C)$ ET $v_p(B) \leq_{lex} v_p(C)$ ET $v_i(B) \geq_{lex} v_i(C)$: on a donc $n_a(A) = n_a(B) = n_a(C)$ ET $n_d(A) = n_d(B) = n_d(C)$ ET $v_p(A) \leq_{lex} v_p(B) \leq_{lex} v_p(C)$ ET $v_i(A) \geq_{lex} v_i(B) \geq_{lex} v_i(C)$, donc $A \succeq C$.

  Dans tous les cas, $A \succeq C$.

Les arguments de valeur () sont les éléments maximaux de $\succeq$: Soit X une feuille de valeur (), on a $n_d(X) = \infty$ et $n_a(X) = 0$. Prenons un argument Y qui n'est pas une feuille, donc dont la valeur $v(Y) \neq ()$, on a alors $n_d(Y) < \infty$ et $n_a(Y) \geq 0$. Comparons X et Y avec l'algorithme 2 page 33 : $v(X) \neq v(Y)$ donc le cas 1 n'est pas utilisé ; puis, on a $n_a(X) = 0 \leq n_a(Y)$ ET $n_d(X) = \infty > n_d(Y)$ donc $X \succ Y$, donc $X \succeq Y$. □

Nous allons ensuite définir précisément les notions d'amélioration/détérioration des défenses/attaques qui composent les principes **P1'** à **P4'** (voir en section 5.1 page 25). Puis, nous prouverons que l'évaluation par tuple vérifie ces principes.

Définition 14 *Soit A un argument ayant pour valeur $v(A) \neq ()$ telle que $v'_p(A) = (x^p_1, \ldots, x^p_n)$ (le tuple des défenses, donc x^p_j est un entier pair, $\forall j \in [1..n]$) et $v'_i(A) = (x^i_1, \ldots, x^i_m)$ (le tuple des attaques, donc x^i_j est un entier impair, $\forall j \in [1..m]$). On définit :*

Une amélioration de la défense *consiste à*
- *soit rajouter une branche de défense à A :* $v'_p(A)$ *devient* $f_c(x^p_1, \ldots, x^p_n, x^p_{n+1})$ *avec* x^p_{n+1} *entier pair ;*
- *soit diminuer une branche de défense de A :* $\exists j \in [1..n]$ *tel que* $v'_p(A)$ *devient* $(x^p_1, \ldots, x^p_{j-1}, x'^p_j, x^p_{j+1}, \ldots, x^p_n)$ *avec* $x'^p_j < x^p_j$ *et* x'^p_j *pair ;*
- *soit supprimer la seule branche de défense menant à A :* *si initialement* $v(A) = (x^p_j)$ *avec* x^p_j *pair et que* $v(A)$ *devient* () *;*

Une amélioration de l'attaque *consiste à*
- *soit rajouter une branche d'attaque à A :* $v'_i(A)$ *devient* $f_c(x^i_1, \ldots, x^i_m, x^i_{m+1})$ *avec* x^i_{m+1} *entier impair ;*
- *soit diminuer une branche d'attaque de A :* $\exists j \in [1..m]$ *tel que* $v'_i(A)$ *devient* $(x^i_1, \ldots, x^i_{j-1}, x'^i_j, x^i_{j+1}, \ldots, x^i_m)$ *avec* $x'^i_j < x^i_j$ *et* x'^i_j *impair ;*

Une détérioration de la défense *consiste à*
- *soit supprimer une branche de défense à A :* $\exists j \in [1..n]$ *tel que* $v'_p(A)$ *devient* $(x^p_1, \ldots, x^p_{j-1}, x^p_{j+1}, \ldots, x^p_n)$ *;*
- *soit rallonger une branche de défense de A :* $\exists j \in [1..n]$ *tel que* $v'_p(A)$ *devient* $(x^p_1, \ldots, x^p_{j-1}, x'^p_j, x^p_{j+1}, \ldots, x^p_n)$ *avec* $x'^p_j > x^p_j$ *et* x'^p_j *pair ;*

Une détérioration de l'attaque *consiste à*

– *soit supprimer une branche d'attaque à A :*
  $\exists j \in [1..m]$ *tel que* $v'_i(A)$ *devient* $(x^i_1, \ldots, x^i_{j-1}, x^i_{j+1}, \ldots, x^i_m)$ *;*
– *soit rallonger une branche d'attaque de A :*
  $\exists j \in [1..m]$ *tel que* $v'_i(A)$ *devient* $(x^i_1, \ldots, x^i_{j-1}, x'^i_j, x^i_{j+1}, \ldots, x^i_m)$ *avec* $x'^i_j > x^i_j$ *et* x'^i_j *impair.*

Définition 15 *Soit A un argument ayant pour valeur* $v(A) = ()$ *(donc* $v'_p(A) = v'_i(A) = ()$*). On définit :*

Une amélioration de l'attaque *consiste à rajouter une branche d'attaque à A :*
$v'_i(A)$ *devient* (x^i_j) *avec* x^i_j *entier impair ;*

Une détérioration de la défense *consiste à rajouter une branche de défense à A :*
$v'_p(A)$ *devient* (x^p_j) *avec* x^p_j *entier pair.*

Propriété 7 *Soit v une évaluation par tuples avec comparaison des nombres de branches définie par l'algorithme 2 page 33, v respecte les principes* **P1'**, **P2'**, **P3'** *et* **P4'**.

Preuve : Le principe **P1'** est vérifié par l'utilisation des règles de calcul 12 page 26 à 13 page 26 et par le fait que () est le max de V (voir propriété 6 page 40).
Le principe **P2'** est vérifié à cause des règles de calcul 12 page 26 à 13 page 26.
Les principes **P3'** et **P4'** sont vérifiés car il suffit de passer en revue l'un après l'autre tous les cas possibles d'amélioration/détérioration de la défense/de l'attaque[9] pour un argument donné (voir définitions 14 page précédente et 15) et de comparer les valeurs de cet argument avant et après l'application de cette amélioration/détérioration de la défense/de l'attaque en utilisant l'algorithme 2 page 33. □

5.5 Prise en compte de la valeur intrinsèque

Soit un argument A ayant :
– une valeur issue du graphe des contrariétés sous la forme d'un tuple $v(A) = (a_1, \ldots, a_n)$,
– une valeur intrinsèque $v_i(A)$.
Comment "mixer" ces 2 informations pour obtenir $v_c(A)$?
Nous allons passer en revue diverses possibilités en analysant leurs avantages et leurs inconvénients.
Remarquons que le contenu de cette section est totalement indépendant du type d'approche par tuples choisi.

5.5.1 Possibilité 1

On pose $v_c(A) = (a'_1, \ldots, a'_n)$ avec $a'_i = \Phi(v_i(A), a_i)$.
Il va alors falloir respecter les contraintes suivantes :
– si a_i est pair alors a'_i doit aussi être pair,
– si a_i est impair alors a'_i doit aussi être impair.

Avantages D'abord, avoir les informations fusionnées permet de les propager ensemble dans le graphe des contrariétés. Et puis, on peut alors garder le même mécanisme de comparaison entre 2 arguments.

Inconvénients/Questions en suspens Peut-on toujours utiliser les a'_i pour tester/comparer les longueurs des branches ? Il semblerait que, puisque les a'_i représentent les longueurs des branches, ils ne doivent pas tous évoluer de la même façon (suivant qu'il s'agit d'une branche de défense ou d'une branche d'attaque !).
Un exemple : soient A_1 et A_2 ayant la même valeur issue du graphe des contrariétés ($v(A_1) = v(A_2) = (a_1, \ldots, a_n)$), mais ayant une valeur intrinsèque différente $v_i(A_1) > v_i(A_2)$. Si on modifie à l'identique les a_i quelle que soit la nature de la branche correspondant, on va se retrouver avec 2 arguments à comparer dont les longueurs de branches ont évolué de la même façon (toutes les branches ont, soit allongé, soit retréci), d'où impossible de trancher entre A_1 et A_2.

[9]On se limite, bien-sûr, à l'étude de ce type de modification simple pour éviter d'avoir à traiter deux problèmes différents en même temps. Par exemple, la modification de la longueur d'une branche qui modifie la nature de la branche (un entier pair remplacé par un entier impair, ou vice-versa) correspond au cas complexe d'une suppression d'une branche d'une nature donnée suivie d'un rajout d'une branche d'une nature différente !

D'autre part, même si on fait un traitement différent suivant la nature de la branche correspondant à a_i, il est très difficile de dire *de combien* il faut "raccourcir" ou "allonger" les branches !

5.5.2 Possibilité 2

Modifier les règles de comparaison afin de prendre en compte la comparaison d'un couple (tuple, valeur_intrinsèque).

Avantages Ne pas toucher au graphe des contrariétés et à l'étiquetage par tuples en cas de modification des valeurs intrinsèques.

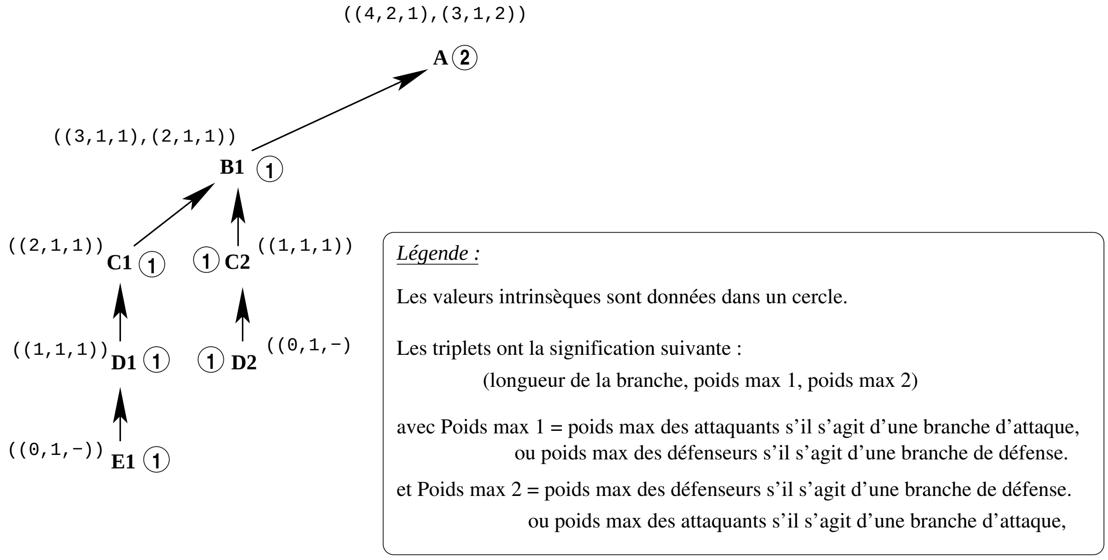
Inconvénients On va alors se retrouver soit à devoir faire des choix supplémentaires (ce qui est obligatoire sinon la valeur intrinsèque ne servirait à rien, mais ce qui va alourdir d'autant le processus de comparaison), soit à ne pas pouvoir trancher quand l'ordre sur les tuples est en contradiction avec l'ordre sur les valeurs intrinsèques.
Autre inconvénient majeur, on ne prend pas en compte les valeurs intrinsèques des éléments "amont" du graphe dans le calcul de la valeur de l'argument (peut-être que dans une branche de défense, les défenseurs ont une valeur intrinsèque beaucoup plus forte que celle de tous les attaquants des branches d'attaque !).

5.5.3 Possibilité 3

Attribuer 2 poids supplémentaires à chaque branche : la valeur intrinsèque maximale des attaquants et la valeur intrinsèque maximale des défenseurs. Puis travailler alors sur des tuples de triplets[10].

Avantages On a toujours, à chaque étape dans le graphe des contrariétés, l'essentiel des informations permettant de juger de la valeur de l'étape courante. D'autre part, on peut conserver à peu-près les mêmes mécanismes de comparaison.

Inconvénients On a une structure lourde à manipuler (en particulier si on veut faire une comparaison lexicographique des triplets, il faut inverser à chaque étape dans le graphe le poids des attaquants et celui des défenseurs) et qui introduit quelques nouveaux cas soit de "forçage d'égalité", soit d'incomparabilité. Par exemple :



On aura alors soit une égalité forcée, soit une incomparabilité entre A et B_1 car, bien que A ait une valeur intrinsèque plus forte que B_1, ce qui le favorise dans le cas de l'attaque (le tuple $((3, 1, 2))$ l'emporte sur le tuple $((3, 1, 1))$), il a quand même une branche de défense plus longue (le tuple $((2, 1, 1))$ l'emporte sur le tuple $((4, 2, 1))$).

[10] On pourrait aussi envisager de conserver la totalité ds informations concernant les valeurs intrinsèques des arguments rencontrés jusqu'alors en parcourant le graphe des contrariétés, mais cela semble extrêmement complexe à gérer !

5.6 Conclusion sur l'approche 2

Cette approche cherche à prendre en compte l'ensemble de la relation de contrariété pour calculer la valeur d'un argument. En cela, elle est dite "globale".
La valeur de l'argument sera donc un tuple reflétant l'ensemble des attaques et l'ensemble des défenses que subit l'argument.
Il existe ensuite plusieurs manières de comparer ces valeurs tuplées. Nous en avons proposé ici deux :
– une qui utilise un taux entre nombre d'attaques et nombre de défenses,
– une qui compare directement les nombres d'attaques et de défenses sans passer par le calcul d'un taux.
Ces deux approches ne fournissent pas toujours les mêmes résultats sur les exemples étudiés. En effet, l'approche 2 avec taux, bien que définissant une relation totale sur l'ensemble des arguments, ne permet pas d'obtenir une relation de pré-ordre sur cet ensemble (elle n'est pas transitive). D'autre part, l'approche 2 avec comparaison du nombre de branches définit une relation de pré-ordre sur l'ensemble des arguments mais, comme elle est beaucoup plus prudente que celle avec taux, cette relation de pré-ordre est seulement partielle.
Enfin, la prise en compte de la valeur intrinsèque dans ce modèle d'évaluation reste entièrement à étudier (même si quelques pistes sont fournies ici).

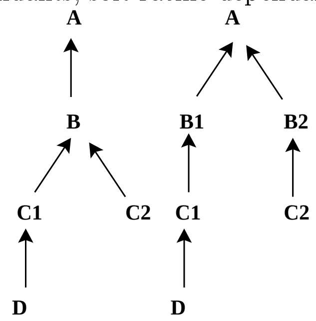
Chapitre 6

Comparaisons

6.1 Comparaison des approches 1 et 2

Bien que ces deux approches reposent sur des principes différents, il y a quand même des cas où l'on obtient les mêmes résultats : c'est lorsque le graphe des contrariétés se réduit à un chemin unique sans circuit (voir les exemples 0 page 7, 1 page 7, 2 page 7, 7 page 8). Les différences apparaîssent dès qu'on peut avoir pour un argument donné à la fois des branches d'attaque et des branches de défense (voir exemples 3 page 8, 4 page 8, 5 page 8, 6 page 8, 8 page 9 et 9 page 9). En effet, l'approche 1 fait l'agrégation des valeurs des attaquants directs avant d'évaluer l'argument, alors que l'approche 2 ne fait aucune agrégation de valeurs et considère chaque branche comme indépendante même si elles ne le sont pas graphiquement.
Cela se voit particulièrement bien sur l'exemple suivant (exemple supplémentaire numéroté 24) :

24. cas où les chemins sont soit dépendants, soit racine-dépendants :



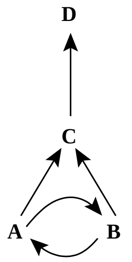
Ici, les valeurs de A par l'approche 2 sont les mêmes, ce qui n'est pas le cas avec l'approche 1.
Quant au cas des circuits, l'approche 1 impose l'existence de points fixes pour les fonctions d'évaluation, ce qui n'est absolument pas le cas de l'approche 2 (voir exemples de 10 page 9 à 17 page 10).

6.2 Le modèle de [Dun95]

Le schéma d'argumentation de [Dun95] permet la définition d'ensembles acceptables d'arguments, appelés extensions, avec différentes sémantiques. Sous la sémantique préférée par exemple, une extension est un ensemble maximal pour l'inclusion qui est sans-conflit et défend tous ses éléments. Un argument est donc accepté (ou non) selon qu'il appartient (ou pas) à une extension préférée. On peut donc considérer qu'à chaque sémantique correspond un type d'évaluation binaire des interactions et sous une sémantique donnée, à chaque extension correspond une évaluation binaire.

Exemple 17 page 10 La seule extension préférée est l'ensemble vide. Cela correspond à une évaluation où tous les arguments sont rejetés.

Exemple supplémentaire Soit le graphe suivant :



Il y a deux extensions préférées $\{A, D\}$ et $\{B, D\}$. L'extension $\{A, D\}$ correspond à une évaluation où les arguments A et D (resp. B et C) sont acceptés (resp. rejetés).

6.3 Le modèle de [JV99]

Dans le modèle proposé par [JV99], un étiquetage d'un ensemble d'arguments permet d'assigner à chaque argument un statut (accepté, rejeté, non décidé) sous la forme d'une étiquette appartenant à l'ensemble $\{+, -, ?\}$. $+$ (resp. $-$, $?$) représente le statut "accepté" (resp. "rejeté", "non décidé"). Intuitivement, un argument étiqueté par ? est à la fois supporté et affaibli.

Définition 16 *Soit $<\mathcal{A}, \mathcal{R}>$ un système d'argumentation. Un étiquetage complet de $<\mathcal{A}, \mathcal{R}>$ est une application $Et\ : \mathcal{A} \to \{+, ?, -\}$ telle que :*

1. *Si $Et(A) \in \{?, -\}$ alors $\exists B \in \mathcal{R}^-(A)$ tel que $Et(B) \in \{+, ?\}$*
2. *Si $Et(A) \in \{+, ?\}$ alors $\forall B \in \mathcal{R}^-(A)$ ou $\in \mathcal{R}^+(A)$, $Et(B) \in \{?, -\}$*

L'intuition est la suivante : un argument ne peut être affaibli (étiquette $-$ ou ?) que si l'un de ses attaquants directs est supporté (condition 1) ; un argument ne peut recevoir un support que si tous ses attaquants directs sont affaiblis et un argument qui est supporté (étiquette $+$ ou ?) affaiblit les arguments qu'il attaque (condition 2).
Conséquences :
- Si A n'a pas de contrariant, $Et(A) = +$.
- Si $Et(A) =?$ alors $\exists B \in \mathcal{R}^-(A)$ tel que $Et(B) =?$.
- Si ($\forall B \in \mathcal{R}^-(A)$, $Et(B) = -$) alors $Et(A) = +$.
- Si $Et(A) = +$ alors $\forall B \in \mathcal{R}^-(A)$ ou $\in \mathcal{R}^+(A)$, $Et(B) = -$.

Tout système d'argumentation admet un étiquetage complet. La sémantique associée est définie par : S est un ensemble d'arguments acceptable ssi il existe Et étiquetage complet de $<\mathcal{A}, \mathcal{R}>$ tel que : $S = \{A | Et(A) = +\}$.
On trouve dans [JV99] d'autres types d'étiquetages et les sémantiques associées parmi lesquelles la sémantique "enracinée" définie à partir des étiquetages "enracinés". L'idée est de ne rejeter que des arguments attaqués par des arguments acceptés. En effet, une attaque par un argument "non décidé" n'est pas "enracinée" puisque un attaquant "non décidé" peut devenir rejeté.

Définition 17 *L'étiquetage complet Et est enraciné ssi $\forall A \in \mathcal{A}$, si $Et(A) = -$ alors $\exists B \in \mathcal{R}^-(A)$ tel que $Et(B) = +$.*

La sémantique enracinée a permis de clarifier les liens entre toutes les sémantiques proposées dans [JV99] et certaines sémantiques de [Dun95].

Sur les exemples On obtient les résultats suivants :

Exemple 0 page 7 $v(A) = +$

Exemple 1 page 7 $v(B) = +$ et $v(A) = -$

Exemple 2 page 7 $v(C) = v(A) = +$ et $v(B) = -$

Exemple 7 page 8 Cas où n est pair : $v(A_n) = v(A_{n-2}) = \ldots = v(A_2) = +$ et $v(A_{n-1}) = v(A_{n-3}) = \ldots = v(A_1) = -$

Cas où n est impair : $v(A_n) = v(A_{n-2}) = \ldots = v(A_1) = +$ et $v(A_{n-1}) = v(A_{n-3}) = \ldots = v(A_2) = -$

Exemple 10 page 9 Ici, on a 3 étiquetages possibles. Soit $v(A) = +$ et $v(B) = -$, soit $v(A) = -$ et $v(B) = +$, soit $v(A) = v(B) =?$

Exemple 14 page 10 $v(A) =?$

Exemple 17 page 10 Ici, 2 étiquetages possibles. Soit $v(A) = v(B) = v(C) = v(D) = ?$, soit $v(A) = v(B) = v(C) = ?$ et $v(D) = -$

Exemple 24 page 45 Dans le premier cas (chemins dépendants), on a : $v(D) = v(C_1) = v(A) = +$ et $v(C_2) = v(B) = -$. Dans le second cas (chemins racine-dépendants), on a : $v(D) = v(C_1) = v(B_2) = +$ et $v(C_2) = v(B_1) = v(A) = -$

Commentaires sur ces résultats :
- Cas d'un graphe sans circuit : Tout argument est étiqueté par $+$ ou $-$. Un argument A est accepté ssi il est (indirectement) défendu contre chacun de ses attaquants directs (donc ssi chacun de ses attaquants directs est rejeté).
- Cas d'un graphe avec circuit : Les arguments présents sur un circuit impair sont nécessairement étiquetés par "non décidé". Si Et est un étiquetage enraciné :
  - A est accepté $(+)$ ssi tous ses attaquants directs sont rejetés $(-)$.
  - A est rejeté ssi l'un de ses attaquants directs est accepté.
  - A est non décidé ssi l'un de ses attaquants directs est non décidé et aucun de ses attaquants directs n'est accepté.

Propriété 8 (Lien avec l'approche 1) *Tout étiquetage enraciné de $<\mathcal{A}, \mathcal{R}>$ peut être défini par une évaluation v telle que : $V = W = \{-, ?, +\}$ avec $- < ? < +$, V_{Min} dénoté par $-$ et V_{Max} dénoté par $+$.*

> **Preuve :** Il suffit de poser : g définie par $g(-) = +$, $g(+) = -$, $g(?) = ?$ et h est la fonction max. □

6.4 La fonction d'agrégation de [BH01]

[BH01] propose un cadre logique pour le traitement d'arguments "déductifs"[1]. L'approche est caractérisée par les points suivants :
- Un argument est structuré sous la forme d'une paire $<$support, conclusion$>$, où le support est un ensemble consistant de formules permettant de prouver la conclusion.
- La relation de contrariété considérée est stricte et les circuits sont exclus.
- Le concept d'arbre d'arguments permet d'avoir une représentation synthétique et exhaustive des attaquants et défenseurs d'un argument donné, racine de l'arbre.
- Une fonction, appelée "catégoriseur", permet d'attacher une valeur à un arbre d'arguments. Cette valeur représente la force relative d'un argument (la racine de l'arbre) compte tenu de ses attaquants et de ses défenseurs.
- Une fonction, appelée "accumulateur", permet de synthétiser les valeurs attachées à tous les arbres d'arguments dont la racine est un argument pour (resp. contre) une conclusion donnée.

L'évaluation des interactions correspond donc à la phase de catégorisation. [BH01] propose la fonction Cat suivante :
- si $\mathcal{R}^-(A) = \emptyset$, alors $Cat(A) = 1$
- si $\mathcal{R}^-(A) \neq \emptyset$ avec $\mathcal{R}^-(A) = \{A_1, \ldots, A_n\}$, $Cat(A) = \frac{1}{1+Cat(A_1)+\ldots+Cat(A_n)}$

Intuitivement, plus l'argument à évaluer a d'attaquants directs, moindre est sa valeur et plus l'argument à évaluer a de défenseurs, plus grande est sa valeur.

Sur les exemples Rappelons que l'on obtient les résultats suivants :

Exemple 0 page 7 $v(A) = 1$

Exemple 1 page 7 $v(B) = 1 > v(A) = \frac{1}{2}$

Exemple 2 page 7 $v(C) = 1 > v(A) = \frac{2}{3} > v(B) = \frac{1}{2}$

Exemple 7 page 8 On retrouve les mêmes résultats qu'en section 4 page 21

Exemple 24 page 45 Dans le premier cas (chemins dépendants), $v(A) = \frac{5}{7}$. Dans le second cas (chemins racine-dépendants), $v(A) = \frac{6}{13}$

Propriété 9 (Lien avec l'approche 1) *La fonction de catégorisation de [BH01] est la restriction à des graphes sans circuit d'une évaluation au sens de l'approche proposée en section 4 page 21.*

[1]"Deductive arguments" en anglais dans [BH01].

Preuve : En effet, posons $V = [0,1]$, $W = [0,\infty[$, $V_{\text{Min}} = 0$ et $V_{\text{Max}} = 1$ et définissons $g : W \to V$ définie par $g(x) = \frac{1}{1+x}$ et h par $h(\{x_1,\ldots,x_n\}) = x_1 + \ldots + x_n$. $\square$

L'évaluation proposée dans la preuve ci-dessus peut aussi s'appliquer à des graphes des contrariétés quelconques. La fonction g définie admet un seul point fixe, noté $\alpha \in [\frac{1}{2},1]$. $\alpha = (\sqrt{5}-1)/2$. C'est l'inverse du nombre d'Or. C'est aussi le seul point fixe de g^2. Par conséquent, tout argument appartenant à un circuit aura ce point fixe pour valeur.

Bibliographie

[AC98] Amgoud (Leila) et Cayrol (Claudette). – On the acceptability of arguments in preference-based argumentation. *In : Proc. of the 14^{th} Uncertainty in Artificial Intelligence*, éd. par Cooper (G. F.) et Moral (S.). pp. 1–7. – Madison, Wisconsin, 1998.

[BH01] Besnard (Philippe) et Hunter (Anthony). – A logic-based theory of deductive arguments. *Artificial Intelligence*, vol. 128 (1-2), 2001, pp. 203–235.

[Dun95] Dung (Phan Minh). – On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, vol. 77, 1995, pp. 321–357.

[JV99] Jakobovits (H.) et Vermeir (D.). – Robust semantics for argumentation frameworks. *Journal of logic and computation*, vol. 9(2), 1999, pp. 215–261.

[KAEF95] Krause (P.), Ambler (S.), Elvang (M.) et Fox (J.). – A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, vol. 11 (1), 1995, pp. 113–131.

[Par97] Parsons (S.). – Normative argumentation and qualitative probability. *LNAI*, vol. 1244, 1997, pp. 466–480.

[PS97] Prakken (H.) et Sartor (G.). – Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, vol. 7, 1997, pp. 25–75.