Prise en compte de Préférences Conditionnelles dans les Systèmes d'Argumentation

Claudette CAYROL **Sylvie DOUTRE**

Rapport IRIT/99-21-R

Octobre 1999

Rapport IRIT/99-21-R Octobre 1999

Prise en compte de Préférences Conditionnelles dans les Systèmes d'Argumentation

Claudette CAYROL **Sylvie DOUTRE**

Institut de Recherche en Informatique de Toulouse (I.R.I.T.)
Université Paul Sabatier
118, route de Narbonne - 31062 TOULOUSE Cedex (FRANCE)
E-mail : (ccayrol, doutre)@irit.fr

Résumé : L'argumentation est un modèle de raisonnement basé sur la construction d'arguments puis sur la sélection des plus acceptables d'entre eux. Le concept d'acceptabilité prend en compte à la fois l'existence de relations de contrariété entre les arguments et l'expression de préférences entre les connaissances. Les préférences sont généralement données a priori et ne sont pas remises en question au cours du raisonnement. Cependant, dans certaines applications, les préférences peuvent dépendre de nombreux facteurs et être elles-mêmes conclusion du raisonnement. Notre objectif est de formaliser le raisonnement sur des préférences conditionnelles. Il s'agit tout d'abord de représenter les priorités conditionnelles comme éléments de la base de connaissances. La prise en compte d'arguments supportant des priorités a nécessité la définition d'une nouvelle relation de contrariété entre arguments et d'un nouveau type de préférence entre arguments. Nous enrichissons ainsi le concept d'acceptabilité.

Mots clés : inconsistance, argumentation, préférences.

Table des matières

1. Introduction.....1

2. Systèmes d'Argumentation de base.....3

3. Systèmes d'Argumentation avec Préférences (SAP) 6

3.1 SAP basés sur des priorités Incontestables (SAPI)..... *7*

3.1.1 Définitions générales..... 7

3.1.2 Exemples de relations de préférences..... 8

3.1.3 Classes d'acceptabilité 9

3.2. SAP basés sur des priorités Conditionnelles (SAPC) *10*

3.2.1 Expression des priorités dans le langage 12

3.2.2 Redéfinition de l'argument et du conflit par Attaque 16

3.2.3 Conséquence : les priorités peuvent être contestées 17

3.2.4 Autre type de conflit : la disqualification..... 18

3.2.5 Relation de préférence entre arguments 20

3.3 Comparaison SAPI / SAPC..... *23*

4. Acceptabilité des arguments dans les SAPC 24

4.1 Problème de circularité..... *25*

4.2 Sémantique de l'ensemble des arguments acceptables..... *27*

4.3 Théorie de la preuve..... *32*

5. Conclusion 38

6. Perspectives 39

Bibliographie 41

1. Introduction

L'argumentation est une composante majeure du raisonnement de sens commun. On essaie souvent de justifier son opinion sur des points particuliers en donnant des raisons renforçant son point de vue. Ces raisons sont appelées "arguments". L'argumentation est en particulier une approche prometteuse pour raisonner avec des bases de connaissances inconsistantes. Ce modèle est basé sur la construction et la comparaison des arguments. Un argument supportant une donnée peut être vu comme une raison de croire en cette donnée. Par contre un "contre-argument" (ou un "contrariant") est une raison de ne pas croire en la donnée.

Une des applications de l'argumentation est le domaine du raisonnement révisable. Plusieurs systèmes d'argumentation ont été développés ([Dun93], [EFK93], [Cay95], [AC97&98], [PS96&97]). A chacun de ces systèmes est associé un processus d'argumentation qui suit généralement les trois étapes suivantes :

1. Construire des arguments en faveur d'un fait ainsi que des contre-arguments ;
2. Déterminer l'ensemble ou les ensembles des arguments qui sont jugés acceptables ;
3. Conclure.

Nous allons essentiellement nous intéresser à la deuxième étape qui consiste à déterminer, parmi l'ensemble des arguments fournis pour et contre un fait, ceux qui seront retenus. L'idée est de différencier les arguments avec une notion d'acceptabilité.

Dans les systèmes d'argumentation de base, un argument est acceptable s'il n'a pas de contrariant. Les relations de contrariété entre arguments sont basées sur la comparaison des structures des arguments (des critères purement logiques). Par exemple, un argument supportant une donnée contrarie un argument supportant la donnée contraire.

Par la suite, les systèmes d'argumentation ont été enrichis avec l'expression de priorités entre les connaissances. Ces priorités permettent de construire des relations de préférence entre arguments. Dans ces systèmes, le concept d'acceptabilité prend en compte à la fois l'existence de relations de contrariété et l'expression de priorités entre les connaissances. Ainsi, un argument sera acceptable s'il est préféré à tous ses contrariants. Généralement, les priorités sont données a priori, sont figées et ne sont pas remises en question au cours du raisonnement, elles sont donc incontestables.

Cependant, dans certaines applications (telles que le raisonnement juridique par exemple), une priorité entre croyances et ainsi une préférence entre deux arguments, peut dépendre de nombreux facteurs et être elle-même conclusion d'un raisonnement. Nous avons donc étudié le problème de la prise en compte de priorités conditionnelles dans le raisonnement argumentatif. En nous inspirant de certains travaux développés dans le cadre du

raisonnement juridique par [Bre94], [PS95] et [PS96&97], nous avons proposé un nouveau système d'argumentation.

Pour comparer les arguments, nous utilisons une relation de contrariété qui généralise celles qui sont habituellement considérées. Nous montrons que les priorités conditionnelles sont contestables et que leur prise en compte dans l'acceptabilité des arguments amène de nombreux problèmes que nous mettons en évidence.

Le rapport est organisé comme suit :

- La section 2 donne une définition générale des systèmes d'argumentation dans le cadre de leur application au traitement de bases de connaissances incohérentes.
- La section 3 présente les Systèmes d'Argumentation avec Préférences (les SAP) ; nous rappelons dans un premier temps les travaux sur les SAP basés sur des priorités Incontestables (les SAPI), puis nous proposons une formalisation pour les SAP basés sur des priorités Conditionnelles (les SAPC).
- La section 4 étudie l'acceptabilité des arguments dans les SAPC.
- La section 5 résume notre travail.
- La section 6 donne des perspectives de recherche.

2. Systèmes d'Argumentation de base

Définition 2.1 [Dun93]

Un ***système d'argumentation*** est un couple <A, R> avec A un ensemble d'arguments, et R une relation binaire représentant une notion de contrariété entre arguments, i.e. R √ A?A.

(A, B) â R ou, d'une manière équivalente, A R B veut dire que l'argument A *contrarie* l'argument B ; A est ainsi un ***R-contrariant*** (ou un *R-contre-argument*) de B. On dit également que A et B sont *en conflit*.

Le *graphe des conflits* de A est le graphe orienté G = (A, U) tel que :
å A, B â A, (A, B) â U ≠ A R B.

Ainsi, dans un système d'argumentation, certains arguments sont contrariés, d'autres non. Un argument est dit *acceptable* s'il n'est pas contrarié. L'ensemble de tous ces arguments acceptables est appelé classe d'acceptabilité. Formellement :

Définition 2.2

La ***classe d'acceptabilité du système*** *<A, R>* est l'ensemble :
C_R = {A â A / d B â A tel que B R A}

Cette notion d'acceptabilité est très prudente ; elle n'accepte qu'un nombre très restreint d'arguments, puisque seuls les arguments en dehors de tout conflit sont acceptables.

Exemple 2.1

Soit le système <A, R> avec A = {A, B, C, D}.
- si R = {(C, D), (D, C)} alors C_R = {A, B} ;
- si R = {(C, D)}, la classe d'acceptabilité du système est alors C_R = {A, B, C}.

Définition 2.3

Soit un système d'argumentation <A, R> et un ensemble S √ A. On dit que :
S est ***sans-R-conflit*** ssi d A, B â S tels que A R B.
En d'autres termes, il n'existe aucun conflit (par R) entre les éléments de S.

Définition 2.4

Un ***système d'argumentation*** <A, R> est ***fini*** ssi chaque argument de A est contrarié (selon R) par un nombre fini d'arguments.

Plaçons-nous à présent dans le cadre de l'application des Systèmes d'Argumentation au traitement de l'incohérence dans les bases de connaissances.

Les bases de connaissances seront construites sur un langage propositionnel L. La relation d'inférence classique est dénotée par ;.

Une ***base de connaissances*** Σ est un ensemble de formules de L, éventuellement inconsistant et qui n'est pas supposé fermé par l'inférence classique.
Comme dans [EFK93], on distingue deux types de connaissances dans Σ. La base va être partagée en deux sous-bases (K, E). K représente le noyau de connaissances qui contient les connaissances sûres, alors que E contient les ***croyances*****,** c'est-à-dire les connaissances qui sont incertaines et contestables. K est supposé consistant, alors que généralement KuE est inconsistant.

Exemple 2.2

Σ = (K, E) avec K = {p} et E = {o–v, p–¬v, p–o} ; la sémantique est la suivante : p est interprété par pingouin, o par oiseau et v par voler.

Définition 2.5 [EFK93]
Etant donné Σ = (K, E) une base de connaissances, un ***argument*** de E dans le contexte K est un couple (H, h) où h est une formule du langage L et H est une sous-base de E vérifiant les conditions suivantes :

i) K u H est consistant (i.e. H est K-consistant)
ii) K u H $\vdash$ h
iii) H est minimal pour l'inclusion pour i) et ii)

H est appelé le ***support*** de l'argument et h sa ***conclusion***.

Notation : A(Σ) désigne l'ensemble des arguments construits sur une base Σ.

Pour qu'ils soient plus faciles à manipuler, les arguments seront le plus souvent nommés à l'aide de lettres majuscules éventuellement indicées. Deux arguments différents seront dénotés par deux noms différents.

Exemple 2.2 (suite)

L'ensemble A(Σ) des arguments contient, entre autres :
A_1 *: ({p–o, o–v}, v),* ***A_2*** *: ({p–¬v}, ¬v),* ***A_3*** *: ({p–o, p–¬v}, ¬(o–v)),* ***A_4*** *: ({p–o, p–¬v}, oÊ¬v).*
A_1 et A_2 sont des arguments pour un littéral, A_3 est un argument pour la négation d'une clause, A_4 est un argument pour une conjonction de littéraux.

Définition 2.6
Soit (H, h) et (H', h') deux arguments construits sur une base Σ.

- (H, h) est un ***sous-argument*** de (H', h') ssi H $\surd$ H' ;
- (H, h) et (H', h') sont ***équivalents*** ssi H = H' et h @ h'.

Exemple 2.2 (suite)

A_2 est un sous-argument de A_3, A_3 et A_4 sont équivalents.

Parmi les relations de contrariété entre arguments qui ont été définies, nous retiendrons celles qui suivent, utilisées dans [EFK93] :

Définition 2.7

Soit $A_1 : (H_1, h_1)$ et $A_2 : (H_2, h_2)$ deux arguments.

- A_1 ***réfute*** A_2 ssi $h_1 @ \neg h_2$

  i.e. un argument est réfuté ssi il existe un argument pour la négation de sa conclusion.

- A_1 ***attaque*** A_2 ssi ™h â H_2 / $h_1 @ \neg h$

  i.e. un argument est attaqué ssi il existe un argument pour la négation d'un élément de son support.

  On dit également que A_1 attaque A_2 sur h.

A noter que la relation de réfutation est une relation symétrique, et que la relation d'attaque est non symétrique.

Exemple 2.2 (suite)

Les arguments A_1 et A_2 se réfutent, et A_3 attaque l'argument A_1 sur (o–v).

Exemple 2.3

*Soit une base $\Sigma = (K, E)$ telle que $K = \emptyset$ et $E=\{a, a–b, \neg a, c\}$. Soit S le sous-ensemble de $A(\Sigma)$ contenant les quatre arguments suivants : **A_1** : ($\{a\}$, a), **A_2** : ($\{a, a–b\}$, b), **A_3** : ($\{\neg a\}$, $\neg$a), **A_4** : ($\{c\}$, c).*

Le graphe des conflits du système $<S,$ Réfute$>$ est :

$$A_1 \; R \; A_3$$

et celui du système $<S,$ Attaque$>$ est :

$$A_1 \; R \; A_3 \quad \downarrow \quad A_2$$

La classe d'acceptabilité $C_{Réfute}$ du système $<S,$ Réfute$>$ contient les arguments A_2 et A_4. La classe d'acceptabilité $C_{Attaque}$ du système $<S,$ Attaque$>$ contient A_4.

3. Systèmes d'Argumentation avec Préférences (SAP)

Exemple 3.1

Sur la base de l'exemple 2.2 sur les oiseaux, nous avons vu que l'on peut construire un argument A_1 en faveur de v et un argument A_2 en faveur de ¬v. Or A_1 et A_2 se réfutent et sont tous deux attaqués. Ils n'appartiennent donc à aucune des deux classes d'acceptabilité $C_{Réfute}$ et $C_{Attaque}$. Or le raisonnement de sens commun permet de conclure que le pingouin ne vole pas, en accordant plus d'importance à la croyance p–¬v qu'à la croyance o–v.

On voit ici que les relations d'inférence des systèmes d'argumentation de base sont trop fortes. Pour les assouplir, une nouvelle notion a été introduite : la notion de "force". On va pouvoir accepter un argument contrarié à condition qu'il soit "plus fort" que son contrariant [AC97&98].

Ainsi, les systèmes d'argumentation utilisant les préférences se définissent à l'aide d'un troisième paramètre : un pré-ordre sur les supports d'arguments qui permet de déterminer une relation de préférence entre les arguments.

Définition 3.1

Un ***Système d'Argumentation avec Préférences (un SAP)*** est un triplet <A(∑), R, Pref> avec :

- A(∑) un ensemble d'arguments, construits sur une base de connaissances ∑ = (K, E),
- R une relation binaire représentant une notion de contrariété entre les arguments,
- Pref un pré-ordre (partiel ou total) sur l'ensemble des sous-bases de E.

Dans le cadre de l'application au traitement des bases de connaissances inconsistantes, le pré-ordre sur les supports est défini à partir d'une relation de priorité entre les connaissances. Dans la première partie de cette section, nous allons considérer le cas où les priorités sont incontestables ; elles sont alors figées et ne peuvent être remises en question. C'est le cas le plus couramment étudié. Cependant il existe d'autres types de priorités entre connaissances, qui sont contestables. Nous étudierons dans une deuxième partie le cas des priorités conditionnelles.

N. B. : Concernant la terminologie, on parlera de *priorités* entre croyances et de *préférences* entre arguments, priorités et préférences renvoyant toutes deux à une même notion de supériorité, de prééminence. Cette simple distinction permet de savoir à quel niveau on se place (croyances ou arguments) quand on parle de préférences.

3.1 SAP basés sur des priorités Incontestables (SAPI)

3.1.1 Définitions générales

On entend par priorités incontestables des priorités qui sont données a priori, qui sont figées et ne sont pas remises en question. Ces priorités peuvent s'exprimer, par exemple, via un pré-ordre entre les croyances, donné de manière exogène.

Dans les Systèmes d'Argumentation avec Préférences basés sur des priorités Incontestables (les ***SAPI***), la relation Pref est un mode d'agrégation permettant de passer du pré-ordre (donné a priori) entre les croyances, à un pré-ordre entre les sous-bases de E (et donc entre les supports d'arguments). A partir de cette relation Pref entre les supports, on définit une relation de préférence entre arguments (une Pref-préférence), notée $>^{Pref}$.

Définition 3.2

Soit <A(∑), R, Pref> un SAPI et A : (H_A, h_A), B : (H_B, h_B) deux arguments de A(∑).

A est ***Pref-préféré*** à B (noté A $>^{Pref}$ B) ssi H_A Pref H_B.

Notations :

L'ordre strict associé à la relation Pref est $\underline{Pref}$, celui associé à la relation $>^{Pref}$ est $>>^{Pref}$.

H_A $\underline{Pref}$ H_B ≠ H_A Pref H_B et non(H_B Pref H_A) .

A est ***strictement Pref-préféré*** à B (noté A $>>^{Pref}$ B) ssi A $>^{Pref}$ B et non(B $>^{Pref}$ A)

ssi H_A $\underline{Pref}$ H_B.

En utilisant cette relation de préférence entre arguments, on modélise une notion de *défense individuelle*. Un argument se défend seul ssi il est préféré à tous ses contrariants. Formellement :

Définition 3.3

Soit <A(∑), R, Pref> un SAPI et A, B deux arguments de A(∑) tels que A R B.

On dit que A ***se défend seul*** contre B ssi A est strictement Pref-préféré à B i.e. A $>>^{Pref}$ B.

Définition 3.4

La ***classe d'acceptabilité*** *du SAPI <A(∑), R, Pref>* est maintenant définie par :

$C_{R,Pref}$ = {A â A(∑) / åB âA(∑), si B R A alors A $>>^{Pref}$ B}

Propriété 3.1 C_R √ $C_{R,Pref}$

Preuve

Immédiate d'après la définition de $C_{R,Pref}$.

Exemple 3.2

Soit $A(\Sigma)$ = {A, B, C, D} et R = {(C, D), (D, C)}. Si on suppose que $C >>^{Pref} D$, alors la classe d'acceptabilité du système $<A(\Sigma), R, Pref>$ est $C_{R,Pref}$ = {A, B, C} car C est contrarié par D mais il se défend seul contre lui.

3.1.2 Exemples de relations de préférences

Lorsque la base de croyances E est munie d'un pré-ordre total, il est équivalent de considérer E stratifiée en E_1 u...u E_n. Chaque E_i est une classe pour la relation d'équivalence associée au pré-ordre. Il est alors commode d'interpréter cette stratification en termes de niveaux de certitude. Les croyances de E_i ont le même niveau de certitude et sont toutes plus certaines que les croyances de E_j avec j>i. Le niveau d'une sous-base H de E est le numéro le plus élevé d'une strate rencontrée par cette sous-base.

• La préférence BDP ([BDP93])

Cette relation de préférence a été proposée dans le cadre de la logique possibiliste. Le niveau de certitude est utilisé pour définir un pré-ordre total entre les sous-bases de E.

Définition 3.5

Etant donné H et H' deux sous-bases K-consistantes de E,

H ***BDP*** H' ssi niveau(H) ≤ niveau(H') ;

H ***BDP*** H' ssi niveau(H) < niveau(H').

Exemple 3.3

Soit Σ = (K, E) une base de connaissances telle que K = Ø et $E = E_1 \cup E_2 \cup E_3$ avec E_1 = {a, b, a–c}, E_2 = {b–¬d}, E_3 = {c–d}. {b, b–¬d} BDP {a, a–c, c–d} car le niveau de certitude de la première sous-base est 2 et celui de la seconde est 3. Ainsi : ({b, b–¬d}, ¬d) $>>^{BDP}$ ({a, a–c, c–d}, d).

Lorsque la base est munie d'un pré-ordre partiel ≥ dont l'ordre strict associé est noté >, la relation BDP peut être définie comme suit :

Définition 3.6

Etant donné H et H' deux sous-bases K-consistantes de E,

H ***BDP*** H' si å r â H, ™ r' â H' tel que r > r'.

• La préférence ELI (ou préférence Elitiste [CRS93])

L'idée de cette relation de préférence est que chaque formule retenue doit être meilleure qu'une formule rejetée. La base de connaissances est supposée munie d'un pré-ordre partiel ≥, dont l'ordre strict associé est noté >.

Définition 3.7

Etant donné H et H' deux sous-bases K-consistantes de E,

H ***ELI*** H' ssi å r â H \ H', ™ r' â H' \ H tel que r > r'.

Exemple 3.4

Soit Σ une base avec K = Ø et E = {a, b, c, d} telle que {a > d, b > c}. Soit H = {a, b} et H' = {c, d} deux supports d'arguments. On a (H BDP H') et (H ELI H').

Lorsque la base est stratifiée (c'est-à-dire que la base E est munie d'un pré-ordre total), la relation stricte associée à la relation ELI peut-être définie en termes de niveaux comme suit :

Définition 3.8

Etant donné H et H' deux sous-bases K-consistantes de E,

H ***ELI*** H' ssi niveau(H \ H') < niveau(H' \ H) (avec niveau(Ø)=0).

Propriété 3.2

Etant donné H et H' deux sous-bases K-consistantes de E, si H BDP H' alors H ELI H'.

La réciproque est vraie ssi niveau(H U H') < niveau(H' \ H).

Preuve

voir [AC96].

Remarque : Cette propriété montre que ELI peut-être considérée comme un raffinement de BDP. Il est donc souvent plus intéressant d'utiliser ELI pour comparer les arguments plutôt que BDP, puisque ELI, en plus de reconnaître toutes les préférences reconnues par BDP, permet de comparer encore d'autres arguments.

Exemple 3.5

Soit la base Σ = (K, E) avec K = Ø et E = E_1 u E_2 u E_3 u E_4 avec E_1 = {p, x–o, x–t}, E_2 = {o–v}, E_3 = {t–¬v} et E_4 = {p–x}.

*Les arguments **A** : ({p, p–x, x–o, o–v}, v) et **B** : ({p, p–x, x–t, t–¬v}, ¬v) sont incomparables par la préférence BDP (aucun des deux n'est strictement BDP-préféré à l'autre), mais par contre A $>>^{ELI}$ B.*

3.1.3 Classes d'acceptabilité

Exemple 3.6

Soit Σ = (K, E) telle que K = Ø et E = E_1 u E_2 u E_3 avec E_1 = {a, a–b}, E_2 = {¬a} et E_3 = {c}. On construit $\mathbf{A_1}$: ({a}, a), $\mathbf{A_2}$: ({a, a–b}, b), $\mathbf{A_3}$: ({¬a}, ¬a), $\mathbf{A_4}$: ({c}, c). Les conflits entre ces arguments sont les mêmes que dans l'exemple 2.3, mais à présent $A_1 >>^{BDP} A_3$ et $A_2 >>^{BDP} A_3$. Ainsi $\{A_1, A_2, A_4\} \surd C_{Réfute,BDP}$ et $\{A_2, A_4\} \surd C_{Attaque,BDP}$.

Exemple 3.1 (suite)

Soit Σ = (K, E) telle que K = Ø et E = E_1 u E_2 u E_3 avec E_1 = {p, p–o}, E_2 = {p–¬v} et E_3 = {o–v}. On peut construire sur cette base un argument pour v et un autre argument pour ¬v : $\mathbf{A_1}$: ({p, p–o, o–v}, v) et $\mathbf{A_2}$: ({p, p–¬v}, ¬v). A_2 est dans $C_{Réfute,BDP}$ car il est strictement BDP-préféré à A_1 qui le réfute. Nous avons

donc, dans le système <A(∑), Réfute, BDP> un argument acceptable pour ¬v (un pingouin ne vole pas). A_2 est également acceptable dans le système <A(∑), Réfute, ELI>.

3.2. SAP basés sur des priorités Conditionnelles (SAPC)

Les relations de préférences entre arguments utilisées jusqu'à présent sont calculées à partir de priorités entre les croyances données de façon exogène, a priori ; elles sont exprimées via un pré-ordre et ne dépendent de rien. Or les priorités peuvent être dépendantes d'informations ; ces informations doivent être dérivées de la base pour que la priorité soit effective. On dit dans ce cas que les priorités sont *conditionnelles*. Ceci entraîne que les priorités peuvent être *contestables*, puisque l'on peut dériver de la base des priorités contradictoires. Les préférences entre arguments étant calculées à partir des priorités entre croyances, ces préférences ne sont plus incontestables.

Nous allons dans ce rapport donner une définition des Systèmes d'Argumentation avec Préférences basés sur des priorités Conditionnelles (les ***SAPC***), en nous intéressant directement à l'application de tels systèmes au traitement de l'incohérence dans les bases de connaissances.

Illustrons tout d'abord ce que sont les priorités conditionnelles en considérant un premier extrait de la loi en Italie [PS97]. A noter que c'est dans le domaine juridique que nous avons trouvé un grand nombre d'exemples qui utilisent des priorités conditionnelles. Ce domaine est plus particulièrement le cadre d'étude de Prakken et Sartor [PS95&96&97].

Exemple 3.7

Soit les deux principes suivants, P_1 concernant la protection civile et P_2 concernant la liberté d'expression :

(P_1) Il est interdit de propager des informations concernant la vie privée d'une personne.

(P_2) Il est permis de propager n'importe quelle information quand la personne concernée est un personnage public.

Si nous avons une information à propos de la vie privée d'un personnage public, que faut-il conclure à propos de la propagation de cette information ?

On peut construire deux arguments, A_1 qui est supporté par P_1 et qui permet de conclure la non propagation, et A_2 supporté par P_2 concluant sur la propagation. Pour résoudre le conflit entre A_1 et A_2, le principe suivant a été ajouté :

(P_3) Si la personne en question est un homme politique et que l'information affecte sa fonction alors le principe de la liberté d'expression sera privilégié par rapport au principe de la protection civile.

Ce principe exprime une préférence entre les principes P_1 et P_2, mais cette préférence est sous condition, elle dépend d'informations qui doivent être déduites de la base. C'est seulement si l'on apprend que le personnage en question est un politicien et que l'information peut affecter sa fonction, que la préférence entre P_1 et P_2 peut

s'établir. L'argument A_1 est alors préféré à l'argument A_2 et l'on peut conclure sur la non propagation de l'information.

L'exemple suivant (un deuxième extrait de la loi en Italie [PS97]) montre que l'utilisation de priorités conditionnelles induit que les préférences entre arguments peuvent devenir contestables.

Exemple 3.8

Soit les deux principes suivants :

(P_1) Si un immeuble est classé alors sa façade ne doit pas être modifiée.

(P_2) Si un immeuble a besoin d'être rénové alors sa façade doit être modifiée.

Dans le cas d'une villa classée ayant besoin de rénovation, que conclure quant à la modification de sa façade ? On peut construire deux arguments A_1 et A_2, A_1 étant de support P_1 et concluant sur la modification, et A_2 de support P_2 concluant sur la non modification. Pour résoudre le conflit que l'on voit apparaître entre A_1 et A_2, un principe P_3 a été instauré :

(P_3) Les règles concernant les immeubles classés sont plus prioritaires que les règles sur les autres types d'immeubles.

Dans ce cas, le principe P_1 est prioritaire sur le principe P_2 et l'argument A_1 est préféré à l'argument A_2. Mais il existe un autre principe de priorité :

(P_4) Les règles les plus récentes sont préférées.

Il se trouve que la règle P_2 est plus récente que la règle P_1. La priorité de P_1 sur P_2 est donc contestée ainsi que la préférence de l'argument A_1 sur l'argument A_2.

Le conflit entre A_1 et A_2 n'est pas levé, on ne peut rien conclure quant à la modification de la façade de la villa.

On voit au travers de ces exemples que la représentation actuelle des priorités entre connaissances est insuffisante pour exprimer des priorités conditionnelles, ceci pour plusieurs raisons :

1. Les priorités sont exprimées via un pré-ordre qui ne dépend de rien.

2. Ce pré-ordre est donné a priori, il est non dérivable. On ne raisonne pas sur les priorités, les priorités sont figées.

3. Le pré-ordre est donné de façon exogène, externe. Les priorités ne sont pas représentées comme des connaissances.

La prise en compte de priorités conditionnelles entre les connaissances doit d'abord passer par l'expression des priorités dans le langage. Ceci nous amènera à modéliser un nouveau type de conflit entre arguments et à définir de nouvelles relations de préférence.

3.2.1 Expression des priorités dans le langage

Pour pouvoir travailler avec des priorités conditionnelles, la première chose à faire est d'intégrer l'expression des priorités dans le langage. D'autres chercheurs se sont intéressés à ce problème, comme par exemple Brewka [Bre94] qui a montré comment raisonner sur des priorités entre défauts dans la logique des défauts. Son travail a ensuite été repris par [DS97]. Prakken et Sartor [PS97] ont quant à eux étudié la représentation et la prise en compte de priorités dans un langage de programmation logique. Dans ce rapport, nous proposons une solution pour l'intégration des priorités, en particulier des priorités conditionnelles, dans un *langage du premier ordre*.

Nous nous inspirons donc des idées de [Bre94], proposées dans le cadre de la logique des défauts. Brewka montre tout d'abord la nécessité de pouvoir faire référence aux défauts de façon explicite. Pour cela, il choisit de nommer les défauts à l'aide de termes constants, plutôt qu'à l'aide de propositions atomiques. Cette référence aux défauts étant faite, il définit un symbole de prédicat spécial pour représenter les priorités entre défauts.
De la même manière, nous choisissons de travailler avec des formules nommées. Nous redéfinissons pour cela le langage, et nous introduisons un symbole de prédicat pour représenter les priorités entre formules.

Le nouveau langage L est un langage du premier ordre. Nous n'utilisons pas tout le pouvoir du premier ordre ; nous ne manipulerons pas, par exemple, de formules construites à l'aide de quantificateurs. Les bases de connaissances sont des ensembles de formules construites sur le nouveau langage L ; elles restent de la forme $\sum = (K, E)$.

Syntaxe du langage

Le langage possède un ***alphabet*** comprenant :

un ensemble dénombrable de constantes	$V_C = \{a, b, c\} \cup \{r_0, r_1, r_2, ...\}$
un ensemble dénombrable de variables	$V_V = \{x, y, z\}$
un ensemble dénombrable de symboles de fonction	$V_F = \{o, p, q, t, v, ...\} \cup \{f_0, f_1, f_2, ...\}$
un ensemble dénombrable de symboles de prédicat	$V_P = \{g, h, ...\} \cup \{>\}$
un ensemble de connecteurs	$\{\neg, \hat{E}, ª, –\}$
des délimiteurs	$\{(,)\}$

Pour nommer les formules et exprimer les priorités dans le langage, on réserve :

un sous-ensemble de constantes $C \subseteq V_C$	$C = \{r_0, r_1, r_2, ...\}$
un sous-ensemble de symboles de fonctions $F \subseteq V_F$	$F = \{f_0, f_1, f_2, ...\}$
un symbole de prédicat	$>$

Le ***langage des termes*** $\mathbf{L_T}$ suit les règles de construction suivantes :

1) Tout symbole de constante est un terme ;

2) Tout symbole de variable est un terme ;

3) Si f est un symbole de fonction d'arité n et si t_1, ..., t_n sont des termes alors $f(t_1, ..., t_n)$ est un terme ;

4) Tout terme s'obtient en appliquant un nombre fini de fois les règles ci-dessus.

Un sous-ensemble de l'ensemble des termes est appelé ***ensemble des noms*** et est dénoté par $\mathbf{L_N}$. Les *noms* sont des termes particuliers qui vont servir à désigner les formules. Les termes de L_N suivent les règles de construction ci-dessous :

1) Tout symbole de constante appartenant à C est un nom ;

2) Si f est un symbole de fonction d'arité n appartenant à F et si t_1, ..., t_n sont des noms ou des symboles de variable alors $f(t_1, ..., t_n)$ est un nom ;

3) Tout nom s'obtient en appliquant un nombre fini de fois les règles ci-dessus.

L'expression des priorités se fait grâce au prédicat binaire > ; un prédicat ne pouvant avoir de formule en argument, l'expression d'une priorité entre deux formules se fait via le nom qui les désigne. Le prédicat > est donc un prédicat typé, puisqu'il ne prend en argument que des termes de L_N. Remarquons qu'étant donné deux noms t_i et t_j avec $i \neq j$, on utilisera le plus souvent la notation infixée $(t_i > t_j)$ plutôt que la notation préfixée $>(t_i, t_j)$.
å$i \neq j$, $(t_i > t_j)$ signifie que la formule désignée par le nom t_i est prioritaire sur la formule désignée par le nom t_j.

Remarque : Brewka [Bre94] utilise un symbole de prédicat "inverse" (i.e. <) pour représenter les priorités ; il exprime que la formule t_i est prioritaire sur la formule t_j en écrivant $(t_i < t_j)$. Ce symbolisme nous apparaît contre-intuitif.

Afin de nous assurer que > décrit toujours un ordre partiel strict, nous rajoutons les deux axiomes suivants dans K (axiome T pour la transitivité, axiome I pour l'irréflexivité) :

(**T**) å x, y, z â L_N $(((x > y) \hat{E} (y > z)) – (x > z))$

(**I**) å x â L_N $\neg(x > x)$

On remarque que l'asymétrie (å x, y â L_N si $(x > y)$ alors $\neg(y > x)$) est conséquence de la transitivité et de l'irréflexivité.

Dans toute la suite, le noyau K de toute base de connaissances contiendra l'ensemble **AXI** constitué des deux axiomes T et I.

L'***ensemble des formules*** $\mathbf{L_F}$ est constitué de trois ensembles : L_{FN}, l'ensemble des formules "normales", L_{FP}, l'ensemble des formules construites uniquement sur le prédicat > (on dira que ce sont des formules "de priorité"), et L_{FM} qui combine formules normales et formules

exprimant des priorités (on parlera de formules "mixtes"). Montrons comment sont construits ces ensembles :

• **L_{FN}** est un sous-ensemble des formules classiquement construites sur un langage du premier ordre. Il suit les règles de construction suivantes :

1) Si p â $V_P \setminus \{>\}$, si p est d'arité n et si $t_1, ..., t_n$ sont des termes alors $p(t_1, ..., t_n)$ est une formule normale dite *formule normale atomique* ;
2) Si <f> est une formule normale, alors (¬ <f>) est une formule normale ;
3) Si <f> et <g> sont des formules normales, alors (<f>Ê<g>), (<f>^{a}<g>) et (<f>–<g>) sont des formules normales ;
4) Toute formule normale s'obtient en appliquant un nombre fini de fois les règles ci-dessus.

• **L_{FP}** contient les formules "de priorité", autrement dit les formules construites uniquement à l'aide du prédicat typé > ; L_{FP} suit les règles de construction suivantes :

1) Si t_i et t_j sont des noms alors ($t_i > t_j$) est une formule de priorité, dite *formule de priorité atomique* ;
2) Si <f>, <g> et <h> sont des formules de priorité *atomiques* alors ((<f>Ê<g>)–<h>) et (¬ <f>) sont des formules de priorité ;
3) Toute formule de priorité s'obtient en appliquant un nombre fini de fois les règles ci-dessus.

• **L_{FM}** enfin, contient les formules "mixtes", c'est-à-dire celles qui combinent formules normales et formules de priorité. La construction de ces formules suit la règle suivante :

Si <f> est une formule normale et <g> est une formule de priorité atomique alors (<f>–<g>) est une formule mixte.

A l'intérieur du langage, il apparaît donc deux types de priorités entre formules. Tout d'abord des priorités "classiques", qui ne dépendent de rien (du type ($t_i > t_j$)). Ces priorités étaient précédemment données de façon exogène, aujourd'hui elles s'expriment dans la théorie. Le deuxième type de priorité, qui jusqu'à présent ne pouvait être exprimé, sont les priorités conditionnelles. Ces priorités dépendent d'informations qui devront être dérivées de la base de connaissances. Elles s'expriment sous la forme d'implications dont le conséquent est une priorité entre formules. A noter que selon notre construction, les formules du langage contiennent au plus un littéral positif construit sur le prédicat > ; toutes les clauses contenant le prédicat > sont donc des clauses de Horn en >. Cette restriction simplifie le travail pour la suite (notamment au niveau de la théorie de la preuve) mais nous envisagerons de la lever dans le chapitre 5 (Perspectives).

Dénomination des formules

Nous inspirant du formalisme de [DS97] qui est défini dans le cadre de la logique des défauts, nous définissons une bijection N : $L_F - L_N$ qui permet de faire correspondre à toute formule qui est une *croyance* (i.e. qui appartient à la sous-base E d'une base de connaissances) un nom. Il n'est pas nécessaire de nommer les formules présentes dans K puisque ces formules sont toutes sûres et aucune ne peut être préférée à une autre.
A remarquer que pour une formule <f> ayant un nom n, on notera plus souvent **n** : <f>, afin de rendre le nom plus explicite.

Pour souligner le fait que l'on travaille avec un ensemble fini de formules distinctes, nous adoptons une hypothèse de fermeture des domaines (DCA) et une hypothèse d'unicité des noms (UNA) vis-à-vis de L_N.

Sémantique du langage

Une interprétation **I** est un couple <D, I> avec :

• **D** = <D_N, D_T> le domaine d'interprétation (ensemble non vide) dans lequel vont être interprétés les termes du langage. Les noms sont des termes particuliers, aussi ils seront interprétés dans la composante $\mathbf{D_N}$ de D ; tous les autres termes seront interprétés dans la composante $\mathbf{D_T}$.

• **I** = <I_C, I_f, I_p, I_V> une collection d'applications telles que :

- $\mathbf{I_C}$ associe à tout élément de $V_C \setminus C$ un élément de D_T et associe à tout élément de C un élément de D_N.

- $\mathbf{I_f}$ associe à tout symbole fonctionnel f de $V_F \setminus F$ d'arité m une application $I_f(f)$ de D^m dans D_T et associe à tout symbole f de F d'arité m une fonction $I_f(f)$ de D^m dans D_N.

- $\mathbf{I_p}$ associe à tout symbole de prédicat p de $V_P \setminus \{>\}$ d'arité m une application $I_p(p)$ de D^m dans {Vrai, Faux} et associe au symbole > une application $I_p(>)$ de $D_N{}^2$ dans {Vrai, Faux}.
Pour t1, t2 â D_N, $I_p(>)$ (t1, t2) = Vrai signifie que la formule référencée par le terme t1 est plus prioritaire que la formule référencée par le terme t2.

- $\mathbf{I_V}$ associe à toute variable de V_V un élément de D.

Les règles d'interprétation usuelles permettent d'associer à toute formule <f> une valeur de vérité I(<f>) et à tout terme <t> un élément I(<t>) de D.

Exemples

Traduisons maintenant l'exemple 3.7 dans le formalisme que nous venons de définir, en supposant que l'information que l'on possède concerne un politicien et qu'elle peut affecter sa fonction.

Exemple 3.7 (suite)

La base ∑ = (K, E) est telle que :

K = AXI u {p, pub, pol, aff},

*E = {**r1** : p–¬pg, **r2** : pub–pg, **r3** : (polÊaff)–(r2>r1)}*

avec p, pub, pol, aff, pg qui sont des symboles de prédicat d'arité 0. Les interprétations de ces symboles sont les suivantes : p est interprété par "privée", pub par "public", pol par "politicien", aff par "affecter fonction" et pg par "propager". Les formules désignées par les termes r1, r2 et r3 traduisent les principes P_1, P_2 et P_3.

Traduisons de même l'exemple 3.8 dans notre formalisme :

Exemple 3.8 (suite)

La base ∑ = (K, E) est telle que :

K = AXI u {classée(villa), à-rénover(villa), règle-classée(f1(x)), règle-autre(f2(x)), plus-récent(f2(x), f1(x))}

*E = {**f1(x)** : classée(x)–¬modifier(x), **f2(x)** : à-rénover(x)–modifier(x),*

*   **f3(x, y)** : (règle-classée(x) Ê règle-autre(y))–(x > y), **f4(x, y)** : plus-récent(x, y)–(x > y)}*

où classé, rénover, modifier, plus-récent sont des symboles de prédicat, tout comme règle-classé, règle-autre et plus-récent, qui eux, remarquons-le, ont pour arguments des noms de formules (> n'est pas le seul prédicat s'appliquant sur des noms). Les formules désignées par les termes f1(x), f2(x), f3(x, y) et f4(x, y) traduisent les principes P_1, P_2, P_3, et P_4.

3.2.2 Redéfinition de l'argument et du conflit par Attaque

La dénomination des croyances entraîne une redéfinition de l'argument. Le support contient maintenant des noms de croyances que l'on note non plus entre accolades mais entre crochets. La conclusion demeure une formule du langage.
Afin d'établir une correspondance entre un ensemble de noms et l'ensemble des formules désignées par les noms de cet ensemble, on définit la fonction suivante :

Définition 3.9

Soit V la fonction définie par :

$V : 2^{L_N} – 2^{L_F}$

$H - \{<f> \hat{a} L_F \mid N(<f>) \hat{a} H\}$

Exemple 3.7 (suite)

V({r2, r3}) = {pub–pg, (polÊaff)–(r2>r1)}

Cette fonction nous permet de redéfinir plus facilement l'argument :

Définition 3.10

Un ***argument*** de E dans le contexte K est un couple (H, h) où h est une formule ($h \hat{a} L_F$) et H est un sous-ensemble fini de noms ($H \surd L_N$) tel que :

i) K u V(H) consistant

ii) $K \cup V(H) \vdash h$

iii) V(H) minimal pour l'inclusion pour i) et ii)

On dira qu'***un argument conclut sur une priorité*** si h est une formule de priorité atomique.

On continuera à désigner les arguments par une lettre majuscule éventuellement indicée, afin de les manipuler plus facilement.

Exemple 3.7(suite)

Les arguments pour la propagation et la non propagation de l'information sont ***A1*** *: ([r1], ¬pg) et* ***A2*** *: ([r2], pg). L'argument* ***A3*** *: ([r3], (r2 > r1)) conclut sur une priorité qui exprime que la formule de nom r2 est prioritaire sur la formule de nom r1.*

Exemple 3.8 (suite)

On construit sur cette base les arguments ***A1*** *: ([f1(villa)], ¬modifier(villa)) et* ***A2*** *: ([f2(villa)], modifier(villa)). L'argument* ***A3*** *: ([f3(f1(villa), f2(villa))], (f1(villa) > f2(villa))) conclut sur le fait que le principe P_1 (traduit par la formule nommée f1(x)) est prioritaire sur le principe P_2 (traduit par la formule nommée f2(x)) lorsque ces deux principes sont appliqués au cas de notre villa.*
L'argument ***A4*** *: ([f4(f2(villa), f1(villa))], (f2(villa) > f1(villa))) conclut sur la priorité contraire.*

La modification de la définition de l'argument entraîne une redéfinition du conflit par attaque :

Définition 3.11

Soit (H_1, h_1) et (H_2, h_2) deux arguments.

On dit que (H_1, h_1) ***attaque*** (H_2, h_2) sur n ssi ™ n â H_2 tel que $h_1 @ \neg N^{-1}(n)$.

Exemple 3.9

*Soit ∑ = (K, E) avec K = AXI et E = {****r3*** *: (aÊb)–(r1 > r2),* ***r4*** *: a,* ***r5*** *: b,* ***r6*** *: a–¬b}.*
L'argument ***A1*** *: ([r4, r6],¬b) attaque l'argument* ***A2*** *: ([r3, r4, r5], (r1 > r2)) sur r5.*

3.2.3 Conséquence : les priorités peuvent être contestées

Les priorités étant intégrées dans le langage, elles peuvent être dérivées de la base de connaissances. Il est alors tout à fait possible que d'une base, on déduise des priorités qui se contestent mutuellement (dans l'exemple 3.8 on déduit les priorités (f1(villa) > f2(villa)) et (f2(villa) > f1(villa))). Une priorité est également contestée si l'argument qui la supporte est attaqué, ce qui est le cas dans l'exemple 3.9 pour la priorité (r1 > r2).

On voit bien que les priorités entre croyances ne sont plus figées et incontestables. Ceci va poser des problèmes lorsque l'on va vouloir résoudre un conflit entre deux arguments à l'aide d'une relation de priorité, puisqu'il faudra s'assurer que la priorité est :

- incontestable (cas idéal), ou bien
- justifiée par un argument jugé acceptable.

Ceci entraîne que l'on ne peut plus définir l'ensemble des arguments acceptables par l'ensemble des arguments préférés à leurs contre-arguments, car la préférence entre arguments peut reposer sur une relation de priorité qui n'est supportée par aucun argument acceptable !

3.2.4 Autre type de conflit : la disqualification

Deux arguments concluant sur une priorité ne peuvent se réfuter puisque les priorités conclusions sont toujours des littéraux positifs. Dans l'exemple 3.8 les arguments A3 et A4 sont intuitivement incompatibles car les priorités qu'ils concluent se contestent, mais on ne peut pas dire qu'ils se réfutent.

Il nous faut une relation qui reconnaisse ce type de conflit. Nous définissons pour cela la relation de disqualification comme suit :

Définition 3.12
Soit $A_1 : (H_1, h_1)$ et $A_2 : (H_2, h_2)$ deux arguments.
A_1 ***disqualifie*** A_2 ssi $V(H_2)$ u $K \vdash \neg h_1$.

Cas particulier :
Si h_1 ne contient pas le prédicat >, A_1 disqualifie A_2 ssi A_1 réfute un sous-argument de A_2. Ce cas avait déjà été suggéré dans [Cay95] sous le nom d'attaque faible.

Exemple 3.8 (suite)

K ⊢ ((f1(villa) > f2(villa))@¬(f2(villa) > f1(villa))), donc
V({f3(f1(villa), f2(villa))}) u K ⊢ ¬(f2(villa) > f1(villa)) et
V({f4(f2(villa), f1(villa))}) u K ⊢ ¬(f1(villa) > f2(villa)).

A3 et A4 se disqualifient mutuellement.

Propriété 3.3
(i) å A_1, A_2 â $A(\Sigma)$, A_1 réfute A_2 = A_1 disqualifie A_2
(ii) å A_1, A_2 â $A(\Sigma)$, A_1 attaque A_2 = A_1 disqualifie A_2

Preuve

(i) $A_1 : (H1, h1)$ réfute $A_2 : (H2, h2)$
= h2 @ ¬h1
= $V(H2)$ u $K \vdash \neg h_1$.

(ii) $A_1 : (H1, h1)$ attaque $A_2 : (H2, h2)$
= ™ r â H2 / h1@ $\neg N^{-1}(\{r\})$
= $V(H2)$ u $K \vdash \neg h_1$.

D'après cette propriété, la relation de disqualification est donc une généralisation des relations de réfutation et d'attaque.

Exemple 3.10

*Soit la base ∑ = (K, E) telle que K = AXI et E = { **r1** : a, **r2** : a–¬b, **r3** : ¬a, **r4** : c, **r5** : c–d, **r6** : d–¬a }.*

*L'argument **A1** : ([r1, r6], ¬d) réfute l'argument **A2** : ([r4, r5], d). L'argument **A3** : ([r4, r5, r6], ¬a) réfute l'argument **A4** : ([r1], a) et attaque l'argument A1 sur r1. Soit S = {A1, A2, A3, A4}.*

Le graphe des conflits du système <S, Réfute> est :

A1 A3
↓↑ ↓↑
A2 A4

et celui du système <S, Attaque> est :

A1 ← A3

De plus, A1 réfute A2 et A2 est un sous-argument de A3. On sent que A1 contrarie A3 mais ce type de contrariété n'est reconnu ni par la relation Réfute, ni par la relation Attaque. Il nous faut faire appel à la relation de disqualification.

Le graphe des conflits du système <S, Disqualifie> est :

A1 ⇄ A3
↑↓ ↑↓
A2 A4

Cet exemple montre que la relation de disqualification généralise la réfutation et l'attaque, et qu'elle permet de reconnaître un plus grand nombre de contrariétés entre arguments.

Définition 3.13

La ***classe d'acceptabilité*** $C_{Disqualifie}$ rassemble tous les arguments non disqualifiés :

$C_{Disqualifie}$ = { A â A(∑) / d B â A(∑) tel que B disqualifie A }

Propriété 3.4

Soit A : (H, h) â A(∑) avec h ne contenant pas ⊥.

A â $C_{Disqualifie}$ ssi aucun sous-argument de A n'est réfuté

ssi tout sous-argument de A appartient à $C_{Réfute}$.

Preuve

Immédiate d'après la définition de la relation Disqualifie.

Etudions la relation stricte associée à la relation Disqualifie :

Définition 3.14

Soit ∑ une base de connaissances et A, B â A(∑).

A ***disqualifie strictement*** B ssi A disqualifie B et B ne disqualifie pas A.

Exemple 3.8 (suite)

Puisque A3 et A4 se disqualifient mutuellement, il n'y a pas de disqualification stricte entre eux.

Propriété 3.5

å A, B â A(∑), A réfute B = A ne disqualifie pas strictement B.

Preuve

Immédiate du fait de la symétrie de la relation Réfute et de la propriété 4.1 (i).

La réciproque de cette propriété est fausse.

3.2.5 Relation de préférence entre arguments

Les relations de préférences entre arguments qui sont présentées dans le cadre des systèmes d'argumentation avec priorités fixes (paragraphe 3.1), sont construites à partir d'une relation Pref sur les supports. Cette relation entre les supports est elle-même induite par une relation de priorité entre les croyances donnée de façon exogène ; on munit par exemple la base de connaissances d'un pré-ordre (total ou partiel) ≥.

Dans les SAP basés sur des priorités Conditionnelles, les priorités entre croyances ne sont plus données a priori ; elles peuvent apparaître comme conclusions d'arguments. La préférence entre supports va ainsi se définir par rapport à un ensemble S d'arguments, plus précisément par rapport à l'ensemble des priorités entre croyances issues des conclusions d'arguments de S. La relation de préférence entre arguments est construite à partir de cette préférence sur les supports.

Définition de la relation de préférence

Définition 3.15

Soit S un ensemble d'arguments.

- **AP_S** désigne l'ensemble des arguments de S concluant sur une priorité :
  AP_S = {(H, h) â S / h est une formule de priorité atomique}
- **$>_S$** désigne l'ensemble des conclusions d'arguments de AP_S :
  $>_S$ = {h / (H, h) â AP_S}, ou de manière équivalente,
  $>_S$ = {r > r' / r > r' est conclusion d'un argument A â S}.

Exemple 3.7 (suite)

*Sur la base de l'exemple 3.7, on a construit : **A1** : ([r1], pg), **A2** : ([r2], pg) et **A3** : ([r3], (r2 > r1)).*
Pour S = {A3}, on a $>_S$ = {r2 > r1}.

Exemple 3.8 (suite)

Sur la base de l'exemple 3.8, on a construit les arguments

***A1** : ([f1(villa)], ¬modifier(villa)), **A2** : ([f2(villa)], modifier(villa)),*

***A3** : ([f3(f1(villa), f2(villa))], (f1(villa) > f2(villa))), **A4** : ([f4(f2(villa), f1(villa))], (f2(villa) > f1(villa))).*

Pour S1 = {A1, A2, A3}, AP_{S1} = {A3} et $>_{S1}$ = {(f1(villa) > f2(villa))}.

Pour S2 = {A1, A2, A3, A4}, AP_{S2} = {A3, A4} et $>_{S2}$ = {(f1(villa) > f2(villa)), (f2(villa) > f1(villa))}.

Propriété 3.6

Soit S1 et S2 deux ensembles d'arguments tels que S1, S2 √ A(Σ).

(i) S1 √ S2 = AP_{S1} √ AP_{S2}

(ii) S1 √ S2 = $>_{S1}$ √ $>_{S2}$

Preuve

(i) Immédiate d'après les définitions. (ii) Immédiate d'après (i).

Définissons maintenant la relation de préférence entre supports induite par S, notée $Pref_S$:

Définition 3.16

$Pref_S$ est une relation binaire sur les supports d'arguments définie à partir de $>_S$.

Pour deux arguments A_1 : (H_1, h_1) et A_2 : (H_2, h_2), on dit que :

A_1 est ***$Pref_S$-préféré*** à A_2 (noté $A_1 >^{\mathbf{Pref_S}} A_2$) ssi H_1 $Pref_S$ H_2.

Notation : La relation stricte associée à $Pref_S$ est dénotée par $\underline{Pref_S}$, celle associée à $>^{Pref_S}$ est dénotée $>>^{Pref_S}$. Si A_1 est strictement $Pref_S$-préféré à A_2, on écrit $A_1 >>^{Pref_S} A_2$.

Propriété 3.7

Soit S1 et S2 deux ensembles d'arguments tels que S1, S2 √ A(Σ).

S1 √ S2 = $Pref_{S1}$ √ $Pref_{S2}$.

Preuve

Immédiate par la définition de $Pref_S$ et la monotonie de $>_S$ (propriété 3.6 (ii)).

Exemples de relations de préférence

Dans les SAPC, le prédicat distingué permettant d'exprimer des priorités entre formules représente un ordre partiel strict (grâce aux axiomes (T) et (I) présents dans K). Nous présentons ci-dessous deux exemples de construction $Pref_S$ en utilisant respectivement la définition 3.6 (BDP-préférence) et la définition 3.7 (ELI-préférence).

Définition 3.17

Soit S un ensemble d'arguments inclus dans A(Σ) et H, H' deux sous-ensembles finis de L_N tels que V(H), V(H') minimaux K-consistants.

H $\mathbf{BDP_S}$ H' ssi å r â H, ™ r' â H' tel que (r > r') â $>_S$

H $\mathbf{ELI_S}$ H' ssi å r â H \ H', ™ r' â H' \ H tel que (r > r') â $>_S$

Exemple 3.7 (suite)

Pour S = {A3}, [r2] $\underline{BDP_S}$ [r1] et [r2] $\underline{ELI_S}$ [r1]. Donc A2 $>>^{BDP_S}$ A1 et A2 $>>^{ELI_S}$ A1.

Exemple 3.8 (suite)

Pour S1 = {A1, A2, A3}, A1 $>>^{BDP_{S1}}$ A2 (et A1 $>>^{ELI_{S1}}$ A2) ; la préférence de A1 sur A2 est stricte, alors que pour S2 = {A1, A2, A3, A4}, A1 $>^{BDP_{S2}}$ A2 et A2 $>^{BDP_{S2}}$ A1 (et A1 $>^{ELI_{S2}}$ A2 et A2 $>^{ELI_{S2}}$ A1). A1 et A2 sont, dans ce deuxième cas, autant préféré l'un que l'autre.

De même que ELI peut être considérée comme un raffinement de BDP (propriété 3.2), ELI_S peut être considérée comme un raffinement de BDP_S.

Exemple 3.11

*• Soit ∑ = (K, E) avec K = AXI et E = { **r1** : a, **r2** : a–b, **r3** : b–d, **r4** : e, **r5** : e–¬d, **r6** : r5 > r3 }.*

*Soit les arguments **A1** : ([r1, r2, r3], d), **A2** : ([r4, r5], ¬d), **A3** : ([r6], r5 > r3).*

Avec S = {A3}, aucun des deux arguments A1, A2 n'est BDP_S-préféré (ni ELI_S-préféré) à son contrariant.

*• Soit ∑' = (K, E') avec K = AXI et E' = { **r1** : a, **r2** : a–b, **r3** : b–d, **r5'** : a–¬d, **r6'** : r5' > r3 }.*

*Soit les arguments **A1'** : ([r1, r2, r3], d), **A2'** : ([r1, r5'], ¬d) et **A3'** : ([r6'], r5' > r3).*

Avec S = {A3'}, A1' et A2' sont incomparables par la préférence BDP_S mais avec la préférence ELI_S, A2' >>^{ELI_S} A1'.

Propriété 3.8

Soit S √ A(∑), A, A' et B trois arguments de A(∑) tels que A' est un sous-argument de A.

(i) A >>^{BDP_S} B = A' >>^{BDP_S} B ;

(ii) A >>^{ELI_S} B = A' >>^{ELI_S} B.

Preuve

(i) Immédiate.

(ii) Soit A : (H_A, h_A) et B : (H_B, h_B) â A(∑).

H' √ H_A = H_B \ H_A √ H_B \ H' et H' \ H_B √ H_A \ H_B

A >>^{ELI_S} B et H' √ H_A = å r' â H' \ H_B, ™ r_Bâ H_B \ H' tel que (r' > r_B) â $>_S$

Propriété 3.9

Soit S √ A(∑).

Si $>_S$ sans cycle alors å A : (H_A, h_A) et B : (H_B, h_B) â A(∑) tels que H_B ∫ H_A,

A ELI_S-préféré à B = non(B ELI_S-préféré à A).

Preuve

Supposons que B ELI_S-préféré à A

= å r_B â H_B \ H_A, ™ r_Aâ H_A \ H_B tel que (r_B > r_A) â $>_S$

On a supposé que H_B ∫ H_A donc H_B \ H_A ≠ Ø

= ™ r_B â H_B \ H_A, ™ r_Aâ H_A \ H_B tel que (r_B > r_A) â $>_S$

et comme A ELI_S-préféré à B, ™ r'$_B$ â H_B \ H_A tel que (r_A > r'$_B$) â $>_S$.

D'où å r_B â H_B \ H_A, ™ r_Aâ H_A \ H_B, ™ r'$_B$ â H_B \ H_A tels que

(r_B > r_A) â $>_S$ et (r_A > r'$_B$) â $>_S$.

- si r'$_B$ = r_B alors (r_B > r_A) â $>_S$ et (r_A > r'$_B$) â $>_S$ = contradiction avec $>_S$ sans cycle.

- sinon on recommence :

™ r'$_A$â H_A \ H_B, ™ r''$_B$ â H_B \ H_A tels que (r'$_B$ > r'$_A$) â $>_S$ et (r'$_A$ > r''$_B$) â $>_S$

Comme H_B \ H_A est fini, on en déduit l'existence d'un cycle dans $>_S$. Contradiction.

Exemple 3.8 (suite)

Nous avons vu que pour S1 = {A1, A2, A3}, on a $>_{S1}$ *= {(f1(villa) > f2(villa))}, ce qui induit une* ELI_{S1}*-préférence stricte de A1 sur A2.*

Mais pour S2 = {A1, A2, A3, A4}, on a $>_{S2}$ *= {(f1(villa) > f2(villa)), (f2(villa) > f1(villa))}, et la préférence de A1 sur A2 n'est plus stricte ; on remarque que* $>_{S2}$ *contient un cycle (de longueur 2).*

Exemple 3.12

Soit Σ *= (K, E) avec E = {* ***r1*** *: a,* ***r2*** *: a–b,* ***r3*** *: c,* ***r4*** *: c–¬b,* ***r5*** *: d,* ***r6*** *: d–(r1 > r4),* ***r7*** *: r2 > r3,* ***r8*** *: r4 > r2,* ***r9*** *: r3 > r1 } et les arguments suivants :* $\boldsymbol{A_1}$ *: ([r1, r2], b),* $\boldsymbol{A_2}$ *: ([r3, r4], ¬b),* $\boldsymbol{A_3}$ *: ([r5, r6], r1 > r4),* $\boldsymbol{A_4}$ *: ([r7], r2 > r3),* $\boldsymbol{A_5}$ *: ([r8], r4 > r2) et* $\boldsymbol{A_6}$ *: ([r9], r3 > r1).*

Pour S1 = {A_3*,* A_4*}, [r1, r2]* $\underline{ELI_{S1}}$ *[r3, r4]. Donc* $A_1 >>^{ELI_S} A_2$.

Pour S2 = {A3, A4, A5, A6}, on a [r1, r2] ELI_{S2} *[r3, r4] et [r3, r4]* ELI_{S2} *[r1, r2]. On n'a donc plus de préférence stricte entre A1 et A2, et l'on peut remarquer que* $>_{S2}$ *contient un cycle (de longueur 4) :*

r1 > r4, r4 > r2, r2 > r3, r3 > r1.

3.3 Comparaison SAPI / SAPC

Résumons sous forme de tableau les différences entre l'expression des préférences dans les SAPI (Systèmes d'Argumentation avec Préférences basés sur des priorités Incontestables) et dans les SAPC (Systèmes d'Argumentation avec Préférences basés sur des priorités Conditionnelles).

	SAPI	**SAPC**
Priorités entre les croyances		
expression : via : sont : statut :	exogène ≥ pré-ordre partiel sur E données a priori incontestable	dans le langage > prédicat du langage déduites de la base contestable
Relation entre les supports		
construite à partir de : dénotée : relation stricte associée:	≥ Pref $\underline{Pref}$	$>_S$: ensemble des priorités supportées par des arguments de S, avec S √ A(Σ) $Pref_S$ avec S √ A(Σ) $\underline{Pref_S}$
Préférence entre arguments		
dénotée : relation stricte associée :	$>^{Pref}$ (Pref-préférence) $>>^{Pref}$	$>^{Pref_S}$ ($Pref_S$-préférence) avec S√A(Σ) $>>^{Pref_S}$ avec S √ A(Σ)

4. Acceptabilité des arguments dans les SAPC

Pour déterminer l'acceptabilité des arguments dans les SAP basés sur des priorités Conditionnelles, nous avons rencontrés plusieurs problèmes, dûs au fait que les priorités sont maintenant dérivables et qu'elles peuvent être contestés. Le problème de circularité est un exemple de difficulté rencontrée. Pour mieux le caractériser, définissons tout d'abord une notion de défense dans les SAPC :

Définition 4.1

Soit <A(∑), R, Pref> un SAPC, soit S √ A(∑) et A, B â A(∑) tels que A R B.

A ***se défend*** contre B grâce à S ssi A $>>^{Pref}_{S}$ B.

Exemple 3.7 (suite)

*L'argument **A2** : ([r2], pg) se défend contre **A1** : ([r1], ¬pg) grâce à S = {**A3**: ([r3], (r2 > r1))} dans le SAPC <{A1, A2, A3}, Disqualifie, ELI>.*

Cette définition généralise la notion de défense introduite dans les SAPI.

Nous allons nous intéresser à trois ensembles d'arguments (partition suggérée dans [PS97]) :

- l'ensemble des *arguments acceptables* ;
- l'ensemble des *arguments éliminés* : un argument sera éliminé s'il est contrarié par un argument acceptable ;
- l'ensemble des *arguments en suspens* : les arguments qui ne sont ni acceptables ni éliminés.

On voit que de la définition de l'ensemble des arguments acceptables découle celle des arguments éliminés et en suspens. Nous allons donc nous concentrer sur la définition de l'ensemble des arguments acceptables, et ce, de deux manières :

- tout d'abord en donnant sa *sémantique* : on définira cet ensemble en donnant des propriétés qui le caractérisent ;
- puis avec une *théorie de la preuve* : on donnera une procédure qui permettra de déterminer si un argument donné appartient à cet ensemble.

N.B. Nous allons dans toute la suite nous placer dans le SAPC particulier <A(∑), Disqualifie, ELI>. Nous choisissons pour ce SAPC la relation de contrariété Disqualifie, puisque cette relation généralise les relations Réfute et Attaque (propriété 3.3) et qu'elle est la seule à reconnaître les conflits entre arguments concluants des priorités. Nous choisissons la préférence ELI puisqu'elle est un raffinement de la relation BDP. Le système <A(∑), Disqualifie, ELI> est d'une certaine manière, une généralisation des autres systèmes que l'on construit à l'aide des relations de contrariété {Réfute, Attaque, Disqualifie} et des relations {ELI, BDP}.

Notation : Le système d'argumentation étant fixé, nous ne parlerons plus pour alléger le vocabulaire, de Disqualifie-contrariants ou d'ensembles sans-Disqualifie-conflit, mais simplement de contrariants et d'ensembles sans-conflit. De même, si un argument A est ELI_S-préféré à un argument B, on dira que A est S-préféré à B ; pour la relation stricte associée, on notera $A >>^S B$ au lieu de $A >>^{ELI_S} B$.

4.1 Problème de circularité

Le problème de circularité, ou problème d'auto-référence, a été mis en évidence sur des exemples. Il apparaît à plusieurs niveaux : dans les formules, dans la construction des arguments et dans la détermination de leur acceptabilité.

Au niveau des formules, l'auto-référence apparaît avec la dénomination ; une formule s'auto-réfère si son nom apparaît dans la formule elle-même.

Exemple 4.2

La formule r1 : (r2 > r1) permet à la formule de nom r2 d'être prioritaire sur la formule de nom r1 ; autrement dit, une première formule peut être prioritaire sur une deuxième grâce à la deuxième formule elle-même !

On peut s'interroger sur la signification de telles formules ; qu'apportent-elles comme connaissance nouvelle ? Même si elles sont conditionnelles, ces formules établissent simplement qu'elles sont plus ou moins prioritaires que d'autres formules. Ces formules "raisonnent sur elles-mêmes" et ne font que statuer leur force ou leur faiblesse par rapport aux autres. Elles n'apportent rien, et pour cela elles ne sont pas souhaitables. De plus, elles peuvent entraîner un problème de circularité dans la construction des arguments et dans leur acceptabilité (voir exemple 4.4).
Pour les interdire, il faudrait redéfinir la fonction d'affectation des noms N de manière à ce que le nom choisi pour une formule n'apparaisse pas dans la formule elle-même.
Néanmoins, il n'est pas gênant de travailler avec des formules qui s'auto-réfèrent, puisque les problèmes de circularité qu'elles entraînent pourront être palliés. Nous choisissons donc de ne pas modifier la fonction N.

Remarque : Il ne faut pas confondre ce problème d'auto-référence avec le cas où les formules sont nommées à l'aide de termes fonctionnels, et que c'est une instance du nom de la formule qui apparaît dans la formule elle-même (voir exemple 4.5).

La circularité peut également apparaître dans la construction des arguments : un argument s'auto-réfère si sa conclusion fait référence à un élément de son support.

Exemple 4.3

*Soit une base Σ = (K, E) avec K = AXI et E = {**r1** : a, **r2** : ¬a, **r3** : a–(r1 > r2)}.*

*Considérons les arguments **A1** : ([r1], a), **A2** : ([r2], ¬a) et **A3** : ([r1, r3], (r1 > r2)).*

L'argument A3 conclut une priorité entre deux formules supportée par l'une des formules elle-même : cet argument s'auto-réfère.

Cet exemple d'auto-référence fait apparaître un problème plus général que nous avons mis en évidence au paragraphe 3.2.3 : pour qu'une défense soit bonne, il faut que les priorités qui permettent la défense soient supportées par des arguments jugés acceptables.

Exemple 4.3 (suite)

A3 permet à A1 de se défendre contre A2 qui le disqualifie. Par suite, on a envie de dire que A1 est acceptable. Or si on s'intéresse de plus près à l'argument A3, on voit qu'il est disqualifié, par A2. La défense de A1 n'est pas bonne pour que A1 soit acceptable.

Enfin, l'auto-référence peut apparaître au niveau de l'acceptabilité des arguments.

Exemple 4.4 [PS97]

*Soit la base Σ = (K, E) avec K = AXI et E = {**r1** : r2 > r1, **r2** : r1 > r2}.*

*Les arguments **A1** : ([r1], r2 > r1) et **A2** : ([r2], r1 > r2) se disqualifient mutuellement, et chacun permet à l'autre de se défendre contre son disqualifiant. Mais pour que A1 soit acceptable, il faut que sa défense soit bonne, donc que A2 soit acceptable. Or, pour que A2 soit acceptable, il faut que A1 soit acceptable : problème de circularité.*

Dans ce cas particulier, le problème vient initialement du fait que les formules de la base s'auto-réfèrent.

Les choix que nous allons faire pour la sémantique de l'ensemble des arguments acceptables devront permettre de pallier ces problèmes de circularité.

Remarque : L'exemple suivant pourrait aussi faire penser à un cas de circularité ; mais dans le cas où les croyances sont nommées à l'aide de termes fonctionnels, il est possible qu'une instance d'une règle fasse référence à une autre instance de cette même règle, sans que l'on puisse réellement parler de problème de circularité.

Exemple 4.5 [PS95]

Dans le domaine légal, les lois concernant les contrats de travail sont plus prioritaires que les lois concernant tout autre type de contrat. Un autre principe de priorité, le principe temporel, précise que les lois, contrats ou règles les plus récents sont préférés aux plus anciens. Il se trouve que le principe temporel est antérieur au premier principe de priorité énoncé.

Etant donné b une loi concernant un contrat de travail et a une loi plus récente concernant un quelconque contrat, traduisons ces connaissances dans la base Σ = (K, E) suivante :

K = { contrat(a), travail(b), plus-récent(a, b), plus-récent(f1(x, y), f2(x, y)) }

*E = { **f1(x, y)** : (contrat(x)Êtravail(y)) – (y > x), **f2(x, y)** : plus-récent(x, y) – (x > y) }*

On peut construire un argument qui donne la priorité à a sur b, et un autre qui donne la priorité à b sur a :

A1 *: ([f1(a, b)], b > a),* ***A2*** *: ([f2(a, b)], a > b).*

Mais A(∑) contient également un argument qui va permettre à A1 de se défendre strictement contre A2 :

A3 *: ([f2(f1(a, b), f2(a, b))], f1(a, b) > f2(a, b)).*

La conclusion de A3 ne fait pas référence à un élément de son support, mais à une instance d'un élément de son support. C'est le principe temporel appliqué à lui-même et au principe sur les contrats qui permet au principe sur les contrats d'être prioritaire sur le principe temporel dans le cas des lois a et b.

4.2 Sémantique de l'ensemble des arguments acceptables

Nous nous inspirons des travaux de Prakken et Sartor [PS97] pour caractériser l'ensemble des arguments acceptables ; comme eux, nous nous plaçons dans le cadre de la théorie du point fixe. Cette sémantique par point fixe nous permettra, entre autres, de résoudre le problème de circularité.

Définition 4.2

Soit <A(∑), R, Pref> un SAPC et **ACC** la fonction définie par :

$ACC : 2^{A(\Sigma)} \rightarrow 2^{A(\Sigma)}$

$S \rightarrow ACC(S) = \{A \in A(\Sigma) / \text{ si } B \in A(\Sigma) \text{ tel que } B\ R\ A, \text{ alors } A >>^{Pref_S} B\}$

i.e. ACC(S) contient les arguments qui se défendent contre leurs contrariants grâce aux préférences issues de $>_S$.

Propriété 4.1

(i) $ACC(\emptyset) = C_R$

(ii) $ACC(\emptyset)$ est sans-R-conflit

(iii) $\forall S \subseteq A(\Sigma), C_R \subseteq ACC(S)$

Preuve

(i) $ACC(\emptyset) = C_R$

ACC(Ø) est l'ensemble des arguments qui "se défendent" contre leurs contrariants "sans l'aide d'aucune préférence" ; c'est donc l'ensemble des arguments qui ne sont pas contrariés (C_R).

(ii) $ACC(\emptyset)$ est sans-R-conflit

Puisque $ACC(\emptyset) = C_R$ et que C_R est sans-R-conflit, $ACC(\emptyset)$ est sans-R-conflit.

(iii) $\forall S \subseteq A(\Sigma), C_R \subseteq ACC(S)$

D'après la définition de ACC.

Exemple 3.7 (suite)

$ACC(\emptyset) \supseteq \{A3\}$ puique A3 n'est pas contrarié ;

$ACC(\{A3\}) \supseteq \{A3, A1\}$ car A1 se défend contre A2 grâce à {A3}.

Propriété 4.2

$\forall S \subseteq A(\Sigma)$, ACC(S) est sans-conflit.

Preuve

Supposons qu'il existe deux arguments A : (H, h) et A' : (H', h') appartenant à ACC(S) tels que A disqualifie A'.

A' â ACC(S) = A' $>>^S$ A

A : (H, h) disqualifie A' : (H', h') = V(H') u K ⊢ ¬h, donc A est disqualifié par un sous-argument de A', soit A".

A" sous-argument de A', donc A" $>>^S$ A (propriété 3.8).

Contradiction avec le fait que A â ACC(S) = A $>>^S$ A".

Exemple 3.8 (suite)

Rappelons les quatre arguments construits sur la base de l'exemple 3.8 :

***A1** : ([f1(villa)], ¬modifier(villa)), **A2** : ([f2(villa)], modifier(villa))*

***A3** : ([f3(f1(villa), f2(villa))], (f1(villa) > f2(villa))), **A4** : ([f4(f2(villa), f1(villa))], (f2(villa) > f1(villa)))*

Les conflits entre ces arguments sont représentés ci-dessous :

$$A2 \rightleftarrows A1 \qquad A3 \rightleftarrows A4$$

• Si S1 = {A2, A3}, $>_{S1}$ = {f1(villa) > f2(villa)} et ACC(S) = {A1}, car A1 $>>^{S1}$ A2.

• Si S2 = {A2, A3, A4}, $>_{S2}$ = {f1(villa) > f2(villa), f2(villa) > f1(villa)} et ACC(S2) = Ø, car maintenant A1 est S2-préféré à A2 et A2 est S2-préféré à A1 ; la préférence de A1 sur A2 n'est plus stricte.

Remarque 1 :

Pour qu'un argument appartienne à ACC(S), il doit se défendre contre ses contrariants en leur étant strictement préféré. Si cette préférence n'était pas stricte, ACC(S), qui est censé caractériser l'ensemble des arguments acceptables, ne serait pas toujours sans conflit.
L'exemple précédent permet d'illustrer ceci. Pour S2 = {A2, A3, A4}, A1 et A2, qui sont en conflit, appartiendraient tous deux à ACC(S).

Remarque 2 :

L'exemple ci-dessus illustre également que pour S √ A(∑), on n'a pas toujours S √ ACC(S) ni ACC(S) √ S.

Remarque 3 :

De plus, l'exemple montre que ACC n'est pas une fonction monotone, puisque S1 √ S2 et ACC(S1) ∫ ACC(S2).

Notons tout de même que dans l'exemple précédent, S2, et en particulier AP_{S2}, ne sont pas sans-conflit, et que ces conflits engendrent un cycle de longueur 2 dans $>_{S2}$.

Pour la propriété qui suit, nous allons considérer la restriction de ACC à des ensembles S d'arguments tels que $>_S$ sans cycle.

Propriété 4.3

Soit le système d'argumentation <A(∑), Disqualifie, ELI> et S, S' √ A(∑).
Si S √ S', et $>_S$, $>_{S'}$ sans cycle alors ACC(S) √ ACC(S').

Preuve

Soit S √ S', $>_S$ et $>_{S'}$ sans cycle et A : (H_A, h_A) â A(∑) tel que A â ACC(S).

Montrons que A â ACC(S').

- Supposons qu'il existe B : (H_B, h_B) â A(∑) tel que B disqualifie A.

Comme A â ACC(S), on sait que A $>>^S$ B, c'est-à-dire A S-préféré à B et non(B S-préféré à A).

- Comme S √ S', d'après la propriété 3.7, A S-préféré à B = A S'-préféré à B. (1)
- non(B S-préféré à A) = ™ r_B â $H_B \setminus H_A$ tel que å r_Aâ $H_A \setminus H_B$, ($r_B > r_A$) á $>_S$

$= H_B \setminus H_A \neq Ø = H_B \int H_A$

d'où, d'après la propriété 3.9, comme $>_{S'}$ sans cycle,

A S'-préféré à B = non(B S'-préféré à A) (2)

- (1) et (2) = A â ACC(S').

Il reste à montrer la propriété suivante pour vérifier les propriétés qui suivent :

å S √ A(∑), si $>_S$ est sans cycle alors $>_{ACC(S)}$ est sans cycle.

Propriété 4.4

Si l'on se restreint aux parties S de A(∑) telles que $>_S$ sans cycle, alors la fonction ACC admet un plus grand et un plus petit point fixe.

Preuve

A partir d'un résultat classique de la théorie du point fixe, $< 2^{A(\Sigma)}$, √ > étant un treillis complet, ACC une fonction (de $2^{A(\Sigma)}$ dans $2^{A(\Sigma)}$) monotone sous les conditions vues à la propriété 4.3.

Propriété 4.5

Soit le système d'argumentation fini <A(∑), Disqualifie, ELI>.
Pour toute suite emboîtée d'ensembles S_1 √ S_2 √ ...√ S_i √ ... tels que å i≥1, S_i √ A(∑) et $>_{Si}$ sans cycle, on a ACC($u_{i\geq 1}$ S_i) = $u_{i\geq 1}$ ACC (S_i).

Preuve

• *$u_{i\geq 1}$ (ACC (S_i)) √ACC($u_{i\geq 1}$ S_i) ?*

Soit A â $u_{i\geq 1}$ ACC (S_i) = il existe un nombre k tel que A â ACC (S_k).

Or S_k √ $u_{i\geq 1}$ S_i. Montrons que $>_{(ui\geq 1\ Si)}$ est sans cycle pour pouvoir appliquer la propriété 4.10. Pour cela, supposons que $>_{(ui\geq 1\ Si)}$ contient un cycle de longueur n :

$r_1 > r_2$, $r_2 > r_3$, ..., $r_{(n-1)} > r_n$, $r_n > r_{(n+1)}$, $r_{(n+1)} > r_1$

alors il existe un nombre j tel que $>_{Sj}$ contient ce cycle, car les S_i sont emboîtés. Contradiction avec l'hypothèse $>_{Sj}$ sans cycle.

D'où, d'après la propriété 4.3, ACC (S_k) √ ACC($u_{i\geq 1}$ S_i).

• $ACC(u_{i\geq 1} S_i) \sqrt{} u_{i\geq 1} (ACC (S_i))$?

Soit A â ACC($u_{i\geq 1} S_i$). Si B disqualifie A alors A $>>^{(ui\geq 1\ Si)}$ B, c'est-à-dire que le support de A est strictement préféré au support de B en utilisant des conclusions d'arguments de $u_{i\geq 1} S_i$.

Comme les supports de A et B sont finis, on n'utilise qu'un nombre fini de priorités.

Donc å B_k contrariant de A, il existe S_k tel que A $>>^{Sk}$ B_k. Comme A a un nombre fini de contrariants, il existe j tel que A $>>^{Sj}$ B, å B. Et donc A â ACC (S_j).

D'où A â $u_{i\geq 1}$ (ACC (S_i)).

Propriété 4.6

Le plus petit point fixe de ACC s'obtient par $u_{i\geq 1}$ ACC^i (Ø).

Preuve

D'après la propriété 4.5 et un résultat classique de la théorie du point fixe.

Définition 4.3

L'***ensemble des arguments acceptables*** est le plus petit point fixe de ACC.

La difficulté de cette construction par point fixe est de déterminer, parmi l'ensemble de tous les arguments construits sur une base de connaissances ∑, l'ensemble de tous les arguments non contrariés (i.e. ACC(Ø)).

Plusieurs points vont nous aider à déterminer cet ensemble :

- le fait qu'un argument (H,h) qui conclut sur une priorité est tel que h est un littéral positif;
- la propriété 3.4 qui va permettre d'éliminer certains arguments ; en effet, si A est un argument disqualifié, alors tout argument qui a A comme sous-argument, est également disqualifié.

Exemple 4.6

*Soit ∑ = (K, E) avec E = { **r1** : a, **r2** : a–b, **r3** : ¬a, **r4** : r3 > r1, **r5** : r1 > r3, **r6** : r4 > r5 }.*

*Les arguments **A1** : ([r1], a) et **A3** : ([r3], ¬a) se disqualifient mutuellement, ainsi que les arguments **A4** : ([r4], r3 > r1) et **A5** : ([r5], r1 > r3). Les arguments **A2** : ([r2], a–b) et **A6** : ([r6], r4 > r5) par contre, ne sont pas disqualifiés. On ne peut construire aucun argument ayant pour sous-argument A2 ou A6. L'ensemble des arguments non disqualifiés est donc :*

$S_1 = ACC(Ø) = \{A2, A6\}$

Le seul argument supportant une priorité est A6 ; c'est lui qui permet à A4 d'être strictement préféré à A5. D'où:

$S_2 = ACC(S_1) = \{A2, A6, A4\}$

L'argument A4 à son tour permet à A3 de se défendre contre A1.

$S_3 = ACC(S_2) = \{A2, A6, A4, A3\}$

Aucun argument de S_3 ne permet de rendre acceptable un autre argument.

$ACC(S_3) = S_3$

S_3 est donc le plus petit point fixe de ACC. {A2, A6, A4, A3} est l'ensemble des arguments acceptables.

L'ensemble des arguments éliminés contient A1 et A5, ainsi que tous les arguments construits sur ∑ qui ont pour sous-argument A1 ou A5. Il n'y a pas d'argument en suspens.

Exemple 4.8 [PS97]

Soit la base ∑ = (K, E) où E est tel que :

*E = { **r4** : r2 > r1, **r5** : r3 > r2, **r6** : r1 > r3, **r7** : r4 > r6, **r8** : r5 > r6 }.*

*Seuls les arguments **A7** : ([r7], r4 > r6) et **A8** : ([r8], r5 > r6) ne sont pas contrariés.*

$S_1 = ACC(\emptyset) = \{A7, A8\}$

$>_{S1}$ *= { r4 > r6, r5 > r6 }, ce qui implique que tous les arguments dont le support contient r4 ou r5 sont strictement* S_1*-préférés aux arguments dont le support contient r6. Ainsi, **A1** : ([r5, r4], r3 > r1) est strictement* S_1*-préféré à tous ses contrariants, qui sont :*

***A2** : ([r4, r6], r2 > r3), **A3** : ([r6, r5], r1 > r2) et **A6** : ([r6], r1 > r3).*

*Les sous-arguments **A4** : ([r4], r2 > r1) et **A5** : ([r5], r3 > r2) de A1 sont, quant à eux, strictement* S_1*-préféré à leurs contrariants A3 et A2. Donc :*

$S_2 = ACC(S_1) = \{ A7, A8, A1, A4, A5 \}$

Aucun autre argument n'est acceptable.

$ACC(S_2) = S_2$

{ A1, A4, A5, A7, A8 } est l'ensemble des arguments acceptables. L'ensemble des arguments éliminés est {A2, A3, A6}, ces trois arguments étant disqualifiés par des arguments acceptables. Aucun argument n'est en suspens.

Cette sémantique permet de pallier les problèmes de circularité des exemples 4.3 et 4.4 :

Exemple 4.3 (suite)

A1 était strictement préféré à A2 grâce à A3. Or l'argument A3 était disqualifié par A2. La défense de A1 contre A2 était donc "mauvaise", A1 ne pouvait être acceptable.

Vérifions si la construction par point fixe de l'ensemble des arguments acceptables confirme ceci :

A1, A2 et A3 sont tous trois contrariés, ils n'apparaissent donc pas dans $S_1 = ACC(\emptyset)$*. A3 étant le seul argument supportant une relation de priorité, aucun argument dans* S_1 *ne permet alors de donner une préférence entre A1 et A2. A1 ne se défend donc pas contre A2 et il n'est pas un argument acceptable.*

Exemple 4.4 (suite)

*Les arguments **A1** : ([r1], r2 > r1) et **A2** : ([r2], r1 > r2) se disqualifient,* $ACC(\emptyset) = \emptyset$*. Ni A1 ni A2 n'est acceptable.*

L'exemple qui suit montre que notre sémantique pourrait être revue :

Exemple 4.7 [PS97]

Soit la base de connaissances ∑ = (K, E) telle que :

K = { a, c, d } et

*E = { **r1** : a–b, **r2** : (aÊc)–¬b, **r3** : d–b, **r4** : (cÊd)–¬b, **r5** : r2 > r1, **r6** : r4 > r3}.*

On peut construire sur cette base plusieurs arguments ayant même support mais avec des conclusions différentes, par exemple ([r1], a–b) et ([r1], b).

Les arguments de support [r1] et [r3] sont disqualifiés par des arguments de support [r2] et [r4]. Les seuls arguments non disqualifiés sont : ***A5*** *: ([r5], r2 > r1) et* ***A6*** *: ([r6], r4 > r3).*

D'où : $S_1 = ACC(\emptyset) = \{A5, A6\}$.

Intéressons-nous plus particulièrement à l'acceptabilité des quatre arguments suivants : ***A1*** *: ([r1], b),* ***A2*** *: ([r2], ¬b),* ***A3*** *: ([r3], b),* ***A4*** *: ([r4], ¬b). A2 $>>^{S1}$ A1 et A4 $>>^{S1}$ A3, mais A2 est disqualifié par A3 , A4 est disqualifié par A1 et on n'a pas A2 $>>^{S1}$ A3 ni A4 $>>^{S1}$ A1.*

Plus généralement, les arguments de support [r2] et [r4] ne se défendent pas contre tous leurs contrariants ; ils ne sont donc pas acceptables. D'où :

$$ACC(S_1) = S_1$$

{A5, A6} est l'ensemble des arguments acceptables, l'ensemble des arguments éliminés est vide et celui des arguments en suspens contient tous les autres arguments de $A(\Sigma)$.

Prakken et Sartor dans [PS97] font remarquer que dans le contexte de l'héritage non-monotone, les arguments A2 et A4 de cet exemple auraient été acceptés. En effet, puisque tous les arguments pour ¬b disqualifient les arguments pour b et leurs sont strictement préférés, alors les arguments pour ¬b, dans ce contexte, sont acceptables.
Cet exemple montre que la sémantique que l'on a choisie est peut-être trop prudente. Il serait intéressant de voir ce qu'il faudrait changer pour obtenir ce résultat.

Intéressons-nous enfin à l'acceptabilité des arguments dans les exemples 3.7 et 3.8 :

Exemple 3.7 (suite)

L'argument ***A3*** *: ([r3], (r2 > r1)) est le seul argument non contrarié ; ainsi, S1 = ACC(Ø) = {A3}. A3 permet à* ***A2*** *: ([r2], pg) de se défendre contre* ***A1*** *: ([r1], ¬pg), donc S2 = ACC(S1) = ({A2, A3}) et S3 = ACC(S2). Il existe ainsi un argument acceptable pour la propagation de l'information (A2). A1 est un argument éliminé.*

Exemple 3.8 (suite)

Tous les arguments sont contrariés : ACC(Ø) = Ø. Aucun argument n'est acceptable, tous sont en suspens.

4.3 Théorie de la preuve

Grâce à la sémantique donnée précédemment, on peut caractériser l'ensemble des arguments acceptables, l'ensemble des arguments rejetés, et l'ensemble des arguments en suspens. On aimerait maintenant avoir une méthode pour connaître le statut d'un argument (accepté, rejeté ou en suspens) sans avoir à calculer ces trois ensembles et voir auquel il appartient. Pour cela, nous proposons une théorie de la preuve, similaire à celle introduite par Prakken et Sartor dans [PS97].

Comme nous l'avons vu, dans les systèmes d'argumentation finis, un argument est acceptable s'il appartient au plus petit point fixe de la fonction ACC, lequel est construit par applications successives de ACC à l'ensemble vide. L'idée de base est que l'on va traverser la séquence résultante $ACC^1(Ø)$, ..., $ACC^n(Ø)$ en sens inverse, l'argument A dont on cherche à déterminer l'acceptabilité se trouvant dans $ACC^n(Ø)$.

On commence la preuve avec A, et pour tout argument B disqualifiant A, on trouve un ensemble C d'arguments inclus dans $ACC^{(n-1)}(Ø)$ qui permettent à A d'être strictement préféré à B ; ainsi A se défend contre B. Ensuite, pour tout argument disqualifiant un argument de C, on trouve un argument dans $ACC^{(n-2)}(Ø)$ permettant à cet argument disqualifié d'être strictement préféré à son contrariant, et ainsi de suite. Puisque le système d'argumentation est fini, la preuve va se terminer avec un ensemble d'arguments inclus dans $ACC^1(Ø)$, c'est-à-dire des arguments non disqualifiés.

La preuve d'un argument prend la forme d'un arbre de dialogue. La racine de l'arbre est l'argument que l'on cherche à prouver, chaque branche est un dialogue entre deux personnes : Pour (**P**) et Contre (**C**). Un coup du dialogue consiste en la donnée d'un ensemble d'arguments. Pour ces ensembles d'arguments, P et C vont avoir des rôles différents :

- Au coup c_i, P va donner des arguments qui permettent à l'argument proposé au coup $c_{(i-2)}$ qui est contrarié par l'argument proposé au coup $c_{(i-1)}$ de lui être strictement préféré.
- C quant à lui, donne au coup c_k, un argument qui disqualifie un argument proposé au coup $c_{(k-1)}$.

Formellement, dialogue et arbre de dialogue sont définis comme suit :

Définition 4.4

Un ***dialogue*** est une séquence non vide finie de coups avec $coup_i = (Joueur_i, Arg_i)$, $(i \geq 1)$ tel que :

1. $Joueur_i = P$ ssi i est impair ; $Joueur_i = C$ ssi i est pair ;
2. $Joueur_1 = P$ et $Arg_1 = A$, A étant l'argument que l'on cherche à prouver ;
3. Si $Joueur_i = Joueur_j = P$ et $i \neq j$, alors $Arg_i \neq Arg_j$;
4. Si $Joueur_i = P$, avec $i > 1$, alors Arg_i est un ensemble d'arguments supportant des priorités permettant à l'argument de $Arg_{(i-2)}$ contrarié par l'argument de $Arg_{(i-1)}$ d'être strictement préféré à son contrariant ;
5. Si $Joueur_i = C$ alors Arg_i est un ensemble contenant un seul argument, lequel disqualifie un argument de $Arg_{(i-1)}$.

La condition 1 exprime le fait que P et C jouent à tour de rôle. La condition 2 impose que c'est le joueur P qui commence le dialogue ; P est à la racine de l'arbre. La condition 3 empêche le joueur P de répéter son attaque, c'est-à-dire de donner le même ensemble d'arguments plusieurs fois.

Définition 4.5

Un ***arbre de dialogue*** est un arbre fini dont chaque branche est un dialogue.

Notons que l'arbre de dialogue peut-être considéré comme un arbre ET/OU. Un noeud correspondant au joueur P est un noeud ET, et un noeud correspondant au joueur C est un noeud OU. Cette distinction entre les noeuds est due au fait qu'un argument est acceptable si les arguments qui servent à le défendre se défendent tous contre leurs contrariants.
Les fils d'un noeud contenant des arguments de P représentent des contrariants, il faut donc les battre tous. Par contre, les fils d'un noeud contenant un argument du joueur C représentent des défenseurs de P ; il suffit donc qu'un seul d'entre eux montre que l'argument donné par P se défend contre celui donné par C.

Définition 4.6

• Le joueur P ***gagne un dialogue*** si l'autre joueur ne peut pas donner d'argument.

• Le joueur P ***gagne un arbre de dialogue*** ssi il existe un sous-arbre contenant tous les fils par noeud ET et un seul fils par noeud OU, et dont toutes les branches sont des dialogues gagnés par P.

Propriété 4.7

Si le joueur P gagne un dialogue alors son dernier coup est un sous-ensemble d'arguments de la classe $C_{Disqualifie}$.

Preuve

Soit A le dernier ensemble d'arguments fourni par P. Puisque P a gagné le dialogue, c'est que C n'a pas d'argument qui disqualifie un argument de A. Donc, d'après la définition de la classe $C_{Disqualifie}$ (ensemble des arguments non disqualifiés), $A \surd C_{Disqualifie}$.

Définition 4.7

Un argument A est ***justifié*** ssi il existe un arbre de dialogue dont la racine est A, et gagné par le joueur P.

Exemple 4.6 (suite)

• Nous avons vu que l'argument A3 â $ACC^3(\emptyset)$, A3 est donc acceptable. Donnons un arbre de dialogue qui le justifie.

P, au premier coup, avance l'argument A3 ;

$\mathbf{P_1}$: { A3 : ([r3], ¬a) }

Or A3 est disqualifié par A1, C avance donc l'argument A1 ;

$\mathbf{C_1}$: { A1 : ([r1], a) }

P va ensuite donner l'argument A4 puisqu'il permet à l'argument A3 d'être strictement préféré à l'argument A1 ;

$\mathbf{P_2}$: { A4 : ([r4], r3 > r1) }

Mais A4 est disqualifié par A5 ;

C2 : { A5 : ([r5], r1 > r3) }

P propose alors l'argument A6, qui permet à A4 d'être strictement préféré à A5 ;

P3 : { A6 : ([r6], r4 > r5) }

A6 n'est disqualifié par aucun argument ; C n'a donc plus d'argument à présenter. L'arbre de dialogue se résume à une branche de dialogue, laquelle est gagnée par P. L'argument A3 est justifié.

Remarquons que A3 â $ACC^3(Ø)$, A4 â $ACC^2(Ø)$, A6 â $ACC(Ø)$.

• Interrogeons-nous sur la justification de l'argument A2.

P va proposer au premier coup A2.

P1 : { A2 : ([r1, r2], b) }

Or A2 est disqualifié par A3 ;

C1 : { A3 : ([r3], ¬a) }

P ne peut pas donner un ensemble d'arguments permettant à A2 d'être strictement préféré à A3. P ne gagne pas l'arbre de dialogue, A2 n'est pas justifié.

Exemple 4.7 (suite)

Comme nous l'avons vu, l'argument A1 est acceptable ; donnons l'arbre de preuve qui le justifie.

P1 : { A1 : ([r5, r4], r3 > r1) }

/ | \

C1 : { A2 : ([r4, r6], r2 > r3) } **C1** : { A6 : ([r6], r1 > r3) } **C1** : { A3 : ([r5, r6], r1 > r2)}

| | |

P2 : { A8 : ([r8], r5 > r6) } **P2** : { A7 : ([r7], r4 > r6), A8 : ([r8], r5 > r6) } **P2** : { A7 : ([r7], r4 > r6) }

P possède un sous-arbre gagnant, A1 est bien justifié.

Exemple 4.8 (suite)

• Montrons que l'argument A1 n'est pas justifié. L'arbre de dialogue s'organise comme suit:

P1 : { A1 : ([r1], b) }

/ \

C1 : { A2 : ([r2], ¬b) } **C1** : { A4 : ([r4], ¬b) }

En effet, A1 possède deux contrariants, A2 et A4. P ne peut plus jouer dans aucune des deux branches puisqu'il n'existe pas d'ensemble d'arguments permettant à A1 d'être strictement préféré à A2 et A4. P ne gagne pas l'arbre de dialogue et A1 n'est pas justifié.

• Regardons maintenant si l'argument A2 est justifié :

P1 : { A2 : ([r2], ¬b) }

/ \

C1 : { A1 : ([r1], b) } **C1** : { A3 : ([r3], b) }

|

P2 : { A5 : ([r5], r2 > r1) }

P gagne la branche de gauche, mais pas celle de droite. L'arbre de dialogue n'est donc pas gagné par P et A2 n'est pas justifié.

Exemple 4.3 (suite)

L'argument A1 ne devrait normalement pas être justifié ; vérifions ceci :

P1 : { A1 : ([r1], a) }
|
C1 : { A2 : ([r2], b) }
|
P2 : { A3 : ([r1, r3], r1 > r2) }
|
C2 : { A2 : ([r2], b) }

P n'ayant pas le droit de reproposer le même ensemble d'arguments, au contraire de C, P ne gagne pas l'arbre de dialogue. A1 n'est pas justifié.

Exemple 4.9

*Soit la base ∑ = (K, E) où E = {**r1** : a, **r2** : a–b, **r3** : c, **r4** : c–¬b, **r5** : r1 > r3, **r6** : r2 > r4, **r7** : r3 > r1, **r8** : r5 > r7}. Le littéral b est-il supporté par un argument justifié ?*

P1 : { A1 : ([r1, r2], b) }
|
C1 : { A2 : ([r3, r4], ¬b) }
|
P2 : { A3 : ([r5], r1 > r3), A4 : ([r6], r2 > r4) }
|
C2 : { A5 : ([r7], r3 > r1) }
|
P3 : { A6 : ([r8], r5 > r7) }

A1 est disqualifié par A2 mais A1 est strictement {A3, A4}-préféré à A2. Le problème est que l'argument A3 est disqualifié par l'argument A5 ; par chance, A3 est strictement {A6}-préféré à A5, et il n'est pas disqualifié. P gagne l'arbre de dialogue, A1 est un argument justifié.

Exemple 3.7 (suite)

Montrons que l'argument A2 pour la propagation de l'information est justifié :

P1 : { A2 : ([r2], ¬pg) }
|
C1 : { A1 : ([r1], pg) }
|
P2 : { A3 : ([r3], r2 > r1) }

Exemple 3.8 (suite)

Rappelons les quatre arguments que nous avons construits dans cet exemple :

***A1** : ([f1(villa)], ¬modifier(villa)), **A2** : ([f2(villa)], modifier(villa)), **A3** : ([f3(f1(villa), f2(villa))], (f1(villa) > f2(villa))), **A4** : ([f4(f2(villa), f1(villa))], (f2(villa) > f1(villa))) .*

Montrons que ni l'argument A1, ni l'argument A2 n'est justifié :

P1 : { A1 }
|
C1 : { A2 }
|
P2 : { A3 }

P1 : { A2}
|
C1 : { A1 }
|
P2 : { A4 }

C2 : { A4 }	**C2** : { A3 }

Si on ajoute aux connaissances de cette base que le principe P_4 est plus récent que le principe P_3, on peut construire un argument qui donne la priorité au principe P_4 sur le principe P_3. La base devient alors :

$\Sigma' = (K', E)$ avec $K' = K \cup \{plus\text{-}récent(f4(x, y), f3(x, y))\}$ et on a :

***A5** : $([f4(f4(f2(villa), f1(villa)), f3(f1(villa), f2(villa)))], f4(f2(villa), f1(villa)) > f3(f1(villa), f2(villa)))$.*

L'ensemble des arguments acceptables de l'ensemble {A1, A2, A3, A4, A5} est tel que :

$S_1 = ACC(\emptyset) = \{A5\}$

$S_2 = ACC(S_1) = \{A5, A4\}$

$S_3 = ACC(S_2) = \{A5, A4, A2\}$

$S_4 = ACC(S_3) = S_3$

Donnons une preuve de l'argument A2 :

P1 : { A2}
|
C1 : { A1 }
|
P2 : { A4 }
|
C2 : { A3 }
|
P3 : { A5 }

L'argument A2 est donc acceptable et justifié ; on a un bon argument pour la modification de la façade de la villa.

Nous avons proposé dans ce paragraphe les bases d'une théorie de la preuve. Une continuation de ce travail consistera à étudier l'adéquation et la complétude, c'est-à-dire à montrer que les arguments acceptables correspondent exactement aux arguments justifiés.

5. Conclusion

Nous nous sommes intéressées dans ce rapport à la prise en compte de priorités conditionnelles dans le raisonnement argumentatif. Ainsi, nous avons posé les bases d'un Système d'Argumentation avec Préférences basé sur des priorités Conditionnelles. Le travail s'est effectué en plusieurs étapes : tout d'abord nous avons montré comment exprimer des priorités conditionnelles entre formules à l'intérieur d'un langage du premier ordre. Les priorités entre formules sont alors dérivées du langage (elles sont notamment conclusion d'arguments) et peuvent être contestées. Nous avons défini une relation de préférence entre arguments basée sur un ensemble d'arguments supportant des priorités, et nous avons montré la nécessité de travailler avec une nouvelle relation de contrariété, la disqualification, qui combine les relations classiquement considérées, la réfutation et l'attaque. Nous avons enfin redéfini l'acceptabilité des arguments en prenant en compte à la fois la relation de disqualification et la nouvelle relation de préférence, et nous avons posé les bases d'une théorie de la preuve.

Les priorités conditionnelles, comme nous l'avons dit, s'avèrent contestables. Pour cette raison, elles ont été peu étudiées jusqu'à présent, la plupart des travaux s'intéressant à des priorités incontestables. La prise en compte de ces priorités s'est faite ici dans le cadre du modèle argumentatif défini par Amgoud et Cayrol dans [AC97&98]. Ce travail est une contribution à l'intégration de priorités contestables dans leur modèle.

Les préférences conditionnelles ne sont pas les seules préférences contestables que l'on peut intégrer aux systèmes d'argumentation, mais leur seule prise en compte donne un aperçu des difficultés à travailler avec des relations de préférence non figées. Il sera intéressant d'approfondir cette étude.

6. Perspectives

Nous avons fait un certain nombre de restrictions, de simplifications concernant les arguments concluant sur des priorités, mais ces dernières n'ont pas suffi pour résoudre les problèmes rencontrés lors de l'étude de l'acceptabilité des arguments. Ces problèmes ont souvent été mis en évidence au travers d'exemples. Nous en avons fait disparaître certains, mais d'importants restent ouverts.

À court terme

• Résoudre le problème de l'acceptabilité des arguments en montrant la propriété suivante :

å S â A(∑), si $>_S$ sans cycle alors $>_{ACC(S)}$ sans cycle.

(Nous avons montré que la fonction ACC n'est monotone que sous une condition particulière : $>_S$ sans cycle.)

• Montrer l'adéquation et la complétude de la théorie de la preuve proposée, c'est-à-dire que les arguments acceptables correspondent exactement aux arguments justifiés.

À moyen terme

• Donner une procédure pour construire l'ensemble des arguments non disqualifiés. On pourra s'aider de la propriété 3.4 et d'une propriété supplémentaire : lorsqu'un argument A disqualifie un argument B de support H, alors il disqualifie tous les arguments de support H. Ce résultat découle de la définition de la relation de disqualification.
Dans l'exemple 4.8 notamment, il est difficile de caractériser tous les arguments de support [r1] et d'étudier individuellement l'acceptabilité de chacun d'eux. Par contre, il est facile de voir qu'au moins l'un d'eux est disqualifié. Grâce au résultat ci-dessus, on élimine d'un coup tous les arguments de support [r1].

À long terme

• Regarder ce qui se passerait si les clauses contenant le prédicat > (i.e. l'expression d'une priorité entre formule) n'étaient pas de Horn en >. Les arguments supportant des priorités pourraient avoir des conjonctions, disjonctions ou implications de formules de priorité en conclusion. Qu'est-ce que cela entraînerait dans la détermination de l'acceptabilité des arguments ? dans le déroulement de la théorie de la preuve ?

• Considérer l'acceptabilité avec un deuxième niveau de contrariété. Dung ([Dun39&95]) a introduit une notion d'acceptabilité conjointe qui prend en compte plusieurs niveaux de contrariété ; dans ce cas, un argument peut être acceptable si ses contrariants sont eux-mêmes

contrariés. Cette notion a été prise en compte dans les Systèmes d'Argumentation avec Préférences basés sur des priorités Incontestables ([AC98b]). Il serait très intéressant de la considérer dans le cas où les priorités sont conditionnelles.

• Choisir une sémantique moins prudente, comme cela a été suggéré par l'exemple 4.8. Un argument pourrait être acceptable s'il est strictement préféré à au moins un de ses contrariants (dans le cas où le contrariant est un Réfute-contrariant).

Bibliographie

[AC97] L. Amgoud, C. Cayrol. Integrating Preference Orderings into Argument-based reasoning. Actes de *International Joint Conference on Qualitative* et *Quantitative Practical Reasoning, ECSQUARU-FAPR'97*. LNAI 1244. pp. 159-170, 1997.

[AC98a] L. Amgoud, C. Cayrol. Les systèmes d'argumentation basés sur les préférences. *Technical Report 98-01-R.* Janvier 1998.

[AC98b] L. Amgoud, C. Cayrol. On the acceptability of arguments in Preference-based argumentation frameworks. Actes de *14th Conference on Uncertainty in Artificial Intelligence, UAI'98.* pp. 1-7, 1998.

[ACL96] L. Amgoud, C. Cayrol, D. Le Berre. Comparing Arguments using Preference Orderings for Argument-based Reasoning. Actes de *8th International Conference on Tools with Artificial Intelligence, ICTAI'96.* pp. 400-403, 1996.

[Bre94] G. Brewka. Reasoning About Priorities in Default Logic. Actes de *National Conference on Artificial Intelligence, AAAI'94*. pp. 940-945, 1994.

[BDP93] S. Benferhat, D. Dubois, H. Prade. Argumentative Inference in Uncertain and Inconsistent Knowledge Bases. Actes de *14th International Conference On Uncertainty in Artificial Intelligence, UAI'93*. pp. 411-419, 1993.

[Cay95] C. Cayrol. On the Relation between Argumentation and Coherence-Based Non-monotonic Entailment. Actes de *14th International Joint Conference on Artificial Intelligence, IJCAI'95*. pp. 1443-1448, 1995.

[CRS93] C. Cayrol, V. Royer, C. Saurel. Management of preferences in Assumption-Based Reasoning. *Lecture Notes in Computer Science (B. Bouchon-Meunier, L. Valverde, R.Y. Yager Eds.), Springer Verlag,* vol. 682, pp. 13-22, 1993.

[Dun93] P. M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning and logic programming. Actes de *13th International Joint Conference on Artificial Intelligence, IJCAI'93*. pp. 852-857, 1993.

[Dun95] P. M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence,* vol. 77. pp. 321-357, 1995.

[DS97] J. P. Delgrande, T. H. Schaub. Compiling reasoning with and about preferences into default logic. Actes de *15th International Joint Conference on Artificial Intelligence, IJCAI'97.* pp. 168-174. 1997.

[EFK93] M. Elvang-Goransson, J. Fox, P. Krause. Acceptability of arguments as logical uncertainty. Actes de *Symbolic and Quantitative Approaches to Reasoning under Uncertainty, ECSQARU'93.* pp. 85-90, 1993.

[PS95] H. Prakken, G. Sartor. On the relation between legal language and legal argument: assumptions, applicability and dynamic priorities. Actes de *5th International Conference in Artificial Intelligence and Law, ICAIL'95.* 1995.

[PS96] H. Prakken, G. Sartor. A System for Defeasible Argumentation, with Defeasible Priorities. Actes de *Formal Approach of Practical Reasoning, FAPR'96*. pp. 510-524, 1996.

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