Cadre Argumentatifs avec Nécessités : Nouvelles Sémantiques et Algorithmes d'étiquetage

Farid Nouioua

LSIS, Université d'Aix-Marseille
Avenue Escadrille Normandie Niemen,
13397, Marseille, Cedex 20

farid.nouioua@lsis.org

Résumé

Les Cadres d'Argumentation avec Nécessités (CANs) proposés dans [17] représentent un type de systèmes d'argumentation bipolaires qui étendent les systèmes de Dung par une relation de support ayant le sens particulier de nécessité. Ce papier est une continuation de ce travail dans deux directions. Nous complétons la panoplie des sémantiques d'acceptabilité en définissant la sémantique basée, la sémantique complète et la sémantique semi-stable pour les CANs. Nous montrons que les sémantiques proposées gardent les mêmes propriétés que celles des systèmes de Dung et représentent des généralisations propres de celles-ci (en l'absence de la relation de nécessité, les sémantiques classiques sont retrouvées). Ensuite, nous montrons comment on peut généraliser les algorithmes d'étiquetage de Caminada en présence de la relation de nécéssité pour calculer les extensions sous les différentes sémantiques étudiées pour les CANs. [1]

Abstract

The Argumentation Frameworks with Necessities (AFNs) proposed in [17] are a kind of bipolar AFs extending Dung AFs with a support relation having the particular meaning of necessity. This paper is a continuation of this work in two respects. First, we complete the acceptability semantics picture by defining the grounded, the complete and the semi-stable semantics for AFNs. We show that the proposed semantics keep the same properties as those given for Dung AFs and represent proper generalizations of them (in absence of the necessity relation, the classical semantics are recovered). Then, we show how to generalize Caminada's labelling algorithms in presence of a necessity relation to compute the extensions under the studied semantics for AFNs.

1. Ce paper a été publié dans les actes de SUM'2013 (International Conference on Scalable Uncertainty Management).

1 Introduction

La théorie d'argumentation abstraite de Dung est aujourd'hui une des théories les plus influentes sur les approches de l'argumentation en Intelligence Artificielle. Une des extensions des cadres argumentatifs (CAs) de Dung vise à prendre en compte des interactions positives (supports) entre arguments en plus des attaques qui représentent des interactions négatives. Une question intéressante est alors de savoir comment représenter ces supports et comment les traiter sans forcément passer par un CA de Dung équivalent.

Les principales approches pour le traitement des supports les représentent explicitement. Ces approches comprennent les cadres argumentatifs bipolaires (CABs) [10] [11], l'approche à supports déductifs [5] et les cadres dialectiques abstraits [6] entre autres. Nous avons présenté dans [17] une discussion des avantages et inconvénients des ces différentes propositions et nous avons introduit les Cadres Argumentatfs avec Nécessitiés (CANs) comme type de CAs bipolaires où la relation de support possède le sens particulier de "nécessité". Il a été montré que grâce à cette spécification du sens du support, il était possible de généraliser quelques unes des sémantiques d'acceptabilité (stable et préférée) de façon naturelle et sans emprunter de techniques issues des programmes logiques (PLs) ou nécessairement utiliser un méta modèle de Dung.

Nous continuons cette ligne de recherche en explorant d'autres sémantiques d'acceptabilité pour les CANs, notamment les sémantiques complète, basée et semi-stable. Nous montrons en particulier comment garder les définitions et les propriétés de "haut niveau" des différentes sémantiques d'acceptabilité pour

les CAs dans le cas des CANs en assurant juste une incorporation adéquate de la relation de nécessité à quelques concepts de "bas niveau". Ainsi, nous généralisons le fait que contrairement à d'autres formalismes d'argumentation bipolaire, pour tirer des conclusions, il n'est pas nécessaire de traduire d'abord le CAN en un CA de Dung ou d'utiliser des techniques issues des PLs. De plus, nous montrons que dans le cas où la relation de nécessité est vide, nous retrouvons exactement les versions originales de ces sémantiques.

Dans [9] [8][16], une caractérisation intéressante des sémantiques d'acceptabilité qui utilise l'idée d'étiquetage a été proposée pour les CAs de Dung. Les extensions sous une sémantique donnée sont caractérisées par des étiquetages satisfaisant certaines conditions qui dépendent de cette sémantique. Cette approche a donné naissance aux *algorithmes d'étiquetage* qui calculent les extensions d'un CA sous une sémantique donnée. Ainsi, une autre contribution de ce papier est de généraliser la caractérisation de Caminada en termes d'étiquetages au cas des CANs et de proposer ensuite de nouveaux algorithmes d'étiquetage pour les sémantiques d'acceptabilité des CANs.

Le plan du papier est comme suit. Dans la section 2., nous rappelons quelques bases des CAs de Dung. La section 3. présente les différentes sémantiques d'acceptabilité pour les CANs. Dans la section 4, l'approche fondée sur les étiquetages pour les sémantiques d'acceptabilié est généralisée aux CANs et la section 5 décrit des algorithmes d'étiquetage adaptés pour les CANs et capable de calculer les différents types d'extensions (basée, admissible, préférée, stable et semi-stable). Dans la section 6, nous concluons et donnons quelques perspectives pour des travaux futurs.

2 Bref rappel des CAs de Dung

Un CA [12] est un couple $\mathcal{A} = \langle AR,\ att\rangle$ où AR est un ensemble d'arguments et att est une relation binaire d'attaque sur AR. Par abus de langage, nous écrivons $S\ att\ a$ (resp. $a\ att\ S$) pour $S \subseteq AR$ et $a \in AR$ pour exprimer qu'il existe $b \in S$ tel que $b\ att\ a$ (resp. $a\ att\ b$). Un sous-ensemble $S \subseteq A$ est sans-conflit s'il n'existe pas $a, b \in S$ tel que $a\ att\ b$, S défend a si pour tout $b \in AR$, si $b\ att\ a$ alors $S\ att\ b$ et S est un ensemble admissible s'il est sans-conflit et défend tous ses éléments. La fonction caractéristique d'un CA $\mathcal{A}$ est une fonction $\mathcal{F}_{\mathcal{A}} : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ où : $\mathcal{F}_{\mathcal{A}}(S) = \{a | S\ defend\ a\}$. Nous notons par S^+ l'ensemble d'arguments attaqués par S : $S^+ = \{a | S\ att\ a\}$.

Différentes sémantiques d'acceptabilité ont été définies pour capter les ensembles d'arguments qui peuvent être acceptés collectivement. Soit $S \subseteq AR$. S est une extension complète si et seulement si elle est admissible et contient tout argument qu'elle défend. S est une extension basée si et seulement s'il est le plus petit point fixe de $\mathcal{F}_{\mathcal{A}}$. S est une extension préférée si et seulement si S est un ensemble admissible $\subseteq$-maximal. S est une extension stable si et seulement si S est un ensemble admissible qui attaque tout argument qui ne lui appartient pas (i.e., $S^+ = AR \setminus S$) et enfin S est une extension semi-stable si et seulement si S est une extension complète qui maximise $S \cup S^+$.

3 Les CANs et leurs sémantiques d'acceptabilité

Un CAN [17] est défini par $\mathcal{G} = \langle AR,\ att,\ \mathcal{N}\rangle$ où AR est un ensemble d'arguments. att est une relation binaire d'attaque interprétée exactement comme dans les CAs de Dung : $a\ att\ b$ veut dire que si a est accepté alors b n'est pas accepté. $\mathcal{N}$ est une relation de nécessité interprétée de façon duale comme suit : Pour $E \subseteq AR$ et $b \in AR$, $E\ \mathcal{N}\ b$ veut dire que si aucun argument de E n'est accepté alors b n'est pas accepté (l'acceptation de b nécessite l'acceptation d'au moins un argument de E). Notons que $E\ \mathcal{N}\ b$ n'implique pas l'existence d'une relation de nécessité "primitive" entre le(s) élément(s) de E et b.

Exemple 1. Considérons la conversation suivante entre deux personnes P_1 et P_2 :

P1 : Je vais prendre la voiture pour une petite promenade. Je sais que la roue n'est pas intacte à cause de la crevaison qui a eu lieu hier, mais je vais utiliser la roue de secours.

P2 : Il y a un autre problème. Le moteur n'est pas fonctionnel. Ce matin je n'ai pas pu démarrer et en inspectant le moteur j'ai détecté une panne.

On peut modéliser cette situation comme dans la figure 1. Les arguments utilisés sont les suivants : **VU** : "la voiture est utilisable" ; **RNI** : "roue normale intacte" ; **RSI** : "roue de secours intacte" ; **CR** : "il y a une crevaison dans la roue normale" ; **MF** : "le moteur est fonctionnel" ; **P** : "Il y a une panne dans le moteur".

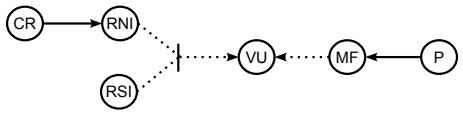

FIGURE 1 – Exemple d'un CAN

On a $CR\ att\ RNI$, $P\ att\ MF$, $\{MF\}\ \mathcal{N}\ VU$ et $\{RNI, RSI\}\ \mathcal{N}\ VU$. l'expression $\{RNI, RSI\}\ \mathcal{N}\ VU$ veut dire que pour la voiture soit utilisable, il faut

avoir au moins la roue normale ou la roue de secours (ou les deux). Cela ne se réduit ni au fait que l'une des deux roues prise toute seule est nécessaire à l'utilisation de la voiture (l'autre roue peut la remplacer), ni au fait que les deux roues sont nécessaires ensemble à l'utilisation de la voiture.

Comme pour la relation *att*, nous ne supposons aucune propriété particulière de la relation $\mathcal{N}$.

Nous généralisons ici les sémantiques d'acceptabilité aux CANs [2]. Les nouvelles sémantiques sont définies de façon très similaire au cas des CAs de Dung sauf qu'au lieu d'utiliser l'absence de conflits comme exigence minimale que toute extension doit garantir, cette notion est renforcée par une notion additionnelle de *cohérence* qui prend en compte la relation $\mathcal{N}$.

Les notions de défense, de fonction caractéristique et d'ensemble d'arguments attaqués par un ensemble donné sont également adaptés pour prendre en compte la relation $\mathcal{N}$.

Définition 1. (de [17]). Soit $S \subseteq AR$, S est *clos sous* $\mathcal{N}^{-1}$ si et seulement si pour tout $a \in S$, s'il existe $E \subseteq AR$ tel que $E \mathcal{N} a$ alors $E \cap S \neq \emptyset$. Un argument $a \in S$ est *Sans-Cycle-N dans* S si et seulement si pour tout $E \subseteq AR$ tel que $E \mathcal{N} a$, soit $E \cap S = \emptyset$, soit il existe $b \in E \cap S$ tel que b est Sans-Cycle-N dans S. S est Sans-Cycle-N si et seulement si tout $a \in S$ est Sans-Cycle-N dans S. S est *cohérent* si et seulement si S est Sans-Cycle-N et clos sous $\mathcal{N}^{-1}$. Finalement, S est *fortement cohérent* si et seulement si S est cohérent et sans-conflict.

Intuitivement, dans un ensemble cohérent S, on satisfait les nécessités de chaque argument tout en assurant l'absence de cycles de nécessités. Nous excluons les cycles de nàcessaità car on entend par une relation $E\mathcal{N}a$ que pour obtenir a, il faut d'abord assurer l'un des arguments de E. Ainsi, l'existence d'un cycle de nécessités reflète une sorte d'interblocage qui ne doit pas être gardé dans l'ensemble des arguments acceptés.

Nous introduisons la notion d'un argument *puissant* pour capter ce sens de cohérence au niveau individuel (i.e., le niveau des arguments).

Définition 2. Soit $\mathcal{G} = \langle AR,\ att,\ \mathcal{N} \rangle$ un CAN et $S \subseteq AR$. Un argument a est puissant dans S si et seulement si $a \in S$ et il existe une séquence $a_0, \ldots, a_k$ d'éléments de S telle que : $a_k = a$, il n'existe pas de $E \subseteq AR$ tel que $E \mathcal{N} a_0$ et pour $1 \leq i \leq k$, pour chaque $E \subseteq AR$, si $E \mathcal{N} a_i$ alors $E \cap \{a_0, \ldots, a_{i-1}\} \neq \emptyset$.

2. Le travail présenté dans [17] s'est intéressé à cette généralisation pour les ensembles admissibles ainsi que les extensions stables et préférée. Ici, nous considérons d'autres sémantiques.

Les ensembles cohérents sont caractérisés en termes d'arguments puissants comme suit :

Proposition 1. Soit $\mathcal{G} = \langle AR,\ att,\ \mathcal{N} \rangle$ un CAN et $S \subseteq AR$. S est cohérent si et seulement si tout $a \in S$ est puissant dans S.

Le second ingrédient dans la généralisation des sémantiques d'acceptabilité aux CANs est de redéfinir les notions de défense, de la fonction caractéristique et des arguments attaqués par un ensemble donné d'arguments (appelés ici les arguments désactivés).

Definition 3. (Défense / fonction caractéristique / arguments désactivés). Soient $S \subseteq AR$ et $a \in AR$. On dit que S défend a si et seulement si $S \cup \{a\}$ est cohérent et pour tout $b \in AR$, si b *att* a alors pour tout sous-ensemble cohérent C de AR qui contient b, S *att* C. En se fondant sur cette notion étendue de défense, la fonction caractéristique d'un CAN est définie comme pour les CAs classiques par $F : 2^{AR} \rightarrow 2^{AR}$ où $F(S) = \{a \mid S \text{ défend } a\}$. L'ensemble d'arguments désactivés par S est défini par $S^+ = \{a \mid S \text{ } att \text{ } a \text{ } ou \text{ } il \text{ } existe \text{ } E \subseteq AR \text{ } tel \text{ } que \text{ } E \mathcal{N} a \text{ } et \text{ } S \cap E = \emptyset\}$.

A présent, nous sommes prêts à définir les différentes sémantiques d'acceptabilité pour les CANs comme suit :

Définition 4. (Sémantiques d'acceptabilité pour les CANs). Soit $S \subseteq AR$. S est : un *ensemble admissible* si et seulement si S est fortement cohérent et défend tous ses arguments ; une extension *complète* si et seulement si S est admissible et contient tout argument qu'il défend ; une extension *basée* si et seulement si S est le plus petit point fixe de F ; une extension *préférée* si et seulement si S est un ensemble admissible maximal ; une extension *stable* si et seulement si S est une extension complète et $S^+ = AR \setminus S$ et une extension *semi-stable* si et seulement si S est une extension complète qui maximise $S \cup S^+$.

Il se trouve que les principales propriétés des différentes sémantiques d'acceptabilité pour les CAs de Dung restent vérifiées dans le cas des CANs.

Proposition 2. Soit $S \subseteq AR$. S est :

- un ensemble admissible si et seulement si $S \subseteq F(S)$; une extension complète si et seulement si $S = F(S)$; une extension basée si et seulement si S est une extension complète minimale ; une extension préférée si et seulement si S est une extension complète maximale.

– Il existe exactement une extension basée ; zéro, une ou plusieurs extensions stables et au moins une extension préférée.
– Toute extension stable est semi-stable et toute extension semi-stable est préférée. L'inverse n'est pas vrai.
– Si des extensions stables existent, alors elles coïncident avec les extensions semi-stables.

Les nouvelles sémantiques pour les CANs sont des généralisations propres des sémantiques correspondantes pour les CAs de Dung :

Proposition 3. Soit $\mathcal{G} = \langle AR,\ att,\ \mathcal{N}\rangle$ avec $\mathcal{N} = \emptyset$ et $S \subseteq AR$. S est un ensemble admissible (resp. extension complète, basée, préférée, stable, semi-stable) de $\mathcal{G}$ si et seulement si S est un ensemble admissible (resp. extension complète, basée, préférée, stable, semi-stable) du CA $\mathcal{F} = \langle AR,\ att\rangle$.

Exemple 2. Considérons les trois CANs représentés dans la figure 2.
$\mathcal{G}_1 = \langle AR = \{a,b,c\}, att = \{(b,c)\}, \mathcal{N} = \{(\{a\},b), (\{b\},a)\}\rangle$,
$\mathcal{G}_2 = \langle AR = \{a,b,c,d\},\ att = \{(a,b),(d,c)\},\ \mathcal{N} = \{(\{b,c\},a)\}\rangle$
$\mathcal{G}_3 = \langle AR = \{a,b,c,d,e,f,g\}, att = \{(a,b), (b,a),(c,d),(d,e),(e,c),(e,f),(g,g)\}, \mathcal{N} = \{(\{b\},c), (\{g\},f)\}\rangle$;

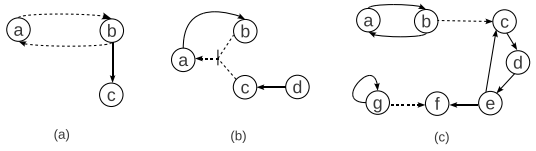

FIGURE 2 – (a) CAN $\mathcal{G}_1$, (b) CAN $\mathcal{G}_2$, (c) CAN $\mathcal{G}_3$

$\mathcal{G}_1$ a deux ensembles admissibles : $\emptyset$ et $\{c\}$. En effet, c est attaqué par b mais il n'existe pas de sous-ensemble cohérent contenant b (le seul ensemble contenant b et clos sous $\mathcal{N}$ est $\{a,b\}$ qui n'est pas cohérent). Cela signifie que $\{c\}$ défend son unique élément c. Nous avons $\mathcal{F}_{\mathcal{G}_1}(\emptyset) = \{c\}$ et $\mathcal{F}(\{c\}) = \{c\}$ et on peut vérifier que $\{c\}$ est l'unique point fixe de $\mathcal{F}_{\mathcal{G}_1}$. Par conséquent, $\{c\}$ est la seule extension complète de $\mathcal{G}_1$ qui est sa seule extension basée et sa seule extension préférée. Elle est également sa seule extension stable et semi-stable puisque $\{c\}^+ = \{a,b\} = AR \setminus \{c\}$.

Les ensemble fortement cohérents de $\mathcal{G}_2$ sont : $\emptyset$, $\{b\}$, $\{c\}$, $\{d\}$, $\{a,c\}$, $\{b,c\}$, $\{b,d\}$. Seuls $\emptyset$ et $\{d\}$ défendent leurs éléments respectifs et sont donc admissibles. On a $\mathcal{F}_{\mathcal{G}_2}(\emptyset) = \{d\}$ et $\mathcal{F}_{\mathcal{G}_2}(\{d\}) = \{d\}$ et on peut vérifier que $\{d\}$ est le seul point fixed de $\mathcal{F}_{\mathcal{G}_2}$. Ainsi, $\{d\}$ est la seule extension complète de $\mathcal{G}_2$ qui est son extension basée et sa seule extension préférée. On a $\{d\}^+ = \{a,c\} \neq AR \setminus \{d\}$. $\mathcal{G}_2$ n'a pas d'extension stable et admet $\{d\}$ comme unique extension semi-stable.

$\mathcal{G}_3$ a quatre ensembles admissibles : $\emptyset$, $\{a\}$, $\{b\}$ et $\{a,d\}$. On a $\mathcal{F}_{\mathcal{G}_3}(\emptyset) = \emptyset$, $\mathcal{F}_{\mathcal{G}_3}(\{a\}) = \{a,d\}$, $\mathcal{F}_{\mathcal{G}_3}(\{b\}) = \{b\}$ et $\mathcal{F}_{\mathcal{G}_3}(\{a,d\}) = \{a,d\}$. Ainsi, $\emptyset$, $\{b\}$ et $\{a,d\}$ sont les extensions complètes $\mathcal{G}_3$. Parmi elles, $\emptyset$ est son extension basée. $\{b\}$ et $\{a,d\}$ sont ses extensions préférées. On a $\{b\}^+ = \{a,f\} \neq AR \setminus \{b\}$ et $\{a,d\}^+ = \{b,c,e,f\} \neq AR \setminus \{a,d\}$. $\mathcal{G}_3$,'a pas d'extension stable et puisque $\{b\} \cup \{b\}^+ \subseteq \{a,d\} \cup \{a,d\}^+$, seulement $\{a,d\}$ est une extension semi-stable extension de $\mathcal{G}_3$.

3.1 CANs et CAs

Etant donné un CAN $\mathcal{G} = \langle AR, att, \mathcal{N}\rangle$, une première question qui nous intéresse est de savoir s'il est toujours possible de trouver un CA de Dung avec exactement les mêmes arguments et qui contient toutes les informations codées dans $\mathcal{G}$. Il a été montré dans [17] que la réponse est positive lorsque la relation de nécéssité est définie entre arguments isolés (pour les CANs où si $E\ \mathcal{N}\ a$ alors E est un singleton). L'idée consiste à ajouter les attaques implicites qui résultent de l'interaction entre les attaques (directes) et les nécessités comme suit : si a attaque b et b est nécessaire pour c alors a attaque indirectement c et si a nécessite b et b attaque c alors a attaque indirectement c.

On montre ici que la réponse est négative dans le cas générale et on peut avoir besoin d'un nombre plus important d'arguments pour coder toute l'information d'un CAN dans un CA. Pour montrer cela, considérons le CAN $\mathcal{G}_2$ de l'exemple 2 et supposons que $F = \langle AR, att'\rangle$ est un CA qui code la même information que $\mathcal{G}_2$. Il est clair que $(a,b),(d,c)$ sont dans att'. Le CA $\langle AR, \{(a,b),(d,c)\}\rangle$ n'a pas les mêmes extensions pour toutes les sémantiques considérées. Hormis ces deux attaques, toute autre attaque possible d'un argument y par un argument x ($x,y \in AR$) n'est pas présente ni directement ni indirectement dans $\mathcal{G}_2$. En particulier, on ne peut pas dire que d attaque a car a peut être obtenu soit en ayant c soit en ayant b et d attaque uniquement b. La solution est de représenter séparément les deux façons différentes d'obtenir a (par b ou par c) comme deux méta arguments, notés par exemple A_1 et A_2. Seulement le second méta argument, impliquant a et c, est attaqué par d.

Plus généralement, étant donné un CAN $\mathcal{G} = \langle AR, att, \mathcal{N}\rangle$ et $a \in AR$, chaque ensemble cohérent $\mathcal{C} \subseteq AR$ contenant a et minimal (aucun sous-ensemble strict de $\mathcal{C}$ contenant a est cohérent) est un méta argument dans le CAN qui code $\mathcal{G}$. Soit AR' l'ensemble de tous ces méta arguments. Notons que tout argument non puissant dans AR ne va pas être représenté dans AR' puisqu'il n'appartient à aucun

ensemble cohérent. Pour $\mathcal{C}_1, \mathcal{C}_2 \in AR'$, $\mathcal{C}_1$ attaque $\mathcal{C}_2$ si et seulement s'il existe $x \in \mathcal{C}_1$ et $y \in \mathcal{C}_2$ avec x *att* y. Soit att' la relation d'attaque qui en résulte.

Théorème 1. Soit $\mathcal{G} = \langle AR, att, \mathcal{N} \rangle$ un CAN, $F = \langle AR', att' \rangle$ son CA correspondant et $S \subseteq AR$. S est une extension basée (resp. complète, préférére, stable, semi-stable) de $\mathcal{G}$ si et seulement si F admet un ensemble $Y = \{\mathcal{C}_1, ..., \mathcal{C}_n\} \subseteq AR'$ comme extension sous la même sémantique avec $S = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n$.

Il se trouve qu'il existe des CANs dont le CA correspondant contient un nombre d'arguments qui est exponentiel par rapport au nombre d'arguments dans le CAN initial. Pour montrer cela, Considérons l'exemple du CAN $\mathcal{G} = \langle AR, att, \mathcal{N} \rangle$ où $AR = \{a\} \cup AR_1 \cup \ldots AR_n$, chaque AR_i contient p arguments ($p > 1$) et $AR_i \; \mathcal{N} \; a$ (pour $1 \leq i \leq n$). Soit F le CA correspondant. Le nombre d'arguments dans $\mathcal{G}$ est $1 + p \times n$. Chaque ensemble $\{a, b_1, \ldots b_n\}$ tel que $b_i \in AR_i$ pour $1 \leq i \leq n$ est un ensemble cohérent minimal contenant a, i.e., est un méta argument dans F. a donne naissance à p^n méta arguments et chaque $x \in AR \setminus \{a\}$ donne naissance à un méta argument ($\{x\}$). Le nombre total des méta arguments est donc $p^n + p \times n$. Ainsi, même si l'information présent dans un CAN peut être toujours codée dans un CA de Dung, l'utilisation d'un CAN en général, peut permettre une représentation significativement plus concise que celle obtenue en utilisant le CA correspondant.

4 Caractérisation en termes d'étiquetages

Dans cette section, L'approches d'étiquetages pour les sémantiques d'acceptabilité introduites pour les CAs de Dung [16] est généralisée aux CANs. Cette approche fournie une caractérisation complète des différente sémantiques d'acceptabilité en se basant sur des étiquettes attribués aux arguments et indiquant leur statuts (accepté, rejeté ou indéfini). En particulier, la vision de la notion de nécessité sous l'angle des d'étiquetages permet de mieux l'appréhender et la comparer à la relation d'attaque.

Soit $\mathcal{G} = \langle AR, \; att, \; \mathcal{N} \rangle$ un CAN. Un étiquetage est une fonction $\mathcal{L} : AR \longrightarrow \{in, out, undec\}$. On pose $in(\mathcal{L}) = \{a \in AR | \mathcal{L}(a) = in\}$, $out(\mathcal{L}) = \{a \in AR | \mathcal{L}(a) = out\}$ et $undec(\mathcal{L}) = \{a \in AR | \mathcal{L}(a) = undec\}$ et on note un étiquetage $\mathcal{L}$ par le triplet $(in(\mathcal{L}), out(\mathcal{L}), undec(\mathcal{L}))$. La notion d'étiquetage légal est généralisée aux CANs comme suit.

Définition 5. Soit a un argument et $\mathcal{L}$ un étiquetage.

- a est légalement *in* si et seulement si a et étiqueté *in* et les deux conditions suivantes sont satisfaites : (1) pour chaque argument b, si b *att* a alors $b \in out(\mathcal{L})$ (tous les attaquants de a sont *out*) et (2) pour tout ensemble d'arguments E, si $E \; \mathcal{N} \; a$ alors $E \cap in(\mathcal{L}) \neq \emptyset$ (au moins un argument de chaque ensemble nécessaire à a est *in*).
- a est légalement *out* si et seulement si a est étiqueté *out* et au moins une des deux conditions suivantes est satisfaite : soit (1) il existe un argument b tel que b *att* a et $b \in in(\mathcal{L})$ (au moins un attaquant de a est *in*) soit (2) il existe un ensemble d'arguments E, tel que $E \; \mathcal{N} \; a$ et $E \subseteq out(\mathcal{L})$ (tous les arguments d'au moins un ensemble d'arguments nécessaire à a sont *out*).
- a est légalement *undec* si et seulement si a est étiqueté *undec* et les trois conditions suivantes sont vérifiées : (1) Pour chaque argument b, si b *att* a alors $b \notin in(\mathcal{L})$ (aucun attaquant de a n'est *in*), (2) Pour tout ensemble $E \subseteq AR$, si $E \; \mathcal{N} \; a$ alors $E \not\subseteq out(\mathcal{L})$ (Aucun des ensembles nécessaire à a n'est tel que tous ces arguments sont *out*) et (3) soit il existe un argument b tel que b *att* a et $b \notin out(\mathcal{L})$ ou il existe $E \subseteq AR$ tel que $E \; \mathcal{N} \; a$ et $E \cap in(\mathcal{L}) = \emptyset$ (soit au moins un attaquant de a n'est pas *out* soit au moins un ensemble nécessaire à a ne contient aucun argument qui est *in*).

Notons que pour $\mathcal{N} = \emptyset$, on trouve exactement les définitions originales des étiquetages légaux données dans [16]. En plus de de la légalité des étiquetages, la présence de la relation de nécessité impose deux contraintes supplémentaires. Tout argument qui n'est pas puissant dans AR n'appartient à aucune extension et doit être étiqueté *out* et comme chaque extension E sous n'importe quelle sémantique doit être cohérente, l'ensemble des arguments *in* de tout étiquetage caractérisant une sémantique d'acceptabilité quelconque pour un CAN doit être cohérent. Les étiquetages satisfaisant ces contraintes sont appelés étiquetages *sûrs*.

Définition 6. Un étiquetage $\mathcal{L}$ est dit *sûr* si et seulement si l'ensemble $in(\mathcal{L})$ est cohérent et pour tout $a \in AR$: si a n'est pas puissant dans A alors $a \in out(\mathcal{L})$.

L'idée étant que pour être sûr, les argument *in* d'un étiqutage $\mathcal{L}$ doivent former un ensemble cohérent, i.e., tous les arguments de $in(\mathcal{L})$ sont puissants dans $in(\mathcal{L})$. Notons qu'en général, ceci peut ne pas être le cas même si tous les arguments de $in(\mathcal{L})$ sont légalement *in*. Aussi, un qui n'est pas puissant dans A peut devenir puissant dans $in(L)$ lorsqu'il est n'est pas Sans-Cycle-N dans A mais Sans-Cycle-N dans $in(\mathcal{L})$.On note que dans un étiquetage sûr, la relation $\mathcal{N}$ ne joue aucun rôle pour déterminer si

un argument est légalement *in* ou pas (la cohérence de $in(\mathcal{L})$ assure la condition (2) pour la légalité *in* d'un argument). La décision dépend uniquement de la relation d'attaque comme dans le cas des CAs classiques. Une fois la notion d'étiquetage est étendue pour tenir compte de la relation de nécessité, les différents types d'étiquetages sont définis comme d'habitude sauf qu'ils doivent toujours être sûrs.

Définition 7. Un étiquetage $\mathcal{L}$ est : *admissible* si et seulement si $\mathcal{L}$ est sûr et ne contient aucun argument qui soit illégalement *in* ou illégalement *out* ; *complet* si et seulement si $\mathcal{L}$ est admissible et ne contient aucun argument qui soit illégalement *undec* ; *basé* si et seulement si $\mathcal{L}$ est complet et $in(\mathcal{L})$ est $\subseteq$-minimal ; *préférée* si et seulement si $\mathcal{L}$ est complet et $in(\mathcal{L})$ est $\subseteq$-maximal ; *stable* si et seulement si $\mathcal{L}$ est complet et $undec(\mathcal{L}) = \emptyset$ et *semi-stable* si et seulement si $\mathcal{L}$ est complet et $undec(\mathcal{L})$ est $\subseteq$-minimal.

Pour les CAs de Dung (i.e. un CAN où $\mathcal{N} = \emptyset$), tout ensemble d'arguments est sûr. Dans ce cas, on obtient exactement les définitions classiques des arguments légalement *in*, *out* et *undec* et des différents types d'étiquetage. La relation entre les étiquetages et les sémantiques d'acceptabilité pour les CANs est donnée comme suit.

Théorème 2. S est un ensemble admissible (resp. extension complète, basée, préférée, stable, semi-stable) si et seulement s'il existe un étiquetage admissible (resp. complet, basé, préféré, stable, semi-stable) $\mathcal{L}$ tel que $S = in(\mathcal{L})$.

Exemple 2 (cont.). Considérons à nouveau le CAN de l'exemple 2.

Considérons les étiquetages : $\mathcal{L}_1 = (\{c\}, \{b\}, \{a\})$, $\mathcal{L}_2 = (\emptyset, \emptyset, \{a, b, c\})$, $\mathcal{L}_3 = (\emptyset, \{a, b\}, \{c\})$, et $\mathcal{L}_4 = (\{c\}, \{a, b\}, \emptyset)$ pour $\mathcal{G}_1$. Dans $\mathcal{L}_1$ c est légalement *in* car $\mathcal{L}_1(b) = out$ mais b est illégalement *out*. De plus, $\mathcal{L}_1$ n'est pas sûr car b n'est pas puissant dans AR mais $a \notin out(\mathcal{L}_1)$. Ainsi, $\mathcal{L}_1$ n'est pas admissible. $\mathcal{L}_2$ n'est pas sûr pour la même raison et donc, il n'est pas admissible. Dans $\mathcal{L}_3$, a et b sont légalement *out* mais c est illégalement *undec*. $\mathcal{L}_3$ est admissible mais pas complet. Dans $\mathcal{L}_4$ c est légalement *in* et a et b sont légalement *out*. De plus, $\mathcal{L}_4$ est sûr et donc il est admissible et complet (pas d'argument illégalement *undec* dans $\mathcal{L}_4$). En résumé, on peut vérifier que $\mathcal{L}_3$ et $\mathcal{L}_4$ sont les étiquetages admissibles de $\mathcal{G}_1$ et $\mathcal{L}_4$ est son unique étiquetage complet qui est aussi son unique étiquetage basé et préféré. De plus, puisque $undec(\mathcal{L}_4) = \emptyset$, $\mathcal{L}_4$ est aussi l'unique étiquetage stable et semi-stable de $\mathcal{G}_1$.

$\mathcal{L}_1 = (\emptyset, \emptyset, \{a, b, c, d\})$ et $\mathcal{L}_2 = (\{d\}, \{c\}, \{a, b\})$ sont les étiquetages admissibles de $\mathcal{G}_2$. $\mathcal{L}_2$ est le seul étiquetage complet de $\mathcal{G}_2$ (d est illégalement *undec* dans $\mathcal{L}_1$) qui est aussi son unique étiquetage basé et préféré. De plus, puisque $undec(\mathcal{L}_2) \neq \emptyset$, $\mathcal{L}_4$ n'est pas un étiquetage stable mais il est un étiquetage semi-stable de $\mathcal{G}_1$.

$\mathcal{L}_1 = (\emptyset, \emptyset, \{a, b, c, d, e, f, g\})$, $\mathcal{L}_2 = (\{a\}, \{b, c\}, \{d, e, f, g\})$, $\mathcal{L}_3 = (\{b\}, \{a\}, \{c, d, e, f, g\})$ et $\mathcal{L}_4 = (\{a, d\}, \{b, c, e\}, \{f, g\})$ sont les étiquetages admissibles de $\mathcal{G}_3$. Parmi eux, seulement $\mathcal{L}_2$ n'est pas complet (d est illégalement *undec* dans $\mathcal{L}_2$). L'étiquetage basé (qui minimise les arguments *in*) est $\mathcal{L}_1$, les étiquetages préférés (qui maximisent les arguments *in*) sont $\mathcal{L}_3$ et $\mathcal{L}_4$. Aucun étiquetage complet n'a d'ensemble vide d'arguments *undec*. Donc, aucun étiquetage n'est stable. Le seul étiquetege semi-stable stable (qui minimise les arguments *undec*) est $\mathcal{L}_4$.

5 Algorithmes d'étiquetage pour les CANs

Dans cette section, nous adaptons les algorithmes d'étiquetage donnés dans [16] pour les extensions basée, préférée, stable et semi-stable au cas des CANs. Nous montrons qu'un traitement approprié de la relation de nécessité permet de garder la principale forme des algorithmes existants proposés initialement pour les CAs de Dung AFs dans le cas des CANs.

5.1 Sémantique basée

Pour les CAs de Dung, L'étiquetage basé est construit par une application successive de la fonction caractéristique en démarrant de l'ensemble vide et en s'arrêtant lorsqu'un point fixe est atteint. D'abord, l'algorithme attribue l'étiquette *in* à tous les arguments qui ne sont pas attaqués. Ensuite, chaque argument attaqué par ces arguments est étiqueté *out*. Après cela, chaque argument dont tous les attaquants sont étiquetés *out* est étiqueté *in*, et ainsi de suite. Lorsqu'aucun argument ne peut être étiqueté *in* ou *out*, l'algorithme attribue aux arguments restants non encore étiquetés l'étiquette *undec* et se termine.

Nous proposons un algorithme similaire pour la sémantique basée dans les CANs avec deux considérations supplémentaires :

- Au tout début de l'algorithme, tous les arguments non puissants dans AR sont étiquetés *out*. La détection des arguments non puissants d'un ensemble S est effectuée par un algorithme itératif simple. Cet algorithme commence par sélectionner tout argument $a \in S$ pour lequel il n'existe aucun $E \subseteq AR$ tel que $E \mathcal{N} a$. En-

suite, à l'itération i, si l'ensemble de tous les arguments obtenus jusqu'ici est H_{i-1} alors $H_i = H_{i-1} \cup \{a | si\ E\ \mathcal{N}\ a\ alors\ E \cap H_{i-1} \neq \emptyset\}$. L'algorithme se termine dès qu'aucun autre argument ne peut être ajouté ($H_i = H_{i-1}$). Les arguments qui restent : $S \setminus H_i$ sont tous les arguments non puissants dans S ;
- l'ajout de nouveaux arguments *in* ou *out* doit prendre en compte aussi bien la relation d'attaque que celle de nécessité.

Le premier point est requis puisque les arguments non puissants ne doivent pas être présents dans l'extension basée et les attaques dont ils représentent les sources ne doivent pas être prises en compte. Une fois cette première opération est effectuée, il est simple de vérifier que l'ensemble des arguments *in* obtenu après chaque itération est cohérent. Pour le deuxième point, la condition d'ajout d'un nouvel argument *in* est renforcée : un argument devient *in* non simplement lorsque ses attaquants sont *out* mais en plus, lorsqu'au moins un argument de chaque ensemble qui est nécessaire pour lui est *in*. La condition d'ajout d'un nouvel argument *out* est affaiblie : un argument devient *out* soit lorsque l'un de ses attaquants est *in* soit lorsque tous les arguments de l'un de ses ensembles nécessaires sont *out*. L'algorithme 1. permet de calculer l'étiquetage basée d'un CAN :

Algorithm 1 Algorithme pour calculer l'étiquetage basée d'un CAN

1: Args-Out $\leftarrow$ arguments non puissants dans AR ;
2: $\mathcal{L}_0 \leftarrow (\emptyset$, Args-Out $, \emptyset)$;
3: **repeat**
4: $in(\mathcal{L}_{i+1}) \leftarrow in(\mathcal{L}_i) \cup \{x | x$ n'est pas étiqueté dans $\mathcal{L}_i$ et $\forall y \in AR$: si $y\ att\ x$ alors $y \in out(\mathcal{L}_i)$ et $\forall E \subseteq AR$: si $E\ \mathcal{N}\ x$ alors $E \cap in(\mathcal{L}_i) \neq \emptyset\}$;
5: $out(\mathcal{L}_{i+1}) \leftarrow out(\mathcal{L}_i) \cup \{x | x$ n'est pas étiqueté dans $\mathcal{L}_i$ et soit $\exists y \in AR : y\ att\ x$ et $y \in in(\mathcal{L}_i)$, soit $\exists E \subseteq AR : E\ \mathcal{N}\ x$ et $E \subseteq out(\mathcal{L}_{i+1})\}$;
6: **until** $(\mathcal{L}_{i+1} = \mathcal{L}_i)$
7: $\mathcal{L}_G \leftarrow (in(\mathcal{L}_i), out(\mathcal{L}_i), AR \setminus (in(\mathcal{L}_i) \cup out(\mathcal{L}_i)))$;

5.2 Sémantiques préférée, stable et semi-stable

Les algorithmes d'étiquetage originaux pour les sémantiques préférée, stable et semi-stable sont adaprtés ici au cas des CANs en injectant deux principales modifications : introduire une nouvelle opération (*opération de nettoyage*) au début de chaque itération de l'algorithme et modifier légèrement ce qu'on appelle l'étape de transition. L'opération de nettoyage d'un étiquetage $\mathcal{L}$ enlève les arguments qui ne sont pas puissants dans $in(\mathcal{L})$ afin d'assurer la cohérence dans les extensions. L'application de l'opération de nettoyage au début de l'algorithme permet d'enlever les argumentn non puissants dans AR comme dans le cas de l'algorithme pour l'extension basée. On a besoin d'utiliser cette opération à nouveau au début de chaque itération car il est possible qu'un sous-ensemble d'un ensemble cohérent soit incohérent.

Définition 8. Soit $\mathcal{L}$ un étiquetage sans arguments illégalement *out*. L'étiquetage qui résulte du *nettoyage* de $\mathcal{L}$, noté $\mathcal{C}(\mathcal{L})$, est obtenu par : (1) pour chaque argument a de $in(\mathcal{L})$ qui est non puissant dans $in(\mathcal{L})$, changer l'étiquette de a de *in* à *out*, (2) ensuite, tant qu'il reste des arguments illégalement *out*, changer leurs étiquettes à *undec*.

La première étape change simultanément à *out* tous les arguments *in* qui ne sont pas puissants dans $\mathcal{L}$. En fait, ces arguments forment collectivement une partie des arguments *in* qui ne doivent pas être gardés. La deuxième étape consiste en une étape de propagation qui assure qu'il ne reste plus d'argument qui est illégalement *out* dans l'étiquetage résultat. Il est évident qu'on a :

Proposition 4. Si $\mathcal{L}$ est un étiquetage alors $\mathcal{C}(\mathcal{L})$ est sûr et ne contient pas d'arguments illégalement *out*.

Dans une étape de transition classique, l'étiquette d'un argument illégalement *in* a est changée à *out*. Ensuite, uniquement a et certains arguments attaqués par a peuvent devenir illégalement *out* et doivent recevoir l'étiquette *undec*. Dans les CANs, à cause de la relation de nécessité, d'autres arguments peuvent devenir illégalement *out* et doivent être étiquetés *undec*. Ainsi, le processus doit être répété jusqu'à ce qu'aucun tel argument reste. Evidemment, si $\mathcal{N} = \emptyset$ alors on trouve la définition classique d'une étape de transition.

Définition 9. Soit $\mathcal{L}$ un étiquetage sûr et a un argument illégalement *in* dans $\mathcal{L}$. Une étape de transition sur a dans $\mathcal{L}$ se compose de ce qui suit : (1) changer l'étiquette de a de *in* à *out* ; ensuite (2) tant qu'il reste un argument qui est illégalement *out* dans $\mathcal{L}$, changer son étiquette à *undec*.

En se basant sur les notions précédentes, une séquence de transitions est définie comme une liste $[\mathcal{C}(\mathcal{L}_0), a_1, \mathcal{C}(\mathcal{L}_1), \ldots, a_n, \mathcal{C}(\mathcal{L}_n)]$ $(n \geq 0)$ où chaque argument a_i $(0 \leq i \leq n)$ est illégalement *in* dans $\mathcal{C}(\mathcal{L}_{i-1})$ et $\mathcal{L}_n$ est l'étiquette qui résulte de l'étape de transition sur a_i dans $\mathcal{C}(\mathcal{L}_{i-1})$. Une séquence de transitions est terminée si et seulement si $\mathcal{C}(\mathcal{L}_n)$ ne contient aucun argument qui est illégalement

in. Comme dans le cas classique, tout au long de la séquence de transitions, le nombre d'arguments étiquetés *in* est non croissant tandis que le nombre d'arguments étiquetés *undec* est non décroissant.

Proposition 5. Soit $[\mathcal{C}(\mathcal{L}_0), a_1, \mathcal{C}(\mathcal{L}_1), \ldots, a_n, \mathcal{C}(\mathcal{L}_n)]$ une séquence de transitions. Pour tout $i \geq 0$: $in(\mathcal{C}(\mathcal{L}_{i+1})) \subseteq in(\mathcal{C}(\mathcal{L}_i))$ et $undec(\mathcal{C}(\mathcal{L}_i)) \subseteq undec(\mathcal{C}(\mathcal{L}_{i+1}))$.

Une conséquence immédiate de la proposition précédente est que si le nombre d'arguments est fini alors, toute séquence de transitions terminée est aussi finie. A présent, le résultat suivant montre que les ensembles admissibles et uniquement eux sont trouvés à partir des séquences de transitions terminées qui commencent par l'étiquetage *Tous-In* (l'étiquetage où tous les arguments sont étiquetés *in*).

Théorème 3. Soit $[\mathcal{C}(\mathcal{L}_0), a_1, \mathcal{C}(\mathcal{L}_1), \ldots, a_n, \mathcal{C}(\mathcal{L}_n)]$ ($n \geq 0$) une séquence de transitions terminée où $\mathcal{L}_0$ est l'étiquetage *Tous-In* alors $\mathcal{C}(\mathcal{L}_n)$ est un étiquetage admissible. Inversement, pout chaque étiquetage admissible $\mathcal{L}$ il existe une séquence de transitions terminée $[\mathcal{C}(\mathcal{L}_0), a_1, \mathcal{C}(\mathcal{L}_1), \ldots, a_n, \mathcal{C}(\mathcal{L}_n)]$ ($n \geq 0$) où $\mathcal{L}_0$ est l'étiquetage *Tous-In* et $\mathcal{C}(\mathcal{L}_n) = \mathcal{L}$.

Les ensembles admissibles sont calculés en construisant un arbre avec *$\mathcal{C}$(Tous-In)* comme racine et pour tout noeud $\mathcal{C}(\mathcal{L})$ et tout a qui est illégalement *in* dans $\mathcal{C}(\mathcal{L})$, si $\mathcal{L}'$ est le résultat de l'étape de transition sur a dans $\mathcal{C}(\mathcal{L})$ alors $\mathcal{C}(\mathcal{L}')$ est un fils de $\mathcal{C}(\mathcal{L})$. Les feuilles de l'arbre (l'ensemble des éléments finaux de toutes les séquences de transitions terminées) correspondent aux ensembles admissibles.

Pour calculer les extensions sous une sémantique donée, quelques techniques d'optimisation sont proposées dans [16] pour les CAs de Dung. Heureusement, ces techniques restent valides pour les CANs. Rappelons ces techniques brièvement. A un point donné de l'algorithme, soit Σ l'ensemble des extensions candidates trouvées jusqu'ici.

Pour la sémantique préférée (resp. semi-stable), on garde les étiquetages admissibles qui maximisent (resp. minimisent) l'ensemble des arguments *in* (resp. *undec*). Ainsi, si l'étiquetage courant $\mathcal{L}$ est tel que $in(\mathcal{L}) \subseteq in(\mathcal{L}')$ (resp. $undec(\mathcal{L}') \subseteq undec(\mathcal{L})$) pour un certain $\mathcal{L}' \in \Sigma$ alors on arrête le développement de la branche courante. En effet, d'après la proposition 4., tout déscendant $\mathcal{L}''$ de $\mathcal{L}$ va être plus mauvais que $\mathcal{L}'$ car $in(\mathcal{L}'') \subseteq in(\mathcal{L}')$ (resp. $undec(\mathcal{L}') \subseteq undec(\mathcal{L}'')$). De plus, si un nouvel étiquetage $\mathcal{L}$ est trouvé, $\mathcal{L}$ est ajouté à l'ensemble des étiquetages candidats et tout étiquetage candidat qui est plus mauvais que $\mathcal{L}$ est supprimé. Pour la sémantique stable, on garde seulement les étiquetages admissibles où l'ensemble des arguments *undec* est vide. Ainsi, si l'étiquetage courant $\mathcal{L}$ est tel que $undec(\mathcal{L}) \neq \emptyset$ on arrête le développement de la branche courante. En effet, d'après la proposition 4., pour tout déscendant $\mathcal{L}''$ de $\mathcal{L}$ on va avoir $undec(\mathcal{L}'') \neq \emptyset$. Donc, la branche courante ne peut pas conduire à un étiquetage stable.

Une troisième technique d'optimisation utilise ce que l'on appelle les arguments *super-illégalement in*. Un argument illégalement *in* est dit super-illégalement *in* si et seulement s'il est attaqué par un argument qui est soit légalement *in* soit légalement *undec*. En présence d'arguments super-illégalement *in*, il suffit d'effectuer une étape de transition sur l'un d'eux au lieu de considérer tous les arguments illégalement *in*. La raison est que si a est super-illégalement *in* dans un étiquetage $\mathcal{L}$ alors, toute séquence de transitions qui commence par $\mathcal{L}$ et qui n'effectue pas d'étape de transition sur a conduit à un étiquetage dans lequel a reste illégalement *in*. Heureusement, cette raison reste valable pour les CANs :

Proposition 6. Soit $\mathcal{L}_0$ un étiquetage sûr où un argument a est super-illégalement *in* et $[\mathcal{L}_0, a_1, \mathcal{C}(\mathcal{L}_1), \ldots, a_n, \mathcal{C}(\mathcal{L}_n)]$ une séquence de transitions où $a \notin \{a_1, \ldots, a_n\}$ alors a est illégalement *in* dans $\mathcal{C}(\mathcal{L}_n)$.

Maintenant, voici l'algorithme qui calcule les étiquetages préférés. Nous décrivons ensuite comment modifier cet algorithme afin de calculer les étiquetages stables et semi-stables.

Pour la sémantique semi-stable, l'arbre est élagué lorsque l'ensemble des arguments étiquetés *undec* dans l'étiquetage courant est un sur-ensemble des arguments étiquetés *undec* dans un candidat existant. Ainsi, la ligne 10 doit être remplacée par : " `if` ($\exists \mathcal{L}' \in$ ens-etiqs : $undec(\mathcal{L}') \subset undec(\mathcal{L})$) `then`". Pour la sémantique stable, l'arbre est élagué dès que l'étiquetage courant a un ensemble vide d'arguments étiquetés *undec*. Ainsi, on doit remplacer la ligne 10 par : " `if` ($undec(\mathcal{L}) \neq \emptyset$) `then`". De plus, puisqu'une extension stable n'est jamais strictement incluse dans une autre, les lignes 14-18 peuvent être enlevées.

6 Conclusion et perspectives

Dans cet article, nous continuons le travail présenté dans [17] sur les CANs dans deux directions. D'abord, nous introduisons de nouvelles sémantiques d'acceptabilité pour les CANs, notamment les sémantiques complète, basée et semi-stable. Nous définissons toutes ces sémantiques de manière très similaire aux sémantiques

Algorithm 2 Algorithme pour calculer les étiquetages préférés d'un CAN

```
1: begin
2: ens-etiqs ← ∅ ;
3: trouve-etiqs(Tous-In) ;
4: print ens-etiqs ;
5: end.
6:
7: procedure trouve-etiqs(Lab)
8: begin
   {nettoyer l'étiquetage Lab}
9: L ← nettoyer(Lab) ;
   {si L est plus mauvais qu'un étiquetage existant,
   alors s'arrêter}
10: if (∃L' ∈ ens-etiqs : in(L) ⊂ in(L')) then
11:    arrêter ;
12: end if
   {si une séquence de transitions est terminée, alors
   enlever les candidats les plus mauvais, ajouter le
   nouvel étiquetage et s'arrêter}
13: if (aucun argument n'est illégalement in dans L)
   then
14:    for (chaque L' ∈ ens-etiqs) do
15:       if in(L') ⊂ in(L) then
16:          ens-etiqs ← ens-etiqs \{L'} ;
17:       end if
18:    end for
       {ajouter L comme nouveau candidat}
19:    ens-etiqs ← ens-etiqs ∪ {L} ;
20:    arrêter ;
21: else
22:    if (∃ des arguments super-illégalement in dans
       L) then
23:       x ← un argument super-illégalement in dans
          L ;
24:       trouve-etiqs(etape-transition(L,x)) ;
25:    else
26:       for (chaque x illégalement in dans L) do
27:          trouve-etiqs(etape-transition(L,x)) ;
28:       end for
29:    end if
30: end if
31: end ;
```

correspondantes définies pour les CAs classiques en injectant la relation de nécessité dans la définition de certaines notions élémentaires. Les sémantiques proposées pour les CANs gardent les même propriétés que celles des sémantiques classiques correspondantes et représentent des généralisations propres de celles-ci (les nouvelles sémantiques coïncident avec les anciennes lorsque la relation de nécessité est vide). Nous avons également montré que tout CAN peut être representé par un CA de Dung équivalent. Cependant, Dans le cas général, un CAN peut exprimer l'information qui ne peut pas être codée dans le CA équivalent sans utiliser un nombre exponentiel d'arguments par rapport au nombre d'arguments dans le CA original. Cela signifie que, du point de vue représentation, il y a des situations où les CANs sont significativement plus concis que les CAs pour la repésentation des mêmes connaissances. La deuxième contribution de ce travail est une généralisation des algorithmes d'étiquetage de Caminada proposés pour les CA de Dung au cas des CANs. Nous avons montré que grâce à un traitement approprié de la relation de nécessité, les modifications requises dans cette généralisation ne sont pas significatives et la forme principale de l'algorithme originale est gardée. De plus, si la relation de nécéssité est vide, on retrouve exactement les algorithmes d'étiquetage originaux utlisés pour les CAs de Dung.

Comme les CAs de Dung, les CANs restent très abstraits et ne supposent aucune structure pour les arguments. Leur utilisation pratique nécessite une étape d'instantiation qui permet de construire des arguments, des attaques et des nécessitiés à partir de bases de connaissances "concrètes". Différents travaux ont été proposés dans ce domaine et la plupart d'eux utilisent des bases de connaissances exprimées soit en logique classique soit comme programmes logiques. La première classe de travaux (voir par exemple [1], [15]) suit l'approche logique d'argumentation [4] et construit des arguments à partir d'une base de connaissances logique $\mathcal{K}$ comme couples de la forme $(support, conclusion)$ où *support* est un ensemble minimal de formules de $\mathcal{K}$ impliquant *claim* qui est une formule quelconque. Des travaux récents (voir [2] [3]) ont sérieusement remis en question l'adéquation des sémantiques d'acceptabilité des CAs de Dung dans des systèmes obtenus par cette instantiation.

D'autres travaux construisent les arguments à partir de bases de connaissances exprimées par des PLs. Dans [12], les arguments sont construits à partir des PLs comme couples (K, c) où c est une conséquence défaisable de K et dans [7] les arguments sont construits comme structures arborescentes de règles. Dans [17] chaque règle représente un argument. Ce type d'instantiations donne une vision concrète des CAs et met

plus de lumière sur les liens possibles entre l'argumentation et les PLs. Certains liens entre les sémantiques d'acceptabilité en argumentation et les sémantiques des PLs ont été déjà établis comme le lien : entre les modèles stables des PLs et les sémantiques stables des CAs [12] ou dans les CANs [17], entre la sémantique bien fondée dans les PLs et la sémantique basée dans les CAs [12] et entre les modèles stables partielles dans les PLs et la sémantique complète dans les CAs [18]. Une perspective naturelle est alors de continuer l'investigation pour d'autres liens possibles entre les autres sémantiques proposées aussi bien en programmation logique (comme la sémantique L-stable [14]) qu'en théorie de l'argumentation (comme la sémantique idéal [13]), en utilisant notre méthode d'instantiation. Une autre perspective est d'utiliser les algorithmes d'étiquetage pour proposer de nouveaux solveurs pour la théorie de l'argumentation et la programmation logique sous d'autres sémantiques que la sémantique stable pour laquelle des solveurs très efficaces sont disponibles aujourd'hui. Enfin, nous voulons généraliser les procédures de preuves dialectiques proposés pour les CAs de Dung (voir [16]) aux CANs et aux PLs. Ceci est bénéfique en pratique lorsque le but est de savoir si un argument (resp. atome) est accepté ou pas sous une sémantique donnée par rapport à un certain critère d'acceptabilité (sceptique, crédule, ...) sans calculer toutes les extensions (resp. modèles) sous la sémantique considérée.

Acknowledgments.

Ce travail a été réalisé dans le cadre du projet ANR (Agence Nationale de la Recherche), ASPIQ. Ref : ANR-12-BS02-0003.

Références

[1] Amgoud, L., Besnard, P. : Bridging the gap between abstract argumentation systems and logic. In : Proceedings of SUM'2009, pp. 12–27. (2009).

[2] Amgoud, L. : Stable Semantics in Logic-Based Argumentation. In : Proceedings of SUM'2012, pp. 58–71. (2012).

[3] Amgoud, L. : The Outcomes of Logic-Based Argumentation Systems under Preferred Semantics. In : Proceedings of SUM'2012, pp. 72-84. (2012).

[4] Besnard, P., Hunter, A. : Elements of Argumentation. The MIT Press (2008).

[5] Boella, G., Gabbay, D.M., Van Der Torre, L., Villata, S. : Support in Abstract Argumentation. In : Proceedings of COMMA'2010, pp. 40–51. (2010).

[6] Brewka, G., Woltran, S. : Abstract Dialectical Frameworks. In : Proceedings of KR'2012, pp. 102–111. Toronto, Canada (2010).

[7] Caminada, M.W.A., Carnielli, W.A., Dunne, P.E. : Semi-Stable Semantics. J. Log. Comp. 22(5) :1207–1254 (2012).

[8] Caminada, M.W.A. : Semi-Stable Semantics. In : Proceedings of COMMA'2006, pp. 121–130. Liverpool, UK (2006).

[9] Caminada, M.W.A. : An Algorithm for Computing Semi-Stable Semantics. In : Proceedings of ECSQARU'2007, pp. 222–234. Hammamet, Tunisia (2007).

[10] Cayrol, C., Lagasquie-Schiex, M.C. : On the acceptability of arguments in bipolar argumentation frameworks. In : Proceedings of ECSQARU'05, pp. 378–389. (2005).

[11] Cayrol, C., Lagasquie-Schiex, M.C. : Coalitions of arguments : A tool for handling bipolar argumentation frameworks. Int. J. Intell. Syst. 25(1), 83–109 (2010).

[12] Dung, P.M. : On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intel. 77, 321–357 (1995).

[13] Dung, P.M., Mancarella, P., Toni, F. : Computing ideal sceptical argumentation. Artif. Intel. 171(10-15) :642–674 (2007).

[14] Eiter, T., Leone, N., Sacca, D. : On the partial semantics for disjunctive deductive a databases. Ann. Math. Artif. Intell. 19(1-2), 59–96 (1997).

[15] Gorogiannis, N., Hunter, A. : Instantiating abstract, argumentation with classical logic arguments : Postulates and properties. Artif. Intel. 175, 1479–1497 (2011).

[16] Modgil, S., Caminada, M.W.A. : Proof Theories and Algorithms for Abstract Argumentation Frameworks. In : I. Rahwan and G. Simari (eds.) Argumentation in Artificial Intelligence, pp. 105–129 (2009).

[17] Nouioua, F., Risch, V. : Argumentation Frameworks with Necessities. In : Proceedings of SUM'2011, pp. 163–176. Dayton, Ohio, USA (2011).

[18] Wu, Y., Caminada, M., Gabbay, D. : Complete Extensions in Argumentation Coincide with 3-Valued Stable Models in Logic Programming. Studia logica. 93(2-3) :383–403, Springer, Heidelberg (2009).