UNIVERSITÉ TOULOUSE III – PAUL SABATIER
U.F.R. Mathématiques Informatique Gestion

THÈSE
pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ TOULOUSE III
Discipline : Informatique

présentée et soutenue
par
Sylvie DOUTRE

le 12 décembre 2002

Autour de la sémantique préférée des systèmes d'argumentation

Directrice de thèse :
Claudette CAYROL

JURY

M. Ph. BESNARD	Directeur de Recherche CNRS/IRIT, Toulouse (Invité)
Mme C. CAYROL	Professeur à l'Université Toulouse III
M. M. HABIB	Professeur à l'Université Montpellier II
Mme M.-Ch. LAGASQUIE-SCHIEX	Maître de Conférences à l'Université Toulouse III
M. J. MENGIN	Maître de Conférences à l'Université Toulouse III
Mme C. SCHWIND	Chargée de Recherche CNRS/MAP, Marseille (Rapporteuse)
M. D. VERMEIR	Professeur à l'Université libre de Bruxelles (Rapporteur)

Remerciements

Je tiens à remercier tout d'abord Camilla SCHWIND et Dirk VERMEIR pour l'intérêt qu'ils ont porté à mon travail en acceptant d'en être les rapporteurs.

Je remercie également Michel HABIB et Philippe BESNARD pour avoir accepté de participer à mon jury de thèse.

Mes remerciements vont aussi à ceux qui m'ont encadrée durant ces trois années de thèse, et en premier lieu, à ma directrice de thèse, Claudette CAYROL. C'est elle et Michel CAYROL qui m'ont fait découvrir l'Intelligence Artificielle, et l'initiation à la recherche que nous avons faite ensemble en DEA est ce qui, en grande partie, m'a donné envie de faire cette thèse.

J'ai pu alors appréhender tous les aspects de la recherche, et, par la collaboration avec Jérôme MENGIN et Marie-Christine LAGASQUIE-SCHIEX, j'ai compris que le travail du chercheur n'est pas seulement une quête solitaire ; c'est souvent en équipe que se construisent les idées les plus originales et que la recherche prend tout son sens. À travers eux trois, j'ai appris que curiosité, rigueur, ouverture d'esprit et dynamisme sont des qualités nécessaires au chercheur.

Et ce n'est pas seulement pour avoir aiguisé mon goût pour la recherche que je tiens à les remercier : ils sont aussi pour beaucoup dans le plaisir que j'ai aujourd'hui à enseigner.

Si ma thèse s'est faite dans d'aussi bonnes conditions, c'est également grâce au soutien et à la convivialité de tous les membres de l'équipe RPDMP et plus largement du troisième étage et du bâtiment U4. Je les en remercie très chaleureusement.

Mes derniers remerciements, et non les moindres, vont à mon compagnon, à ma sœur, à mes parents et à mes amis, pour le soutien précieux qu'ils m'ont apporté.

Encore merci à tous.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction

L'Intelligence Artificielle (l'IA) est une discipline de l'informatique qui s'intéresse à l'automatisation de tâches demandant des connaissances et des processus de raisonnement complexes. Les problématiques de l'IA concernent, entre autres, la représentation des connaissances, leur acquisition, la formalisation et la mécanisation de différents types de raisonnement, l'aide à la décision collective.

L'IA étudie ces problèmes sous deux grands aspects : un aspect modélisation et un aspect calculatoire. Les problèmes peuvent être étudiés à différents niveaux d'abstraction. Lorsqu'un niveau très abstrait est choisi, c'est pour établir des résultats très généraux qui pourront s'appliquer aux instances les plus concrètes du problème.

L'*argumentation* est étudiée en IA en tant que mode de raisonnement qui consiste à construire et à évaluer des arguments, le plus souvent conflictuels. Ce mode de raisonnement s'applique notamment au traitement de connaissances incohérentes et à la prise de décision sous incertitude. Il est aussi utilisé dans les processus dialectiques mis en œuvre dans les discussions entre agents.

L'argumentation est également étudiée dans d'autres disciplines comme la langue et la cognition : psychologie, linguistique ou philosophie sont concernées par ce sujet.

Dans ce mémoire, c'est d'un point de vue formel, en lien avec les problématiques de l'IA, que nous allons étudier l'argumentation.

Les arguments et les conflits entre arguments mis en jeu dans le processus d'argumentation peuvent être représentés dans ce que l'on appelle un *système d'argumentation*. Celui proposé par Dung dans [Dun95] est très abstrait et permet de saisir les caractéristiques de bon nombre d'autres systèmes d'argumentation, mais également de systèmes logiques utilisés pour formaliser le raisonnement de sens commun. Pour cette raison, c'est ce système que nous avons choisi comme cadre de notre étude.

Le but du processus d'argumentation est d'évaluer les arguments afin de déterminer les plus acceptables. Il existe plusieurs manières d'évaluer les arguments ; chacune de ces manières définit une sémantique. L'acceptabilité peut être conjointe, ce sont alors des ensembles d'arguments qui sont acceptables ; ces ensembles doivent généralement être sans conflit, et le plus souvent maximaux.

À une sémantique donnée peuvent être associés des problèmes de décision. Le *problème de décision crédule* consiste à déterminer si un argument appartient à au moins un ensemble d'arguments acceptables sous une sémantique donnée ; le *problème de décision sceptique* consiste à déterminer si un argument appartient à tous les ensembles d'arguments acceptables sous la sémantique.

La sémantique probablement la plus répandue est la sémantique stable, c'est aussi l'une des plus restrictives. Sa définition très intuitive donne des résultats problématiques pour plusieurs systèmes d'argumentation ; en particulier, il existe des systèmes d'argumentation qui ne possèdent pas d'ensembles d'arguments acceptables sous cette sémantique. Dung a proposé dans [Dun95] une autre sémantique, la *sémantique préférée*, qui permet d'éviter plusieurs des inconvénients de la sémantique stable. Alors que la sémantique stable a largement été étudiée, la sémantique préferée a reçu moins d'attention.

Dans ce mémoire, nous allons étudier la sémantique préférée sous deux aspects : un aspect preuves et algorithmes, et un aspect modélisation.

D'un point de vue preuves et algorithmes, nous allons nous intéresser non seulement au *calcul* des ensembles d'arguments acceptables sous la sémantique préférée, mais également aux problèmes de décision crédule et sceptique sous cette sémantique. Une caractéristique des problèmes de décision étudiés en Intelligence Artificielle est que bien souvent une réponse «oui» ou «non» n'est pas suffisante : on attend en général d'un système intelligent qu'il soit capable d'expliquer ses réponses. En argumentation, une façon élégante d'expliquer les réponses est de les fournir sous forme d'un dialogue entre deux agents ; de telles *preuves* de l'acceptabilité d'un argument sont dites *dialectiques*.

D'un point de vue modélisation, on peut remarquer que généralement dans les systèmes d'argumentation, aucune distinction n'est faite entre les arguments, dans le sens qu'il n'y a aucune restriction sur leur utilisation. Or, dans certains contextes, une telle restriction est nécessaire. Par exemple, dans une discussion entre deux agents où chacun essaie de convaincre l'autre de la validité de son point de vue, un agent peut hésiter à utiliser un argument particulier parce qu'il révèle des informations trop personnelles ; il préfère alors utiliser un argument moins personnel. Les arguments qui ne doivent être utilisés que si cela est absolument nécessaire, c'est-à-dire dans le but de «supporter» d'autres arguments qui seraient sinon rejetés, sont dits «restreints». Cette restriction va nous amener à enrichir le système d'argumentation de Dung et à modéliser une nouvelle sémantique pour ce système qui *raffine* la sémantique préférée : un ensemble d'arguments sera acceptable seulement s'il contient un minimum d'arguments «restreints» pour un maximum d'arguments non restreints.

Pour présenter nos travaux, nous avons choisi de découper ce mémoire en trois parties.

Dans la première partie, nous exposons le contexte des travaux en présentant plus précisément le domaine de l'argumentation (chapitre 1), le système d'argumentation de Dung et ses sémantiques (chapitre 2), ainsi que des cadres formels liés au système de Dung (chapitre 3).

Dans la deuxième partie, nous étudierons les problèmes d'acceptabilité crédule et sceptique sous la sémantique préférée de manière à établir des théories de la preuve dialectiques (chapitre 4) et des algorithmes qui calculent les preuves de ces théories (chapitre 5). Nous nous intéresserons également à d'autres aspects algorithmiques de la sémantique préférée (chapitre 6).

Dans la troisième partie, nous chercherons à modéliser le raffinement du système d'argumentation de Dung et de la sémantique préférée (chapitre 7) et nous réfléchirons à une méthode de calcul de la nouvelle sémantique (chapitre 8).

Le schéma de la figure 1 donne un guide de lecture des chapitres : une flèche d'un chapitre vers un autre indique que la lecture du premier est nécessaire à la compréhension du deuxième.

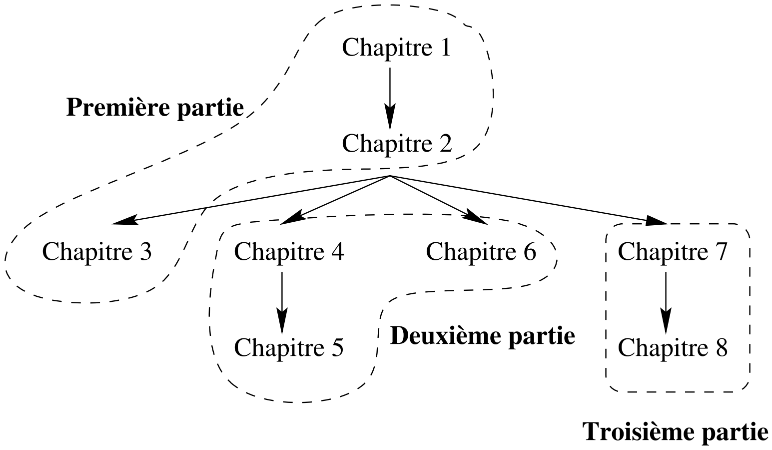


FIG. 1 – Guide de lecture.

Première partie

Cadre des travaux

Chapitre 1

L'argumentation

Chacun de nous s'est déjà retrouvé dans la situation de devoir convaincre un auditoire de la validité de son opinion, de son point de vue, lors d'un entretien d'embauche ou de la présentation d'un projet par exemple. Dans une telle situation, nous avançons des raisons expliquant notre point de vue. L'ensemble des raisons avancées constitue ce que l'on appelle un *argument* ; cet argument peut prendre différentes formes, avoir différentes forces, être plus ou moins pertinent vis-à-vis de la thèse défendue. L'auditoire étant rarement acquis à notre cause, il est fréquent qu'il oppose à nos arguments des *contre-arguments.* Ces contre-arguments peuvent venir contrarier notre thèse ou une des raisons avancées pour la soutenir. Comment déterminer alors les arguments les plus acceptables ?

Le *processus d'argumentation* est là pour répondre à cette question. Grâce à une représentation des arguments et des contrariétés entre arguments sous forme d'un système d'argumentation, le processus d'argumentation détermine l'acceptabilité des arguments.

Nous allons présenter plus en détail le processus d'argumentation, puis quelques domaines d'application de l'argumentation, avant de préciser la place de notre travail dans le domaine.

1.1 Le processus d'argumentation

1.1.1 Le système d'argumentation

De manière abstraite, les *systèmes d'argumentation* permettent de représenter les arguments et les contrariétés qui peuvent exister entre eux. Les arguments peuvent être vus comme des entités abstraites qui ont un type d'interaction particulier, l'attaque (voir le schéma de la figure 1.1 : les rectangles sont les arguments, une flèche d'un argument vers l'autre indique que le premier argument attaque le deuxième).

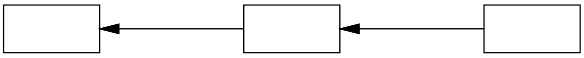


FIG. 1.1 – Représentation intuitive des arguments et des attaques entre arguments.

De manière plus concrète, la forme d'un argument peut être une preuve déductive, ou bien analogique, inductive ou abductive. Cette preuve peut prendre la forme d'un arbre d'inférence comme dans [Vre97, LS89], d'une séquence d'inférences (*i.e.* d'une déduction) comme dans [PS97, Pol87], ou d'une paire support-conclusion comme dans [SL92, EGKF93, KAEGF95, Amg99]. Le système d'argumentation se base alors sur un langage logique et une notion de conséquence associée, généralement monotone. Par exemple, dans le système de Pollock [Pol87], c'est la logique du premier ordre qui est utilisée ; dans le système [PS97], c'est un langage de programmation logique.

La structure interne d'un argument peut tout aussi bien être un ensemble de liens entre des morceaux de texte ; c'est le cas pour des applications dans les systèmes de médiation, où les arguments peuvent par exemple avoir la forme de messages électroniques écrits par des humains (système Zeno [GK97]). D'autres travaux, comme [Ver99, SWCP95], représentent les arguments sous forme de diagrammes.

La relation de contrariété, quant à elle, peut exprimer plusieurs types d'attaque comme :
- deux arguments qui soutiennent des thèses contradictoires s'attaquent mutuellement ;
- un argument soutenant une thèse qui contredit une raison qu'un deuxième argument utilise pour soutenir une autre thèse attaque ce deuxième argument.

Ces deux types d'attaque sont formalisés et étudiés séparément dans [Amg99], mais sont combinés dans [Pol87, Vre97, PS97].

La relation de contrariété peut être renforcée par la prise en compte d'une notion de préférence entre arguments. Les préférences peuvent être extraites des connaissances utilisées pour construire les arguments (grâce au principe de spécificité de [SL92] par exemple), ou bien être intégrées dans les connaissances comme dans [PS97, Dou99], ou encore être établies par une relation exogène comme dans [Amg99].

Pour illustrer ces notions, considérons un exemple très classique en Intelligence Artificielle :

Exemple 1.1 (Zoé l'autruche). Un oiseau, nommé Zoé, a été observé. L'observation permet de conclure que Zoé est une autruche. Concernant la capacité à voler de Zoé, plusieurs arguments s'affrontent :

a. Zoé est un oiseau et les oiseaux volent, donc Zoé vole.
b. Zoé est une autruche et les autruches ne volent pas, donc Zoé ne vole pas.
c. L'observation que Zoé est une autruche a été faite avec des instruments défectueux, donc on ne peut pas être sûr que Zoé est une autruche.

Les arguments a et b supportent des conclusions contradictoires, ils s'attaquent donc l'un l'autre. Néanmoins, le fait que Zoé est une autruche est plus spécifique que le fait que Zoé est un oiseau ; on dit alors que l'attaque de b sur a est plus forte que l'attaque de a sur b, et on ne retient que la première attaque. Pour ce qui est de l'argument c, il attaque l'argument b.

Étant donné ces arguments et ces contrariétés entre arguments exprimées en langage naturel, le schéma de la figure 1.1 page précédente peut être instancié comme indiqué sur la figure 1.2. ♢

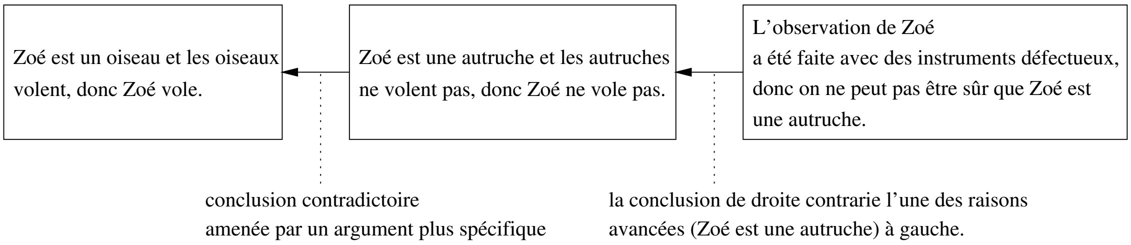


Fig. 1.2 – Instanciation du schéma de la figure 1.1 avec l'exemple *Zoé l'autruche*.

1.1.2 L'acceptabilité des arguments

Dans les systèmes d'argumentation, il existe deux notions d'acceptabilité : une notion *individuelle* et une notion *conjointe*.

L'*acceptabilité individuelle* consiste à attribuer un niveau d'acceptabilité à chaque argument en fonction de l'existence de contrariants. Les arguments ayant le même niveau sont ensuite regroupés en classes ([EGKF93, EGH95]). On distingue généralement trois grandes classes : celle des arguments acceptés, celle des arguments rejetés et celle des arguments en suspens. Des travaux récents s'attachent à dépasser ces trois classes pour déterminer une acceptabilité graduelle ([BH01, CLS02] par exemple).

Par opposition, la notion d'*acceptabilité conjointe* (ou collective) permet de caractériser des ensembles d'arguments qui sont conjointement acceptables. De tels ensembles sont appelés des extensions. L'acceptabilité peut alors être définie de deux manières :

– soit sous une *forme déclarative*, qui permet de caractériser les extensions ; cette définition correspond à la *sémantique* du système d'argumentation ;
– soit sous une *forme procédurale*, qui permet de déterminer si un argument appartient à une extension ; cette définition correspond à la *théorie de la preuve* du système d'argumentation.

Dans un cadre abstrait, les sémantiques peuvent avoir, entre autres, les caractéristiques suivantes :

1. un argument appartient à une extension si aucun des arguments qui l'attaque n'appartient à l'extension ;
2. un argument est rejeté d'une extension s'il est attaqué par un argument de l'extension ;
3. un argument appartient à une extension si lorsqu'il est attaqué par un argument hors de l'extension, cet attaquant est à son tour attaqué par un argument de l'extension (cette notion est celle de *défense* introduite par Dung [Dun95]).

À l'aide de ces caractéristiques, on peut définir sans difficulté l'extension de systèmes simples, comme celui de l'exemple *Zoé l'autruche* :

EXEMPLE 1.1 (SUITE). c n'est attaqué par aucun argument, donc c appartient à toute extension. b est attaqué par un argument qui appartient à toute extension, il est donc rejeté de toute extension. L'attaquant b de a est attaqué par c ; on dit que c défend a. a appartient donc à toute extension.

L'appartenance d'un argument à une extension (ou son rejet de cette extension) est définie sans ambiguïté ; il y a donc une seule extension pour ce système : l'ensemble $\{a, c\}$. $\diamondsuit$

Dans des systèmes plus complexes, en particulier ceux où des arguments s'attaquent mutuellement, il peut exister plusieurs extensions. C'est le cas sur l'exemple suivant (communément appelé exemple de Nixon) :

EXEMPLE 1.2 (NIXON). On s'est longtemps interrogé sur les convictions pacifistes de Nixon. Deux arguments s'affrontent :

a. Nixon était républicain et les républicains sont connus pour ne pas être pacifistes, donc Nixon n'était pas pacifiste.

b. Nixon était quaker et les quakers sont connus pour être pacifistes, donc Nixon était pacifiste.

Les arguments a et b supportent des conclusions contradictoires et s'attaquent donc l'un l'autre. Si a appartient à une extension, b doit en être rejeté, et inversement. Il y a donc au moins deux extensions pour ce système : $\{a\}$ et $\{b\}$. $\diamondsuit$

En présence de plusieurs extensions, la notion d'acceptabilité d'un argument peut être raffinée de deux manières :

– une approche prudente (ou sceptique) vise à ne considérer comme acceptables que les arguments qui appartiennent à toutes les extensions.
– une approche plus aventureuse (ou crédule) consiste à considérer comme acceptables les arguments qui appartiennent à au moins une extension.

Notons qu'il est possible de rencontrer des arguments qui s'attaquent eux-mêmes, comme en témoigne l'exemple ci-dessous :

EXEMPLE 1.3 (LE PARADOXE DU MENTEUR). Frère Jean-Baptiste est un moine. En ce jour de pénitence, il fait une drôle de déclaration : «Je suis un menteur, dit-il». Pour soutenir le fait qu'il est effectivement un menteur, l'argument suivant peut être établi :

a. Frère Jean-Baptiste dit qu'il est un menteur. Puisque les moines sont connus pour toujours dire la vérité, ce qu'il dit est vrai, et il est donc un menteur.

Mais cet argument se contrarie lui-même, puisque la conclusion que Frère Jean-Baptiste est un menteur contredit la raison qui la supporte, qui est qu'il dit la vérité ! ♢

Selon Dung, de tels arguments n'appartiennent jamais à aucune extension. Dans cet exemple, comme dans l'exemple suivant où les arguments forment une «boucle» de longueur impaire (alors que dans l'exemple 1.2 page précédente, la «boucle» était de longueur paire), la seule extension selon Dung est l'ensemble vide :

EXEMPLE 1.4 (L'AFFAIRE DES PEINTURES). (adapté de [JV99a]) Antoine a vendu sa maison à Alice. Alice, souhaitant effectuer quelques rénovations dans la maison, a fait appel à Marc, un ouvrier du bâtiment.

Pendant les travaux, Marc a cassé un plafond et a ainsi découvert une collection de peintures authentiques d'une grande valeur. Une discussion s'est engagée entre Alice et Marc, pour savoir à qui doivent revenir ces peintures :

a. Alice avance que les peintures ont été trouvées dans sa propriété, et que donc elles lui appartiennent.

b. Marc soutient que c'est seulement parce qu'il a cassé le plafond et découvert les peintures qu'Alice connaît leur existence. C'est par son travail que Marc a trouvé les peintures, et puisque c'est lui qui les a trouvées, elles doivent lui revenir.

c. Alice rétorque que lorsque Marc a découvert les peintures, il travaillait pour elle, et que donc le fruit du travail de Marc lui appartient.

d. Marc répond à cela que les peintures ne sont pas la propriété d'Alice. Le précédent propriétaire lui a seulement vendu la maison, les biens mobiliers restent sa propriété.

Il est évident que dans cette discussion, Marc s'est contredit. En effet, il revendique par l'argument b que les peintures sont sa propriété et par l'argument d, il soutient que les peintures sont la propriété de l'ancien propriétaire. L'argument b attaque les arguments a et d, d attaque c et c attaque b : les arguments b, d et c forment une «boucle».

Dans cette situation, la seule extension est l'ensemble vide. En effet, si on considère l'un des trois arguments de la boucle, b par exemple, alors :

- si b appartient à une extension, alors les arguments qui sont attaqués par b (*i.e.* d et a) en sont rejetés. Par ailleurs, puisque b appartient à une extension et qu'il est attaqué par c, c doit être à son tour attaqué par un argument de l'extension. Donc d doit appartenir à l'extension. Contradiction.
- si b n'appartient pas à une extension donnée, alors les arguments qui sont attaqués uniquement par b (*i.e.* a et d) doivent appartenir à l'extension. Comme c est attaqué par un argument de l'extension, il en est rejeté. Mais comme c est le seul attaquant de b et que b est rejeté de l'extension, c devrait appartenir à l'extension. Contradiction.

♢

1.2 Domaines d'application

L'argumentation est un outil utile pour l'étude des problèmes suivants :

- explication dans les raisonnements,
- modélisation du raisonnement de sens commun,
- prise en compte d'incertitude, d'incohérences, échange d'informations pour une prise de décision collective dans les systèmes d'aide à la décision,
- négociation et plus généralement interactions entre agents,
- ...

L'argumentation peut donc prendre place aussi bien dans un cadre mono-agent que dans un cadre multi-agents. Ses applications correspondent à des domaines variés, comme le commerce électronique, la communication, l'enseignement, etc.

Nous allons présenter plus en détail l'utilisation de l'argumentation dans la modélisation du raisonnement révisable, et sa prise en compte dans le processus de négociation entre agents.

1.2.1 Modélisation du raisonnement de sens commun

Les raisonnements de sens commun, que l'Homme effectue au quotidien, s'accommodent d'informations incomplètes, incertaines, voire incohérentes. De tels processus de raisonnement dépassent les capacités de représentation et d'inférence de la logique classique ; cette dernière ne permet de raisonner que sur des informations complètes, cohérentes et bivalentes (certainement vraies ou certainement fausses).

Pour pallier ces limites, l'IA a formalisé différentes extensions ou affaiblissements du raisonnement déductif classique, dans le but de propager l'incertitude associée à des informations, de pouvoir produire des conclusions plausibles en cas d'informations incomplètes, ou encore d'être capable de prendre en compte des informations contradictoires.

Les conclusions plausibles établies en cas d'informations *incomplètes* sont *révisables* puisqu'elles pourront être remises en cause à l'arrivée d'une nouvelle information.

Par exemple, si dans un premier temps on a comme informations que Zoé est un oiseau et que les oiseaux volent, on pourra justifier que Zoé vole et cette conclusion sera plausible. Si dans un deuxième temps on apprend de nouvelles informations, à savoir que Zoé est une autruche et que les autruches ne volent pas, on pourra justifier une deuxième conclusion (Zoé ne vole pas) qui vient contredire la première et la remet en cause.

Les mécanismes qui permettent de prendre en compte ce type de raisonnement proposent une procédure d'inférence qui ne satisfait pas la propriété de monotonie[1] du mécanisme d'inférence classique. Ces modes de raisonnement non-classiques sont dits *non-monotones*.

Les informations *contradictoires* peuvent apparaître pour plusieurs raisons. Comme on vient de le voir, si les informations dont on dispose sont incomplètes, l'ajout d'une nouvelle information peut venir contrarier des conclusions déjà établies. Par ailleurs, les informations dont on dispose viennent souvent de sources multiples, de fiabilités inégales, et leur mise en commun conduit souvent à des incohérences.

Deux attitudes peuvent être adoptées face à des informations contradictoires : soit elles sont éliminées dans des processus de «fusion» d'informations si on souhaite restaurer la cohérence ; soit elles sont prises en compte dans des processus d'argumentation si le but est de préciser des informations, éventuellement de comprendre les intentions de l'autre ; ces situations se rencontrent souvent dans le cadre de dialogues, où chaque parti ne dispose pas des mêmes informations.

Pour saisir le raisonnement révisable, plusieurs approches formelles peuvent être utilisées : citons, entre autres, la logique des défauts de Reiter [Rei80] définie pour traiter des connaissances avec exceptions, la programmation logique avec négation par l'échec (présentée, entre autres, dans [EK89]), la circonscription de McCarthy [McC80], la logique des défauts de Poole [Poo88] basée sur le raisonnement hypothétique, la logique autoépistémique [Moo85], les logiques modales non-monotones [McD82] et plus récemment, des modèles à base d'argumentation.

Ces modèles à base d'argumentation (et plus généralement le processus d'argumentation) permettent de déterminer si une conclusion est acceptée ou rejetée sur la base des arguments pour et contre elle, et sur le fait que ces arguments peuvent être attaqués par d'autres.

Dans ces modèles, un argument prend généralement la forme d'une preuve pour une conclusion ; cette preuve n'est jamais remise en cause au cours du raisonnement. C'est l'acceptabilité des arguments qui peut être remise en cause : l'ajout d'informations peut amener la construction de nouveaux arguments qui viennent attaquer des arguments existants. C'est à cause de ces nouvelles attaques qu'un argument précédemment accepté risque d'être rejeté.

Parmi les systèmes d'argumentation formels pour le raisonnement révisable, citons le système de Pollock [Pol87], celui de Lin et Shoham [LS89], celui de Simari et Loui [SL92], celui de Dung

---

[1] La propriété de *monotonie* signifie qu'un résultat acquis n'est jamais remis en cause.

[Dun95], celui de Bondarenko, Dung, Kowalski et Toni [BDKT97], celui de Prakken et Sartor [PS97] et celui de Vreeswijk [Vre97].

Des systèmes d'argumentation formels ont également été établis pour traiter des informations *incertaines*. L'idée de base derrière l'argumentation est la possibilité d'en dire plus sur la certitude d'un fait particulier. Plutôt que de le quantifier uniquement par un nombre entre 0 et 1 (comme dans les approches probabilistes), on évalue la raison pour laquelle le fait est vrai en construisant des arguments en sa faveur. L'avantage de construire des arguments pour des faits et de les utiliser pour qualifier leur certitude est la possibilité de raisonner sur ces arguments eux-mêmes. Citons en particulier les travaux de [EGKF93] dans le domaine médical, et ceux de [EGH95].

1.2.2 Négociation dans les systèmes multi-agents

Plusieurs applications logicielles ont été conçues et implémentées en utilisant la notion d'agent autonome. Ces applications varient du commerce électronique à de larges applications industrielles. Dans la plupart de ces applications, les composantes autonomes ont besoin d'interagir. Elles ont besoin de communiquer afin de résoudre les différences d'opinions et les conflits d'intérêts, de travailler ensemble afin de résoudre des dilemmes, trouver des preuves, ou tout simplement pour échanger des informations. En particulier, elles peuvent avoir besoin de négocier.

La *négociation* est le processus par lequel des agents communiquent pour tenter d'arriver à un accord acceptable sur un problème donné (problème au sujet duquel ils avaient au préalable des opinions différentes). Les problèmes peuvent être par exemple des conflits de buts ou de ressources.

Parsons, Jennings et Sierra se sont intéressés dans [PSJ98] à l'utilisation de l'argumentation pour modéliser des dialogues de négociation entre agents. Ils ont montré qu'un système d'argumentation peut être utilisé à deux niveaux :

- au niveau d'un agent pour son propre raisonnement ;
- pour modéliser le processus de négociation.

Pour comprendre ce que peut être un dialogue de négociation qui utilise l'argumentation, prenons un exemple. Supposons que deux agents A_1 et A_2 dialoguent. A_1 propose un argument pour un but. A_2 peut être en désaccord avec cet argument si :

- le but soutenu par l'argument de A_1 est en conflit avec l'un des buts de A_2 ; dans ce cas, A_2 peut faire une contre-proposition en avançant un argument pour son but attaqué.
- un des éléments du support de l'argument proposé par A_1 est en conflit avec un but ou avec une croyance de A_2 ; dans ce cas, A_2 retourne un attaquant et A_1 peut tenter de proposer un autre argument pour le but qu'il tente de soutenir.

Parsons, Sierra et Jennings concluent dans [PSJ98] que l'argumentation offre un cadre suffisamment large pour permettre la modélisation de différents types de dialogues (en particulier, ceux identifiés par Walton et Krabbe dans [WK95]). C'est cette modélisation qu'ont effectuée Amgoud, Maudet et Parsons dans [AMP00].

1.3 Buts de l'étude

Des modèles formels d'argumentation ont émergé ces dernières années (voir [PV02]) et les aspects calculatoires de l'argumentation trouvent un intérêt croissant. Cette thèse contribue à l'étude *algorithmique* de l'argumentation, ainsi qu'à la *modélisation* d'une nouvelle sémantique pour l'acceptabilité des arguments.

Le système d'argumentation de Dung, que nous allons présenter dans le chapitre suivant, se place à un niveau d'abstraction très intéressant. En effet, du fait que les arguments et la relation de contrariété y sont non spécifiés, ce système permet de saisir les caractéristiques de bon nombre d'autres systèmes d'argumentation. De plus, il est aisé de faire le lien entre ce système et la théorie des graphes, et d'utiliser ainsi des résultats établis en théorie des graphes pour résoudre des problèmes relatifs à ce système.

Les problèmes que nous allons étudier dans ce mémoire concernent une sémantique particulière, la sémantique préférée de Dung [Dun95]. Nous comparerons dans le chapitre suivant cette sémantique à d'autres, et nous justifierons ainsi ce choix d'étude.

Chapitre 2

Le système d'argumentation de Dung

Le système d'argumentation de Dung [Dun95] est un système abstrait, dans lequel les arguments et les conflits entre arguments sont des primitives. Ainsi, ce système permet de se concentrer sur les interactions entre arguments.

Nous rappelons tout d'abord la définition formelle du système de Dung et de ses sémantiques. Nous faisons ensuite le lien avec des notions de la théorie des graphes, puis nous étudions quelques propriétés intéressantes. Nous décrivons enfin des problèmes d'acceptabilité relatifs aux sémantiques, avant de donner quelques résultats de complexité. Dans ce mémoire, nous allons porter notre attention sur la sémantique préférée. Quelques critiques sur cette sémantique seront formulées à la fin du chapitre.

2.1 Définition et sémantiques

La définition formelle du système d'argumentation de Dung est la suivante :

Définition 2.1 (Système d'argumentation [Dun95]).
Un système d'argumentation *est une paire* (A, R) *où* A *est un ensemble d'*arguments *et* R *est une relation binaire sur* A *qui représente une notion de* contrariété *ou d'*attaque *entre arguments* $(R \subseteq A \times A)$*. Étant donné deux arguments* a *et* b*,* $(a, b) \in R$ *ou* aRb *signifie que* a attaque b*, ou encore que* b est attaqué par a*.*

Si dans un système d'argumentation, on trouve un argument a tel que a attaque a, on dit que *a s'auto-attaque*.

Notation 2.1.

- A^* *dénote l'ensemble des séquences finies d'éléments de* A *;*
- 2^A *dénote l'ensemble de tous les sous-ensembles de* A *;*
- *l'ensemble des arguments qui s'auto-attaquent dans un système d'argumentation* (A, R) *est dénoté par* $\mathrm{Refl}(A, R)$*, ou simplement* Refl *lorsqu'il n'y a pas d'ambiguïté sur* (A, R)*.*

Définition 2.2 (Système d'argumentation fini).
Un système d'argumentation (A, R) *est dit* fini *si* R *est fini.*

Cette définition est différente de celle de Dung [Dun95] pour qui un système est fini si tout argument du système est attaqué par un nombre fini d'arguments du système. Notre définition est plus restrictive. Lorsque la définition de Dung peut être utilisée, nous le préciserons.

Exemple 2.1. Le système d'argumentation (A, R) qui correspond à :

- l'exemple 1.1 (*Zoé l'autruche*) est tel que :

  $A = \{a, b, c\}$,
  $R = \{(c, b), (b, a)\}$;
- l'exemple 1.2 (*Nixon*) est tel que :
  $A = \{a, b\}$,
  $R = \{(a, b), (b, a)\}$;
- l'exemple 1.3 (*Le paradoxe du menteur*) est tel que :
  $A = \{a\}$,
  $R = \{(a, a)\}$ (l'argument a s'auto-attaque) ;
- l'exemple 1.4 (*L'affaire des peintures*) est tel que :
  $A = \{a, b, c, d\}$,
  $R = \{(b, a), (b, d), (d, c), (c, b)\}$.

Ces quatre systèmes d'argumentation sont finis. ♢

L'acceptabilité dans les sémantiques de Dung est basée sur la notion de défense (présentée en section 1.1.2). L'acceptabilité est collective, ce sont des ensembles d'arguments qui sont acceptés. Ces ensembles ne doivent généralement pas s'attaquer eux-mêmes. Ces notions ont été formalisées comme suit :

Définition 2.3 (Ensemble sans-conflit, défense collective, défense individuelle [Dun95]). *Soit (A, R) un système d'argumentation, $S \subseteq A$ un ensemble d'arguments et $a \in A$ un argument :*
- *S est* sans-conflit *si et seulement si $\nexists a, b \in S$ tels que aRb.*
- *S* défend collectivement *a si et seulement si $\forall b \in A$, si $bRa, \exists c \in S$ tel que cRb*[1]*. S* défend tous ses éléments *si et seulement si S défend collectivement tout élément de S.*
- *$b \in A$ est un* défenseur individuel *de a si et seulement s'il existe une séquence finie $x_0, \ldots, x_{2n}$ d'eléments de A telle que :*
  1. *$a = x_0$, $b = x_{2n}$, $n \neq 0$ et*
  2. *$\forall i, 0 \leq i \leq (2n-1), x_{i+1} R x_i$.*

  Si $n = 1$, b est un défenseur individuel direct *de a, sinon c'est un* défenseur individuel indirect *de a.*

Exemple 2.1 (suite). Considérons le système d'argumentation *Zoé l'autruche* ; l'ensemble $\{a, c\}$ est sans-conflit, l'ensemble $\{a, b\}$ ne l'est pas. L'argument c est un défenseur individuel direct de l'argument a. L'ensemble $\{a, c\}$ défend collectivement a et c, $\{a, c\}$ défend donc tous ses éléments. ♢

Nous ajoutons quelques définitions supplémentaires (l'attaque individuelle est inspirée par [Dun95]) :

Définition 2.4 (Ensembles en conflit, attaque collective, attaque individuelle). *Soit (A, R) un système d'argumentation, $S \subseteq A$ un ensemble d'arguments et $a \in A$ un argument :*
- *S est* en conflit *avec un ensemble d'arguments $S' \subseteq A$ si et seulement si $\exists s \in S$, $\exists s' \in S'$ tels que sRs' ou $s'Rs$.*
- *S* attaque collectivement *a (ou S attaque a) si et seulement si $\exists b \in S$ tel que bRa.*
- *$b \in A$ est un* attaquant individuel *de a si et seulement si il existe une séquence finie $x_0, \ldots, x_{2n+1}$ d'éléments de A telle que :*
  1. *$a = x_0$ et $b = x_{2n+1}$, et*
  2. *$\forall i, 0 \leq i \leq 2n, x_{i+1} R x_i$.*

  Si $n = 0$, b est un attaquant individuel direct *de a (ou simplement, b est un* attaquant *de a), sinon c'est un* attaquant individuel indirect *de a.*

Exemple 2.1 (suite). Considérons le système d'argumentation *Zoé l'autruche*. Les ensembles $\{a, c\}$ et $\{b\}$ sont en conflit. L'ensemble $\{b\}$ attaque l'argument a, b est un attaquant de a. ♢

---

[1] Ce que nous appelons ici *défense collective* est ce qui est appelé par Dung dans [Dun95] *acceptabilité d'un argument par rapport à un ensemble d'arguments.*

Dung a défini plusieurs sémantiques de l'acceptabilité pour son système. En voici trois qui mènent généralement à plusieurs extensions :

Définition 2.5 (Extension stable, ensemble admissible, extension préférée [Dun95]). *Soit* (A, R) *un système d'argumentation et* $S \subseteq A$ *un ensemble d'arguments :*
- S *est une* extension stable *si et seulement si* S *est sans-conflit et* S *attaque tout argument de* A *qui n'appartient pas à* S*.*
- S *est un* ensemble admissible *si et seulement si* S *est sans-conflit et* S *défend tous ses éléments.*
- S *est une* extension préférée *si et seulement si* S *est maximal pour l'inclusion ensembliste parmi les ensembles admissibles.*

Dans toute la suite, la *sémantique stable* fera référence à l'ensemble des extensions stables, la *sémantique admissible* à l'ensemble des ensembles admissibles, la *sémantique préférée* à l'ensemble des extensions préférées.

Notation 2.2. *Étant donné un système d'argumentation* (A, R) *:*
- *l'ensemble des ensembles admissibles de* (A, R) *est dénoté par* $\mathrm{Adm}(A, R)$ *;*
- *l'ensemble des extensions préférées de* (A, R) *est dénoté par* $\mathrm{Pref}(A, R)$*.*

Plusieurs logiques pour le raisonnement par défaut donnent à l'acceptabilité la sémantique stable. Une extension stable reflète d'une certaine manière un point de vue maximal cohérent : tout argument qui n'est pas accepté l'est parce qu'il est attaqué par un argument accepté. Cette définition amène plusieurs inconvénients.

Tout d'abord, étant donné une extension stable, tout argument doit soit appartenir à l'extension, soit être attaqué par elle. Ceci implique que pour déterminer si un argument donné appartient à une extension stable, il faut calculer une extension stable contenant l'argument et donc explorer tout le système d'argumentation pour décider de l'acceptabilité de tous les arguments du système. Ceci est difficilement calculable dans beaucoup de cas.

De plus, certains systèmes ne possèdent pas d'extension stable. C'est souvent le cas lorsque le système contient un argument qui s'auto-attaque. Intuitivement, si un tel argument n'attaque et n'est attaqué par aucun autre argument, il n'influe en rien sur l'acceptabilité des autres arguments ; pourtant, aucun des autres arguments du système n'est alors acceptable.

Ces inconvénients de la sémantique stable ont motivé la définition de la sémantique admissible.

Un ensemble admissible reflète le fait qu'un agent rationnel est capable de défendre tous les arguments qu'il accepte. Pour déterminer si un argument appartient à au moins un ensemble admissible, il suffit de s'intéresser aux arguments qui l'attaquent directement ou indirectement. La recherche d'un ensemble admissible est donc locale, seule une partie du système d'argumentation est explorée.

De plus, il existe toujours au moins un ensemble admissible : l'ensemble vide.

Une extension préférée est un ensemble maximal d'arguments qu'un agent rationnel est capable de défendre. La sémantique préférée distingue donc les ensembles admissibles maximaux.

Dung a montré que toute extension stable est une extension préférée. Cette propriété ainsi que de nouvelles propriétés sont résumées ci-dessous :

Propriété 2.1. *[Dun95]*

1. *Tout ensemble admissible est contenu dans une extension préférée.*
2. *Tout système d'argumentation possède au moins une extension préférée.*
3. *Toute extension stable est également préférée.*

Les preuves de ces propriétés se trouvent dans [Dun95].

Exemple 2.1 (extensions préférées et stables). Les extensions préférées et stables des systèmes d'argumentation de l'exemple sont les suivantes :

- *Zoé l'autruche* : $\{a, c\}$ est la seule extension préférée ; elle est stable. Les autres ensembles admissibles sont $\emptyset$ et $\{c\}$.
- *Nixon* : $\{a\}$ et $\{b\}$ sont les deux extensions préférées du système ; elles sont également stables. Le seul autre ensemble admissible du système est l'ensemble vide.
- *Le paradoxe du menteur* : $\emptyset$ est la seule extension préférée, il n'y a pas d'extension stable.
- *L'affaire des peintures* : $\emptyset$ est la seule extension préférée, il n'y a pas d'extension stable.

$\diamondsuit$

Dung a défini une autre sémantique susceptible de mener à plusieurs extensions : la *sémantique complète*. La notion d'extension complète capture le fait qu'un agent rationnel accepte tous les arguments qu'il peut défendre. Alors qu'un ensemble admissible est *contenu* dans l'ensemble des arguments qu'il défend, une extension complète est *identique* à l'ensemble des arguments qu'elle défend.

Définition 2.6 (Extension complète [Dun95]).
Un ensemble admissible d'arguments S est une extension complète *si et seulement si tout argument qui est défendu collectivement par S appartient à S.*

En d'autres termes, un ensemble sans-conflit d'arguments S est une extension complète *si et seulement si S est un point fixe*[2] *de l'application $\mathcal{F}$ définie comme suit :*

$$\begin{array}{lcl} \mathcal{F} & : & 2^A \rightarrow 2^A \\ \mathcal{F}(S) & = & \{a \mid a \textit{ est défendu collectivement par } S\} \end{array}$$

Il est montré dans [Dun95] que l'application $\mathcal{F}$ est monotone.

Propriété 2.2. *[Dun95]*

1. *Toute extension préférée est une extension complète et la réciproque est fausse.*
2. *Tout système d'argumentation possède au moins une extension complète.*

EXEMPLE 2.1 (EXTENSIONS COMPLÈTES). Les extensions complètes des systèmes d'argumentation sont les suivantes :

- *Zoé l'autruche* : $\{a, c\}$ est la seule extension complète. Elle est identique à l'extension préférée et à l'extension stable du système.
- *Nixon* : $\emptyset$, $\{a\}$ et $\{b\}$ sont les trois extensions complètes du systèmes. On peut remarquer que l'ensemble $\emptyset$ n'est pas une extension préférée, au contraire des deux autres ensembles.
- *Le paradoxe du menteur* : l'ensemble $\emptyset$ est la seule extension complète.
- *L'affaire des peintures* : l'ensemble $\emptyset$ est la seule extension complète. On peut remarquer que l'ensemble $\{a, b, c, d\}$ est un point fixe de $\mathcal{F}$, mais il n'est pas sans-conflit : ce n'est donc pas une extension complète.

$\diamondsuit$

Le schéma de la figure 2.1 page ci-contre résume les inclusions entre les différentes sémantiques présentées jusqu'ici.

Dung a également défini une sémantique qui ne produit qu'un seul ensemble d'arguments acceptables ; c'est la *sémantique basique* (qui ne comprend qu'une seule extension basique). Elle correspond à la plus petite extension complète.

Intuitivement, si un conflit entre deux arguments ne peut être résolu (comme par exemple dans le système *Nixon*, où a attaque b et b attaque a), la sémantique basique laisse les deux arguments hors de l'extension. Dans les autres sémantiques, plusieurs extensions sont générées, et chaque argument est tantôt accepté, tantôt rejeté (par exemple, dans le système *Nixon*, $\{a\}$ et $\{b\}$ sont les deux extensions stables, préférées et complètes).

L'extension basique est caractérisée comme suit :

---

[2] Un *point fixe* de $\mathcal{F}$ est un élément S de 2^A tel que $\mathcal{F}(S) = S$.

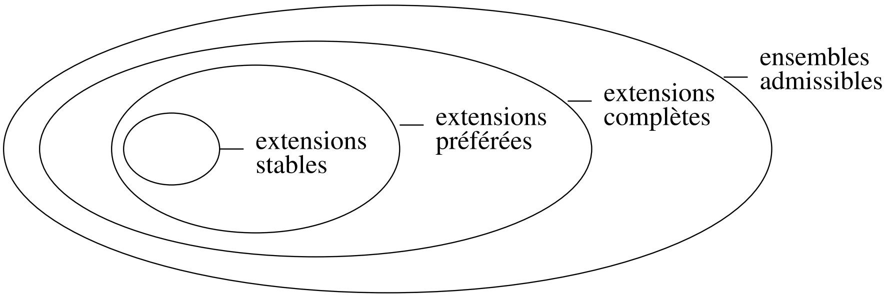


Fig. 2.1 – Inclusion entre les sémantiques stable, admissible, préférée et complète

Définition 2.7 (Extension basique [Dun95]).
*L'*extension basique *d'un système d'argumentation* (A, R) *est le plus petit point fixe de la fonction* $\mathcal{F}$*. Ce plus petit point fixe coïncide avec l'intersection des extensions complètes.*

Dung a montré dans [Dun95] que dans le cas particulier où le système d'argumentation est fini[3], $\mathcal{F}$ est continue et donc le plus petit point fixe de $\mathcal{F}$ peut être calculé en appliquant itérativement la fonction $\mathcal{F}$ à l'ensemble vide :

Propriété 2.3. *[Dun95] Si* (A, R) *est un système d'argumentation fini*[4]*, le plus petit point fixe de* $\mathcal{F}$ *est égal à* $\bigcup_{i \geq 1} \mathcal{F}^i(\emptyset)$.

Exemple 2.1 (extension basique). L'extension basique pour chacun des systèmes d'argumentation est la suivante :

- *Zoé l'autruche* :
$$\begin{array}{lcll} \mathcal{F}(\emptyset) & = & \{a\} & \text{car } a \text{ n'est pas attaqué ;} \\ \mathcal{F}(\{a\}) & = & \{a, c\} & \text{car } c \text{ est défendu par } a \text{ ;} \\ \mathcal{F}(\{a, c\}) & = & \{a, c\}. & \end{array}$$
  L'extension basique de ce système est donc $\{a, c\}$, elle est identique à l'extension préférée, à l'extension stable et à l'extension complète.
- *Nixon* : $\mathcal{F}(\emptyset) = \emptyset$.
  En effet, ni a, ni b, ne sont défendus par l'ensemble vide : a est défendu par $\{a\}$ et b est défendu par $\{b\}$. L'extension basique est donc $\emptyset$. Elle correspond à l'intersection des extensions préférées.
- *Le paradoxe du menteur* : l'extension basique est l'ensemble vide.
- *L'affaire des peintures* : l'extension basique est l'ensemble vide.

$\diamondsuit$

L'extension basique et les autres types d'extensions sont liées par les propriétés suivantes :

Propriété 2.4. *[Dun95, BDKT97]*

1. *Toute extension préférée (resp. stable, complète) contient l'extension basique.*
2. *L'extension basique d'un système d'argumentation est contenue dans l'intersection des extensions préférées du système.*

Notons qu'un ensemble admissible ne contient pas forcément l'extension basique. De plus, l'extension basique et l'intersection des extensions préférées ne coïncident généralement pas (contrairement à ce qui peut être suggéré dans les exemples ci-dessus). Le système d'argumentation (A, R) tel que :

$A = \{a, b, c, d, e\}$,

---

[3] *au sens de Dung*

[4] *au sens de Dung*

$R = \{(d,c),(c,d),(d,b),(c,b),(b,a),(b,e),(e,e)\}$;
en est un contre-exemple : ses extensions préférées sont $\{d,a\}$ et $\{c,a\}$, leur intersection est $\{a\}$, et l'extension basique est $\emptyset$.

2.2 Expression en théorie des graphes

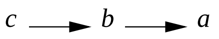
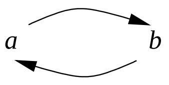
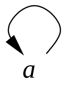
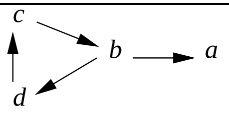
Dans cette section, nous allons établir le lien entre le système d'argumentation de Dung, certaines de ses sémantiques, et des notions de la théorie des graphes. C'est ce lien qui nous permettra d'étudier les aspects algorithmiques de ces sémantiques dans les chapitres suivants. Le lien entre la théorie des graphes et des formalismes qui saisissent le raisonnement révisable (logique des défauts, programmation logique avec négation par l'échec...) a déjà été établi et utilisé dans plusieurs travaux ([DT96, DMP97, DM94, CL91]) ; nous présenterons ces travaux et le lien qui peut être fait avec le travail de Dung dans le chapitre 3.

D'un point de vue mathématique, le système d'argumentation de Dung est un graphe orienté. Plusieurs notions définies par Dung trouvent une correspondance immédiate avec des notions de la théorie des graphes. Cette correspondance a déjà été implicitement utilisée dans plusieurs travaux ([DBC01b, DBC01a, DBC02] par exemple), mais n'a jamais clairement été établie. C'est ce que nous faisons dans cette section. Pour cela, nous rappelons quelques définitions, notations et propriétés de la théorie des graphes (voir [Ber73, DT96]).

Définition 2.8 (Graphe orienté).
Un graphe orienté est une paire (V,E) *où* V *est un ensemble dont les éléments sont appelés des* sommets *et* $E \subseteq V \times V$ *est un ensemble dont les éléments sont appelés des* arcs. *Étant donné un arc* $(v,v') \in E$, v *est appelé* sommet origine *de* (v,v') *et* v' *est appelé* sommet but *de* (v,v'). *Un arc* $(v,v') \in E$ *est dit* symétrique *si l'arc* $(v',v) \in E$.

Un système d'argumentation (A,R) est donc un graphe orienté dont les sommets sont les arguments de A et les arcs correspondent aux éléments de la relation de contrariété R.

EXEMPLE 2.1 (REPRÉSENTATION GRAPHIQUE). Les systèmes d'argumentation sont représentés sous forme de graphe dans le tableau 2.1. ◊

Zoé l'autruche	Nixon	Le paradoxe du menteur	L'affaire des peintures
			

TAB. 2.1 – Représentation sous forme de graphe des systèmes d'argumentation de l'exemple 2.1

Définition 2.9 (Chemin).
Un chemin *est une suite finie* $e_1, e_2, \ldots, e_n$ *d'arcs tels que* $\forall i, 1 \leq i \leq n-1$, *le sommet but de* e_i *est le sommet origine de* e_{i+1}. *Le sommet origine de* e_1 *est appelé* origine du chemin, *le sommet but de* e_n *est appelé* but du chemin. *La* longueur d'un chemin *est égale au nombre d'arcs qui composent le chemin.*

EXEMPLE 2.1 (SUITE). Étant donné le système d'argumentation *Zoé l'autruche*, $(c,b)(b,a)$ est un chemin d'origine c et de but a. Étant donné le système d'argumentation *Nixon*, $(a,b)(b,a)$ est un chemin d'origine et de but a. ◊

La défense individuelle (définition 2.3) et l'attaque individuelle (définition 2.4) peuvent être reformulées de manière immédiate en termes de chemins :

Propriété 2.5 (Défense individuelle, attaque individuelle). *Soit* (A, R) *un système d'argumentation,* a *et* b *deux arguments de* A*.* b défend individuellement a *s'il existe un chemin d'origine* b *et de but* a *de longueur paire.* b attaque individuellement a *s'il existe un chemin d'origine* b *et de but* a *de longueur impaire.*

Il existe des chemins particuliers qui correspondent aux notions de «boucle» et d'attaque symétrique introduites de manière informelle en section 1.1.2. Ces chemins sont ceux qui sont fermés, ils sont appelés des cycles[5].

Définition 2.10 (Cycle).
Un cycle *est un chemin dont l'origine et le but sont les mêmes. La* longueur d'un cycle *est égale au nombre d'arcs qui composent le cycle. Un cycle est* simple *s'il ne contient pas deux fois le même arc. Un cycle est* élémentaire *si deux arcs distincts du cycle n'ont pas le même sommet origine*[6]*.*

Exemple 2.1 (suite). Le système d'argumentation *Le paradoxe du menteur* (voir tableau 2.1) comprend un seul argument qui s'auto-attaque : il forme un cycle de longueur 1.

Le cycle $(a, b)(b, a)$ de longueur 2 du système d'argumentation *Nixon* est simple et élémentaire. Le cycle $(a, b)(b, a)(a, b)(b, a)$ de ce même système n'est ni simple, ni élémentaire.

Le système *L'affaire des peintures* comprend un cycle simple et élémentaire de longueur 3 : $(b, d)(d, c)(c, b)$. ♢

Les notations suivantes sont directement issues de la théorie des graphes.

Notation 2.3. *Soit* (V, E) *un graphe orienté et* $v \in V$ *un sommet :*

- $R^+(v) = \{v' \in V \mid (v, v') \in E\}$ *dénote l'ensemble des* successeurs *de* v *;*
- $R^-(v) = \{v' \in V \mid (v', v) \in E\}$ *dénote l'ensemble des* prédécesseurs *de* v *;*
- $R^\pm(v)$ *dénote l'ensemble* $R^+(v) \cup R^-(v)$*.*

Étant donné un ensemble $V' \subseteq V$ *et* $\varepsilon \in \{+, -, \pm\}$, $R^\varepsilon(V') = \bigcup_{v' \in V'} R^\varepsilon(v')$*.*

Dans un système d'argumentation (A, R), la notion de successeur d'un argument a correspond à celle d'argument *attaqué par* a, celle de prédécesseur à celle d'*attaquant (individuel direct)* de a.

Définition 2.11 (Indépendance, dominance, semi-dominance).
Soit (V, E) *un graphe orienté et* $V' \subseteq V$*.* V' *est dit :*

- indépendant *s'il n'existe pas d'arc entre les éléments de* V'*,* i.e. $E \cap (V' \times V') = \emptyset$*, ou* $R^+(V') \cap V' = \emptyset$*, ou encore* $R^-(V') \cap V' = \emptyset$ *;*
- dominant *si tout sommet de* $V \setminus V'$ *possède un prédécesseur dans* V'*,* i.e. *si* $\forall v \in V \setminus V'$*,* $\exists v' \in V'$ *tel que* $(v', v) \in E$*,* i.e. $(V \setminus V') \subseteq R^+(V')$ *;*
- semi-dominant *si tout sommet de* $V \setminus V'$ *qui possède un successeur dans* V'*, possède aussi un prédécesseur dans* V'*,* i.e. $\forall v \in V \setminus V'$ *tel que* $(v, v') \in E$ *avec* $v' \in V'$*, il existe* $v'' \in V'$ *tel que* $(v'', v) \in E$*,* i.e. $R^-(V') \subseteq R^+(V')$*.*

Définition 2.12 (Noyau, semi-noyau, semi-noyau maximal [DT96]).
Soit (V, E) *un graphe orienté et* $V' \subseteq V$*.* V' *est un :*

- noyau *si* V' *est indépendant et dominant ;*
- semi-noyau *si* V' *est indépendant et semi-dominant*[7] *;*
- semi-noyau maximal *si* V' *est un semi-noyau et si* $\nexists V'' \subseteq V$ *tel que* $V' \subseteq V''$*,* $V' \neq \emptyset$ *et* V'' *est un semi-noyau.*

Nous caractérisons les notions d'extension stable, d'ensemble admissible et d'extension préférée en termes de théorie des graphes :

---

[5] C'est le terme *circuit* qui est employé dans la littérature française pour désigner un tel chemin dans un graphe orienté ; dans un graphe non-orienté, c'est le terme *cycle* qui est utilisé. Dans la littérature anglaise, les terminologies sont inversées. Par abus, nous adoptons à notre tour la terminologie anglaise.

[6] Dans la littérature anglaise, ce qui est appelé ici cycle *élémentaire* est appelé cycle *simple.*

[7] Dans [Ber73], les terminologies *noyau* et *semi-noyau* sont inversées.

Propriété 2.6 (Extension stable, ensemble admissible, extension préférée). *Soit* (A, R) *un système d'argumentation et* $S \subseteq A$ *un ensemble d'arguments :*

- S *est une* extension stable *si et seulement si* S *satisfait les deux conditions suivantes :*
  1. $R^+(S) \cap S = \emptyset$ *(*S *est sans-conflit) ;*
  2. $A \setminus S \subseteq R^+(S)$ *(*S *attaque tout argument hors de* S*) ;*
- S *est un* ensemble admissible *si et seulement si* S *satisfait les deux conditions suivantes :*
  1. $R^+(S) \cap S = \emptyset$ *(*S *est sans-conflit) ;*
  2. $R^-(S) \subseteq R^+(S)$ *(*S *défend tous ses éléments) ;*
- S *est une* extension préférée *si et seulement si* S *est un ensemble admissible qui satisfait la condition suivante :*

  pour tout ensemble non vide $X \subseteq A \setminus S, X \cap R^+(S \cup X) \neq \emptyset$ *ou* $R^-(X) \not\subseteq R^+(S \cup X)$.

Preuve – La caractérisation des extensions stables et des ensembles admissibles est une traduction immédiate de leur définition en théorie des graphes. La propriété de maximalité qui caractérise les extensions préférées est moins évidente.

Pour la vérifier, considérons $S \subseteq A$ un ensemble admissible. S est une extension préférée si pour tout $X \subseteq A \setminus S$ tel que $X \neq \emptyset$, $S \cup X$ n'est pas admissible, autrement dit si :

$$R^+(S \cup X) \cap (S \cup X) \neq \emptyset \text{ ou } R^-(S \cup X) \not\subseteq R^+(S \cup X). \tag{2.1}$$

Puisque $S \cap R^+(S) = \emptyset$ et $R^-(S) \subseteq R^+(S)$, nous avons les équivalences suivantes :

$$\begin{array}{cc} & R^+(S \cup X) \cap (S \cup X) \neq \emptyset \\ \Leftrightarrow & R^+(X) \cap X \neq \emptyset \text{ ou } R^+(S) \cap X \neq \emptyset \text{ ou } R^+(X) \cap S \neq \emptyset \\ \Leftrightarrow & R^+(X) \cap X \neq \emptyset \text{ ou } R^+(S) \cap X \neq \emptyset \text{ ou } R^-(S) \cap X \neq \emptyset \\ \Leftrightarrow & R^+(X) \cap X \neq \emptyset \text{ ou } R^+(S) \cap X \neq \emptyset \\ \Leftrightarrow & R^+(X \cup S) \cap X \neq \emptyset \end{array}$$

De plus, puisque $R^-(S) \subseteq R^+(S)$, $R^-(S \cup X) \not\subseteq R^+(S \cup X)$ si et seulement si $R^-(X) \not\subseteq R^+(S \cup X)$. □

De cette caractérisation découlent les équivalences suivantes :

Corollaire 2.1. *Les notions de noyau, semi-noyau et semi-noyau maximal sont respectivement équivalentes aux notions d'extension stable, d'ensemble admissible et d'extension préférée.*

On peut montrer de manière immédiate que tout noyau est aussi un semi-noyau maximal. Ceci correspond au point 3 de la propriété 2.1 établie par Dung [Dun95].

Dans toute la suite, nous utiliserons indifféremment les termes de la théorie des graphes ou ceux de Dung pour travailler avec les systèmes d'argumentation.

2.3 Propriétés

Certains des problèmes que nous allons étudier (en particulier le problème d'acceptabilité sceptique sous la sémantique préférée que nous présenterons en section 2.4) sont difficiles à résoudre, mais sont plus faciles lorsque les systèmes d'argumentation satisfont des propriétés particulières.

Ces propriétés sont la cohérence [Dun95], l'unicité de l'extension préférée et la coïncidence de l'extension basique et de l'intersection des extensions préférées (système relativement basique selon [Dun95]). Nous donnons leurs définitions et nous montrons leurs caractéristiques graphiques (présence de cycles de longueur paire, impaire).

2.3.1 Cohérence

Nous verrons que certains problèmes qui sont difficiles à résoudre sous la sémantique préférée peuvent s'avérer plus simples à résoudre sous la sémantique stable. La notion de cohérence définie par Dung [Dun95] saisit la coïncidence entre les deux sémantiques :

Définition 2.13 (Système cohérent [Dun95]).
Un système d'argumentation (A, R) *est dit* cohérent *si toute extension préférée de* (A, R) *est stable.*

EXEMPLE 2.1 (SUITE). Les systèmes d'argumentation *Zoé l'autruche* et *Nixon* sont cohérents. ♢

On peut identifier une classe de systèmes cohérents : ceux qui ne possèdent pas de cycle de longueur impaire. Cette classe comprend les systèmes sans cycle, identifiés par Dung sous le nom de systèmes bien-fondés :

Définition 2.14 (Système bien-fondé [Dun95]).
Un système d'argumentation (A, R) *est dit* bien-fondé *si et seulement si il n'existe pas de séquence infinie* $a_0, a_1, \ldots, a_n, \ldots$ *telle que* $\forall i, a_i \in A$ *et* $a_{i+1} R a_i$ *; si* (A, R) *est fini,* (A, R) *est bien-fondé si et seulement si il ne contient pas de cycle.*

EXEMPLE 2.1 (COHÉRENCE). Des quatre systèmes d'argumentation indiqués dans l'exemple, seul *Zoé l'autruche* est bien-fondé. ♢

Une classe plus large de systèmes cohérents a été identifiée par Dunne et Bench-Capon dans [DBC01a, DBC02] :

Propriété 2.7. *[DBC01a, DBC02]Un système d'argumentation sans cycle élémentaire de longueur impaire est cohérent.*

La preuve repose sur le théorème suivant établi par Berge [Ber73] en théorie des graphes :

Théorème 2.1. *Un graphe sans cycle élémentaire de longueur impaire possède un semi-noyau maximal non vide.*

PREUVE (PROPRIÉTÉ 2.7) – [8] Soit (A, R) un système non cohérent. Soit $S \subseteq A$ une extension préférée non stable de (A, R).

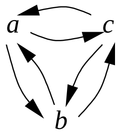
Supposons que $S = \emptyset$. Étant donné que deux extensions préférées distinctes ne peuvent être incluses l'une dans l'autre, et puisque d'après le théorème 2.1, les seuls systèmes à posséder une extension préférée vide sont ceux qui ont un cycle élémentaire de longueur impaire, alors (A, R) possède un cycle élémentaire de longueur impaire.

Supposons que $S \neq \emptyset$. Distinguons trois sous-ensembles d'arguments dans A : l'ensemble S, l'ensemble $R^+(S)$ des arguments attaqués par S, et l'ensemble des arguments restants $Z = A \setminus (S \cup R^+(S))$. Puisque S n'est pas stable, $Z \neq \emptyset$. Considérons le sous-graphe sur Z, (Z, R') où $R' = R \cap (Z \times Z)$. Toute extension préférée T de ce sous-graphe devrait pouvoir être ajoutée à S de telle manière que $T \cup S$ soit admissible dans (A, R), puisque $A \setminus Z$ et Z ne sont pas en conflit. Or S est maximal admissible, et donc la seule extension préférée de (Z, R') doit être l'ensemble vide. Donc, d'après le théorème 2.1, (Z, R'), et donc (A, R), contient un cycle élémentaire de longueur impaire. □

La réciproque de cette propriété (un système cohérent ne possède pas de cycle de longueur impaire) est fausse. Le système ci-dessous en est un contre-exemple : il est cohérent puisque ses trois extensions préférées $\{a\}$, $\{b\}$, et $\{c\}$ sont stables, et il possède plusieurs cycles de longueur impaire.

---

[8] Cette preuve nous a été communiquée par Dunne et Bench-Capon.



Dung a défini dans [Dun95] une classe de systèmes cohérents, *via* les notions d'argument controversé et de système limité controversé. Cette classe est proche de celle identifiée par Dunne et Bench-Capon.

Définition 2.15 (Argument controversé [Dun95]).
Soit (A, R) un système d'argumentation. $b \in A$ est dit controversé *vis-à-vis de $a \in A$ si b est un défenseur individuel (direct ou indirect) de a et b est un attaquant individuel (direct ou indirect) de a.*

Le système précédent donne un exemple d'argument controversé vis-à-vis d'un autre : a attaque et défend individuellement b (contre c).

Définition 2.16 (Système limité controversé [Dun95]).
Un système d'argumentation (A, R) est limité controversé *s'il n'existe aucune séquence infinie d'arguments $x_0, \dots, x_n, \dots$ telle que x_{i+1} est controversé vis-à-vis de x_i.*

Théorème 2.2. *[Dun95] Tout système d'argumentation limité controversé est cohérent.*

Nous montrons que la classe de systèmes cohérents identifiée par Dunne et Bench-Capon comprend les systèmes d'argumentation limités controversés :

Théorème 2.3. *Un système d'argumentation limité controversé ne possède pas de cycle de longueur impaire.*

Lemme 2.1. *Soit (A, R) un système d'argumentation. Si (A, R) possède un cycle élémentaire de longueur impaire, alors tout argument de ce cycle est controversé vis-à-vis d'un autre argument de ce cycle.*

PREUVE – Soit (A, R) un système d'argumentation qui possède un cycle élémentaire de longueur impaire. Ce cycle est de la forme $(x_0, x_1), (x_1, x_2), \dots, (x_{2n}, x_0)$ où $\forall i, 0 \leq i \leq (2n-1), (x_i, x_{i+1}) \in R$ et $(x_{2n}, x_0) \in R$; les x_i sont tous différents car le cycle est élémentaire.

Montrons que $\forall i, 0 \leq i < 2n$, $\forall j, i < j \leq 2n$, il existe un chemin de longueur impaire et un autre de longueur paire d'origine x_i et de but x_j. Nous savons qu'il existe une séquence d'arcs (*i.e.* un chemin) C d'origine x_i et de but x_j incluse dans $(x_0, x_1), (x_1, x_2), \dots, (x_{2n}, x_0)$. Considérons deux cas :

- si C est de longueur paire, alors C prolongé par le chemin de longueur impaire d'origine x_j et de but x_j (c'est le cycle) est un chemin de longueur impaire.
- si C est de longueur impaire, alors C prolongé par le chemin de longueur impaire d'origine x_j et de but x_j (c'est le cycle) est un chemin de longueur paire.

D'après la propriété 2.5 page 17, x_i est donc un défenseur individuel et un attaquant individuel de x_j, et donc x_i est controversé vis-à-vis de x_j. □

PREUVE (THÉORÈME 2.3) – Si un système d'argumentation possède un cycle de longueur impaire, alors d'après le lemme 2.1, il existe dans le système une séquence infinie d'arguments $x_0, \dots, x_n, \dots$ telle que x_{i+1} est controversé vis-à-vis de x_i. Une telle séquence ne peut apparaître dans un système limité controversé. Donc un système limité controversé ne possède pas de cycle de longueur impaire. □

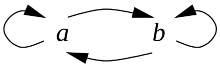
2.3.2 Unicité de l'extension préférée

Certains systèmes ne sont pas cohérents mais possèdent une seule extension préférée. C'est le cas de systèmes qui ne possèdent pas de cycle de longueur paire. Pour les problèmes qui ont en paramètre l'ensemble des extensions préférées, c'est une propriété très intéressante.

Propriété 2.8. *[DBC01b] Un système possède une seule extension préférée s'il ne contient aucun cycle de longueur paire.*

Exemple 2.1 (unicité de l'extension préférée). Les systèmes d'argumentation *Zoé l'autruche*, *Le paradoxe du menteur* et *L'affaire des peintures* ne contiennent aucun cycle de longueur paire et possèdent effectivement une seule extension préferée. ♢

La réciproque de cette propriété est fausse. Le système suivant en est un contre-exemple : il possède une seule extension préférée (l'ensemble ∅) et un cycle de longueur paire.



D'après la propriété 2.8, les systèmes bien-fondés possèdent donc une seule extension préférée. D'après la propriété 2.7, cette extension préférée est stable. Ce résultat a été établi dans [Dun95] par d'autres voies et a été généralisé :

Propriété 2.9. *[Dun95] Tout système d'argumentation bien-fondé possède une et une seule extension préférée, qui est également stable, complète et basique.*

C'est bien ce que nous avons constaté sur le système d'argumentation bien-fondé *Zoé l'autruche*.

Parmi les systèmes qui ont une seule extension préférée, on trouve les systèmes qui ont pour seule extension préférée l'ensemble vide. De tels systèmes, non seulement ne sont pas cohérents, mais en plus ne possèdent aucune extension stable.

Définition 2.17 (Système trivial [DBC01b]).
Un système d'argumentation (A, R) est dit trivial *si sa seule extension préférée est l'ensemble vide.*

Exemple 2.1 (trivialité). Les systèmes d'argumentation *Le paradoxe du menteur* et *L'affaire des peintures* sont triviaux. ♢

Propriété 2.10. *[DBC01b] Si un système d'argumentation (A, R) satisfait les deux conditions suivantes :*

1. *(A, R) ne contient aucun cycle de longueur paire, et*
2. *$\forall x \in A$, $\exists y \in A$ tel que $(y, x) \in R$,*

alors (A, R) est trivial.

Le théorème 2.1 page 19 induit le corollaire suivant :

Corollaire 2.2. *Seuls les systèmes ayant un cycle de longueur impaire peuvent être triviaux.*

La propriété suivante permet d'établir qu'un système d'argumentation trivial ne possède pas d'extension stable, et donc ne peut être cohérent, puisque l'ensemble vide ne peut être une extension stable, sauf dans le cas où l'ensemble des arguments du système est vide.

Propriété 2.11. *Soit (A, R) un système d'argumentation tel que $A \neq \emptyset$. Une extension stable de (A, R) n'est jamais vide.*

> Preuve – Soit S une extension stable de (A, R). Supposons que $S = \emptyset$. Alors $A \setminus S = A$, et donc, par définition, tout argument de A doit être attaqué par $\emptyset$. Ceci n'est possible que si $A = \emptyset$. Puisque par hypothèse $A \neq \emptyset$, $S \neq \emptyset$. □

Propriété 2.12. *Un système d'argumentation trivial n'est pas cohérent.*

2.3.3 Coïncidence de l'extension basique et de l'intersection des extensions préférées

Il est intéressant de voir pour les problèmes qui mettent en jeu l'intersection des extensions préférées, les conditions sous lesquelles cette intersection coïncide avec l'extension basique. Des systèmes qui satisfont cette propriété sont définis par Dung [Dun95] comme suit :

Définition 2.18 (Système d'argumentation relativement basique [Dun95]).
Un système d'argumentation est relativement basique *si son extension basique coïncide avec l'intersection de ses extensions préférées.*

Dung a montré cette coïncidence dans deux cas. Tout d'abord, si le système d'argumentation est *bien-fondé*, la propriété 2.9 page précédente montre que l'extension basique est identique à l'unique extension préférée. Ensuite, cette coïncidence apparaît dans le cas où le système d'argumentation est *non-controversé*.

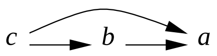
Définition 2.19 (Système d'argumentation non-controversé [Dun95]).
Un système d'argumentation est non-controversé *si aucun de ses arguments n'est controversé.*

Théorème 2.4. *[Dun95] Un système d'argumentation non-controversé est cohérent et relativement basique.*

Exemple 2.1 (relativement basique). Les systèmes d'argumentation *Zoé l'autruche* et *Nixon* sont non-controversés. Les systèmes d'argumentation *Le paradoxe du menteur* et *L'affaire des peintures* sont relativement basiques mais ils ne sont pas non-controversés. ◇

Un système non-controversé est un cas particulier de système limité controversé. Ceci implique qu'un système d'argumentation qui possède un cycle de longueur impaire n'est pas non-controversé.

Cependant, un système sans cycle de longueur impaire (et *a fortiori* sans cycle) peut ne pas être non-controversé. Le système d'argumentation suivant l'illustre : il ne possède pas de cycle, et l'argument c est controversé vis-à-vis de a.



Un système d'argumentation trivial n'étant pas cohérent, il n'est pas non-controversé.

Ces considérations ne nous donnent pas beaucoup plus d'informations sur les caractéristiques graphiques d'un système d'argumentation non-controversé, et plus généralement d'un système d'argumentation relativement basique.

2.3.4 Synthèse

Les liens entre les propriétés particulières d'un système d'argumentation et ses caractéristiques graphiques sont indiqués sur le tableau 2.2 page ci-contre. Pour un système d'argumentation possédant la caractéristique donnée, la légende est la suivante :

- $+$ signifie que la propriété est toujours satisfaite ;
- $-$ signifie que la propriété n'est jamais satisfaite ;
- $\pm$ signifie que la propriété peut être satisfaite.

Un cycle pair (resp. impair) désigne un cycle de longueur paire (resp. impaire).

On peut noter également que l'existence des extensions stables est garantie dans un système sans cycle de longueur impaire (et *a fortiori* dans un système sans cycle) et qu'elle est impossible dans un système vérifiant la propriété 2.10 page précédente.

Le schéma de la figure 2.2 page ci-contre résume les liens entre les propriétés des systèmes d'argumentation.

Caractéristique graphique	Cohérence	Unicité ext. préférée	Coïncidence ext. basique/ $\bigcap$ ext. préférées
sans cycle	+	+	+
sans cycle impair	+	±	±
sans cycle pair	±	+	±
sans cycle pair et vérifiant prop. 2.10	−	+	±
avec cycle impair	±	±	±
avec cycle pair	±	±	±

Tab. 2.2 – Résumé des propriétés des systèmes d'argumentation impliquées par leurs caractéristiques graphiques.

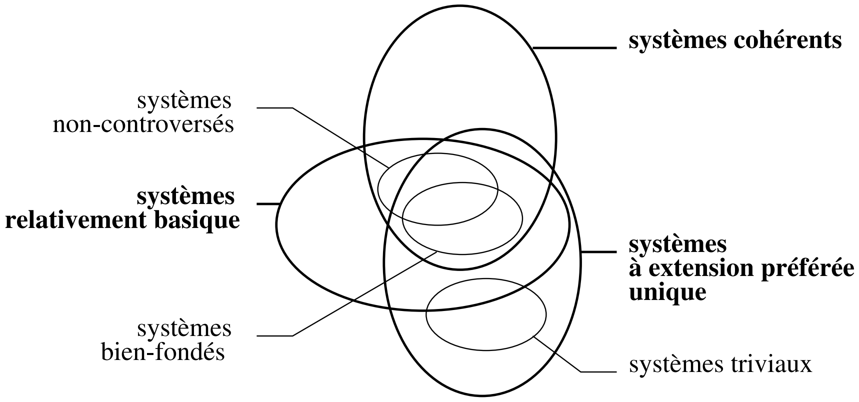


Fig. 2.2 – Liens entre les différentes propriétés des systèmes d'argumentation.

2.4 Problèmes de décision

Dans cette section, nous allons définir les problèmes de décision auxquels nous allons nous intéresser d'un point de vue théorie de la preuve et algorithmique dans la suite de ce mémoire.

2.4.1 Définition

De manière générale, un problème est défini comme étant une question générique qui s'applique à un ensemble d'éléments. Une instance du problème est la question posée pour un élément particulier de l'ensemble. À chaque instance du problème est associée une réponse.

Un *problème de décision* est un problème dont la réponse est *oui* ou *non*, quelle que soit l'instance du problème. Le problème complémentaire d'un problème de décision P est le problème dont la réponse est *oui* quand celle de P est *non* et vice-versa.

Dans les systèmes d'argumentation, deux problèmes de décision sont classiquement associés aux sémantiques qui mènent à plusieurs extensions. Le premier consiste à déterminer si un argument donné appartient à au moins une extension de la sémantique. Ce problème est connu sous le nom de problème de décision crédule, ou selon la terminologie de [DBC01b], *problème d'acceptabilité crédule*. Le deuxième consiste à déterminer si un argument donné appartient à toutes les extensions de la sémantique. Ce problème est connu sous le nom de problème de décision sceptique, ou prudent, ou encore *problème d'acceptabilité sceptique* selon [DBC01b].

Définition 2.20 (Acceptabilité crédule, acceptabilité sceptique).
Étant donné un système d'argumentation (A, R), un argument $a \in A$ et une sémantique :

- *a est* crédulement accepté *sous la sémantique si et seulement si a est contenu dans au moins une extension de la sémantique (autrement dit, si a est contenu dans l'union des extensions de la sémantique) ;*
- *a est* sceptiquement accepté *sous la sémantique si et seulement si a est contenu dans toutes les extensions de la sémantique (autrement dit, si a est contenu dans l'intersection des extensions de la sémantique).*

On dit aussi que s'interroger sur l'acceptabilité crédule (resp. sceptique) d'un argument revient à poser une *requête crédule* (resp. *sceptique*) sur cet argument. Ainsi, le problème d'acceptabilité crédule (resp. sceptique) est aussi appelé *problème de réponse à une requête crédule* (resp. *sceptique*).

Le complémentaire du problème d'acceptabilité sceptique consiste à déterminer s'il existe au moins une extension de la sémantique qui ne contient pas un argument donné. Nous verrons que pour résoudre le problème d'acceptabilité sceptique, il est parfois plus facile de résoudre son complémentaire.

Pour les sémantiques qui ne comportent qu'une seule extension, s'interroger sur l'appartenance d'un argument à l'unique extension de la sémantique est un problème d'acceptabilité à la fois crédule et sceptique. Pour cette raison, nous parlerons simplement du problème d'acceptabilité (sans préciser crédule ou sceptique) associé à une telle sémantique.

EXEMPLE 2.1 (ACCEPTABILITÉ CRÉDULE ET SCEPTIQUE). Considérons les problèmes d'acceptabilité sous la sémantique préférée pour chacun des systèmes d'argumentation :

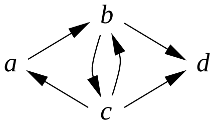
- *Zoé l'autruche* : a et c sont crédulement et sceptiquement acceptés, b n'est pas crédulement accepté (et *a fortiori* b n'est pas sceptiquement accepté) ;
- *Nixon* : a et b sont crédulement acceptés, mais ni l'un ni l'autre ne sont sceptiquement acceptés ;
- *Le paradoxe du menteur* : a n'est pas crédulement accepté (et *a fortiori* il n'est pas sceptiquement accepté) ;
- *L'affaire des peintures* : aucun des arguments n'est crédulement accepté.

$\diamondsuit$

Dans ce mémoire, nous allons essentiellement étudier les problèmes d'acceptabilité sous la sémantique préférée.

Remarquons que le problème d'acceptabilité sceptique sous la sémantique admissible a peu d'intérêt : l'ensemble vide étant toujours un ensemble admissible, aucun argument n'est sceptiquement accepté. De plus, le problème d'acceptabilité crédule sous la sémantique préférée et celui sous la sémantique admissible sont équivalents, puisque tout ensemble admissible est inclus dans une extension préférée.

Sous la sémantique préférée, un plus grand nombre d'arguments sont sceptiquement acceptés que sous la sémantique basique (puisque l'extension basique est incluse dans l'intersection des extensions préférées). Dans certains cas, certains arguments sont rejetés de l'extension basique alors qu'intuitivement ils semblent acceptables. Par exemple, le système d'argumentation suivant possède une extension préférée unique : $\{c\}$. L'argument c est donc sceptiquement accepté sous la sémantique préférée. Ceci se comprend aisément puisque l'argument c se défend contre son attaquant, qui lui-même est attaqué et ne peut être défendu. Or, l'extension basique de ce système est vide, c n'est donc pas accepté sous la sémantique basique.



La sémantique basique ne permet donc pas de saisir certains arguments qui sont intuitivement sceptiquement acceptables.

Cette remarque s'applique également à la sémantique complète, puisque l'extension basique coïncide avec l'intersection des extensions complètes. Dans le système d'argumentation ci-dessus, les extensions complètes sont $\emptyset$ et $\{c\}$, l'argument c n'est donc pas sceptiquement accepté.

Ainsi, il est plus intéressant de considérer le problème d'acceptabilité sceptique sous la sémantique préférée que sous la sémantique basique, la sémantique complète ou la sémantique admissible.

La remarque faite pour le problème d'acceptabilité sceptique sur le fait que certains arguments qui sont intuitivement acceptables sont parfois rejetés de l'extension basique, s'applique au problème d'acceptabilité crédule. Pour cette raison, le problème d'acceptabilité crédule sous la sémantique préférée (équivalent au problème sous la sémantique admissible) est plus intéressant que celui sous la sémantique basique.

Notons enfin que répondre au problème d'acceptabilité crédule sous les sémantiques préférée, complète, basique ou stable, permet de résoudre le problème sous la sémantique admissible. En effet, les extensions sous chacune de ces sémantiques sont des ensembles admissibles ; si un argument appartient à l'une de ces extensions, alors il appartient à un ensemble admissible.

2.4.2 Résolution

Pour expliquer qu'un argument est crédulement ou sceptiquement accepté sous une sémantique donnée, nous allons définir dans le chapitre 4 des **preuves**.

De manière générale, une preuve pour un énoncé prend la forme d'un ensemble d'informations, qui permettent de prouver l'énoncé. La preuve peut parfois inclure la manière dont les informations doivent être articulées pour effectivement prouver l'énoncé, mais pas toujours. Si l'articulation n'est pas donnée, l'idée est que, étant donné l'ensemble des informations, il n'est alors pas trop difficile de reconstruire la preuve. Ce qui est intéressant dans ce cas, c'est que les preuves fournissent une sorte d'explication de pourquoi l'énoncé est vrai, qui est généralement concise et qui évite les informations inutiles.

Une façon agréable de décrire les preuves est la forme **dialectique** : la preuve est un dialogue entre deux joueurs, un partisan et un opposant. Le partisan tente de prouver qu'un argument est accepté en essayant de le défendre contre toutes les attaques de l'opposant.

D'autres formes de preuve existent, comme celles qu'a proposées Pollock dans [Pol95] pour son système basé sur la sémantique préférée. Néanmoins, ces preuves s'appliquent à des domaines spécifiques comme la médiation ou l'enseignement ; l'argumentation y est utilisée comme une composante de systèmes de dialogues basés sur les actes de langage. On retrouve parmi ces systèmes, les systèmes de négociation de [PSJ98, AMP00], les systèmes intelligents [MM99], les systèmes de génération de discours [GCJ00] ou les modèles de procédures légales [BC98].

Pour résoudre les problèmes d'acceptabilité, nous définirons dans les chapitre 5 et 6 des **algorithmes**. On dit qu'un algorithme résout un problème s'il peut être appliqué à toute instance du problème pour fournir une réponse en un temps fini.

Nous allons proposer dans le chapitre 5 des algorithmes qui calculent des preuves dialectiques, et dans le chapitre 6 des algorithmes qui calculent des preuves plus concises.

2.4.3 Problème d'acceptabilité crédule sous la sémantique préférée

Pour le problème d'acceptabilité crédule sous la sémantique préférée, la preuve qu'un argument est accepté peut être simplement un ensemble admissible contenant l'argument. En effet, d'après le point 1 de la propriété 2.1 page 13, tout ensemble admissible est contenu dans une extension préférée. Par exemple, une preuve que l'argument a du système d'argumentation *Zoé l'autruche* (exemple 2.1 page 11) est crédulement accepté est l'ensemble admissible $\{a, c\}$.

La preuve peut également expliquer pourquoi l'ensemble donné est admissible : il faut alors préciser contre quels arguments l'ensemble se défend, et de quelle manière. Par exemple, pour prouver que l'argument a du système d'argumentation *Zoé l'autruche* est crédulement accepté, on peut fournir en plus de l'ensemble admissible $\{a, c\}$, l'ensemble $\{b\}$ contenant l'argument contre lequel $\{a, c\}$ se défend.

On peut affiner la preuve en prenant en compte un critère de concision : l'ensemble admissible ne doit contenir que des arguments qui défendent directement ou indirectement a ; ces arguments sont dits *en faveur* de a. De plus, les arguments qui attaquent l'ensemble admissible doivent tous attaquer directement ou indirectement a ; ces arguments sont ceux *contre* a. La preuve de l'acceptabilité de a dans le système d'argumentation *Zoé l'autruche* donnée dans le paragraphe précédent est concise.

Dans une preuve dialectique, les arguments du partisan seront donc les arguments en faveur de a, ceux de l'opposant seront les arguments contre a. Par exemple, pour prouver que l'argument a du système d'argumentation *Zoé l'autruche* (exemple 2.1) est crédulement accepté, le partisan commencera un dialogue en avançant l'argument a. L'opposant répondra avec l'argument b, attaquant de a. Le partisan défendra alors a grâce à l'argument c. L'opposant ne pourra pas contre-attaquer, et le partisan gagnera le dialogue puisqu'il aura alors défendu a contre toutes les attaques de l'opposant.

Nous proposerons deux types de preuves dialectiques pour le problème d'acceptabilité crédule dans le chapitre 4.

Pour ce qui est des algorithmes pour répondre au problème d'acceptabilité crédule, nous en proposerons deux : le premier calculera une preuve dialectique (section 5.1), le deuxième calculera une preuve qui sera simplement un ensemble admissible contenant l'argument dont l'acceptabilité est à prouver (section 6.5.2).

2.4.4 Problème d'acceptabilité sous la sémantique basique

Sous la sémantique basique, le problème d'acceptabilité revient à déterminer si un argument donné d'un système d'argumentation appartient à l'extension basique du système.

Nous avons vu que dans le cas particulier où le système d'argumentation est fini[9], l'extension basique se construit par applications successives de l'application $\mathcal{F}$ à l'ensemble vide. Pour

---

[9] au sens de Dung

déterminer l'acceptabilité d'un argument a appartenant à $\mathcal{F}^n(\emptyset)$, l'idée est de chercher pour tout argument b qui attaque a, un argument c dans $\mathcal{F}^{n-1}(\emptyset)$ qui défend a.

Amgoud et Cayrol [AC02] et Prakken et Sartor [PS97] ont montré qu'il suffit de chercher des défenseurs qui ne sont pas attaqués par les attaquants de a, autrement dit des défenseurs stricts de a. Le processus est répété jusqu'à ce qu'il n'y ait plus d'attaquant ou de défenseur.

Dans le cas d'un système d'argumentation bien-fondé, l'extension basique est identique à l'unique extension préférée. Une preuve de l'appartenance d'un argument à l'extension basique peut donc être une preuve de l'acceptabilité crédule de l'argument sous la sémantique préférée, et inversement.

Plus généralement, un argument appartient à l'extension basique seulement s'il appartient à l'intersection des extensions préférées, autrement dit s'il est sceptiquement accepté sous la sémantique préférée. Lorsque cette intersection coïncide avec l'extension basique, la réciproque est également vraie. Les méthodes de preuve pour le problème d'acceptabilité sceptique sous la sémantique préférée peuvent alors s'appliquer au problème d'acceptabilité sous la sémantique basique, et inversement.

Dans le chapitre 4, nous décrirons les preuves dialectiques de [PS97, AC02] pour le problème d'acceptabilité sous la sémantique basique.

2.4.5 Problème d'acceptabilité sceptique sous la sémantique préférée

Pour le problème d'acceptabilité sceptique, prouver qu'un argument est accepté est une tâche plus difficile. La raison à cette difficulté est qu'il faut prouver que l'argument se trouve dans *tous* les ensembles admissibles *maximaux*, pas simplement dans un ensemble admissible. Cependant, le problème peut être plus facile à résoudre lorsque le système satisfait certaines propriétés.

Tout d'abord, lorsque le système d'argumentation possède *une seule extension préférée*, le problème d'acceptabilité sceptique se ramène au problème d'acceptabilité crédule. En effet, tous les ensembles admissibles sont inclus dans cette unique extension, et donc pour décider si un argument est sceptiquement accepté, il suffit de trouver un ensemble admissible le contenant.

Ensuite, passer par la résolution du problème complémentaire peut être plus facile. La propriété ci-dessous donne des conditions sous lesquelles un argument n'est pas sceptiquement accepté :

Propriété 2.13. *Étant donné (A, R) un système d'argumentation et $a \in A$ un argument, si a s'auto-attaque, ou possède un attaquant qui est crédulement accepté sous la sémantique préférée, alors a n'est pas sceptiquement accepté sous la sémantique préférée.*

PREUVE – Supposons que a est sceptiquement accepté.

Supposons que b est un attaquant de a et considérons une extension préférée S du système d'argumentation. Puisque a est sceptiquement accepté, $a \in S$, et puisque S est sans-conflit, $b \notin S$. Les attaquants de a ne sont donc pas crédulement acceptés.

De plus, on sait que le système a au moins une extension préférée S. Donc $a \in S$ et puisque S est sans-conflit, a ne s'auto-attaque pas. □

Cette propriété pour la non-acceptabilité est correcte, mais elle n'est généralement pas complète. Par exemple, l'argument a du système d'argumentation *L'affaire des peintures* (exemple 2.1 page 11) n'est pas sceptiquement accepté, mais il ne s'auto-attaque pas et son attaquant n'est pas crédulement accepté.

La propriété suivante montre que la propriété 2.13 est complète pour un système cohérent :

Propriété 2.14. *Étant donné (A, R) un système d'argumentation cohérent, si un argument $a \in A$ n'est pas sceptiquement accepté pour la sémantique préférée, alors a a un attaquant qui est crédulement accepté sous la sémantique préférée.*

Preuve – Si a n'est pas sceptiquement accepté, alors il existe une extension préférée S telle que $a \notin S$. S est stable, donc $a \in R^+(S)$: cela veut dire que a a un attaquant b qui, étant dans S, est crédulement accepté. □

Les propriétés 2.13 et 2.14 indiquent qu'une preuve de la non-acceptabilité sceptique d'un argument a peut donc être un attaquant b de a accompagné d'une preuve de l'acceptabilité crédule de b.

Pour déterminer si un argument est sceptiquement accepté, on peut aussi utiliser le lien entre extension basique et intersection des extensions préférées :

Propriété 2.15. *Étant donné (A, R) un système d'argumentation et $a \in A$ un argument, si a appartient à l'extension basique de (A, R), alors a est sceptiquement accepté sous la sémantique préférée.*

Preuve – La propriété 2.4 page 15 indique que l'extension basique est incluse dans l'intersection des extensions préférées. □

Cette méthode est complète dans le cas où le système est relativement basique :

Propriété 2.16. *Étant donné (A, R) un système d'argumentation et $a \in A$ un argument, si (A, R) est relativement basique, alors a est sceptiquement accepté sous la sémantique préférée si et seulement si a appartient à l'extension basique de (A, R).*

Preuve – D'après le théorème 2.4 page 22, si un système d'argumentation est relativement basique, alors l'extension basique et l'intersection des extensions préférées coïncident. □

Les propriétés 2.15 et 2.16 indiquent qu'une preuve de l'acceptabilité sceptique d'un argument sous la sémantique préférée peut donc être une preuve de son acceptabilité sous la sémantique basique.

Dans le chapitre 4, nous montrerons comment les preuves dialectiques pour le problème d'acceptabilité crédule peuvent être utilisées pour le problème d'acceptabilité sceptique dans les cas particuliers où le système d'argumentation possède une seule extension préférée ou est cohérent. Nous verrons également que les preuves dialectiques pour le problème d'acceptabilité sous la sémantique basique peuvent être utilisées.

Nous proposerons en section 5.2 des algorithmes qui résolvent le problème en utilisant les preuves dialectiques dans ces cas particuliers. En section 6.5.3, nous proposerons un algorithme qui résout le problème d'acceptabilité sceptique dans le cas général.

2.5 Quelques résultats de complexité et de comptage

Dunne et Bench-Capon ont établi plusieurs résultats de complexité et de comptage pour les sémantiques de Dung. Avant d'énoncer ces résultats, nous donnons quelques rappels succincts en complexité et en comptage.

2.5.1 Rappels de complexité

La *complexité temporelle*[10] consiste à caractériser la façon dont évolue le temps de résolution d'un problème (de décision, de calcul...) suivant la taille des données, et ceci dans le pire cas, afin d'obtenir une borne supérieure de la complexité réelle.

Une distinction est habituellement faite entre les problèmes dits «faciles», dont la complexité en pire cas est bornée par une fonction polynomiale de la taille des données, et les autres. Ces

---

[10] C'est seulement à ce type de complexité que nous allons nous intéresser. Il existe un autre type de complexité : la complexité spatiale (voir [GJ79] pour des notions plus complètes).

problèmes dits «faciles» sont résolvables par un algorithme déterministe polynomial. Dans la théorie de la complexité, ils appartiennent à la classe P.

Pour les autres problèmes, la théorie de la complexité introduit la notion d'algorithme non déterministe polynomial. Les problèmes résolvables par ces algorithmes appartiennent à la classe NP. Parmi ces algorithmes, on trouve la classe des algorithmes les plus difficiles de NP : la classe des algorithmes NP-complets.

Notons que pour les problèmes de décision, lorsqu'un problème appartient à une classe X, alors le problème complémentaire appartient à la classe CO-X.

En plus de la distinction P/NP, la théorie de la complexité comprend une hiérarchie de classes de problèmes de plus en plus difficiles : Π_2^p, Δ_3^p,...

Plusieurs conjectures existent en théorie de la complexité, en particulier P $\neq$ NP et NP $\neq$ CO-NP. Elles sont sous-entendues dans les résultats qui suivent.

2.5.2 Rappels de comptage

Dans les *problèmes de comptage* (ou *d'énumération*), contrairement aux problèmes de calcul et de décision, on cherche à connaître le *nombre de solutions* et non seulement une solution ou l'existence d'une solution.

Il existe aussi une hiérarchie des classes de comptage. Les plus connues sont :

FP : le nombre de solutions est calculé en temps polynomial par un algorithme déterministe ;

#P : le nombre de solutions est calculé en temps polynomial par un algorithme non déterministe.

Il existe aussi des problèmes #P-complets qui sont les problèmes d'énumération les plus difficiles de #P.

2.5.3 Résultats pour le système d'argumentation de Dung

Pour les propriétés particulières des systèmes d'argumentation (cohérence, trivialité), Dunne et Bench-Capon ont prouvé les résultats suivants :

Théorème 2.5.
- *[DBC02] Décider si un système d'argumentation est cohérent est un problème Π_p^2-complet.*
- *[DBC01b] Décider si un système d'argumentation est trivial est un problème* CO-NP*-complet.*

Pour l'unicité de l'extension préférée, la complexité du problème qui consiste à décider si un graphe orienté contient un cycle de longueur paire n'est pas connue.

Concernant la complexité des problèmes d'acceptabilité, Dunne et Bench-Capon ont montré dans [DBC01b, DBC01a, DBC02] les résultats résumés dans le tableau 2.3.

Sémantique	Acceptabilité crédule	Acceptabilité sceptique
stable	NP-complet	CO-NP-complet
préférée	NP-complet	Π_2^p-complet CO-NP-complet si cohérence NP-complet si unicité de l'extension préférée

TAB. 2.3 – Complexité des problèmes d'acceptabilité sous les sémantiques stable et préférée

Le résultat de complexité pour le problème d'acceptabilité crédule sous la sémantique préférée est mauvais. Celui pour le problème d'acceptabilité sceptique dans le cas général est pire. Néanmoins, ces mauvais résultats peuvent être nuancés, car ils sont établis en pire cas ; nous voyons que lorsque le système vérifie certaines propriétés (unicité de l'extension préférée, cohérence), le problème d'acceptabilité sceptique est plus facile à résoudre.

En lien avec les problèmes de décision crédules et sceptiques, Dunne et Bench-Capon ont montré dans [DBC01b] que le problème de comptage qui consiste à déterminer, sous la sémantique préférée ou sous la sémantique stable, le nombre d'extensions qui contiennent un argument donné, est #P-complet.

Dunne et Bench-Capon ont montré des résultats de complexité pour d'autres problèmes de décision que nous n'étudions pas dans ce mémoire. Nous les donnons à titre d'information :

Théorème 2.6. *[DBC01b]*

- *Décider si un système d'argumentation possède au moins une extension stable est un problème* NP*-complet.*
- *Décider si un ensemble d'arguments est une extension stable ou un ensemble admissible est calculable en temps polynomial.*
- *Décider si un ensemble d'arguments est une extension préférée est un problème* CO-NP*-complet.*

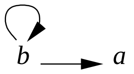
Dans ce mémoire, nous allons proposer des algorithmes pour répondre aux problèmes d'acceptabilité crédule et sceptique sous la sémantique preférée, et pour calculer les extensions préférées. Nous n'allons pas étudier ici la complexité de ces algorithmes : ceci fera l'objet de travaux futurs.

2.6 Critiques des sémantiques de Dung

Une critique courante sur les sémantiques de Dung est leur incapacité à «bien» gérer les arguments qui s'auto-attaquent. Nous présenterons dans un premier temps ces critiques, puis nous en émettrons une autre sur le niveau de désirabilité des défenseurs. Cette critique donnera naissance dans la troisième partie à une nouvelle sémantique, la sémantique *défense minimale.*

2.6.1 Les arguments qui s'auto-attaquent

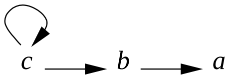
Considérons le système d'argumentation ci-dessous :



D'après [Dun95], [BDKT97] et [Pol94], le seul ensemble d'arguments acceptables (quelle que soit la sémantique) de ce système est l'ensemble vide. Or, [Dun95] et [BDKT97] soutiennent que le seul ensemble d'arguments acceptables devrait être l'ensemble $\{a\}$. En effet, l'argument b s'auto-attaque, il est «irrationnel» et doit donc être «éliminé». De plus, un argument qui s'auto-attaque ne devrait pas avoir suffisament de pouvoir pour rendre inacceptables d'autres arguments.

La sémantique présentée par [KMD94] dans le contexte de la programmation logique, tente d'éliminer les arguments qui s'auto-attaquent ainsi que leurs effets sur d'autres arguments. Aussi, [Dun95] et [BDKT97] suggèrent que cette sémantique devrait être généralisée à l'argumentation.

On peut avoir une intuition opposée. Considérons le système d'argumentation ci-dessous :



Le seul ensemble admissible de ce système est l'ensemble vide. La raison à cela est que l'argument c s'auto-attaque (il ne peut donc appartenir à aucun ensemble sans-conflit, et *a fortiori* à aucun ensemble admissible), l'argument b est attaqué par c et ne peut être défendu contre cette attaque (c «invalide» b) ; quant à l'argument a, il est attaqué par l'argument b et il est défendu par l'argument c, mais c ne pouvant appartenir à aucun ensemble admissible, a ne peut appartenir à aucun ensemble admissible.

Dans [JV96, JV99b], Jakobovits et Vermeir émettent une critique sur ce raisonnement : si c invalide b, pourquoi n'invalide-t-il pas aussi l'attaque de b sur a, et ainsi, pourquoi l'ensemble $\{a\}$ n'est-il pas acceptable ?

Si l'ensemble $\{a\}$ n'est pas acceptable selon Dung, cela vient de sa définition de l'admissibilité. En effet, elle est telle que :

- si un argument appartient à un ensemble admissible, alors ses attaques sur les autres arguments peuvent être utilisées pour défendre d'autres éléments de l'ensemble admissible ;
- par contre, si un argument n'appartient pas à un ensemble admissible, alors ses attaques ne peuvent être utilisées pour défendre un élément de l'ensemble admissible.

Jakobovits et Vermeir pensent qu'il faudrait modifier cette définition de manière à ce que l'existence de l'attaque de c sur b soit reconnue, et que l'attaque de b sur a soit invalidée. Ainsi, non seulement l'ensemble vide, mais également l'ensemble $\{a\}$, seraient acceptables.

Dans ce but, Jakobovits et Vermeir proposent une notion de défense plus faible que celle de Dung. Cette notion de défense est définie comme suit :

Définition 2.21 (Défense faible [JV96]).
Étant donné (A, R) un système d'argumentation, $S \subseteq A$ un ensemble sans-conflit et $T \subseteq A$, on dit que T défend faiblement *S si et seulement si :*

1. *T n'attaque pas S et S n'attaque pas T ;*
2. *$S \cap T = \emptyset$;*
3. *si $a \in A$ attaque $T \cup S$ alors $T \cup S$ attaque a.*

Un ensemble sans-conflit d'arguments est dit faiblement défendable *si et seulement si il existe un ensemble qui le défend faiblement.*

Notons qu'un ensemble d'arguments T qui défend faiblement un ensemble d'arguments S n'est pas nécessairement sans-conflit. Ainsi, l'ensemble $\{c\}$ défend faiblement l'ensemble $\{a\}$. L'ensemble $\{a\}$ est donc faiblement défendable, il est acceptable.

Jakobovits et Vermeir proposent dans [JV96, JV99b] de nouvelles sémantiques basées sur cette notion de défense affaiblie.

2.6.2 Le niveau de désirabilité des défenseurs

Une caractéristique commune des sémantiques de Dung est qu'elles essaient toujours (à l'exception de la sémantique admissible) de maximiser d'une certaine manière l'ensemble des arguments acceptables.

Une critique qui peut être faite sur ces sémantiques est que tous les arguments, lorsqu'ils doivent être utilisés comme défenseurs, sont également désirables. En d'autres termes, si un argument peut être indépendemment défendu par deux arguments différents, alors on choisit n'importe lequel des deux, voire même les deux s'ils ne s'attaquent pas, pour défendre le premier argument.

Or, dans certaines applications, il peut être nécessaire de faire une distinction entre les défenseurs ; l'utilisation de certains peut être plus restreinte que d'autres. Plaçons-nous par exemple dans le cas où les arguments «appartiennent» à deux agents ; l'un, le partisan, soutient un point de vue que l'autre, l'opposant, essaie de réfuter. Parmi les arguments du partisan, il peut en être certains qui ne doivent être révélés qu'en cas d'absolue nécessité, parce qu'ils contiennent des informations très personnelles ou parce qu'ils pourraient mettre en danger le partisan. Nous verrons de tels exemples dans le chapitre 7.

Ainsi, nous proposerons dans la troisième partie d'étendre le système d'argumentation de Dung dans le but de représenter ce que l'on appelle des arguments «restreints» : ces arguments devront être présents dans un ensemble d'arguments acceptables seulement si c'est absolument nécessaire, c'est-à-dire s'ils défendent des arguments qui seraient autrement rejetés de l'ensemble acceptable.

Nous modifierons en conséquence la sémantique préférée de Dung : un ensemble d'arguments sera acceptable seulement s'il contient un minimum d'arguments restreints pour un maximum d'arguments non restreints. La sémantique qui saisira ces conditions sera appelée *sémantique défense minimale.*

Chapitre 3

Cadres formels liés au système de Dung

Plusieurs cadres formels sont fortement liés au système de Dung. Nous avons déjà vu dans le chapitre précédent le lien entre le système de Dung et la théorie des graphes.

Nous allons à présent nous intéresser aux liens avec le système d'argumentation de Bondarenko, Dung, Kowalski et Toni [BDKT97] (appelé dans la suite *système d'argumentation BDKT*) et avec des cadres formels qui permettent de modéliser le raisonnement révisable : la *programmation logique avec négation par l'échec* et la *logique des défauts*.

Ainsi, nous allons voir comment Dung [Dun95] a exprimé un programme logique dans son système et quelle est la correspondance entre les sémantiques de la programmation logique et les notions d'acceptabilité de Dung.

Nous verrons également que, parallèlement au travail de Dung, la programmation logique a été exprimée en théorie des graphes, et que l'équivalence entre ses sémantiques et des notions de la théorie des graphes ont été montrées par Dimopoulos et Torres [DT96].

Par ailleurs, Dimopoulos et Magirou [DM94] ont montré comment une théorie des défauts peut être exprimée sous forme d'un graphe orienté, et quelle est la correspondance entre sa sémantique et une notion de la théorie des graphes. Ce graphe orienté correspondant à une théorie des défauts est directement transposable dans le système de Dung.

Toutes ces connections induisent que les résultats établis dans ce mémoire pour le système d'argumentation de Dung peuvent être utilisés pour les cadres formels qui se ramènent au système de Dung.

3.1 Le système d'argumentation BDKT

Bondarenko, Dung, Kowalski et Toni ont proposé dans [BDKT97] un système d'argumentation. Ce système est apparu après le développement de bon nombre de systèmes d'argumentation qui ont tenté de capturer le raisonnement non monotone. L'innovation principale du système BDKT est qu'il fournit un cadre et un vocabulaire qui permettent de capturer les caractéristiques générales des autres systèmes, et saisissent ainsi de manière remarquablement efficace les logiques non monotones existantes.

La notion de base du système BDKT est celle de *supposition*. Les suppositions sont utilisées pour étendre des ensembles de prémisses «ordinaires» à la condition que l'on ne puisse prouver le contraire de telles suppositions. Ce qui fait que le système BDKT est argumentatif est que cette condition est formalisée en termes de suppositions qui s'attaquent les unes les autres. En d'autres termes, d'après [BDKT97], un argument est un ensemble de suppositions.

Présentons plus en détail le système BDKT avant de le comparer au système de Dung.

Le système BDKT est basé sur un *système déductif* monotone $(\mathcal{L}, \mathcal{R})$ tel que :

- $\mathcal{L}$ est un langage formel comprenant un nombre dénombrable de formules, et
- $\mathcal{R}$ est un ensemble de règles d'inférence de la forme :

$$\frac{\alpha_1, \ldots, \alpha_n}{\alpha}$$

où $\alpha, \alpha_1, \ldots, \alpha_n \in \mathcal{L}$ et $n \geq 0$.

On appelle *théorie* tout ensemble de formules $T \subseteq \mathcal{L}$. Une *déduction* dans une théorie T est une séquence $\beta_1, \ldots, \beta_m$ où $m > 0$, telle que $\forall i = 1, \ldots, m$,

- $\beta_i \in T$, ou
- il existe une règle d'inférence $\frac{\alpha_1, \ldots, \alpha_n}{\beta_i} \in \mathcal{R}$ telle que $\alpha_1, \ldots, \alpha_n \in \{\beta_1, \ldots, \beta_{i-1}\}$.

$T \vdash \alpha$ signifie qu'il existe une déduction dans T dont le dernier élément est α. $Th(T)$ est l'ensemble $\{\alpha \in \mathcal{L} \mid T \vdash \alpha\}$.

$(\mathcal{L}, \mathcal{R})$ est monotone car si $T \subseteq T'$ alors $Th(T) \subseteq Th(T')$.

Définition 3.1 (Système d'argumentation BDKT).
Soit $(\mathcal{L}, \mathcal{R})$ un système déductif monotone. Un système d'argumentation BDKT est un triplet $(T, Ab, ^{-})$ tel que :

- *$T \subseteq \mathcal{L}$ est une théorie qui représente un ensemble de* croyances *;*
- *$Ab \subseteq \mathcal{L}$, $Ab \neq \emptyset$, est un ensemble de* suppositions *qui peut être utilisé pour étendre T ;*
- *$^{-}$ est une application de Ab dans $\mathcal{L}$ qui permet de dénoter le contraire d'une supposition : $\overline{\alpha}$ dénote le contraire de α.*

L'attaque entre ensembles de suppositions est définie comme suit :

Définition 3.2 (Attaque).
Soit $(T, Ab, ^{-})$ un système d'argumentation. Un ensemble de suppositions $\Delta \subseteq Ab$ attaque un autre ensemble de suppositions $\Delta' \subseteq Ab$ s'il existe $\alpha \in \Delta'$ tel que $T \cup \Delta \vdash \overline{\alpha}$.

L'attaque entre ensembles de suppositions n'est pas nécessairement symétrique. Les sémantiques stable, admissible et préférée sont définies comme suit :

Définition 3.3 (Ensemble de suppositions stable).
Un ensemble de suppositions $\Delta \subseteq Ab$ est stable si et seulement si :

- *Δ est fermé,* i.e $\Delta = \{\alpha \in Ab \mid T \cup \Delta \vdash \alpha\}$,
- *Δ ne s'attaque pas lui-même, et*
- *Δ attaque toute supposition α telle que $\alpha \notin \Delta$.*

Définition 3.4 (Ensemble de suppositions admissible).
Un ensemble de suppositions $\Delta \subseteq Ab$ est admissible *si et seulement si, $\forall \Delta' \subseteq Ab$, si Δ' attaque Δ alors Δ attaque $\Delta' \setminus \Delta$ (Δ se défend contre Δ').*

Définition 3.5 (Ensemble de suppositions préféré).
Un ensemble de suppositions $\Delta \subseteq Ab$ est préféré *si et seulement si Δ est admissible, maximal pour l'inclusion ensembliste.*

De nombreux résultats ont été établis pour le système BDKT. Dans [DKT96], Dung, Kowalski et Toni ont proposé une procédure pour calculer la sémantique admissible. Des théories de la preuve dialectiques ont été développées par Kakas et Toni dans [KT99]. Des procédures de preuve pour les problèmes d'acceptabilité ont été proposées par Dung, Mancarella et Toni dans [DMT01]. La complexité des problèmes d'acceptabilité pour les sémantiques stable et préférée a été étudié par Dimopoulos, Nebel et Toni dans [DNT99, DNT00, DNT02].

Certains travaux ([PV02] par exemple) présentent le système d'argumentation de Dung comme étant la version la plus abstraite du système BDKT. Sur certains points, il est pourtant difficile de faire une correspondance immédiate entre les deux systèmes.

S'il semble naturel de faire correspondre au système BDKT, le système (A, R) de Dung tel que :

– les arguments de A sont des ensembles de suppositions et
– les éléments de la relation R sont les attaques entre ensembles de suppositions,
il est par contre plus difficile de faire correspondre les sémantiques.

En effet, sous une sémantique donnée, Dung définit des ensembles d'arguments acceptables, alors que BDKT définit des ensembles de suppositions acceptables. Si l'on fait correspondre au système BDKT le système d'argumentation ci-dessus, c'est, au sens de Dung, un ensemble d'ensembles de suppositions qui est acceptable sous une sémantique donné.

Le système d'argumentation de Dung [Dun95] est plus abstrait que le système BDKT, et peut ainsi être utilisé dans des contextes autres que le raisonnement non monotone. Non seulement il permet de saisir, comme le système BDKT, les caractéristiques générales d'autres systèmes d'argumentation et de logiques non monotones (nous le verrons dans la section suivante), mais il s'applique également à d'autres types de systèmes d'argumentation, dans lesquels les arguments ne peuvent être exprimés à l'aide de suppositions.

L'argumentation est par exemple utilisée comme une composante des protocoles de négociation ; les arguments pour une offre doivent persuader les autres parties d'accepter l'offre [AMP00, PSJ98]. L'argumentation est aussi une composante de certains modèles formels récents pour la médiation et elle a été utilisée dans des systèmes pour l'enseignement intelligent, comme Belvedere [SWCP95] qui enseigne le raisonnement scientifique ou les systèmes CATO [AA97] et Argue ! [Ver99]. Dans les applications de ce type, les arguments ont une structure qui ne peut pas être exprimée à l'aide de suppositions. Un argument est par exemple un ensemble de parties liées d'un texte en langage naturel non structuré (systèmes Belvedere ou ZENO [GK97]), ou il consiste en l'utilisation analogique de précédents (système CATO). Pour de tels systèmes, le système d'argumentation de Dung, plus abstrait que le système BDKT, est mieux approprié puisqu'il laisse non spécifiées la structure interne des arguments et la relation de contrariété.

3.2 La programmation logique

Dans cette section, nous allons voir le lien entre le système d'argumentation de Dung et la programmation logique établi par Dung dans [Dun95], et le lien entre la théorie des graphes et la programmation logique établi par Dimopoulos et Torres dans [DT96].

Avant de présenter ces liens, nous allons formellement définir la programmation logique ; la définition de ses sémantiques permettra de bien établir leur correspondance avec les sémantiques de Dung et les notions de la théorie des graphes.

3.2.1 Définition

Les programmes logiques correspondent à un sous-langage de la logique du premier ordre. Nous donnons tout d'abord la syntaxe de ce langage puis ses sémantiques.

Définissons formellement un langage du premier ordre $\mathcal{L}$.

Définition 3.6 (Langage du premier ordre).
Le vocabulaire *de $\mathcal{L}$ comprend plusieurs ensembles dénombrables de symboles de* constantes*,* variables*,* fonctions *et* prédicats*. L'ensemble des fonctions est noté V_f, l'arité des fonctions est donnée par l'application $A_f : V_f \mapsto \mathbb{N}^*$. L'ensemble des prédicats est noté V_p, l'arité des prédicats est donnée par l'application $A_p : V_p \mapsto \mathbb{N}^*$.*

Le vocabulaire comprend également :
– *des* connecteurs *: $\neg$ (la* négation*), $\wedge$ (la* conjonction*), $\vee$ (la* disjonction*) et $\leftarrow$ (le* si*) ;*
– *des* délimiteurs *: (et) ;*
– *des* quantificateurs *: $\forall$ (quantificateur* universel*) et $\exists$ (quantificateur* existentiel*).*

Le langage des termes *de $\mathcal{L}$ est construit à l'aide des règles suivantes :*

1. *Tout symbole de constante est un terme.*
2. *Tout symbole de variable est un terme.*

3. *Si $f \in V_f$ et $t_1, \ldots, t_{A_f(f)}$ sont des termes, alors $f(t_1, \ldots, t_{A_f(f_i)})$ est un terme.*
4. *Tout terme s'obtient en appliquant un nombre fini de fois les trois règles ci-dessus.*

Le langage des formules *de $\mathcal{L}$ est construit à l'aide des règles suivantes :*

1. *Si $p \in V_p$ et $t_1, \ldots, t_{A_p(p)}$ sont des termes, alors $p(t_1, \ldots, t_{A_p(p)})$ est une formule (dite* formule atomique *ou* atome*).*
2. *Si φ et une formule, alors $(\neg\varphi)$ est une formule.*
3. *Si φ_1 et φ_2 sont des formules, alors $(\varphi_1 \wedge \varphi_2)$, $(\varphi_1 \vee \varphi_2)$ et $(\varphi_1 \leftarrow \varphi_2)$ sont des formules.*
4. *Si φ est une formule et $v \in V_v$ alors $(\forall v)\varphi$ et $(\exists v)\varphi$ sont des formules.*
5. *Toute formule s'obtient en appliquant un nombre fini de fois les quatre règles ci-dessus.*

Les termes de base *sont les termes qui ne contiennent aucun symbole de variable. Les* atomes de base *sont les atomes qui ne contiennent aucun symbole de variable.*

Définition 3.7 (Littéral).
Un littéral *est une formule qui est soit un atome (on parle alors de* littéral positif*), soit la négation d'un atome (on parle de* littéral négatif*).*

Définition 3.8 (Programme logique).
Un programme logique *est un ensemble de formules (appelées des* règles*) de la forme :*

$$\alpha \leftarrow \gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_m$$

où $m \geq 0$, α est un atome et $\forall i, 1 \leq i \leq m$, γ_i est un littéral appelé sous-but*. α est appelé* tête *de la règle et l'ensemble des sous-buts constitue le* corps *de la règle. Toutes les variables sont implicitement universellement quantifiées.*

Le langage associé *à un programme logique P est constitué des symboles apparaissant dans P et est dénoté par $\mathcal{L}_P$.*

Définition 3.9 (Univers de Herbrand, base de Herbrand).
Soit P un programme logique.

- *L'*univers de Herbrand *de $\mathcal{L}_P$ est l'ensemble des termes de base de $\mathcal{L}_P$*[1]*.*
- *La* base de Herbrand *de $\mathcal{L}_P$ (notée $\mathcal{HB}(\mathcal{L}_P)$, ou $\mathcal{HB}$ si le programme logique est implicite) est l'ensemble des atomes de base construits sur l'univers de Herbrand de $\mathcal{L}_P$ et les prédicats de $\mathcal{L}_P$.*

On dénote par $\mathcal{HB}_\neg(\mathcal{L}_P)$ (ou $\mathcal{HB}_\neg$ si le programme logique est implicite) l'ensemble $\{\neg\beta \mid \beta \in \mathcal{HB}(\mathcal{L}_P)\}$.

Définition 3.10 (Instanciation de Herbrand).
*L'*instanciation de Herbrand *d'un programme P (notée P_{inst}) est le programme obtenu en remplaçant les variables de P de toutes les manières possibles par des termes de l'univers de Herbrand de $\mathcal{L}_P$.*

Dans le langage associé à un programme logique, la négation a une signification particulière : étant donné un atome β, $\neg\beta$ signifie que l'*on suppose que β est faux, à moins que l'on arrive à prouver β.* On parle de *négation par l'échec.* Les littéraux négatifs sont ainsi appelés des suppositions. Cette interprétation a été introduite notamment par Eshghi et Kowalski dans [EK89]. Formellement :

Définition 3.11 (Supposition, hypothèse).
Une supposition *est un littéral de $\mathcal{HB}_\neg$, une* hypothèse *est un ensemble de suppositions.*

---

[1] Si l'ensemble des constantes de $\mathcal{L}_P$ est vide, l'univers de Herbrand de $\mathcal{L}_P$ est l'ensemble des termes de base d'une extension de $\mathcal{L}_P$, obtenue en ajoutant à $\mathcal{L}_P$ un symbole de constante.

Exemple 3.1. Traduisons l'exemple *Zoé l'autruche* (exemple 1.1 page 4) sous forme d'un programme logique. Le vocabulaire choisi comprend une constante *Zoé*, une variable x, et quatre symboles de prédicats d'arité 1 : o, pour dénoter un oiseau, a pour dénoter une autruche, v pour dénoter la capacité à voler et d pour dénoter une observation faite avec des instruments défectueux. Étant donné ce langage, le programme logique P qui correspond à l'exemple est le suivant :

$$\begin{array}{lll} v(x) & \leftarrow & o(x) \wedge \neg a(x) \\ o(Zoé) & \leftarrow & \\ a(x) & \leftarrow & \neg d(x) \\ d(Zoé) & \leftarrow & \end{array}$$

La première règle signifie que x vole si x est un oiseau et si l'on suppose que x n'est pas une autruche. La deuxième règle signifie que *Zoé* est un oiseau. La troisième règle signifie que x est une autruche si l'on suppose que l'observation qui a été faite de x ne l'a pas été avec des instruments défectueux. La dernière règle signifie que *Zoé* a été observée avec des instruments défectueux.

L'univers de Herbrand de $\mathcal{L}_P$ est $\{Zoé\}$, la base de Herbrand de $\mathcal{L}_P$ est :

$$\mathcal{HB} = \{o(Zoé), a(Zoé), v(Zoé), d(Zoé)\}.$$

Il existe une seule instanciation de Herbrand de P : c'est le programme obtenu en remplaçant la variable x par la constante *Zoé*. L'ensemble des suppositions du programme est :

$$\mathcal{HB}_\neg = \{\neg o(Zoé), \neg a(Zoé), \neg v(Zoé), \neg d(Zoé)\}.$$

♢

Les sémantiques de la programmation logique sont basées sur la notion d'interprétation de Herbrand.

Définition 3.12 (Interprétation de Herbrand).
Une interprétation de Herbrand *I pour un programme logique P est un sous-ensemble de $\mathcal{HB}(\mathcal{L}_P) \cup \mathcal{HB}_\neg(\mathcal{L}_P)$ tel que $I \cap \{\neg\beta \mid \beta \text{ est un atome}, \beta \in I\} = \emptyset$.*

Une interprétation de Herbrand assigne la valeur *vrai* à tout atome de base ou supposition qui se trouve dans I, et la valeur *faux* à tout atome ou supposition qui ne se trouve pas dans I. Notons que l'on ne peut pas avoir à la fois $\neg\alpha$ et α dans une interprétation. La valeur de vérité des autres formules est définie de la manière usuelle.

Définition 3.13 (Modèle de Herbrand).
Un modèle de Herbrand *pour un programme logique P est une interprétation de Herbrand qui assigne la valeur* vrai *à toute règle du programme.*

Définition 3.14 (Modèle stable [GL88]).
M est un modèle stable *d'un programme logique P si et seulement si M est le plus petit modèle de Herbrand du programme logique P_M obtenu en éliminant de P :*
- *toutes les règles qui ont des conditions de la forme $\neg\gamma$ telle que $\gamma \in M$,*
- *tous les littéraux négatifs des règles restantes.*

D'autres sémantiques ont été définies pour la programmation logique, comme les *modèles partiels stables* ([SZ90]), le *modèle bien-fondé* ([VGRS88]), les *modèles stationnaires* ([Prz91]).

Exemple 3.1 (suite). L'interprétation

$$I = I_{\Delta_A} \cup I_{\Delta_C} = \{\neg a(Zoé), v(Zoé), d(Zoé), o(Zoé)\}$$

est un modèle stable, un modèle partiel stable et le modèle bien-fondé de P. ♢

3.2.2 Lien avec le système d'argumentation de Dung

Un programme logique peut être vu comme un système d'argumentation. Pour montrer cela, nous allons définir une notion d'hypothèse support et une notion d'attaques entre hypothèses. Les définitions qui suivent sont inspirées de [DT96] :

Définition 3.15 (Hypothèse support).
Soit P un programme, Δ une hypothèse et α un atome de $\mathcal{L}_P$. Δ est un support *pour α (dénoté $\Delta \mapsto \alpha$) s'il existe une séquence d'atomes de base $(\alpha_0, \alpha_1, \ldots, \alpha_n)$ avec :*

- *$\alpha_n = \alpha$, et*
- *pour tout α_i :*
  - *soit $\alpha_i \leftarrow$ est une règle de P_{inst},*
  - *soit α_i est la tête d'une règle $\alpha_i \leftarrow \gamma_1, \ldots, \gamma_t, \neg\gamma_{t+1}, \ldots, \neg\gamma_{t+r}$ de P_{inst} telle que les littéraux positifs $\gamma_1, \ldots, \gamma_t$ sont des membres précédents de la séquence et les littéraux négatifs $\neg\gamma_{t+1}, \ldots, \neg\gamma_{t+r}$ appartiennent à Δ.*

On dénote par $\Delta^{\mapsto}$ l'ensemble des atomes supportés par Δ. Δ est minimal *pour α (dénoté $\Delta \mapsto_{min} \alpha$) si aucun sous-ensemble strict de Δ ne supporte α.*

Exemple 3.1 (suite).

- L'hypothèse $\Delta_A = \{\neg a(Zoé)\}$ est un support pour $v(Zoé)$, $o(Zoé)$ et $d(Zoé)$; c'est un support minimal pour $v(Zoé)$.
- L'hypothèse $\Delta_B = \{\neg d(Zoé)\}$ est un support pour $a(Zoé)$, $o(Zoé)$ et $d(Zoé)$; c'est un support minimal pour $a(Zoé)$.
- L'hypothèse $\Delta_C = \emptyset$ est un support pour $d(Zoé)$ et $o(Zoé)$; c'est un support minimal pour ces deux atomes.

♢

Définition 3.16 (Attaque entre hypothèses).
Soit P un programme, Δ et Δ' deux hypothèses.

- Δ attaque Δ' *(noté $\Delta \rightsquigarrow \Delta'$) s'il existe β tel que $\Delta \mapsto \beta$ et $\neg\beta \in \Delta'$.*
- Δ attaque minimalement Δ' *(noté $\Delta \rightsquigarrow_{min} \Delta'$) si $\Delta \mapsto_{min} \beta$ et $\neg\beta \in \Delta'$.*

Exemple 3.1 (suite). L'hypothèse Δ_C attaque l'hypothèse Δ_B. L'hypothèse Δ_B s'attaque elle-même, et attaque également l'hypothèse Δ_A. L'hypothèse Δ_A attaque l'hypothèse Δ_B.

L'hypothèse Δ_B attaque minimalement l'hypothèse Δ_A et Δ_C attaque minimalement Δ_B. ♢

Définition 3.17 (Hypothèse sans-conflit).
Une hypothèse Δ d'un programme P est sans-conflit *si elle ne s'attaque pas elle-même.*

Exemple 3.1 (suite). Les hypothèses Δ_A et Δ_C sont sans-conflit. L'hypothèse Δ_B s'attaque elle-même, mais elle ne s'attaque pas minimalement. ♢

Étant donné un programme logique P, Δ et Δ' deux hypothèses, α et α' deux atomes de base, selon Dung [Dun95], le système d'argumentation correspondant à P, noté $\mathcal{AF}_P = (A, R)$ est tel que :

$$\begin{array}{rcl} A & = & \{(\Delta, \alpha) \mid \Delta \text{ supporte } \alpha\} \cup \{(\{\neg\alpha\}, \neg\alpha) \mid \alpha \text{ est un atome de base }\} \\ R & = & \{((\Delta, \alpha), (\Delta', \alpha')) \mid \Delta \rightsquigarrow \Delta'\} \end{array}$$

Un argument $(\{\neg\alpha\}, \neg\alpha)$ saisit l'idée que α sera conclu faux si aucun argument ne supporte α. Dung définit les arguments comme des paires «support-conclusion», où à chaque hypothèse est associée un atome supporté par l'hypothèse. Remarquons que la relation R est établie seulement à partir des attaques entre supports.

Exemple 3.1 (suite). Le système d'argumentation $\mathcal{AF}_P = (A, R)$ qui correspond à P est tel que :

- A contient les arguments : $(\Delta_A, v(Zoé))$, $(\Delta_A, o(Zoé))$, $(\Delta_A, d(Zoé))$, $(\Delta_B, a(Zoé))$, $(\Delta_B, o(Zoé))$, $(\Delta_B, d(Zoé))$, $(\Delta_C, d(Zoé))$, $(\Delta_C, o(Zoé))$, $(\{\neg a(Zoé)\}, \neg a(Zoé))$, $(\{\neg o(Zoé)\}, \neg o(Zoé))$, $(\{\neg d(Zoé)\}, \neg d(Zoé))$, $(\{\neg v(Zoé)\}, \neg v(Zoé)) \dots$;
- quant à la relation R, tous les arguments de support Δ_C attaquent ceux de support Δ_B, ceux de support Δ_B s'attaquent eux-mêmes et attaquent les arguments de support Δ_A ; les arguments de support Δ_A attaquent ceux de support Δ_B.
  L'argument $(\{\neg v(Zoé)\}, \neg v(Zoé))$ est attaqué par tous les arguments de support Δ_A. L'argument $(\{\neg a(Zoé)\}, \neg a(Zoé))$ est attaqué par tous les arguments de support Δ_B. Les arguments $(\{\neg o(Zoé)\}, \neg o(Zoé))$ et $(\{\neg d(Zoé)\}, \neg d(Zoé))$ sont attaqués par tous les arguments de support Δ_A ou Δ_B ou Δ_C.

Sur le graphe de la figure 3.1 sont représentés les supports des arguments de A énoncés ci-dessus et les attaques entre ces supports. ◇

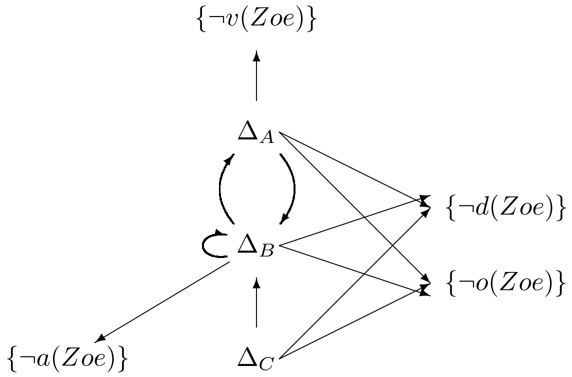


FIG. 3.1 – Représentation partielle du système d'argumentation correspondant, selon Dung, au programme logique de l'exemple 3.1 page 37.

[Dun95] et [BDKT97] ont montré que les sémantiques de la programmation logique trouvent une correspondance avec les sémantiques de Dung :

- les extensions stables de $\mathcal{AF}_P$ correspondent aux modèles stables de P [Dun95] ;
- les ensembles admissibles de $\mathcal{AF}_P$ correspondent aux interprétations bien-fondées de P [Dun95, BDKT97] ;
- les extensions préférées de $\mathcal{AF}_P$ correspondent aux modèles partiels stables de P [Dun95, BDKT97] ;
- l'extension basique de $\mathcal{AF}_P$ correspond au modèle bien-fondé de P [Dun95] ;
- les extensions complètes de $\mathcal{AF}_P$ correspondent aux modèles stationnaires de P [BDKT97].

En particulier, la correspondance entre les extensions stables et les modèles stables montrée par Dung dans [Dun95] est la suivante :

Théorème 3.1. *[Dun95] Soit P un programme logique. Une interprétation M est un modèle stable de P si et seulement si il existe une extension stable E de $\mathcal{AF}_P$ telle que :*

$$M = \{\beta \mid \exists(\Delta, \beta) \in E\}.$$

La correspondance entre le modèle bien-fondé de P et l'extension de base de $\mathcal{AF}_P$ montrée par Dung est la suivante :

Théorème 3.2. *[Dun95] Étant donné un programme logique P, M le modèle bien-fondé de P et E l'extension basique de $\mathcal{AF}_P$, alors :*

$$M = \{\beta \mid \exists(\Delta, \beta) \in E\}.$$

Dung a montré dans [Dun95] que l'ensemble des arguments du système d'argumentation $\mathcal{AF}_P$ associé à un programme logique P est non récursivement décidable, c'est-à-dire qu'il n'existe pas d'algorithme qui termine et qui décide pour toute paire (Δ, α) où Δ est une hypothèse et α est un atome de base de $\mathcal{L}_P$, si elle appartient ou non à l'ensemble des arguments de $\mathcal{AF}_P$. De plus, l'ensemble des extensions stables et celui des extensions préférées du système d'argumentation associé à un programme logique sont généralement non calculables.

Exemple 3.1 (suite). $\mathcal{AF}_P$ a une seule extension préférée E, qui est également stable et basique :

$$E \supseteq \{(\Delta_A, v(Zoé)), (\Delta_A, d(Zoé)), (\Delta_C, d(Zoé)), (\Delta_A, o(Zoé)), (\Delta_C, o(Zoé)), (\{\neg a(Zoé)\}, \neg a(Zoé))\}.$$

Les ensembles $\{(\Delta_C, d(Zoé))\}$ et $\emptyset$ sont admissibles, mais ils ne sont pas maximaux. À partir de E, on retrouve le modèle stable et le modèle bien-fondé M de P :

$$M = \{\beta \mid \exists(\Delta, \beta) \in E\} = \{v(Zoé), d(Zoé), o(Zoé), \neg a(Zoé)\}.$$

$\diamondsuit$

3.2.3 Lien avec la théorie des graphes

Parallèlement aux travaux de Dung, Dimopoulos et Torres [DT96] ont étudié le lien entre programmation logique et théorie des graphes. Étant donné un programme logique P, Δ et Δ' deux hypothèses, α un atome de base, Dimopoulos et Torres ont fait correspondre à P le graphe orienté (V, E) tel que :

$$\begin{array}{rcl} V & = & \{\Delta \mid \exists\alpha \text{ tel que } \Delta \hookrightarrow_{min} \alpha\}, \\ E & = & \{(\Delta, \Delta') \mid \Delta \rightsquigarrow_{min} \Delta'\}. \end{array}$$

Remarquons qu'à une supposition Δ correspond un seul nœud dans le graphe de [DT96], contrairement au graphe de [Dun95], où il y a autant de nœuds de support Δ que d'atomes supportés par Δ. De plus, les attaques entre support doivent être minimales dans le graphe de [DT96] ; il y a donc moins d'attaques et donc moins d'arcs que dans le graphe de [Dun95]. Ceci se voit clairement sur l'exemple :

Exemple 3.1 (suite). Le graphe orienté (V, E) associé à P selon [DT96] est tel que :

$$\begin{array}{rcl} V & \supseteq & \{\Delta_A, \Delta_B, \Delta_C\}, \\ E & \supseteq & \{(\Delta_B, \Delta_A), (\Delta_C, \Delta_B)\}. \end{array}$$

Ce graphe est représenté sur la figure 3.2. $\diamondsuit$

Fig. 3.2 – Représentation partielle du système d'argumentation correspondant, selon [DT96], au programme logique de l'exemple 3.1.

Dimopoulos et Torres ont montré dans [DT96] qu'il existe une correspondance entre les modèles stables, les interprétations bien-fondées et les modèles partiels stables d'un programme logique P, et respectivement les noyaux, semi-noyaux et semi-noyaux maximaux d'un graphe orienté (V, E) associé à P. Ils ont également montré qu'il existe une correspondance entre le modèle bien-fondé de P et la notion de *partie initiale acyclique*[2] de la théorie des graphes.

Exemple 3.1 (suite). L'ensemble $\{\Delta_A, \Delta_C\}$ est inclus dans le seul semi-noyau maximal du graphe orienté associé au programme logique selon [DT96]. L'interprétation supportée de l'ensemble $\Delta_A \cup \Delta_C$ est $\{\neg a(Zoé, v(Zoé), d(Zoé), o(Zoé)\}$. Cet ensemble correspond au modèle partiel stable et au modèle stable du programme. ♢

3.3 La logique des défauts

Dans cette section, nous allons présenter le lien établi par Dimopoulos et Magirou dans [DM94] entre la logique des défauts de Reiter et la théorie des graphes. Le graphe orienté représentant une théorie avec défauts peut être vu comme une instanciation du système d'argumentation de Dung.

Nous allons tout d'abord définir la logique des défauts avant d'évoquer son lien avec la théorie des graphes.

3.3.1 Définition

La logique des défauts a été introduite par Reiter dans [Rei80] dans le but d'augmenter la logique du premier ordre avec un ensemble d'assertions par défaut. Elle permet de traiter des connaissances avec exceptions.

Définition 3.18 (Théorie avec défauts).
Une théorie avec défauts *est une paire* (W, D) *avec :*

- W *un ensemble fini de formules fermées du premier ordre, et*
- D *un ensemble fini de* défauts.

Un défaut a la forme générale :

$$u(x) : v_1(x), \ldots, v_n(x)/w(x)$$

où $u(x)$, $v_1(x) \ldots v_n(x)$ *et* $w(x)$ *sont des formules. Intuitivement, un tel défaut signifie que :*

> *si je crois* $u(x)$*, et que rien ne m'empêche de croire* $v_1(x) \ldots v_n(x)$*, alors je crois* $w(x)$*.*
> $u(x)$ *est appelé* prérequis *du défaut,* $v_1(x) \ldots v_n(x)$ *sont les* justifications *du défaut et* $w(x)$ *est le* conséquent*.*

Exemple 3.2. En gardant le même vocabulaire que dans l'exemple 3.1 page 37, l'exemple *Zoé l'autruche* (exemple 1.1 page 4) peut être traduit dans une théorie avec défauts (W, D) telle que :

- $W = \{o(Zoé), d(Zoé)\}$, et
- $D = \{o(x) : \neg a(x)/v(x), \top : \neg d(x)/a(x)\}$[3].

Le défaut $o(x) : \neg a(x)/v(x)$ signifie que si x est un oiseau, et si rien n'empêche de croire que x n'est pas une autruche, alors x vole. Le défaut $\top : \neg d(x)/a(x)$ signifie que si rien n'empêche de croire que l'observation qui a été faite de x ne l'a pas été avec des instruments défectueux, alors x est une autruche. ♢

À partir d'une théorie des défauts, on veut construire des ensembles de croyances consistants, que l'on appelle des *extensions*.

Une théorie est dite *fermée* si tous les défauts sont fermés (*i.e.* il n'y a pas de variable libre).

Les extensions peuvent être caractérisées comme suit :

Théorème 3.3 (Extensions d'une théorie avec défauts). *Étant donné* (W, D) *une théorie fermée,* E *est une extension de* (W, D) *si et seulement si* $E = \bigcup_{i \geq 0} E_i$ *avec :*

- $E_0 = W$*, et*

---

[2] Voir [DT96] pour la définition de la notion de *partie initiale acyclique*.

[3] La négation a une signification classique : $\neg v(x)$ signifie que $v(x)$ est faux.

- $E_{i+1} = Th(E_i) \cup \{w \mid u : v_1, \ldots, v_n/w \in D \text{ et } u \in E_i, \neg v_j \notin E \text{ pour } 1 \leq j \leq n\}$ *où* $Th(E_i)$ *est l'ensemble des conséquences logiques de* E_i.

Exemple 3.2 (suite). La théorie (W, D) possède une seule extension construite comme suit :

- $E_0 = \{o(Zoé), d(Zoé)\}$;
- Le prérequis des deux défauts appartient à E_0, mais comme $d(Zoé) \in Th(E_0)$, la justification du défaut $\top : \neg d(Zoé)/a(Zoé)$ est contredite, ce défaut n'est donc pas «applicable». Quant à l'autre défaut :
  - Si on suppose que $a(Zoé) \notin E$, alors $E_1 = Th(E_0) \cup \{v(Zoé)\}$. Par suite, $E_2 = Th(E_1)$. Donc $E = \bigcup_{i \geq 0} E_i = Th(E_0 \cup \{v(Zoé)\})$ est une *extension* de la théorie.
  - Si on suppose que $a(Zoé) \in E$, alors $E_1 = Th(E_0)$. Par suite, $E_2 = E_1$ et $E = \bigcup_{i \geq 0} E_i = Th(E_0)$. Mais $a(Zoé) \notin E$: contradiction avec la supposition qui avait été faite. Donc $Th(E_0)$ n'est pas une extension de la théorie.

♢

D'autres sémantiques ont été proposées pour la logique des défauts. En particulier, celles proposées par Dimopoulos et Torres [DT96] généralisent les sémantiques de la programmation logique.

3.3.2 Lien avec la théorie des graphes

Dimopoulos et Magirou se sont intéressés dans [DM94] au lien entre la théorie des graphes et des théories avec défauts sans-prérequis et sans disjonction. Ils ont associé à une théorie avec défauts (fermée) (W, D) un graphe $G_{(W,D)} = (V, E)$ tel que :

- W est un sommet et chaque défaut de D est un sommet ;
- étant donné deux défauts d_i et d_j, un arc $(d_i, d_j) \in E$ si la négation d'une proposition dans le conséquent du défaut d_i appartient à la justification de d_j ; de plus, il existe un arc entre W et un défaut d si la négation d'un littéral de W apparaît dans la justification de d.

Dimopoulos et Magirou ont montré en particulier que les extensions de (W, D) peuvent être calculées à partir des noyaux de $G_{(W,D)}$ (le nœud W appartient au noyau puisqu'il n'a pas de prédécesseur ; l'extension de (W, D) est égale à l'union de W et des conséquents des défauts qui appartiennent au noyau). Dimopoulos et Torres [DT96] ont étendu ces résultats pour leurs nouvelles sémantiques.

Exemple 3.2 (suite). Le graphe associé à la théorie avec défauts est tel que :

$$\begin{tikzcd}
\{o(Zoé), d(Zoé)\} \arrow[d] \\
\top : \neg d(Zoé)/a(Zoé) \arrow[d] \\
o(Zoé) : \neg a(Zoé)/v(Zoé)
\end{tikzcd}$$

Le noyau de ce graphe est composé des sommets $\{o(Zoé), d(Zoé)\}$ et $o(Zoé) : \neg a(Zoé)/v(Zoé)$. L'extension de la théorie est $\{o(Zoé), d(Zoé), v(Zoé)\}$. ♢

3.4 Résumé des liens entre les différents formalismes

Nous avons vu en section 3.1 le lien entre le système BDKT et le système de Dung, ainsi qu'avec les formalismes pour le raisonnement révisable.

Nous avons décrit en section 3.2.2 le lien entre les sémantiques de Dung et celles de la programmation logique établi par [Dun95] ; en section 3.2.3, nous avons montré le lien établi entre la programmation logique et la théorie des graphes par [DT96]. En section 2.2, nous avions établi le lien entre les sémantiques de Dung et la théorie des graphes.

Nous avons par ailleurs évoqué en section 3.3 le lien entre la sémantique classique de la logique des défauts et une notion de la théorie des graphes.

Ces différents liens sont résumés sur le schéma de la figure 3.3. Une flèche entre deux blocs indique que les items numérotés x de chaque bloc correspondent à des notions équivalentes.

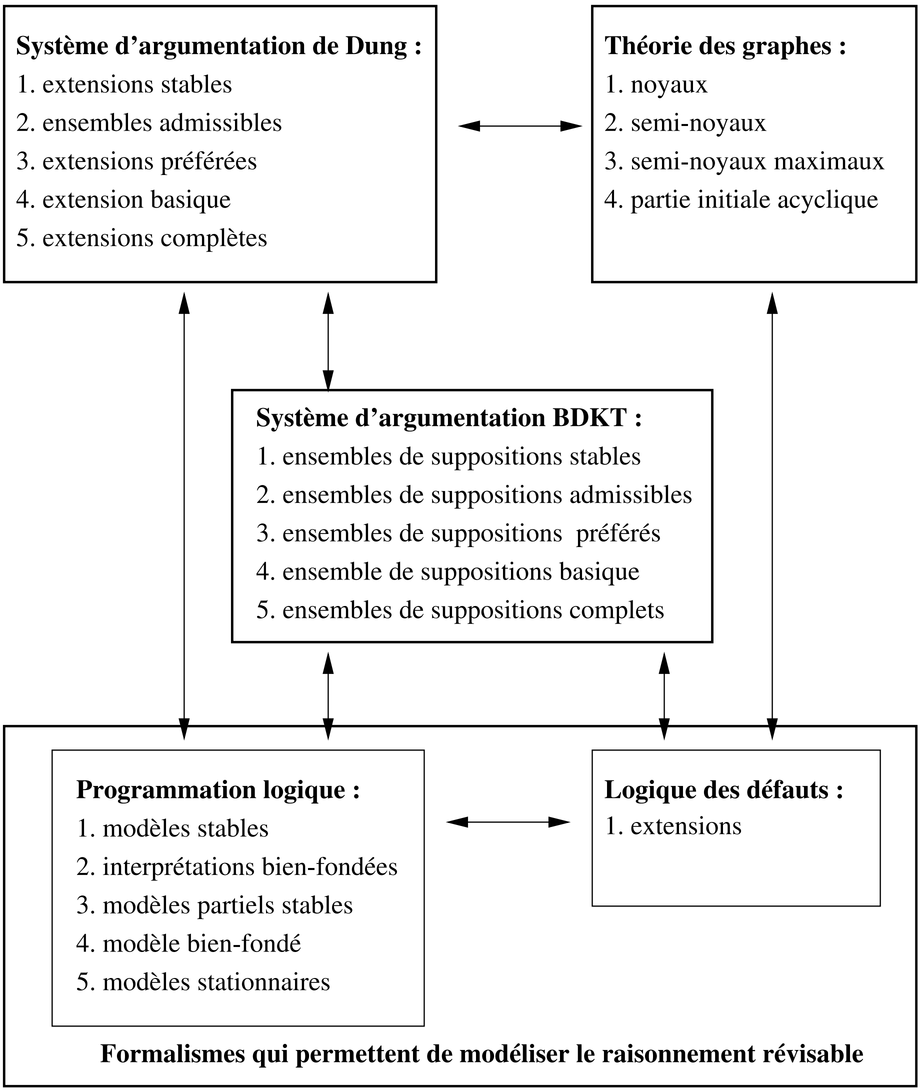


FIG. 3.3 – Liens entre les sémantiques de Dung, celles de BDKT, celles de formalismes pour le raisonnement révisable et la théorie des graphes.

Deuxième partie

Théories de la preuve et algorithmes

Chapitre 4

Théories de la preuve dialectiques

Pour répondre aux problèmes d'acceptabilité crédule et sceptique sous la sémantique préférée dans le système d'argumentation de Dung, nous proposons dans ce chapitre des théories de la preuve dialectiques, conformes à la structure de preuve suggérée en section 2.4 : les preuves prennent la forme d'un dialogue entre un partisan et un opposant ; le partisan commence par proposer l'argument dont il faut prouver l'acceptabilité, et tente de le défendre contre toute attaque venant de l'opposant. Ce dialogue est en fait un jeu entre deux joueurs, dont les règles précises et les conditions de gain dépendent du problème à résoudre.

Prakken a résumé les caractéristiques générales des théories de la preuve dialectiques dans [Pra01] et a proposé un cadre dialectique qui satisfait ces caractéristiques. Il illustre ce cadre en donnant une théorie de la preuve pour le problème d'acceptabilité sous la sémantique basique (proche de la théorie qu'il avait proposée dans [PS97]). Amgoud et Cayrol ont également proposé dans [AC02] une théorie de la preuve pour la sémantique basique. Nous comparerons ces deux théories de la preuve dans la section 4.3.

Jakobovits et Vermeir ont formalisé dans [JV99a] des théories de la preuve dialectiques qui permettent de prouver l'acceptabilité des arguments sous les sémantiques «robuste» et «défendable». Le cadre formel de Jakobovits et Vermeir précise davantage les conditions de gain que celui de [Pra01].

Vreeswijk et Prakken ont proposé dans [VP00] des théories de la preuve dialectiques (appelées «disputes entre deux partis avec réponse immédiate», ou TPI-disputes) pour les problèmes d'acceptabilité crédule et sceptique sous la sémantique préférée. Ces théories de la preuve dialectiques ont été formalisées par Dunne et Bench-Capon dans [DBC01c], mais leur formalisation ne s'inscrit pas dans le cadre de [Pra01], ni dans celui de [JV99a].

En simplifiant quelques définitions de [JV99a], nous décrivons en section 4.1 un cadre dialectique formel qui va nous permettre de donner en section 4.2 deux théories de la preuve pour le problème d'acceptabilité crédule. En section 4.3, nous donnons deux théories de la preuve pour la sémantique basique. En section 4.4, nous utilisons ces théories de la preuve pour répondre, dans plusieurs cas particuliers, au problème d'acceptabilité sceptique. En section 4.5, nous présentons quelques travaux voisins et nous comparons nos théories de la preuve avec celles de [VP00] ; nous montrerons en particulier que les preuves que l'on obtient sont un peu moins explicites mais plus courtes que les TPI-disputes.

Ces travaux ont été présentés dans [CDM01a, CDM01b, CDM02].

4.1 Cadre dialectique formel

Selon [JV99a], une *théorie de la preuve dialectique* est la combinaison d'un *type de dialogue* et d'un *critère de gain*. Elle permet de définir des *preuves dialectiques* ; de telles preuves prennent la forme d'un dialogue ou d'un ensemble de dialogues. La forme des coups et les coups autorisés dans un dialogue sont définis par le type de dialogue. Le critère de gain donne les conditions sous lesquelles le dialogue (ou l'ensemble de dialogues) définit une preuve.

Dans tout ce chapitre, nous faisons l'hypothèse que les systèmes d'argumentation avec lesquels nous travaillons sont finis.

4.1.1 Type de dialogue

Les deux joueurs qui dialoguent sont le partisan, nommé dans la suite PRO, et l'opposant, nommé OPP. Un type de dialogue est défini ainsi :

Définition 4.1 (Type de dialogue).
Un type de dialogue *est un triplet* (A, R, ϕ) *où* (A, R) *est un système d'argumentation et* $\phi : A^* \to 2^A$ *est une fonction appelée* fonction coup-légal*.*

Un coup *dans* A *est une paire* $[J, X]$ *où* $J \in \{\mathrm{PRO}, \mathrm{OPP}\}$ *et* $X \in A$*. Étant donné un coup* $\mu = [J, X]$*, on utilise* $\mathrm{jo}(\mu)$ *pour dénoter* J *et* $\mathrm{arg}(\mu)$ *pour dénoter* X*.*

Un dialogue d *de* (A, R, ϕ) *(ou* ϕ-dialogue*) est une séquence dénombrable* $\mu_0\mu_1\ldots$ *de coups dans* A *telle que :*

1. $\mathrm{jo}(\mu_0) = \mathrm{PRO}$ *;*
2. $\mathrm{jo}(\mu_i) \neq \mathrm{jo}(\mu_{i+1})$ *;*
3. $\mathrm{arg}(\mu_{i+1}) \in \phi(\mathrm{arg}(\mu_0)\ldots\mathrm{arg}(\mu_i))$*.*

On dit que d porte sur *l'argument* $\mathrm{arg}(\mu_0)$*. Si* d *est fini, la* longueur *de* d *est égale au nombre de coups qui composent* d*.*

Dans un ϕ-dialogue, les joueurs jouent donc à tour de rôle, et c'est PRO qui commence le jeu. La fonction coup-légal définit pour chaque coup les arguments qui peuvent être utilisés pour continuer un dialogue. Lorsque l'ensemble des arguments retourné par cette fonction est vide, c'est que le dialogue ne peut pas être poursuivi.

À noter que dans la définition d'origine de [JV99a], ce sont des ensembles d'arguments sans-conflits qui sont avancés dans les coups alors que nous imposons dans la définition ci-dessus qu'un seul argument à la fois soit avancé. La raison de cette restriction est que nous voulons rendre nos dialogues proches des TPI-disputes afin de pouvoir aisément les comparer.

Nous introduisons à présent plusieurs notations.

Notation 4.1. *Étant donné* $d = \mu_0\mu_1\ldots\mu_i$ *un* ϕ*-dialogue fini :*
- μ_i *est dénoté par* $\mathrm{der}(d)$ *;*
- $\phi(\mathrm{arg}(\mu_0)\ldots\mathrm{arg}(\mu_i))$ *est dénoté par* $\phi(d)$ *;*
- $\mathrm{PRO}(d)$ *dénote l'ensemble des arguments avancés par* PRO *durant* d *;*
- *l'extension de* d *avec un coup* μ *dans* A *telle que* $\mu_0\mu_1\ldots\mu_i\mu$ *est un* ϕ*-dialogue est dénotée par la juxtaposition* $d.\mu$*.*

La relation R du système d'argumentation va intervenir dans la définition de la fonction coup-légal. Cette relation va permettre d'imposer des restrictions comme par exemple :
- un argument avancé dans un coup doit attaquer l'argument avancé dans le coup précédent ;
- PRO ne peut répéter un argument qu'il a déjà avancé ;
- aucun joueur ne peut avancer un argument qui s'auto-attaque ;
- ...

On peut noter que la première restriction est imposée dans [JV99a]. Nous ne l'imposons pas afin de pouvoir décrire des preuves séquentielles pour le problème d'acceptabilité crédule ; en effet, il faudra que OPP puisse attaquer un argument précédemment joué dans le dialogue par PRO, sans que ce soit obligatoirement le dernier argument.

4.1.2 Critère de gain

Le type de dialogue étant défini, il faut donner à présent des *critères de gain* pour déterminer les arguments qui peuvent être défendus avec succès par les ϕ-dialogues. On considérera seulement deux critères de gain parmi ceux proposés par [JV99a] :

1. un dialogue donné portant sur un argument x peut être gagné par PRO, ou
2. il peut exister une stratégie gagnante pour x, c'est-à-dire une façon pour PRO de défendre x contre toutes les attaques de OPP.

Ces critères concernent des dialogues *finis*, ils vont nous permettre de comparer nos travaux avec d'autres travaux[1].

Le premier critère est formellement défini comme suit :

Définition 4.2 (Dialogue gagné par PRO [JV99a]).
Soit (A, R, ϕ) *un type de dialogue. Un* ϕ*-dialogue* d *est* gagné par PRO *si et seulement si* d *est fini,* d *ne peut être continué (*i.e. $\phi(d) = \emptyset$*) et* $\text{jo}(\text{der}(d)) = \text{PRO}$.

Pour définir le deuxième critère de gain, nous avons besoin de quelques notions supplémentaires :

Définition 4.3 (Préfixe, extension).
La séquence y *est un* préfixe *de la séquence* x *ou* x *est une* extension *de* y *si et seulement si il existe une séquence* z *telle que* x *est obtenue par la concaténation de* y *et* z*,* i.e. $x = y.z$. *Un ensemble* S *de séquences est* préfixe-fermé *si* $\forall x \in S$*,* $\forall y$ *préfixe de* x*,* $y \in S$.

Le deuxième critère de gain, appelé stratégie gagnante, a été défini dans [JV99a] comme suit :

Définition 4.4 (Stratégie gagnante selon [JV99a]).
Soit (A, R, ϕ) *un type de dialogue. Une* stratégie gagnante *pour un argument* $x \in A$ *est un ensemble* S *fini, non vide, préfixe-fermé, de* ϕ*-dialogues finis portant sur* x *tel que :*

1. $\forall d \in S$*,* $\exists d'$ *extension de* d *tel que* d' *est gagné par* PRO *;*
2. $\forall d.[\text{PRO}, a] \in S$*,* $\forall b \in \phi(d.[\text{PRO}, a])$*,* $\exists d' \in S$ *tel que* d' *est gagné par* PRO *et* d' *est une extension de* $d.[\text{PRO}, a].[\text{OPP}, b]$.

La condition 1 n'est pas nécessaire dans cette définition puisque la condition 2 implique la condition 1 lorsque l'ensemble S est préfixe-fermé. En voici la preuve :

> PREUVE – Supposons que S soit un ensemble fini préfixe-fermé de ϕ-dialogues portant sur x satisfaisant la condition 2. Soit $d \in S$.
>
> Si $\text{jo}(\text{der}(d)) = \text{PRO}$ (autrement dit, si le dernier coup de d est joué par PRO) et d ne peut être continué, alors on prend $d' = d$.
>
> Si $\text{jo}(\text{der}(d)) = \text{PRO}$ et $\phi(d) \neq \emptyset$, d est de la forme $d_1.[\text{PRO}, a]$. Puisque la condition 2 est satisfaite, $\exists b \in \phi(d)$, $\exists d' \in S$ tel que d' est gagné par PRO et d' est une extension de $d.[\text{OPP}, b]$. Donc il existe une extension de d gagnée par PRO.
>
> Si $\text{jo}(\text{der}(d)) = \text{OPP}$, d est de la forme $d_1.[\text{OPP}, b]$ avec $d_1 \in S$ puisque S est préfixe-fermé. $\text{jo}(\text{der}(d_1)) = \text{PRO}$ et $b \in \phi(d_1)$; donc, d'après la condition 2, il existe une extension de $d_1.[\text{OPP}, b]$ (*i.e.* une extension de d) gagnée par PRO. □

Prenant en compte cette remarque, nous proposons une définition plus simple pour la stratégie gagnante dans laquelle il n'est pas non plus nécessaire que S soit préfixe-fermé :

Définition 4.5 (Stratégie gagnante).
Soit (A, R, ϕ) *un type de dialogue. Une* stratégie gagnante *(ou* ϕ-stratégie gagnante*) pour un argument* $x \in A$ *est un ensemble* S *fini non vide de* ϕ*-dialogues portant sur* x *tels que :* $\forall d \in S, \forall d'$ *préfixe de* d *tel que* $\text{jo}(\text{der}(d')) = \text{PRO}$*,* $\forall y \in \phi(d')$*,* $\exists d'' \in S$ *tel que* d'' *est gagné par* PRO *et* d'' *est une extension de la juxtaposition* $d'.[\text{OPP}, y]$.

---

[1] Le troisième critère de gain de [JV99a] autorise des coups gagnants même pour des dialogues infinis ; ce cas sort du cadre de notre étude.

Les définitions 4.4 et 4.5 sont équivalentes.

PREUVE – Nous montrons que les assertions suivantes sont équivalentes :

1. Il existe un ensemble S fini non vide de ϕ-dialogues finis portant sur x tel que : $\forall d \in S, \forall d'$ préfixe de d tel que $\text{jo}(\text{der}(d')) = \text{PRO}$, $\forall y \in \phi(d')$, $\exists d'' \in S$ tel que d'' est gagné par PRO et d'' est une extension de la juxtaposition $d'.[\text{OPP}, y]$.
2. Il existe un ensemble S' fini non vide préfixe-fermé de ϕ-dialogues finis portant sur x tel que : $\forall d.[\text{PRO}, a] \in S'$, $\forall b \in \phi(d.[\text{PRO}, a])$, $\exists d' \in S'$ tel que d' est gagné par PRO et d' est une extension de $d.[\text{PRO}, a].[\text{OPP}, b]$.

(1 ⇒ 2) On définit S' comme étant l'ensemble des ϕ-dialogues portant sur x qui sont préfixes d'un élément de S. S' est préfixe-fermé.

Soit $d.[\text{PRO}, a] \in S'$ et $b \in \phi(d.[\text{PRO}, a])$. Il existe $d_1 \in S$ tel que $d.[\text{PRO}, a]$ est un préfixe de d_1. En appliquant 1, il existe $d'' \in S$, d'' gagné par PRO et d'' extension de $d.[\text{PRO}, a].[\text{OPP}, b]$. Par définition de S', d'' appartient également à S'.

(2 ⇒ 1) Prenons $S = S'$.

Soit $d \in S$. Soit d' préfixe de d tel que $\text{der}(d')$ est joué par PRO. Puisque S est préfixe-fermé, $d' \in S$. Par 2, $\forall b \in \phi(d')$, $\exists d'' \in S$ tel que d'' est gagné par PRO et d'' est une extension de $d'.[\text{OPP}, b]$. □

Le résultat suivant donne encore une autre caractérisation des stratégies gagnantes, dans laquelle seuls les dialogues qui ne peuvent être continués sont considérés :

Propriété 4.1. *Il existe une ϕ-stratégie gagnante pour x si et seulement si il existe un ensemble S fini non vide de ϕ-dialogues finis portant sur x gagnés par* PRO *tel que : $\forall d \in S$, $\forall d'$ préfixe de d tel que* $\text{der}(d')$ *est joué par* PRO*, $\forall y \in \phi(d')$, $\exists d'' \in S$ tel que d'' est une extension de la juxtaposition* $d'.[\text{OPP}, y]$.

PREUVE – La preuve de la condition suffisante est évidente. Pour la condition nécessaire, supposons qu'il existe une ϕ-stratégie gagnante S pour x. Nous définissons $S' = \{d \in S \mid \phi(d) = \emptyset\}$. Il est suffisant de prouver que pour chaque dialogue $d \in S'$, $\text{der}(d)$ est joué par PRO.

Supposons qu'il existe $d \in S'$ tel que $\text{der}(d)$ est joué par OPP, *i.e.* $d = d'.[\text{OPP}, b]$. Par définition de S', $d \in S$. d' est un préfixe de d, $\text{der}(d')$ est joué par PRO et $b \in \phi(d')$. Par conséquent, $\exists d'' \in S$ tel que d'' est gagné par PRO et d'' est une extension de $d'.[\text{OPP}, b] = d$. d'' est différent de d puisque $\text{der}(d'')$ est joué par PRO. Contradiction avec le fait que $\phi(d) = \emptyset$. □

Cette dernière caractérisation des stratégies gagnantes est la plus simple. C'est celle que nous allons utiliser dans toute la suite.

4.1.3 Théorie de la preuve

Le type de dialogue et les critères de gain étant définis, précisons ce que sont une théorie de la preuve et une preuve :

Définition 4.6 (ϕ-théorie de la preuve, ϕ-preuve).
Une ϕ-théorie de la preuve *est la donnée d'un type de dialogue (A, R, ϕ) et d'un critère de gain. Étant donné une ϕ-théorie de la preuve, une* ϕ-preuve *est un ϕ-dialogue ou un ensemble de ϕ-dialogues qui satisfont le critère de gain. S'il n'y a pas d'ambiguïté sur le type de dialogue, on parle simplement de théorie de la preuve ou de preuve.*

Pour définir une théorie de la preuve pour un problème donné, il faut donc préciser la fonction coup-légal et choisir un critère de gain.

Dans la section 4.2, nous allons définir des théories de la preuve pour le problème d'acceptabilité crédule. Nous préciserons tout d'abord la fonction coup-légal du type de dialogue, puis en section 4.2.1, nous définirons une théorie de la preuve qui utilise le critère de gain «dialogue gagné par PRO». En section 4.2.2 nous définirons une théorie de la preuve qui utilise le deuxième critère de gain, *i.e.* les stratégies gagnantes.

4.1.4 Représentation des preuves

Les théories de la preuve qui utilisent le critère de gain «stratégie gagnante» donnent des preuves qui sont des ensembles de dialogues. Nous allons représenter ces ensembles sous forme d'*arbres.* Pour cela, nous rappelons brièvement la terminologie de base sur les arbres.

Définition 4.7 (Arbre).
Un arbre *est un ensemble d'éléments appelés* nœuds *dont un se singularise comme la* racine*, liés par une relation (dite de «parenté») induisant une structure hiérarchique parmi ces nœuds. Un arbre est défini grâce aux deux règles suivantes :*

1. *Un nœud unique, par lui-même, est un arbre. Dans ce cas, il est aussi la racine de l'arbre.*
2. *Si n est un nœud et $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ sont des arbres de racine respective $n_1, n_2, \ldots, n_k$, on peut construire un nouvel arbre en associant comme parent unique aux nœuds $n_1, n_2, \ldots, n_k$ le nœud n. Dans cet arbre, n est la racine et $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ sont les* sous-arbres *de cette racine. Les nœuds $n_1, n_2, \ldots, n_k$ sont appelés les* fils *de n, et n est appelé le* parent *de $n_1, n_2, \ldots, n_k$.*

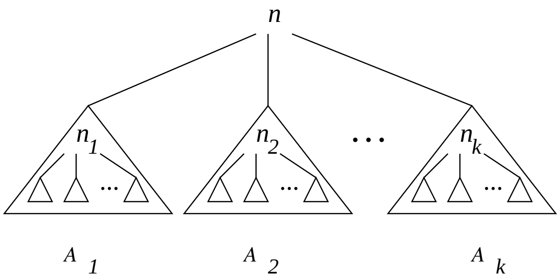


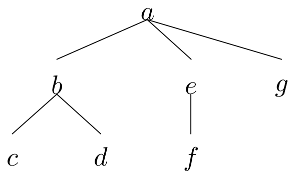
Fig. 4.1 – Un arbre $\mathcal{A}$.

Définition 4.8 (Branche et feuille d'un arbre).
Si $n_1, n_2, \ldots, n_k$ est une suite de nœuds d'un arbre telle que n_i est le parent de n_{i+1} pour $1 \leq i < k$, cette suite est appelée chemin *ou* branche *du nœud n_1 au nœud n_k. On dit que n_1 est un* ascendant *de n_k et réciproquement que n_k est un* descendant *de n_1. La* longueur *d'une branche est égale au nombre de nœuds qu'elle contient moins un.*

Dans un arbre, seule la racine n'a pas d'ascendant. Un nœud sans descendant est appelé une feuille.

La profondeur *d'un nœud est la longueur de la branche entre la racine et le nœud, la profondeur de la racine est 0.*

Exemple 4.1. Considérons l'arbre suivant :



La racine de cet arbre est a. abc est une branche de a à c de longueur 2. ag est une branche de a à g de longueur 1. Les feuilles de l'arbre sont c, d, f et g. c est un nœud de profondeur 2, g est un nœud de profondeur 1, a est de profondeur 0. ♢

Une stratégie gagnante pour un argument x va être représentée sous forme d'un arbre tel que :

- les nœuds de l'arbre seront des coups : les nœuds de profondeur paire seront des coups joués par PRO, ceux de profondeur impaire seront des coups joués par OPP ;
- la racine sera le coup $[\text{PRO}, x]$;
- les feuilles seront toutes de profondeur paire, autrement dit, ce seront toutes des coups joués par PRO ;
- les dialogues qui composent la stratégie correspondront aux chemins de la racine aux feuilles de l'arbre.

Un dialogue gagné par PRO peut être représenté sous forme d'un arbre qui satisfait les conditions ci-dessus mais qui est réduit à une branche.

4.2 Acceptabilité crédule sous la sémantique préférée

Le problème d'acceptabilité crédule consiste à décider si un argument donné x appartient à au moins une extension préférée. Comme nous l'avons déjà mentionné en section 2.4, une preuve que x est crédulement accepté est un ensemble d'arguments admissible qui contient x. Pour que la preuve soit plus explicite, cet ensemble admissible est accompagné de l'ensemble des attaquants contre lesquels il se défend. Les preuves que nous allons proposer dans cette section mettent en évidence la manière dont l'ensemble admissible se défend contre ses attaquants.

Les ϕ-dialogues introduits dans la section précédente permettent de distinguer ces deux ensembles : dans un ϕ-dialogue qui établit une preuve crédule, les arguments de l'ensemble admissible seront ceux joués par PRO, les arguments qui attaquent cet ensemble admissible seront ceux joués par OPP. Pour établir une preuve de l'acceptabilité crédule d'un argument x, la fonction ϕ d'un type de dialogue devra satisfaire les conditions suivantes :

1. tout coup $[\text{OPP}, z]$ sera tel que z attaque individuellement directement ou indirectement x ;
2. tout coup $[\text{PRO}, y]$ (excepté le premier coup) sera tel que y défend individuellement directement ou indirectement x ;
3. tout coup $[\text{OPP}, z]$ sera tel qu'il existe $[\text{PRO}, y]$ un coup précédemment avancé dans le dialogue et z attaque y ;
4. tout coup $[\text{PRO}, w]$ sera tel que le coup qui le précède immédiatement dans le dialogue est de la forme $[\text{OPP}, z]$ et w attaque z.

La condition 4 souligne que les arguments avancés par PRO devront attaquer les arguments de OPP immédiatement précédents ; comme les arguments de OPP attaquent des arguments déjà avancés par PRO (condition 3), cela signifie que les arguments de PRO défendent les arguments de PRO précédemment avancés. On peut dire que les arguments de OPP, en attaquant ceux de PRO, obligent PRO à montrer comment il se défend contre ses attaquants.

Les figures 4.2 page suivante et 4.3 page ci-contre indiquent des schémas de ϕ-dialogues qui respectent les conditions ci-dessus. Une flèche d'un argument vers un autre indique que le premier argument attaque le deuxième.

La fonction ϕ peut être raffinée, afin d'arriver au plus vite (*i.e.* en minimisant le nombre de coups) à ce que l'ensemble des arguments de PRO soit un ensemble admissible contenant x.

Tout d'abord, une condition nécessaire pour qu'un ensemble soit admissible, est qu'il soit sans-conflit. L'ensemble des arguments avancés par PRO dans un dialogue doit donc être sans-conflit. Ainsi, pour continuer un ϕ-dialogue fini d, PRO doit proposer un argument qui est sans-conflit avec l'ensemble des arguments qu'il a déjà proposés dans d. Par conséquent, PRO ne doit avancer ni un argument qui attaque ou qui est attaqué par un argument qu'il a déjà avancé dans d, ni un argument qui s'auto-attaque.

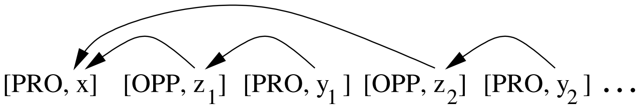


FIG. 4.2 – Schéma d'un ϕ-dialogue pour la preuve de l'acceptabilité crédule d'un argument x, où OPP répond à un coup précédent de PRO.

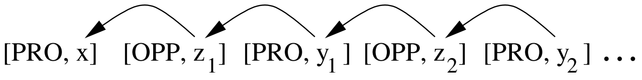


FIG. 4.3 – Schéma d'un ϕ-dialogue pour la preuve de l'acceptabilité crédule d'un argument x, où OPP répond au coup immédiatement précédent de PRO.

Par ailleurs, les arguments avancés par OPP doivent attaquer les arguments de PRO pour forcer ces derniers à montrer comment ils se défendent contre toute attaque. Néanmoins, pour continuer un ϕ-dialogue fini, il est inutile que OPP avance un argument qui est déjà attaqué par PRO, puisque PRO se défend déjà contre cet argument.

Sous cette condition, pour continuer un ϕ-dialogue fini d de longueur impaire, on peut noter que OPP ne répétera jamais un argument qu'il a déjà avancé. En effet, si OPP a joué dans d un argument z et si PRO a répondu en proposant un attaquant de z, alors l'argument z est attaqué et OPP ne peut pas le proposer à nouveau.

De plus, si PRO a avancé dans un ϕ-dialogue d un argument y pour attaquer un argument z joué par OPP, aucun des autres arguments qui sont attaqués par y ne seront joués par OPP dans la suite du dialogue, puisque OPP n'a pas à proposer un argument qui est déjà attaqué par PRO. Donc l'argument y, et plus généralement les arguments avancés par PRO dans d, n'ont pas à être répétés par PRO pour continuer d.

En résumé, pour continuer un ϕ-dialogue d fini, PRO choisira ses arguments dans l'ensemble POSS(d) défini comme suit :

$$\mathrm{POSS}(d) = A \setminus (\mathrm{PRO}(d) \cup R^{\pm}(\mathrm{PRO}(d)) \cup \mathrm{Refl}).$$

Concernant les arguments qui s'auto-attaquent, il n'est clairement pas possible de trouver un ensemble admissible les contenant. Il est donc inutile de chercher à construire une preuve de l'acceptabilité de tels arguments.

Ces contraintes étant posées, nous définissons deux théories de la preuve.

4.2.1 La ϕ_1-théorie de la preuve

Dans les ϕ_1-preuves, il n'est pas nécessaire que l'argument proposé par OPP réponde à l'argument précédemment proposé par PRO. Les ϕ_1-dialogues respectent donc le schéma de dialogue de la figure 4.2.

Définition 4.9 (Fonction coup-légal de type ϕ_1).
Soit (A, R) un système d'argumentation. $\phi_1 : A^* \to 2^A$ est définie par :

- *si d est un ϕ_1-dialogue de longueur impaire (*i.e. *le prochain coup doit être joué par* OPP*),*

$$\phi_1(d) = R^-(\mathrm{PRO}(d)) \setminus R^+(\mathrm{PRO}(d)) \ ;$$

- *si d est un ϕ_1-dialogue de longueur paire (*i.e. *le prochain coup doit être joué par* PRO*),*

$$\phi_1(d) = R^-(\mathrm{arg}(\mathrm{der}(d))) \cap \mathrm{POSS}(d).$$

En d'autres termes :

- dans les coups joués par OPP, l'argument avancé doit attaquer un argument précédemment avancé par PRO dans le dialogue (pas nécessairement le dernier) mais il ne doit être attaqué par aucun argument déjà avancé par PRO.
- dans les coups joués par PRO, l'argument avancé doit attaquer l'argument qui le précède (proposé par OPP), mais il ne doit ni avoir déjà été proposé par PRO, ni s'auto-attaquer, ni être en conflit avec les arguments déjà proposés par PRO.

La combinaison d'un type de dialogue (A, R, ϕ_1) et du critère de gain de la définition 4.2 page 49 permet d'obtenir des ϕ_1-théories de la preuve. Les ϕ_1-preuves sont définies ainsi :

Définition 4.10 (ϕ_1-preuve).
Soit un type de dialogue (A, R, ϕ_1) et un argument $x \in A$ qui ne s'auto-attaque pas. Une ϕ_1-preuve *pour x est un ϕ_1-dialogue portant sur x gagné par* PRO*.*

Les résultats suivants établissent la correction et la complétude des ϕ_1-preuves pour le problème d'acceptabilité crédule sous la sémantique préférée :

Propriété 4.2 (Correction des ϕ_1-preuves). *Si d est une ϕ_1-preuve pour un argument x, alors* PRO(d) *est un ensemble admissible contenant x.*

Lemme 4.1. *Étant donné un type de dialogue (A, R, ϕ_1) et un ϕ_1-dialogue d,* PRO(d) *est sans-conflit.*

Preuve – Nous prouvons ce résultat par induction sur le nombre d'éléments avancés par PRO dans un ϕ_1-dialogue.

- Si PRO a avancé un seul argument dans un ϕ_1-dialogue, alors l'ensemble des arguments avancés par PRO dans ce ϕ_1-dialogue est évidemment sans-conflit.
- Supposons maintenant que la propriété est vraie pour tout ϕ_1-dialogue tel que l'ensemble des arguments avancés par PRO contient au plus $n - 1$ éléments, avec $n > 1$ (*hypothèse d'induction*). Soit d un ϕ_1-dialogue tel que PRO(d) contient n éléments.

  Supposons tout d'abord que le dernier coup de d est joué par PRO ; d est donc de la forme $d = d'.[\text{OPP}, x].[\text{PRO}, y]$ où :
  - $x \in \phi_1(d') = R^-(\text{PRO}(d')) \setminus R^+(\text{PRO}(d'))$;
  - $y \in \phi_1(d'.[\text{OPP}, x]) = R^-(x) \cap \text{POSS}(d'.[\text{OPP}, x]) = R^-(x) \setminus (\text{PRO}(d') \cup \text{Refl} \cup R^{\pm}(\text{PRO}(d'))$.

  Alors $\text{PRO}(d) = \text{PRO}(d') \cup \{y\}$, et $y \notin \text{PRO}(d')$, et donc d' contient strictement moins de n éléments. Ainsi, par l'hypothèse d'induction, PRO(d') est sans-conflit. Puisque $y \notin R^{\pm}(\text{PRO}(d'))$, PRO$(d)$ est également sans-conflit.

  Supposons maintenant que le dernier élément de d est joué par OPP ; d est donc de la forme $d = d'.[\text{OPP}, x]$ où d' est un ϕ_1-dialogue tel que $\text{PRO}(d') = \text{PRO}(d)$. Par conséquent, PRO$(d')$ contient n éléments. Nous venons de prouver que dans ce cas PRO(d') est sans-conflit, donc PRO(d) est sans-conflit.

□

Preuve (propriété 4.2) – Le premier coup de d est $[\text{PRO}, x]$, donc $x \in \text{PRO}(d)$. Le lemme 4.1 prouve que PRO(d) est sans-conflit. De plus, $\phi_1(d) = \emptyset = R^-(\text{PRO}(d)) \setminus R^+(\text{PRO}(d))$, donc $R^-(\text{PRO}(d)) \subseteq R^+(\text{PRO}(d))$. Par conséquent, PRO$(d)$ est admissible. □

Propriété 4.3 (Complétude des ϕ_1-preuves). *Si un argument x appartient à une extension préférée d'un système d'argumentation (A, R) alors il existe une ϕ_1-preuve pour x.*

Lemme 4.2. *Soit (A, R, ϕ_1) un type de dialogue et d un ϕ_1-dialogue dont le dernier coup est joué par* PRO*. Soit $S \subseteq A$ un ensemble minimal tel que S est admissible et contient* PRO(d)*. Si $S \neq$* PRO(d)*, alors il existe $x, y \in A$ tels que le dialogue $d' = d.[\text{OPP}, x].[\text{PRO}, y]$ est un ϕ_1-dialogue et S est minimal admissible contenant* PRO(d')*.*

Preuve (lemme 4.2) – Puisque PRO(d) $\neq S$ et S contient PRO(d), la minimalité de S implique que PRO(d) n'est pas admissible. D'après le lemme 4.1, on sait que PRO(d) est sans-conflit, donc $R^-(\text{PRO}(d)) \not\subset R^+(\text{PRO}(d))$. Soit $x \in R^-(\text{PRO}(d)) \setminus R^+(\text{PRO}(d))$; puisque PRO(d) $\subset S$, et puisque S est admissible, il existe $y \in S$ tel que $x \in R^+(y)$. Puisque $x \notin R^+(\text{PRO}(d))$, $y \notin \text{PRO}(d)$. De plus, S est sans-conflit, donc $y \notin R^{\pm}(\text{PRO}(d))$ et $y \notin \text{Refl}$, et par conséquent $y \in \phi_1(d.[\text{OPP}, x])$. Puisque $\phi_1(d) = R^-(\text{PRO}(d)) \setminus R^+(\text{PRO}(d))$, $x \in \phi_1(d)$.

Donc $d' = d.[\text{OPP}, x].[\text{PRO}, y]$ est un ϕ_1-dialogue. On sait que $PRO(d') = PRO(d) \cup \{y\} \subset S$ car $y \in S$. Il reste à prouver qu'aucun ensemble admissible S' strictement inclus dans S contient d'. Supposons que S' est un sous-ensemble strict de S tel que PRO(d') $\subset S'$; alors PRO(d) $\subset S$ car PRO(d) = PRO(d') $\setminus \{y\}$, et donc puisque S est minimal admissible contenant PRO(d), S' n'est pas admissible. □

Preuve (propriété 4.3) – Soit S un sous-ensemble minimal de A tel que S est admissible et contient x. Soit N le nombre d'éléments de S, soit $d_1 = [\text{PRO}, x]$ et soit $(d_n)_{2 \leq n \leq N}$ une séquence telle que par $d_n = d_{n-1}.[\text{OPP}, y].[\text{PRO}, z]$, pour $2 \leq n \leq N$, où $y \in R^-(\text{PRO}(d_{n-1})) \setminus R^+(\text{PRO}(d_{n-1}))$ et $z \in S \cap R^-(y)$. Le lemme 4.2 prouve qu'une telle séquence existe et que d_N est un ϕ_1-dialogue portant sur x gagné par PRO (puisque PRO(d_N) $= S$ et $R^-(S) \subset R^+(S)$, $\phi_1(d_N) = \emptyset$). □

Illustrons à présent cette ϕ_1-théorie de la preuve sur quelques exemples. Dans le premier exemple, les ϕ_1-dialogues obtenus ne mettent pas en évidence que OPP n'est pas obligé de répondre au dernier coup de PRO car OPP n'a pas d'autre choix que d'attaquer le dernier argument de PRO. Nous verrons dans le deuxième exemple des ϕ_1-dialogues où OPP n'attaque pas le dernier argument de PRO. Les deux exemples illustrent les autres contraintes imposées par la fonction ϕ_1.

Exemple 2.1 (ϕ_1-preuves).

- *Zoé l'autruche* : un seul ϕ_1-dialogue portant sur a peut être construit :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_1(\mu_0) = \{b\} \\ \mu_1 = [\text{OPP}, b] & \phi_1(\mu_0\mu_1) = \{c\} \\ \mu_2 = [\text{PRO}, c] & \phi_1(\mu_0\mu_1\mu_2) = \emptyset \end{array}$$

  Ce ϕ_1-dialogue $\mu_0\mu_1\mu_2$ est gagné par PRO ; c'est donc une ϕ_1-preuve de l'acceptabilité crédule de a.

- *Nixon* : un seul ϕ_1-dialogue portant sur a peut être construit :

$$\mu_0 = [\text{PRO}, a] \quad \phi_1(\mu_0) = \emptyset$$

  L'argument b n'appartient pas à $\phi_1(\mu_0)$ car, même si b attaque a, b est attaqué par a. Ceci illustre la contrainte que OPP *ne propose pas un argument qui est déjà attaqué par* PRO. Ce ϕ_1-dialogue μ_0 est gagné par PRO, c'est une ϕ_1-preuve de l'acceptabilité crédule de a.

- *Le paradoxe du menteur* : il n'est pas possible de construire un ϕ_1-dialogue portant sur l'argument a de ce système, puisque *a s'auto-attaque.* a n'est pas crédulement accepté.

- *L'affaire des peintures* : un seul ϕ_1-dialogue portant sur a peut être construit :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_1(\mu_0) = \{b\} \\ \mu_1 = [\text{OPP}, b] & \phi_1(\mu_0\mu_1) = \{c\} \\ \mu_2 = [\text{PRO}, c] & \phi_1(\mu_0\mu_1\mu_2) = \{d\} \\ \mu_3 = [\text{OPP}, d] & \phi_1(\mu_0\mu_1\mu_2\mu_3) = \emptyset \end{array}$$

  L'argument b n'appartient pas à $\phi_1(\mu_0\mu_1\mu_2\mu_3)$ car, en dépit du fait que b attaque d, b attaque l'argument a proposé par PRO au coup μ_0 et b est attaqué par c, déjà proposé par PRO au coup μ_2. Ceci illustre bien que PRO *n'a pas le droit de proposer un argument qui attaque ou qui est attaqué par* PRO. Le ϕ_1-dialogue $\mu_0\mu_1\mu_2\mu_3$ n'est pas gagné par PRO, il n'existe donc pas de ϕ_1-preuve pour a. a n'est pas crédulement accepté.

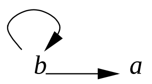
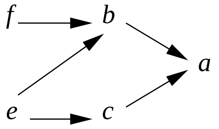
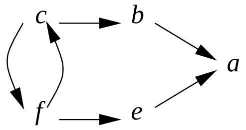
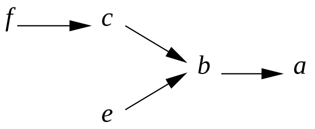
  Pour l'argument b, un seul ϕ_1-dialogue peut être construit :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, b] & \phi_1(\mu_0) = \{c\} \\ \mu_1 = [\text{OPP}, c] & \phi_1(\mu_0\mu_1) = \emptyset \end{array}$$

L'argument d n'appartient pas à $\phi_1(\mu_0\mu_1)$ car, même si d attaque c, d est attaqué par b, un argument déjà avancé par PRO au coup μ_0. PRO ne peut donc pas proposer cet argument. Le ϕ_1-dialogue $\mu_0\mu_1$ n'est pas gagné par PRO, b n'est donc pas crédulement accepté.

♢

Exemple 4.2. Considérons les systèmes d'argumentation indiqués dans le tableau 4.1.

$\mathcal{SA}_1$	$\mathcal{SA}_2$	$\mathcal{SA}_3$	$\mathcal{SA}_4$
			

Tab. 4.1 – Systèmes d'argumentation $\mathcal{SA}_1$, $\mathcal{SA}_2$, $\mathcal{SA}_3$ et $\mathcal{SA}_4$.

$\mathcal{SA}_1$ Le seul ϕ_1-dialogue portant sur a est le suivant :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_1(\mu_0) = \{b\} \\ \mu_1 = [\text{OPP}, b] & \phi_1(\mu_0\mu_1) = \emptyset \end{array}$$

Le dialogue ne peut être continué car PRO doit donner un argument qui attaque le dernier argument avancé par OPP, et dans ce cas, le seul attaquant possible est b ; *cet argument s'auto-attaque,* PRO *ne peut donc pas l'avancer.* Ce dialogue n'est pas gagné par PRO, a n'est pas crédulement accepté.

$\mathcal{SA}_2$ L'argument a est crédulement accepté. Pour établir une preuve de cette acceptabilité, trois ϕ_1-preuves peuvent être construites.

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_1(\mu_0) = \{b, c\} \\ \mu_1 = [\text{OPP}, b] & \phi_1(\mu_0\mu_1) = \{f, e\} \\ \mu_2 = [\text{PRO}, f] & \phi_1(\mu_0\mu_1\mu_2) = \{c\} \\ \mu_3 = [\text{OPP}, c] & \phi_1(\mu_0\mu_1\mu_2\mu_3) = \{e\} \\ \mu_4 = [\text{PRO}, e] & \phi_1(\mu_0\mu_1\mu_2\mu_3\mu_4) = \emptyset \end{array}$$

Remarquons que, dans ce ϕ_1-dialogue $d_{\mathcal{SA}_2} = \mu_0\mu_1\mu_2\mu_3\mu_4$, l'argument c avancé au coup μ_2 *n'attaque pas l'argument précédemment proposé* par PRO (*i.e.* f), mais l'argument du coup μ_0 (*i.e.* a). Ce ϕ_1-dialogue peut être représenté comme suit, ce qui correspond à une instanciation du schéma de la figure 4.2 page 53 :

$d_{\mathcal{SA}_2}$ est une ϕ_1-preuve de l'acceptabilité crédule de a. Construisons un deuxième ϕ_1-dialogue :

$$\begin{array}{ll} \mu'_0 = [\text{PRO}, a] & \phi_1(\mu'_0) = \{b, c\} \\ \mu'_1 = [\text{OPP}, b] & \phi_1(\mu'_0\mu'_1) = \{f, e\} \\ \mu'_2 = [\text{PRO}, e] & \phi_1(\mu'_0\mu'_1\mu'_2) = \emptyset \end{array}$$

L'argument e attaque à la fois b et c. Dans ce ϕ_1-dialogue $d'_{\mathcal{SA}_2} = \mu'_0\mu'_1\mu'_2$, au coup μ'_2, PRO utilise e pour défendre a contre b ; a est donc également défendu contre son deuxième attaquant, *i.e.* c. Pour cette raison, OPP n'a plus d'argument à avancer, le dialogue ne peut être continué. $d'_{\mathcal{SA}_2}$ est une deuxième ϕ_1-preuve de l'acceptabilité crédule de a, *plus courte* que la ϕ_1-preuve $d_{\mathcal{SA}_2}$. Il en existe une troisième :

$$\begin{array}{ll} \mu''_0 = [\text{PRO}, a] & \phi_1(\mu''_0) = \{b, c\} \\ \mu''_1 = [\text{OPP}, c] & \phi_1(\mu''_0\mu''_1) = \{e\} \\ \mu''_2 = [\text{PRO}, e] & \phi_1(\mu''_0\mu''_1\mu''_2) = \emptyset \end{array}$$

Dans ce ϕ_1-dialogue $d''_{\mathcal{SA}_2} = \mu''_0\mu''_1\mu''_2$, OPP a avancé c au coup μ''_1. PRO n'a pas eu d'autre choix, au coup μ''_2, que d'avancer l'argument e, seul attaquant de c. Mais de cette manière, PRO a également attaqué b. OPP n'a alors plus eu d'argument à avancer. $d''_{\mathcal{SA}_2}$ est la troisième ϕ_1-preuve de l'acceptabilité crédule de a.

$\mathcal{SA}_3$ [2]Deux ϕ_1-dialogues portant sur a peuvent être construits :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_1(\mu_0) = \{b, e\} \\ \mu_1 = [\text{OPP}, b] & \phi_1(\mu_0\mu_1) = \{c\} \\ \mu_2 = [\text{PRO}, c] & \phi_1(\mu_0\mu_1\mu_2) = \{e\} \\ \mu_3 = [\text{OPP}, e] & \phi_1(\mu_0\mu_1\mu_2\mu_3) = \emptyset \end{array}$$

L'argument f n'appartient pas à $\phi_1(\mu_0\mu_1\mu_2)$ car, en dépit du fait que f attaque c, f est déjà attaqué par c. De plus, f n'appartient pas à $\phi_1(\mu_0\mu_1\mu_2\mu_3)$ car f attaque et est attaqué par c. Notons également que l'argument e avancé au coup μ_3 par OPP n'attaque pas le dernier argument de PRO. Ce premier ϕ_1-dialogue $d_{\mathcal{SA}_3} = \mu_0\mu_1\mu_2\mu_3$ n'est pas gagné par PRO. Le deuxième ϕ_1-dialogue qui peut être construit est le suivant :

$$\begin{array}{ll} \mu'_0 = [\text{PRO}, a] & \phi_1(\mu'_0) = \{b, e\} \\ \mu'_1 = [\text{OPP}, e] & \phi_1(\mu'_0\mu'_1) = \{f\} \\ \mu'_2 = [\text{PRO}, f] & \phi_1(\mu'_0\mu'_1\mu'_2) = \{b\} \\ \mu'_3 = [\text{OPP}, b] & \phi_1(\mu'_0\mu'_1\mu'_2\mu'_3) = \emptyset \end{array}$$

Ce ϕ_1-dialogue $d'_{\mathcal{SA}_3} = \mu'_0\mu'_1\mu'_2\mu'_3$ n'est pas non plus gagné par PRO. *Tous les ϕ_1-dialogues portant sur a étant perdus par* PRO, il n'existe *pas de ϕ_1-preuve pour a.* a n'est pas crédulement accepté.

Pour ce qui est de l'argument c, un seul ϕ_1-dialogue peut être construit :

$$\mu_0 = [\text{PRO}, c] \quad \phi_1(\mu_0) = \emptyset$$

Ce ϕ_1-dialogue est gagné par PRO, c'est une ϕ_1-preuve de l'acceptabilité crédule de c.

Quant à l'argument b, un seul ϕ_1-dialogue peut être construit :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, b] & \phi_1(\mu_0) = \{c\} \\ \mu_1 = [\text{OPP}, c] & \phi_1(\mu_0\mu_1) = \{f\} \\ \mu_2 = [\text{PRO}, f] & \phi_1(\mu_0\mu_1\mu_2) = \emptyset \end{array}$$

Ce ϕ_1-dialogue est gagné par PRO, c'est une ϕ_1-preuve de l'acceptabilité crédule de b.

$\mathcal{SA}_4$ Deux ϕ_1-dialogues portant sur a peuvent être construits :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_1(\mu_0) = \{b\} \\ \mu_1 = [\text{OPP}, b] & \phi_1(\mu_0\mu_1) = \{c, e\} \\ \mu_2 = [\text{PRO}, c] & \phi_1(\mu_0\mu_1\mu_2) = \{f\} \\ \mu_3 = [\text{OPP}, f] & \phi_1(\mu_0\mu_1\mu_2\mu_3) = \emptyset \end{array}$$

L'argument c ne peut être défendu contre f, c est donc un «mauvais» défenseur de a. Ce premier ϕ_1-dialogue $d_{\mathcal{SA}_4} = \mu_0\mu_1\mu_2\mu_3$ est perdu par PRO. PRO, au coup μ_2, aurait pu avancer l'argument e. Le ϕ_1-dialogue qui découle de cet autre choix est le suivant :

---

[2] Ce système d'argumentation est issu de [VP00].

$$\begin{array}{ll} \mu'_0 = [\mathrm{PRO}, a] & \phi_1(\mu'_0) = \{b\} \\ \mu'_1 = [\mathrm{OPP}, b] & \phi_1(\mu'_0\mu'_1) = \{c, e\} \\ \mu'_2 = [\mathrm{PRO}, e] & \phi_1(\mu'_0\mu'_1\mu'_2) = \emptyset \end{array}$$

Ce ϕ_1-dialogue $d'_{\mathcal{SA}_4} = \mu'_0\mu'_1\mu'_2$ est gagné par PRO, c'est une ϕ_1-preuve pour a. a est donc crédulement accepté. Cet exemple souligne que pour qu'un argument soit crédulement accepté, *il suffit qu'un ϕ_1-dialogue* portant sur cet argument *soit gagné par* PRO.

Les ϕ_1-dialogues décrits pour les arguments a des systèmes $\mathcal{SA}_2$, $\mathcal{SA}_3$ et $\mathcal{SA}_4$ sont récapitulés dans le tableau 4.2. ◇

$\mathcal{SA}_2$			$\mathcal{SA}_3$		$\mathcal{SA}_4$	
$d_{\mathcal{SA}_2}$	$d'_{\mathcal{SA}_2}$	$d''_{\mathcal{SA}_2}$	$d_{\mathcal{SA}_3}$	$d'_{\mathcal{SA}_3}$	$d_{\mathcal{SA}_4}$	$d'_{\mathcal{SA}_4}$
$[\mathrm{PRO}, a]$ | $[\mathrm{OPP}, b]$ | $[\mathrm{PRO}, f]$ | $[\mathrm{OPP}, c]$ | $[\mathrm{PRO}, e]$	$[\mathrm{PRO}, a]$ | $[\mathrm{OPP}, b]$ | $[\mathrm{PRO}, e]$	$[\mathrm{PRO}, a]$ | $[\mathrm{OPP}, c]$ | $[\mathrm{PRO}, e]$	$[\mathrm{PRO}, a]$ | $[\mathrm{OPP}, b]$ | $[\mathrm{PRO}, c]$ | $[\mathrm{OPP}, e]$	$[\mathrm{PRO}, a]$ | $[\mathrm{OPP}, e]$ | $[\mathrm{PRO}, f]$ | $[\mathrm{OPP}, b]$	$[\mathrm{PRO}, a]$ | $[\mathrm{OPP}, b]$ | $[\mathrm{PRO}, c]$ | $[\mathrm{OPP}, f]$	$[\mathrm{PRO}, a]$ | $[\mathrm{OPP}, b]$ | $[\mathrm{PRO}, e]$
gagné par PRO	gagné par PRO	gagné par PRO	perdu par PRO	perdu par PRO	perdu par PRO	gagné par PRO
⇓ ϕ_1-preuve ⇓ a crédulement accepté	⇓ ϕ_1-preuve ⇓ a crédulement accepté	⇓ ϕ_1-preuve ⇓ a crédulement accepté	⇓ a non crédulement accepté			⇓ ϕ_1-preuve ⇓ a crédulement accepté

TAB. 4.2 – ϕ_1-dialogues pour les arguments a des systèmes $\mathcal{SA}_2$, $\mathcal{SA}_3$ et $\mathcal{SA}_4$.

4.2.2 La ϕ_2-théorie de la preuve

Dans le but de comparer notre travail avec d'autres théories de la preuve dialectiques (les TPI-disputes en particulier), nous présentons une théorie de la preuve d'une forme plus classique, où, à chaque étape de la preuve, l'argument proposé attaque directement le précédent. De telles preuves sont obtenues à l'aide de la fonction coup-légal suivante :

Définition 4.11 (Fonction coup-légal de type ϕ_2).
Soit (A, R) un système d'argumentation. $\phi_2 : A^* \to 2^A$ est définie par :

- *si d est un ϕ_2-dialogue de longueur impaire (*i.e. *le prochain coup doit être joué par* OPP*),*

$$\phi_2(d) = R^-(\arg(\mathrm{der}(d))) \setminus R^+(\mathrm{PRO}(d)) \ ;$$

- *si d est un ϕ_2-dialogue de longueur paire (*i.e. *le prochain coup doit être joué par* PRO*),*

$$\phi_2(d) = \phi_1(d) = R^-(\arg(\mathrm{der}(d))) \cap \mathrm{POSS}(d).$$

ϕ_2 est une restriction de ϕ_1 puisque, d'après ϕ_2, OPP doit avancer un argument qui attaque l'argument avancé par PRO au coup précédent. Les ϕ_2-dialogues respectent donc le schéma de dialogue de la figure 4.3 page 53.

La combinaison d'un type de dialogue (A, R, ϕ_2) et du critère de gain de la définition 4.5 page 49 (ou de sa reformulation dans la propriété 4.1 page 50) permet d'obtenir des ϕ_2-théories de la preuve :

Définition 4.12 (ϕ_2-preuve).
Soit un type de dialogue (A, R, ϕ_2) et un argument $x \in A$ qui ne s'auto-attaque pas.

Une ϕ_2-preuve *pour x est une ϕ_2-stratégie gagnante S pour x telle que $\bigcup_{d \in S}(\mathrm{PRO}(d))$ est sans-conflit.*

Le résultat suivant permet d'établir que la ϕ_2-théorie de la preuve est correcte et complète pour le problème d'acceptabilité crédule :

Propriété 4.4 (Équivalence de la ϕ_1- et de la ϕ_2-théorie de la preuve). *Il existe une ϕ_1-preuve pour un argument x si et seulement si il existe une ϕ_2-preuve pour x.*

PREUVE (CONDITION NÉCESSAIRE) – Dans toute la suite, d dénotera un ϕ_1-dialogue, et δ dénotera un ϕ_2-dialogue.

Étant donné d une ϕ_1-preuve pour l'argument x, construisons une ϕ_2-stratégie gagnante pour x en considérant tous les attaquants autorisés par ϕ_2 pour chaque argument de $\mathrm{PRO}(d)$. De fait, pour chaque coup $\mu_{2k} = [\mathrm{PRO}, a_{2k}]$ joué par PRO dans d, pour chaque $b \in \phi_2(\mu_0 \dots \mu_{2k})$:
- soit b apparaît dans un coup μ_{2p+1} de d et le coup μ_{2p+2} de d sera le fils de $[\mathrm{OPP}, b]$ dans un ϕ_2-dialogue ;
- soit b n'apparaît pas dans d car b est attaqué par un argument a avancé par PRO dans d ; dans ce cas, $[\mathrm{PRO}, a]$ sera le fils de $[\mathrm{OPP}, b]$ dans un ϕ_2-dialogue.

Nous avons besoin d'une notation supplémentaire : étant donné d un dialogue et $x \in \mathrm{PRO}(d)$, $\mathrm{OPP}(d, x)$ dénote l'ensemble des arguments qui attaquent l'argument x et qui sont avancés par OPP dans d après le coup $[\mathrm{PRO}, x]$. Formellement, si $\mu_{2k} = [\mathrm{PRO}, x]$ est un coup de d,

$$\mathrm{OPP}(d, x) = \{y \mid y \in R^-(x) \text{ et } \mu_{2p+1} = [\mathrm{OPP}, y] \text{ avec } k = p \text{ est un coup de } d\}.$$

Soit d une ϕ_1-preuve pour l'argument x. d est de la forme $d = \mu_0\mu_1, \dots \mu_{2n}$ avec $\arg(\mu_0) = x$, $\mathrm{der}(d)$ est joué par PRO et $\phi_1(d) = \emptyset$. Nous construisons tout d'abord par induction sur p un ensemble fini S_p de ϕ_2-dialogues portant sur x de longueur $2p$, tels que pour tout $\delta \in S_p$, $\mathrm{PRO}(\delta) \subseteq \mathrm{PRO}(d)$.
- Si $p = 0$, $S_0 = \{\mu_0\}$.
- Supposons que S_p est défini et satisfait les conditions ci-dessus. Étant donné $\delta \in S_p$, $\mathrm{der}(\delta)$ est de la forme $[\mathrm{PRO}, a]$ et $[\mathrm{PRO}, a]$ est un coup de d (μ_{2k} par exemple). Pour tout $b \in \phi_2(\delta)$, $b \in R^-(a) \setminus R^+(\mathrm{PRO}(\delta))$.
  - Si $b \in \mathrm{OPP}(d, a)$, alors il existe $i \geq k$ avec $b = \arg(\mu_{2i+1})$. Définissons $s(b) = \arg(\mu_{2i+2})$. Le dialogue $\delta.[\mathrm{OPP}, b].[\mathrm{PRO}, s(b)]$ est une extension de δ de longueur $2p+2$, avec $[\mathrm{PRO}, s(b)] \in \mathrm{PRO}(d)$.
  - Si $b \notin \mathrm{OPP}(d, a)$, puisque $\phi_1(d) = \emptyset$, $b \in R^-(a)$ et $a \in \mathrm{PRO}(d)$, alors $b \in R^+(\mathrm{PRO}(d))$. Il existe donc μ_{2i} dans d, $i < k$, tel que $\arg(\mu_{2i}) \in R^-(b)$. Définissons $s(b) = \arg(\mu_{2i})$. Le dialogue $\delta.[\mathrm{OPP}, b].[\mathrm{PRO}, s(b)]$ est une extension de δ de longueur $2p + 2$, avec $[\mathrm{PRO}, s(b)] \in \mathrm{PRO}(d)$.

Prouvons maintenant que le dialogue $\delta.[\mathrm{OPP}, b].[\mathrm{PRO}, s(b)]$ est un ϕ_2-dialogue. D'après la définition des ϕ_1-dialogues, $s(b) \in R^-(b)$. Puisque $b \notin R^+(\mathrm{PRO}(\delta))$, on a $s(b) \notin \mathrm{PRO}(\delta)$. De plus, $s(b) \notin R^{\pm}(\mathrm{PRO}(d))$. $\mathrm{PRO}(\delta) \subseteq \mathrm{PRO}(d)$, donc $s(b) \notin R^{\pm}(\mathrm{PRO}(\delta))$. Par conséquent, $s(b) \in \mathrm{POSS}(\delta)$ et $s(b) \in \phi_2(\delta[\mathrm{OPP}, b])$.

Nous définissons :

$$S_{p+1} = \bigcup_{\delta \in S_p} \{\delta.[\mathrm{OPP}, b].[\mathrm{PRO}, s(b)], b \in \phi_2(\delta)\}.$$

Montrons que la construction ci-dessus termine.

Étant donné $\delta \in S_p$, si δ n'est pas gagné par PRO, δ peut être étendu en $\delta' \in S_{p+1}$: $\delta' = \delta.[\mathrm{OPP}, b].[\mathrm{PRO}, s(b)]$, avec $b \in \phi_2(\delta)$. Puisque $s(b) \notin \mathrm{PRO}(\delta)$, $\mathrm{PRO}(\delta)$ est strictement inclus dans $\mathrm{PRO}(\delta')$ et $\mathrm{PRO}(\delta') \subseteq \mathrm{PRO}(d)$. d est une ϕ_1-preuve, donc est fini et $\mathrm{PRO}(d)$ est fini. Donc, il existe p tel que pour tout $\delta \in S_p$, δ ne peut être continué (*i.e.* δ est gagné par PRO). Alors $S_{p+1} = S_p$, et on définit $S = S_0 \cup \ldots \cup S_p$. Par construction de S, pour tout $\delta \in S$, il existe une extension de δ dans S_p.

Nous prouvons enfin que S est une ϕ_2-stratégie gagnante pour x. Soit $\delta \in S$ et δ' préfixe de δ tel que $\mathrm{der}(\delta')$ est joué par PRO. Par construction de S, $\delta \in S_i$, et $\delta' \in S_k$, avec $k \leq i \leq p$.

Nous prouvons que pour tout b de $\phi_2(\delta')$, $\exists \delta'' \in S$ tel que δ'' est gagné par PRO et δ'' est une extension de $\delta'.[\mathrm{OPP}, b]$. Supposons que $b \in \phi_2(\delta')$. Alors $k < p$ et $\delta'.[\mathrm{OPP}, b].[\mathrm{PRO}, s(b)] \in S_{k+1}$. Donc il existe $\delta'' \in S_p$ tel que δ'' est une extension de $\delta'.[\mathrm{OPP}, b].[\mathrm{PRO}, s(b)]$. δ'' est gagné par PRO et est aussi une extension de $\delta'.[\mathrm{OPP}, b]$.

De plus, $\bigcup_{\delta \in S}(\mathrm{PRO}(\delta))$ est sans-conflit puisque il est inclus dans $\mathrm{PRO}(d)$. □

PREUVE (CONDITION SUFFISANTE) – Dans toute la suite, d dénotera un ϕ_1-dialogue, et δ dénotera un ϕ_2-dialogue.

Supposons qu'il existe une ϕ_2-stratégie gagnante S pour x telle que $\bigcup_{\delta \in S}(\mathrm{PRO}(\delta))$ est sans-conflit. On peut supposer que chaque dialogue de S est gagné par PRO (il faut remplacer S par $S' = \{\delta \in S \mid \phi_2(\delta) = \emptyset\}$).

S est non vide. Étant donné $\delta \in S$, construisons une ϕ_1-preuve d pour x en donnant des extensions successives de δ.

– Si δ est un ϕ_1-dialogue gagné par PRO, δ est une ϕ_1-preuve pour x et $d = \delta$.
– Si $\phi_1(\delta) \neq \emptyset$, on choisit $b \in R^-(\mathrm{PRO}(\delta)) \setminus R^+(\mathrm{PRO}(\delta))$. Considérons δ' préfixe de δ de longueur maximale tel que $\mathrm{der}(\delta') = [\mathrm{PRO}, a]$, $b \in R^-(a)$. $b \notin R^+(\mathrm{PRO}(\delta'))$ donc $b \in \phi_2(\delta')$. Puisque $\phi_2(\delta) = \emptyset$, $b \notin \phi_2(\delta)$ et $\delta' \neq \delta$.

S est une ϕ_2-stratégie gagnante pour x, donc il existe une extension de $\delta'.[\mathrm{OPP}, b]$ dans S. En d'autres termes, il existe un élément de S de la forme $\delta'.[\mathrm{OPP}, b].\delta''$.

Nous étendons δ en considérant $\delta_1 = \delta.[\mathrm{OPP}, b].\delta''$. Puisque $\bigcup_{\delta \in S}(\mathrm{PRO}(\delta))$ est sans-conflit, nous sommes sûrs que l'argument du premier coup de δ'' appartient à $\mathrm{POSS}(\delta)$. Donc δ_1 est un ϕ_1-dialogue qui étend δ.

On itère la construction (δ_1 est étendu en considérant $\delta_2, \ldots$) jusqu'à ce que l'extension obtenue d satisfasse $\phi_1(d) = \emptyset$. Alors, d est une ϕ_1-preuve pour x.

L'itération s'arrête puisqu'à chaque étape $\mathrm{PRO}(\delta_i) \subseteq \bigcup_{\delta \in S}(\mathrm{PRO}(\delta))$ et $\mathrm{PRO}(\delta_i)$ est strictement inclus dans $\mathrm{PRO}(\delta_{i+1})$ lorsque $\phi_1(\delta_i) \neq \emptyset$. Gardons en mémoire que S est un ensemble fini de dialogues finis. □

Une ϕ_2-preuve va prendre la forme d'un ensemble de ϕ_2-dialogues gagnés par PRO, que nous allons représenter sous forme d'un arbre.

EXEMPLE 2.1 (ϕ_2-PREUVES). Les ϕ_1-dialogues établis dans l'exemple page 55 sont identiques aux ϕ_2-dialogues que l'on peut construire. Pour chacun des systèmes d'argumentation de cet exemple, lorsqu'une ϕ_1-preuve existe, il existe une unique ϕ_2-preuve qui est l'ensemble constitué de la ϕ_1-preuve. ♢

Pour éviter toute confusion avec les ϕ_1-dialogues que nous notions d dans les exemples précédents, nous allons noter δ les ϕ_2-dialogues dans l'exemple suivant.

EXEMPLE 4.2 (ϕ_2-PREUVES).

$\mathcal{SA}_1$ Le seul ϕ_2-dialogue pour a est identique au ϕ_1-dialogue pour a perdu par PRO. Puisqu'il n'existe pas de ϕ_2-dialogue gagné par PRO, il n'existe pas de ϕ_2-preuve pour a. a n'est donc pas crédulement accepté.

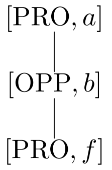
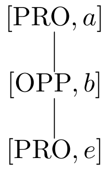
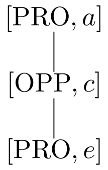
$\mathcal{SA}_2$ Trois ϕ_2-dialogues $\delta_{\mathcal{SA}_2}$, $\delta'_{\mathcal{SA}_2}$ et $\delta''_{\mathcal{SA}_2}$ portant sur a et gagnés par PRO peuvent être construits. $\delta'_{\mathcal{SA}_2}$ et $\delta''_{\mathcal{SA}_2}$ sont respectivement identiques aux ϕ_1-dialogues $d'_{\mathcal{SA}_2}$ et $d''_{\mathcal{SA}_2}$. Le ϕ_1-dialogue $d_{\mathcal{SA}_2}$ n'est pas un ϕ_2-dialogue car l'un des arguments de OPP n'attaque pas l'argument du coup immédiatement précédent. Le ϕ_2-dialogue $\delta_{\mathcal{SA}_2}$ est :

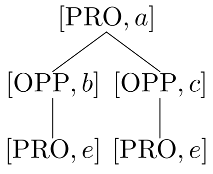
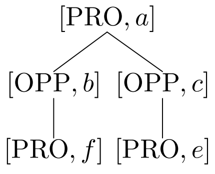
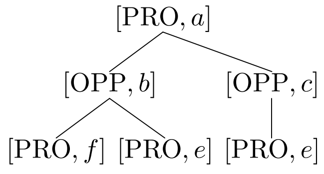
$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_2(\mu_0) = \{b, c\} \\ \mu_1 = [\text{OPP}, b] & \phi_2(\mu_0\mu_1) = \{f, e\} \\ \mu_2 = [\text{PRO}, f] & \phi_2(\mu_0\mu_1\mu_2) = \emptyset \end{array}$$

Ce ϕ_2-dialogue $\delta_{\mathcal{SA}_2} = \mu_0\mu_1\mu_2$ ne peut être continué avec le coup $[\text{OPP}, c]$ car c n'attaque pas directement f.

L'ensemble $\{\delta''_{sa_2}, \delta_{sa_2}\}$ est une ϕ_2*-stratégie gagnante*. En effet, pour chacun des deux dialogues δ''_{sa_2} et δ_{sa_2}, le seul préfixe auquel OPP peut répondre est $[\text{PRO}, a]$. Les deux coups possibles pour OPP sont alors $[\text{OPP}, b]$ et $[\text{OPP}, c]$. $[\text{PRO}, a][\text{OPP}, b]$ et $[\text{PRO}, a][\text{OPP}, c]$ ont tous deux une extension gagnée par PRO dans $\{\delta''_{sa_2}, \delta_{sa_2}\}$. Puisque $\text{PRO}(\delta''_{sa_2}) \cup \text{PRO}(\delta_{sa_2})$ est sans-conflit, $\{\delta''_{sa_2}, \delta_{sa_2}\}$ est une ϕ_2*-preuve* pour a ; a est crédulement accepté.

Il existe deux autres ϕ_2-preuves pour a : $\{\delta'_{sa_2}, \delta''_{sa_2}\}$ et $\{\delta'_{sa_2}, \delta''_{sa_2}, \delta_{\mathcal{SA}_2}\}$. Les ϕ_2-dialogues et les ϕ_2-preuves sont représentés sous forme d'arbre dans le tableau 4.3.

ϕ_2-dialogues gagnés par PRO		
$\delta_{\mathcal{SA}_2}$	$\delta'_{\mathcal{SA}_2} = d'_{\mathcal{SA}_2}$	$\delta''_{\mathcal{SA}_2} = d''_{\mathcal{SA}_2}$
		

ϕ_2-preuves		
$\{\delta'_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$	$\{\delta_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$	$\{\delta_{\mathcal{SA}_2}, \delta'_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$
		

TAB. 4.3 – ϕ_2-dialogues gagnés par PRO et ϕ_2-preuves pour l'argument a du système $\mathcal{SA}_2$.

$\mathcal{SA}_3$ Les ϕ_1-dialogues $d_{\mathcal{SA}_3}$ et $d'_{\mathcal{SA}_3}$ ne sont pas des ϕ_2-dialogues car OPP ne répond pas toujours au coup immédiatement précédent de PRO. Les deux ϕ_2-dialogues portant sur a qui peuvent être construits sont les suivants :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_2(\mu_0) = \{b, e\} \\ \mu_1 = [\text{OPP}, b] & \phi_2(\mu_0\mu_1) = \{c\} \\ \mu_2 = [\text{PRO}, c] & \phi_2(\mu_0\mu_1\mu_2) = \emptyset \end{array}$$

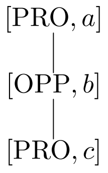
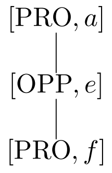
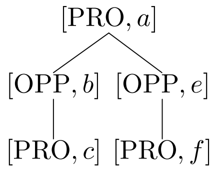
Remarquons que l'argument e n'appartient pas à $\phi_2(\mu_0\mu_1\mu_2)$ car e n'attaque pas directement c. Ce ϕ_2-dialogue $\delta_{\mathcal{SA}_3} = \mu_0\mu_1\mu_2$ est gagné par PRO.

Le deuxième ϕ_2-dialogue est :

$$\begin{array}{ll} \mu'_0 = [\text{PRO}, a] & \phi_2(\mu'_0) = \{b, e\} \\ \mu'_1 = [\text{OPP}, e] & \phi_2(\mu'_0\mu'_1) = \{f\} \\ \mu'_2 = [\text{PRO}, f] & \phi_2(\mu'_0\mu'_1\mu'_2) = \emptyset \end{array}$$

L'argument b n'appartient pas à $\phi_2(\mu'_0\mu'_1\mu'_2)$ car b n'attaque pas directement f. Ce deuxième ϕ_2-dialogue $\delta'_{\mathcal{SA}_3} = \mu'_0\mu'_1\mu'_2$ est lui aussi gagné par PRO.

$\delta_{\mathcal{SA}_3}$ et $\delta'_{\mathcal{SA}_3}$ sont gagnés par PRO, l'ensemble $\{\delta_{\mathcal{SA}_3}, \delta'_{\mathcal{SA}_3}\}$ est une ϕ_2-*stratégie gagnante* pour a. Cependant, $\mathrm{PRO}(\delta_{\mathcal{SA}_3}) \cup \mathrm{PRO}(\delta'_{\mathcal{SA}_3}) = \{a, c, f\}$ n'est pas sans-conflit (c et f s'attaquent l'un l'autre). Donc cette ϕ_2-stratégie gagnante *n'est pas une* ϕ_2-*preuve* pour a ; a n'est pas crédulement accepté. Les ϕ_2-dialogues portant sur a et la ϕ_2-stratégie gagnante pour a sont indiquées sur le tableau 4.4.

ϕ_2-dialogues gagnés par PRO		ϕ_2-stratégie gagnante
$\delta_{\mathcal{SA}_3}$	$\delta'_{\mathcal{SA}_3}$	$\{\delta_{\mathcal{SA}_3}, \delta'_{\mathcal{SA}_3}\}$
		 c et f sont en conflit, donc $\{\delta_{\mathcal{SA}_3}, \delta'_{\mathcal{SA}_3}\}$ n'est pas une ϕ_2-preuve

TAB. 4.4 – ϕ_2-dialogues gagnés par PRO et ϕ_2-stratégie gagnante pour a du système $\mathcal{SA}_3$.

$\mathcal{SA}_4$ Deux ϕ_2-dialogues portant sur a peuvent être construits ; ils sont identiques aux ϕ_1-dialogues. Le ϕ_1-dialogue gagné par PRO est un ϕ_2-dialogue gagné par PRO. L'ensemble réduit à ce ϕ_2-dialogue est une ϕ_2-preuve pour a.

◇

4.2.3 Comparaison des ϕ_1-preuves et des ϕ_2-preuves

Nous avons montré dans la section précédente que les deux théories de la preuve que nous avons proposées sont équivalentes.

Notons que ni les ϕ_1-, ni les ϕ_2-théories de la preuve ne font apparaître tous les attaquants de PRO. Par exemple, la ϕ_1-preuve pour l'acceptabilité de a dans le système d'argumentation *Nixon* décrite dans l'exemple 2.1 (ϕ_1-preuves) page 55 ne fait apparaître ni l'attaquant b de a, ni la façon dont a est défendu contre b ; de même dans le ϕ_2-dialogue portant sur a qui compose la ϕ_2-preuve pour a.

Néanmoins, les ϕ_2-preuves font parfois apparaître plus d'attaquants de PRO que les ϕ_1-preuves. La raison à cela est que OPP est obligé d'essayer toutes les attaques possibles contre PRO dans les différents dialogues d'une ϕ_2-preuve, afin que l'ensemble de dialogues soit «complet». Par exemple, la façon dont l'argument a du système $\mathcal{SA}_2$ est défendu contre b dans la ϕ_1-preuve $\delta''_{\mathcal{SA}_2}$ (voir tableau 4.2 page 58) n'est pas clairement exprimée ; il faut voir que a est défendu parce que l'argument e attaque non seulement c, mais aussi b. Ceci apparaît clairement dans les trois ϕ_2-preuves (voir tableau 4.3 page précédente).

De plus, du point de vue de la forme des preuves, les ϕ_2-preuves ont l'avantage sur les ϕ_1-preuves de permettre d'identifier immédiatement l'argument qui est attaqué par l'argument avancé par OPP (l'argument du coup précédent).

Pour ces raisons, les ϕ_2-preuves sont plus explicites que les ϕ_1-preuves.

Remarquons que les ϕ_2-preuves peuvent contenir plus de dialogues que nécessaire pour établir la preuve de l'acceptabilité crédule. Par exemple, la ϕ_2-preuve $\{\delta'_{sa_2}, \delta''_{sa_2}, \delta_{\mathcal{SA}_2}\}$ décrite dans le tableau 4.3 page précédente pourrait être réduite : les arbres obtenus en coupant la feuille $[\mathrm{PRO}, f]$ ou la feuille $[\mathrm{PRO}, e]$ correspondent aux ϕ_2-preuves «plus petites» (*i.e.* qui contiennent moins de nœuds) $\{\delta''_{\mathcal{SA}_2}, \delta_{\mathcal{SA}_2}\}$ et $\{\delta'_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$. Ceci est dû au fait que que la séquence de coups $[\mathrm{PRO}, a][\mathrm{OPP}, b]$ a deux extensions possibles qui sont des ϕ_2-dialogues gagnés par PRO ; les deux ϕ_2-dialogues sont construits indépendamment, et bien qu'un seul d'entre eux soit nécessaire pour établir une ϕ_2-preuve, la définition d'une ϕ_2-stratégie gagnante permet aux deux d'appartenir à une même ϕ_2-preuve. Cette définition des ϕ_2-preuves est un inconvénient pour la taille des preuves (elles peuvent contenir plus de nœuds que nécessaire), mais elle a l'avantage de montrer plusieurs façons dont un argument de PRO se défend contre une attaque de OPP.

Du fait que les ϕ_2-dialogues d'une ϕ_2-stratégie gagnante sont construits indépendamment, l'ensemble des arguments avancés par PRO n'est pas garanti sans-conflit. En revanche, les ϕ_1-dialogues gagnés par PRO garantissent que l'ensemble des arguments avancés par PRO est sans-conflit. C'est là un avantage de la ϕ_1-théorie de la preuve sur la ϕ_2-théorie de la preuve.

Remarquons enfin que si l'acceptabilité d'un argument a est à prouver, ni la définition des ϕ_1-preuves, ni celle des ϕ_2-preuves, n'obligent l'ensemble des arguments avancés par PRO à être *minimal admissible contenant* a, c'est-à-dire qu'il peut exister un sous-ensemble strict de l'ensemble admissible contenant a qui est admissible. Considérons par exemple les ϕ_1-preuves $d_{\mathcal{SA}_2}$ et $d'_{\mathcal{SA}_2}$ indiquées dans le tableau 4.2 page 58. La preuve $d'_{\mathcal{SA}_2}$ est plus courte que la preuve $d_{\mathcal{SA}_2}$. Les ensembles $\mathrm{PRO}(d_{\mathcal{SA}_2})$ et $\mathrm{PRO}(d'_{\mathcal{SA}_2})$ sont admissibles, mais $\mathrm{PRO}(d'_{\mathcal{SA}_2})$, au contraire de $\mathrm{PRO}(d_{\mathcal{SA}_2})$, est minimal contenant a. Pour les ϕ_2-preuves indiquées dans le tableau 4.3 page 61, seule $\{\delta'_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$ est telle que $\mathrm{PRO}(\delta'_{\mathcal{SA}_2}) \cup \mathrm{PRO}(\delta''_{\mathcal{SA}_2})$ est minimal admissible contenant a.

Imaginons que l'on veuille produire uniquement des preuves minimales (*i.e.* telles que l'ensemble admissible est minimal contenant l'argument dont l'acceptabilité est prouvée). Sur l'exemple du système d'argumentation $\mathcal{SA}_2$, cela implique de «voir» qu'en avançant l'argument e, PRO est défendu contre ses deux attaquants à la fois. Ceci semble difficilement réalisable.

4.3 Acceptabilité sous la sémantique basique

Rappelons que le problème d'acceptabilité sous la sémantique basique consiste à déterminer si un argument donné d'un système d'argumentation appartient à l'extension basique du système.

Prakken et Sartor ont proposé dans [PS97] une théorie de la preuve dialectique pour répondre à ce problème. Cette théorie a été reprise dans [Pra01], conformément au cadre dialectique qui y est décrit. Amgoud et Cayrol ont également proposé dans [AC02] une théorie de la preuve dialectique, plus générale que celle de [PS97]. Nous présentons ici ces deux théories de la preuve et nous les réexprimons dans le cadre formel de Jakobovits et Vermeir [JV99a].

Amgoud et Cayrol n'ont pas formalisé leur théorie de la preuve dans le cadre formel de Jakobovits et Vermeir. Néanmoins, plusieurs notions qu'elles utilisent se retrouvent dans ce cadre.

Dans [AC02], la preuve qu'un argument est acceptable prend la forme d'un arbre de dialogues fini, dont chaque branche est un dialogue. Un dialogue est une séquence non vide de coups, définis comme dans le cadre de [JV99a]. Les coups autorisés dans un dialogue sont tels que :

- l'argument avancé dans un coup joué par OPP doit attaquer l'argument qui le précède immédiatement dans le dialogue ;
- l'argument avancé dans un coup joué par PRO doit attaquer «strictement» (*i.e.* doit attaquer sans être attaqué par) l'argument qui le précède immédiatement dans le dialogue, et ne doit pas avoir déjà été proposé par PRO.

Exemple 4.2 (arbres de dialogues).

$\mathcal{SA}_2$ Les arbres correspondants aux ϕ_2-preuves dans le tableau 4.3 page 61 sont des arbres de dialogues au sens de [AC02].

◊

La définition des coups autorisés peut être exprimée dans une fonction coup-légal (que nous dirons de type ϕ_{db}) comme suit :

Définition 4.13 (Fonction coup-légal de type ϕ_{db}).
Soit (A, R) *un système d'argumentation.* $\phi_{db} : A^* \rightarrow 2^A$ *est définie par :*

- *si* d *est un* ϕ_{db}*-dialogue de longueur impaire (*i.e. *le prochain coup doit être joué par* OPP*),*

$$\phi_{db}(d) = R^-(\mathrm{arg}(\mathrm{der}(d)))\ ;$$

- *si* d *est un* ϕ_{db}*-dialogue de longueur paire (*i.e. *le prochain coup doit être joué par* PRO*),*

$$\phi_{db}(d) = R^-(\mathrm{arg}(\mathrm{der}(d))) \setminus (R^+(\mathrm{arg}(\mathrm{der}(d)) \cup \mathrm{PRO}(d)).$$

Pour [AC02], un dialogue est gagné par PRO si OPP ne peut plus jouer (*i.e.* PRO termine le dialogue) ; cette définition est équivalente au premier critère de gain de [JV99a] (définition 4.2 page 49).
Le critère de gain relatif à un arbre de dialogues est plus difficile à définir.

Dans [PS97], un arbre de dialogues est gagné par un joueur si et seulement si il gagne toutes les branches de l'arbre (autrement dit s'il gagne tous les dialogues qui composent l'arbre). Pour [AC02], ce critère de gain est très strict et s'applique en pratique à des arbres de dialogues très particuliers, où chaque coup joué par PRO a un seul fils dans l'arbre. [AC02] propose donc un critère de gain plus général, qui s'applique à des arbres de dialogues arbitraires.

Pour définir ce critère de gain, [AC02] considère l'arbre de dialogues comme un arbre Et/Ou. Les nœuds correspondants au joueur PRO sont des nœuds Et, ceux correspondants au joueur OPP sont des nœuds Ou. Cette distinction entre les nœuds est due au fait qu'un argument est acceptable s'il est défendu contre tous ses attaquants. Les fils d'un nœud contenant un argument de PRO représentent ses attaquants, et doivent donc *tous* être attaqués. En revanche, les fils d'un nœud contenant un argument de OPP représentent des défenseurs de PRO, et il est suffisant qu'*un seul* d'entre eux attaque l'argument de OPP.

[AC02] définit un *sous-arbre candidat* comme étant un sous-arbre de l'arbre de dialogues contenant tous les fils de chaque nœud Et et exactement un fils de chaque nœud Ou. Un *sous-arbre solution* est un sous-arbre candidat dont les branches sont toutes gagnées par PRO.

PRO gagne un arbre de dialogues si et seulement si l'arbre de dialogues a un sous-arbre solution. Un arbre de dialogues est *complet* si et seulement si chaque nœud correspondant à un argument x joué par PRO a pour fils exactement tous les attaquants de x.

Exemple 4.2 (sous-arbres solutions).

$\mathcal{SA}_2$ Les arbres correspondant aux ϕ_2-preuves représentées dans le tableau 4.3 page 61 sont des arbres de dialogues complets. Les arbres correspondant aux ϕ_2-preuves $\{\delta_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$ et $\{\delta'_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$ sont des sous-arbres solution de l'arbre correspondant à la ϕ_2-preuve $\{\delta_{\mathcal{SA}_2}, \delta'_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$. Ces trois arbres de dialogue sont gagnés par PRO.

◇

Comme on peut le constater avec l'exemple précédent, la notion de sous-arbre solution d'un arbre de dialogues complet est une notion de stratégie gagnante particulière (définition 4.5 page 49) dans le sens où elle autorise qu'un nœud joué par OPP ait plusieurs fils (cette caractéristique a été soulignée en section 4.2.3).

Néanmoins, l'existence d'un arbre gagné par PRO selon [AC02] est équivalente à l'existence d'une stratégie gagnante.

[AC02] ont établi la proposition suivante pour l'acceptabilité sous la sémantique basique :

Propriété 4.5. *Un argument a est* acceptable sous la sémantique basique *si et seulement si il existe un arbre de dialogue complet dont la racine est le coup* [PRO, a] *et gagné par* PRO.

La correction et la complétude de cette proposition sont montrées dans [AC02].

Exemple 4.2 (acceptabilité sous la sémantique basique).

$\mathcal{SA}_2$ L'argument a est acceptable sous la sémantique basique car les trois arbres de dialogues correspondant aux ϕ_2-preuves indiquées dans le tableau 4.3 page 61 sont complets et gagnés par PRO.

◇

En considérant la proposition 4.5 page précédente et l'équivalence entre les notions d'arbre complet gagné par PRO et de stratégie gagnante, conformément à l'idée de [AC02], la preuve de l'acceptabilité d'un argument sous la sémantique basique est une ϕ_{db}-preuve définie comme suit :

Définition 4.14 (ϕ_{db}-preuve selon [AC02]).
Soit un type de dialogue (A, R, ϕ_{db}) et un argument $x \in A$. Une ϕ_{db}-preuve pour x est une ϕ_{db}-stratégie gagnante S pour x.

[PS97] impose que chaque coup joué par PRO ait un seul fils dans l'arbre de dialogue gagné par PRO. En d'autres termes, une ϕ_{db}-preuve selon [PS97] est une ϕ_{db}-stratégie gagnante telle que tout préfixe d'un dialogue de la stratégie dont le dernier coup est joué par OPP a une seule extension dans la stratégie. Ceci peut être exprimé formellement :

Définition 4.15 (ϕ_{db}-preuve selon [PS97]).
Soit un type de dialogue (A, R, ϕ_{db}) et un argument $x \in A$. Une ϕ_{db}-preuve pour x est une ϕ_{db}-stratégie gagnante S pour x telle que : $\forall d \in S$, $\forall d'$ préfixe de d tel que der(d') *est joué par* OPP, *il existe un et un seul dialogue $d'' \in S$ tel que d'' est une extension de d'.*

EXEMPLE 2.1 (ϕ_{db}-PREUVES). L'extension basique de chacun des systèmes d'argumentation de cet exemple est indiquée dans l'exemple 2.1 (Extension basique) page 15 (la représentation graphique des systèmes est dans le tableau 2.1 page 16). Intéressons nous à l'acceptabilité sous la sémantique basique, en considérant la notion de ϕ_{db}-preuve selon [AC02] :

- *Zoé l'autruche* : construisons un ϕ_{db}-dialogue portant sur a :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_{db}(\mu_0) = \{b\} \\ \mu_1 = [\text{OPP}, b] & \phi_{db}(\mu_0\mu_1) = \{c\} \\ \mu_2 = [\text{PRO}, c] & \phi_{db}(\mu_0\mu_1\mu_2) = \emptyset \end{array}$$

  Ce ϕ_{db}-dialogue $d = \mu_0\mu_1\mu_2$ est gagné par PRO. L'ensemble $\{d\}$ est une *ϕ_{db}-preuve* de l'appartenance de a à l'extension basique du système.

- *Nixon* : construisons un ϕ_{db}-dialogue portant sur a :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_{db}(\mu_0) = \{b\} \\ \mu_1 = [\text{OPP}, b] & \phi_{db}(\mu_0\mu_1) = \emptyset \end{array}$$

  a n'appartient pas à $\phi_{db}(\mu_0\mu_1)$ car même si a attaque b, a est attaqué par b et a déjà été proposé par PRO. Ce ϕ_{db}-dialogue est perdu par PRO, il n'existe pas d'autre ϕ_{db}-dialogue. Il n'existe donc pas de ϕ_{db}-stratégie gagnante, et *a fortiori* pas de ϕ_{db}-preuve de l'appartenance de a à l'extension basique.

- *Le paradoxe du menteur* : construisons un ϕ_{db}-dialogue portant sur a :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, a] & \phi_{db}(\mu_0) = \{a\} \\ \mu_1 = [\text{OPP}, a] & \phi_{db}(\mu_0\mu_1) = \emptyset \end{array}$$

  Ce ϕ_{db}-dialogue est perdu par PRO. Il n'existe pas d'autre ϕ_{db}-dialogue portant sur a. Aucune ϕ_{db}-preuve pour a n'existe, donc a n'appartient pas à l'extension basique.

- *L'affaire des peintures* : construisons un ϕ_{db}-dialogue portant sur b :

$$\begin{array}{ll} \mu_0 = [\text{PRO}, b] & \phi_{db}(\mu_0) = \{c\} \\ \mu_1 = [\text{OPP}, c] & \phi_{db}(\mu_0\mu_1) = \{d\} \\ \mu_2 = [\text{PRO}, d] & \phi_{db}(\mu_0\mu_1\mu_2) = \{b\} \\ \mu_3 = [\text{OPP}, b] & \phi_{db}(\mu_0\mu_1\mu_2\mu_3) = \{c\} \\ \mu_4 = [\text{PRO}, c] & \phi_{db}(\mu_0\mu_1\mu_2\mu_3\mu_4) = \{d\} \\ \mu_5 = [\text{OPP}, d] & \phi_{db}(\mu_0\mu_1\mu_2\mu_3\mu_4\mu_5) = \emptyset \end{array}$$

b n'appartient pas à $\phi_{db}(\mu_0\mu_1\mu_2\mu_3\mu_4\mu_5)$ car, même si b attaque d, b a déjà été proposé par PRO au coup μ_0. Ce ϕ_{db}-dialogue portant sur b est unique et est perdu par PRO. Il n'existe donc pas de ϕ_{db}-stratégie gagnante portant sur b, et *a fortiori* pas de ϕ_{db}-preuve pour b, donc b n'appartient pas à l'extension basique.

♢

La différence entre la notion de ϕ_{db}-preuve selon [AC02] et celle selon [PS97] peut être vue sur l'exemple suivant :

Exemple 4.2 (ϕ_{db}-preuves).

$\mathcal{SA}_2$ L'extension basique de ce système est $\{a, e, f\}$. Les trois ϕ_{db}-dialogues portant sur a qui peuvent être construits sont identiques aux ϕ_2-dialogues $\delta_{\mathcal{SA}_2}$, $\delta'_{\mathcal{SA}_2}$ et $\delta''_{\mathcal{SA}_2}$ (voir le tableau 4.3 page 61). Les trois ϕ_2-preuves sont des ϕ_{db}-preuves au sens de [AC02], mais seules les ϕ_2-preuves $\{\delta'_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$ et $\{\delta_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$ sont des ϕ_{db}-preuves au sens de [PS97]. En effet, l'ensemble $\{\delta_{\mathcal{SA}_2}, \delta'_{\mathcal{SA}_2}, \delta''_{\mathcal{SA}_2}\}$ est une ϕ_{db}-stratégie gagnante, mais le dialogue $[\text{PRO}, a][\text{OPP}, b]$ a deux extensions dans l'ensemble : $\delta_{\mathcal{SA}_2}$ et $\delta'_{\mathcal{SA}_2}$.

♢

Notons que dans le cas d'un système d'argumentation bien-fondé, l'extension basique est identique à l'unique extension préférée. Il est donc possible dans ce cas d'utiliser les ϕ_1- et ϕ_2-théories de la preuve pour déterminer si un argument appartient à l'extension basique.

Prenons un dernier exemple :

Exemple 4.3. Considérons le système d'argumentation de la figure 4.4.

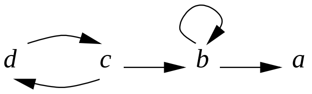


Fig. 4.4 – Système d'argumentation relativement basique, non cohérent, ayant plusieurs extensions préférées.

Ce système a deux extensions préférées : $\{d\}$ et $\{c, a\}$. Seul l'ensemble $\{c, a\}$ est une extension stable ; ce système n'est donc pas cohérent. L'extension basique est l'ensemble $\emptyset$, qui correspond à l'intersection des extensions préférées. Le système est donc relativement basique.

a n'appartient pas à l'extension basique, mais pour s'en convaincre, construisons le seul ϕ_{db}-dialogue portant sur a :

$$\begin{array}{ll}
\mu_0 = [\text{PRO}, a] & \phi_{db}(\mu_0) = \{b\} \\
\mu_1 = [\text{OPP}, b] & \phi_{db}(\mu_0\mu_1) = \{c\} \\
\mu_2 = [\text{PRO}, c] & \phi_{db}(\mu_0\mu_1\mu_2) = \{d\} \\
\mu_3 = [\text{OPP}, d] & \phi_{db}(\mu_0\mu_1\mu_2\mu_3) = \emptyset
\end{array}$$

Ce ϕ_{db}-dialogue est perdu par PRO, il n'existe donc pas de ϕ_{db}-stratégie gagnante, et *a fortiori* pas de ϕ_{db}-preuve de l'acceptabilité de a (ni au sens de [AC02], ni au sens de [PS97]). ♢

4.4 Acceptabilité sceptique sous la sémantique préférée

Nous avons vu en section 2.4 que le problème d'acceptabilité sceptique est difficile à résoudre. Nous allons montrer comment les théories de la preuve établies dans les sections 4.2 et 4.3 peuvent être utilisées pour répondre au problème dans plusieurs cas particuliers.

Tout d'abord, lorsqu'on sait que le système d'argumentation possède *une seule extension préférée*, nous avons montré que décider si un argument est sceptiquement accepté revient à décider s'il est crédulement accepté. Les ϕ_1-preuves et les ϕ_2-preuves peuvent donc être utilisées pour résoudre le problème.

Exemple 2.1 (suite). Les systèmes d'argumentation *Zoé l'autruche*, *Le paradoxe du menteur* et *L'affaire des peintures* possèdent une seule extension préférée. Les arguments acceptés crédulement sont tous acceptés sceptiquement dans ces systèmes. Les ϕ_1-preuves et les ϕ_2-preuves établies dans l'exemple 4.2 page 56 sont donc, dès qu'on sait qu'il n'y a qu'une seule extension préférée, des preuves de l'acceptabilité sceptique des arguments sur lesquels elles portent. ♢

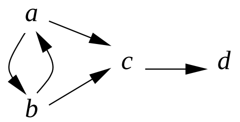
Exemple 4.2 (suite). Les systèmes $\mathcal{SA}_1$, $\mathcal{SA}_2$ et $\mathcal{SA}_4$ ont une seule extension préférée. Les ϕ_1-preuves établies dans l'exemple 4.2 page 56 et les ϕ_2-preuves établies dans l'exemple 4.2 (ϕ_2-preuves) page 60 sont donc des preuves de l'acceptabilité sceptique des arguments sur lesquels elles portent. ♢

Dans le cas général, un argument qui s'auto-attaque n'est bien sûr pas acceptable sous la sémantique préférée. La propriété 2.13 page 27 montre que si un argument ne s'auto-attaque pas, il est possible de prouver sa non-acceptabilité en donnant un de ses attaquants et une preuve de l'acceptabilité crédule de ce dernier, c'est-à-dire une ϕ_1-preuve ou une ϕ_2-preuve par exemple. Cette méthode de preuve est correcte, mais elle n'est pas complète dans tous les cas. On la sait complète dans le cas où le système est cohérent (propriété 2.14 page 27).

Exemple 2.1 (suite). Dans le système d'argumentation *Nixon*, l'argument b est crédulement accepté. La donnée de la ϕ_1-preuve de l'acceptabilité crédule de l'argument a qui attaque b, établie dans l'exemple page 55, et de a, montre que b n'est pas sceptiquement accepté. ♢

Exemple 4.2 (suite). Dans le système d'argumentation $\mathcal{SA}_3$, l'argument b a un seul prédécesseur, l'argument c. Dans l'exemple 4.2 page 56, nous avons établi une ϕ_1-preuve pour c et une ϕ_1-preuve pour b. La donnée de la ϕ_1-preuve pour c et de l'argument c montre que b n'est pas sceptiquement accepté. ♢

Exemple 4.4. Considérons le système d'argumentation suivant :



Il n'existe pas de ϕ_1-preuve de l'acceptabilité crédule de l'argument c. Puisque le système d'argumentation est cohérent, on peut en conclure que l'argument d est sceptiquement accepté. ♢

Dans le cas où le système d'argumentation est relativement basique, il est possible d'utiliser la ϕ_{db}-théorie de la preuve pour la sémantique basique pour déterminer si un argument est sceptiquement accepté.

Exemple 4.3 (suite). Nous avons vu qu'il n'existe pas de ϕ_{db}-dialogue gagné par PRO portant sur a. Le système d'argumentation étant relativement basique, cela permet de conclure que a n'est pas sceptiquement accepté sous la sémantique préférée. ♢

Exemple 4.4 (suite). Il n'existe pas de ϕ_{db}-dialogue portant sur d (d n'appartient pas à l'extension de base du système). Le système n'étant pas relativement basique, on ne peut pas conclure que d n'est pas sceptiquement accepté (et c'est heureux, puisque d est effectivement sceptiquement accepté). ♢

Pour ce qui est des systèmes non cohérents, qui ont plusieurs extensions préférées et qui ne sont pas relativement basiques, nous n'avons pas établi de théorie de la preuve dialectique pour déterminer l'acceptabilité sceptique sous la sémantique préférée.

Néanmoins, nous verrons dans le chapitre 6 un algorithme qui répond au problème d'acceptabilité sceptique dans le cas général.

4.5 Travaux voisins

Vreeswijk et Prakken ont défini dans [VP00] une théorie de la preuve appelée «Dispute entre deux partis avec réponse immédiate» ou TPI-disputes. Pour établir une preuve de l'**acceptabilité crédule** sous la sémantique préférée d'un argument a, le partisan doit exhiber un ensemble admissible contenant a. Pour cela, il dialogue avec l'opposant qui attaque de toutes les manières possibles les arguments du partisan. Le jeu est une succession de coups qui doivent satisfaire les conditions suivantes :

- PRO commence le jeu.
- Une *ligne de dispute* est une succession de coups tels que les joueurs jouent à tour de rôle et attaquent l'argument avancé au coup précédent par l'autre joueur. Dans une même ligne de dispute, OPP ne peut répéter un argument qu'il a déjà avancé, mais il peut répéter un argument déjà avancé par PRO.
- Chaque joueur peut revenir en arrière[3], ce qui consiste à ouvrir une nouvelle ligne de dispute. OPP peut se répéter dans différentes lignes de dispute. PRO peut se répéter, mais il ne peut répéter un argument déjà avancé par OPP.

OPP gagne une dispute si PRO ne peut répondre au dernier coup de OPP, ou si le dernier coup de OPP est un argument déjà proposé par PRO et sur lequel il n'y a pas eu de backtrack. PRO gagne une dispute si OPP ne gagne pas. Il est montré dans [VP00] qu'un argument a est crédulement accepté si et seulement si chaque dispute commençant par a est gagnée par PRO. Autrement dit, une preuve de l'acceptabilité crédule d'un argument est *l'ensemble des* TPI*-disputes* portant sur cet argument, qui doivent toutes être gagnées par PRO.

Dans [DBC01c], Dunne et Bench-Capon ont «formalisé» les TPI-disputes afin d'étudier leur longueur. La formalisation qu'ils ont proposée ne s'inscrit ni dans le cadre formel de [JV99a], ni dans celui de [Pra01]. Il nous est donc difficile de faire ici une comparaison formelle entre leurs travaux et les nôtres.

Nous faisons simplement une comparaison informelle entre les ϕ_1- et les ϕ_2-théories de la preuve, et les TPI-disputes, au travers de plusieurs exemples. Cette comparaison met en évidence le fait que les stratégies que nous utilisons permettent d'établir des preuves plus courtes, mais moins explicites que les TPI-disputes.

Exemple 4.5. Dans le tableau 4.5 sont indiquées deux preuves pour l'acceptabilité de a dans le système d'argumentation *Nixon* (exemple 2.1 page 11) : une ϕ_1-preuve, qui est celle décrite dans l'exemple page 55, et une TPI-dispute gagnée par PRO. Dans ce tableau sont également indiquées deux «preuves» de la non-acceptabilité crédule de b dans le système *L'affaire des peintures* : le seul ϕ_1-dialogue, qui est celui décrit dans l'exemple page 55, et une TPI-dispute gagnée par OPP. Pour le système *Le paradoxe du menteur*, la seule TPI-dispute pour l'argument a est également indiquée dans le tableau.

- *Nixon* : la TPI-dispute pour a est plus longue que la ϕ_1-preuve. La raison à cela est que les règles du jeu des TPI-disputes n'empêchent pas OPP d'avancer un argument qui est déjà attaqué par PRO : dans la TPI-dispute pour a, en réponse au coup $[\text{PRO}, a]$, OPP avance donc l'argument b, et PRO répète ensuite l'argument a.
- *L'affaire des peintures* : la TPI-dispute pour b est plus longue que l'unique ϕ_1-dialogue portant sur b. Cela vient du fait que dans les TPI-disputes, rien n'empêche PRO d'avancer un argument qui est déjà attaqué par un argument qu'il a précédemment proposé ; dans la TPI-dispute pour b, en réponse au coup $[\text{OPP}, c]$, PRO avance donc l'argument d et OPP repropose ensuite l'argument b. OPP montre ainsi qu'il y a un conflit dans l'ensemble des arguments proposés par PRO, et PRO ne peut plus jouer.
- *Le paradoxe du menteur* : puisque a s'auto-attaque, il n'existe aucun ϕ_1-dialogue portant sur a ; ceci montre que a n'est pas crédulement accepté. Pour montrer ceci, il existe, dans la théorie de Vreeswijk et Prakken, une TPI-dispute gagnée par OPP.

◇

---

[3] *backtracker* en anglais

Nixon : acceptabilité crédule de a		*L'affaire des peintures* : non-acceptabilité crédule de b		*Le paradoxe du menteur* : non-acceptabilité crédule de a	
ϕ_1-preuve	TPI-dispute	ϕ_1-dialogue	TPI-dispute	ϕ_1-dialogue	TPI-dispute
$[\text{PRO}, a]$	$[\text{PRO}, a]$ $[\text{OPP}, b]$ $[\text{PRO}, a]$	$[\text{PRO}, b]$ $[\text{OPP}, c]$	$[\text{PRO}, b]$ $[\text{OPP}, c]$ $[\text{PRO}, d]$ $[\text{OPP}, b]$	aucun	$[\text{PRO}, a]$ $[\text{OPP}, a]$

TAB. 4.5 – Comparaison de ϕ_1-dialogues et de TPI-disputes.

De manière générale, les ϕ_1 et les ϕ_2-théories de la preuve amènent des preuves plus courtes que les TPI-disputes pour une raison simple : les arguments qui sont attaqués par des arguments avancés par PRO dans le dialogue sont éliminés des coups possibles de PRO et OPP.

En retour, les TPI-disputes ont l'avantage de traiter les arguments qui s'auto-attaquent, et de montrer explicitement pourquoi ils ne sont pas crédulement acceptés.

Pour ce qui est du *niveau d'explication* donné par une preuve, nous dirons qu'une preuve peu explicite de l'acceptabilité crédule d'un argument est simplement la donnée d'un ensemble admissible contenant l'argument. Une preuve moyennement explicite est un ensemble admissible contenant l'argument et la façon dont cet ensemble admissible se défend contre ses attaquants. Une preuve très explicite est un ensemble admissible contenant l'argument, accompagné de la façon dont chaque argument de l'ensemble admissible se défend contre chacun de ses attaquants.

Les TPI-disputes sont des preuves très explicites, puisqu'elles mettent en évidence tous les attaquants des arguments avancés par PRO et une façon dont PRO se défend contre chacun.

Les ϕ_1 et ϕ_2-preuves sont quant à elles moyennement explicites. Elles ne mettent pas en évidence tous les attaquants de PRO, mais seulement la façon dont l'ensemble admissible se défend contre ses attaquants : les arguments avancés par OPP sont ceux contre lesquels PRO ne se défend pas encore au moment où ils sont avancés. Les ϕ_2-preuves sont néanmoins plus explicites que les ϕ_1-preuves car, comme nous l'avons vu en section 4.2.3, elles montrent plus d'attaquants de l'ensemble admissible et peuvent mettre en évidence plusieurs façons dont un argument de PRO se défend contre l'un de ses attaquants.

Les différents types de preuve pour l'acceptabilité crédule d'un argument sous la sémantique préférée peuvent être classés comme suit, du moins explicite au plus explicite :

$$\text{ensemble admissible seul} \quad < \quad \phi_1\text{-preuve} \quad < \quad \phi_2\text{-preuve} \quad < \quad \text{TPI-dispute}$$

Il est des situations où avoir des preuves explicites est plus important qu'avoir des preuves courtes.

Par exemple, les preuves données dans un dialogue engagé dans un contexte éducatif ne doivent pas être nécessairement courtes ; il faut surtout qu'elles soient claires, et c'est un avantage si elles sont très explicites.

Cependant, les contraintes de temps font souvent que l'enseignant ne peut passer tout le temps qu'il souhaiterait pour expliquer les preuves. Aussi, il fera le choix dans ces cas-là de donner des preuves courtes et rapides, parfois au détriment de leur expressivité.

Concernant le problème d'**acceptabilité sceptique**, Vreeswijk et Prakken définissent une théorie de la preuve qui est correcte et complète seulement dans le cas où le système est cohérent [VP00]. Ils montrent alors qu'un argument n'est pas sceptiquement accepté si et seulement si il ne peut pas être défendu dans toute TPI-dispute ou si l'un de ses attaquants peut être défendu dans toute TPI-dispute (et donc est crédulement accepté).

Vreeswijk et Prakken ne considèrent pas le cas où l'intersection des extensions préférées coïncide avec l'extension basique. Ainsi, ils ne peuvent déterminer l'acceptabilité sceptique d'aucun argument sous la sémantique préférée du système non cohérent mais relativement basique de l'exemple 4.3 page 66.

Chapitre 5

Algorithmes pour le calcul de preuves dialectiques

Nous avons étudié dans le chapitre précédent les aspects théoriques de preuves dialectiques pour les problèmes d'acceptabilité crédule et sceptique sous la sémantique préférée. Nous allons à présent proposer des algorithmes pour calculer ces preuves. Ces algorithmes ont été présentés dans l'article [CDM02].

5.1 Acceptabilité crédule sous la sémantique préférée

Rappelons tout d'abord que le problème d'acceptabilité crédule sous la sémantique préférée consiste à déterminer si un argument d'un système d'argumentation donné appartient à au moins une extension préférée du système.

Nous avons proposé deux théories de la preuve pour répondre à ce problème : la ϕ_1-théorie de la preuve et la ϕ_2-théorie de la preuve.

Ces deux théories de la preuve étant correctes et complètes pour le problème d'acceptabilité crédule sous la sémantique préférée, le problème d'acceptabilité crédule peut être reformulé comme suit : étant donné un système d'argumentation (A, R) et un argument $a \in A$, existe-t-il une ϕ_1-preuve (resp. ϕ_2-preuve) pour a ?

Une ϕ_2-preuve est représentée sous forme d'arbre, alors qu'une ϕ_1-preuve est séquentielle (c'est en fait un arbre réduit à une branche). Parce qu'il est plus aisé pour un algorithme de rechercher une preuve séquentielle qu'une preuve sous forme d'arbre, et parce que les deux théories de la preuve sont équivalentes, nous proposons un algorithme qui recherche une ϕ_1-preuve de l'acceptabilité d'un argument.

L'algorithme construit une preuve en énumérant récursivement tous les prédécesseurs (*i.e.* les attaquants directs) des arguments avancés par PRO : ces prédécesseurs sont les arguments de OPP dans la preuve en construction. Pour chaque argument b de OPP qui n'est pas encore attaqué par PRO, l'algorithme choisit l'un des prédécesseurs de b comme défenseur des arguments attaqués par b ; ce défenseur est avancé par PRO.

Les défenseurs choisis par l'algorithme peuvent s'avérer être de mauvais défenseurs, dans le sens où ils ne pourront pas être défendus face à de nouvelles attaques de OPP. Dans ce cas, l'algorithme, lorsqu'il s'en aperçoit, revient en arrière et cherche à proposer un autre défenseur. S'il n'en existe aucun, l'algorithme conclut qu'aucune ϕ_1-preuve ne peut être trouvée pour l'argument a. S'il en existe au moins un, PRO propose un nouveau défenseur et note que le précédent était mauvais (car avec lui PRO ne peut gagner un ϕ_1-dialogue), afin de ne pas perdre de temps à l'utiliser de nouveau. Le fait que PRO ne peut gagner un ϕ_1-dialogue avec ce mauvais défenseur est montré par le résultat suivant :

Théorème 5.1. *Soit* $d.[\mathrm{PRO}, x]$ *un* ϕ_1*-dialogue tel qu'aucun* ϕ_1*-dialogue qui a* $d.[\mathrm{PRO}, x]$ *en préfixe n'est gagné par* PRO. *Alors aucun* ϕ_1*-dialogue* d' *gagné par* PRO *tel que* $x \in \mathrm{PRO}(d')$ *n'a* d *comme préfixe.*

Preuve – Supposons qu'il existe un ϕ_1-dialogue d' gagné par PRO tel que $x \in \mathrm{PRO}(d')$ et tel que d est un préfixe de d'. Alors $S = \mathrm{PRO}(d')$ est admissible et contient $\mathrm{PRO}(d.[\mathrm{PRO}, x])$. D'après le lemme 4.2, $d.[\mathrm{PRO}, x]$ peut être étendu en un dialogue d'' tel que $\mathrm{PRO}(d'') \subseteq S$ et d'' est gagné par PRO. □

Présentons à présent notre algorithme pour le problème d'acceptabilité crédule. Nous allons nous placer selon le point de vue de PRO. Rappelons que le but de PRO est de construire un ensemble admissible contenant l'argument a dont l'acceptabilité est à prouver.

Supposons qu'à une étape donnée, un ϕ_1-dialogue d portant sur a a été construit, et que c'est au tour de OPP d'avancer un coup. À ce moment, OPP doit choisir un attaquant d'un argument de $\mathrm{PRO}(d)$ qui n'est pas déjà attaqué par $\mathrm{PRO}(d)$, c'est-à-dire un élément x de $R^-(\mathrm{PRO}(d)) \setminus R^+(\mathrm{PRO}(d))$. S'il n'en existe aucun, PRO a gagné le dialogue. S'il en existe un, PRO doit avancer un attaquant y de cet x (*i.e.* un défenseur de l'argument que x attaque), qui doit satisfaire plusieurs contraintes : y doit appartenir à l'ensemble $\mathrm{POSS}(d) = A \setminus (\mathrm{PRO}(d) \cup R^\pm(\mathrm{PRO}(d)) \cup \mathrm{Refl})$ et y ne doit pas être un attaquant qui a déjà été montré «mauvais».

On dénote par O l'ensemble des arguments qui, à une étape donnée, n'ont pas encore été avancés par PRO dans la branche de la preuve que l'algorithme est entrain de parcourir, et ne doivent pas être avancés par PRO ; O contient les arguments de $R^\pm(\mathrm{PRO}(d)) \cup \mathrm{Refl}$ et les «mauvais» défenseurs de a, ceux qui ont été essayés précédemment sans succès. Si un attaquant x de PRO a tous ses prédécesseurs dans O, alors PRO ne peut proposer un attaquant y de x, et donc PRO a perdu le dialogue.

L'algorithme qui permet de calculer des ϕ_1-preuves est celui de la fonction $\mathtt{CredQA}_{\mathtt{rec}}$. Cette fonction prend en entrée un ϕ_1-dialogue de longueur impaire d et un ensemble O disjoint de $\mathrm{PRO}(d)$. Elle retourne un ϕ_1-dialogue d' gagné par PRO, qui étend d, et tel que $\mathrm{PRO}(d')$ est disjoint de O, si un tel d' existe ; la fonction retourne $\perp$ si un tel d' n'existe pas.

La fonction $\mathtt{CredQA}_{\mathtt{rec}}$ est appelée par la fonction principale `CredQA`, qui vérifie avant toute chose que l'argument a dont on cherche à prouver l'acceptabilité, ne s'auto-attaque pas. Si effectivement a ne s'auto-attaque pas, $\mathtt{CredQA}_{\mathtt{rec}}$ est appelé avec $O = \mathrm{Refl} \cup R^\pm(a)$ et $d = [\mathrm{PRO}, a]$, sinon `CredQA` retourne directement $\perp$.

Exemple 5.1. Considérons les systèmes $\mathcal{SA}_2$, $\mathcal{SA}_3$ et $\mathcal{SA}_4$ de l'exemple 4.2 page 56 :

$\mathcal{SA}_2$ Démarrons l'algorithme :

$$\mathtt{CredQA}(R, a). \tag{5.1}$$

$a \notin \mathrm{Refl}$, donc le résultat de cet appel est celui de :

$$\mathtt{CredQA}_{\mathtt{rec}}(R, [\mathrm{PRO}, a], \{b, c\}). \tag{5.2}$$

Les deux prédécesseurs de a, *i.e.* b et c, n'ont aucun de leurs prédécesseurs dans O. Intéressons-nous à c ; on pose $x = c$. Le seul prédécesseur de c est e, donc $y = e$. Le résultat de la variable *res* est celui de l'appel suivant :

$$\mathtt{CredQA}_{\mathtt{rec}}(R, [\mathrm{PRO}, a][\mathrm{OPP}, c][\mathrm{PRO}, e], \{b, c\}). \tag{5.3}$$

Tous les prédécesseurs des arguments avancés par PRO sont successeurs des arguments de PRO. Par conséquent, le ϕ_1-dialogue $d''_{\mathcal{SA}_2} = [\mathrm{PRO}, a][\mathrm{OPP}, c][\mathrm{PRO}, e]$ est retourné.

$d''_{\mathcal{SA}_2}$ est le résultat de la variable *res* de l'appel 5.2, appel qui retourne donc $d''_{\mathcal{SA}_2}$. Ce résultat est celui de l'appel 5.1.

$d''_{\mathcal{SA}_2}$ est donc une ϕ_1-preuve de l'acceptabilité crédule de a.

Algorithme 1 `CredQA`$_{\texttt{rec}}(R, d, O)$

Entrée:

- une relation binaire R,
- un ϕ_1-dialogue de longueur impaire d (le prochain coup doit être joué par OPP)
- un ensemble O disjoint de PRO(d)

Sortie:

un ϕ_1-dialogue d' gagné par PRO qui étend d, tel que PRO(d') $\cap O = \emptyset$, si un tel d' existe,
$\perp$ sinon

```
Si il existe $x \in R^-(\mathrm{PRO}(d)) \setminus R^+(\mathrm{PRO}(d))$ tel que $R^-(x) \subseteq O$ Alors
  retourner $(\perp)$ ;
Sinon Si il existe $x \in R^-(\mathrm{PRO}(d)) \setminus R^+(\mathrm{PRO}(d))$ tel que $R^-(x) \not\subseteq O$ Alors
  $y \leftarrow$ un élément de $R^-(x) \setminus O$ ;
  $res \leftarrow \texttt{CredQA}_{\texttt{rec}}(R, d.[\mathrm{OPP}, x].[\mathrm{PRO}, y], O \cup R^{\pm}(y))$ ;
  Si $res \neq \perp$ Alors
    retourner $(res)$ ;
  Sinon
    retourner $(\texttt{CredQA}_{\texttt{rec}}(R, d, O \cup \{y\}))$ ;
  Fin Si
Sinon
  retourner $(d)$ ;
Fin Si
```

Algorithme 2 `CredQA`(R, a)

Entrée:

- une relation binaire R,
- un argument a

Sortie:

une ϕ_1-preuve pour a si a est crédulement accepté,
$\perp$ sinon

```
Si $a \in \mathrm{Refl}$ Alors
  retourner $(\perp)$ ;
Sinon
  retourner $(\texttt{CredQA}_{\texttt{rec}}(R, [\mathrm{PRO}, a], \mathrm{Refl} \cup R^{\pm}(a)))$ ;
Fin Si
```

$\mathcal{SA}_3$ Démarrons l'algorithme :

$$\texttt{CredQA}(R, a). \tag{5.4}$$

$a \notin$ Refl, donc le résultat de cet appel est celui de :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\mathrm{PRO}, a], \{b, e\}). \tag{5.5}$$

Les deux prédécesseurs de a sont b et e. Aucun de ces deux arguments n'a tous ses prédécesseurs dans O. Intéressons-nous à b ; on pose $x = b$. b a pour seul prédécesseur c, donc $y = c$. Le résultat de la variable *res* est celui de l'appel suivant :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\mathrm{PRO}, a][\mathrm{OPP}, b][\mathrm{PRO}, c], \{b, e, f\}). \tag{5.6}$$

L'argument e attaque l'argument a avancé par PRO, il n'est pas attaqué par PRO, et son seul prédécesseur est f, un argument qui est dans O (f est dans O car il attaque et est attaqué par un argument déjà avancé par PRO, *i.e.* c). Ceci signifie que l'argument c, initialement choisi par PRO comme défenseur de a, ne peut être défendu contre son attaquant. C'est donc un «mauvais» défenseur. Par conséquent, $\perp$ est retourné.

$\perp$ est le résultat de la variable *res* de l'appel 5.5. Le résultat de l'appel 5.5 est donc celui de l'appel :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\mathrm{PRO}, a], \{b, e, c\}). \tag{5.7}$$

Notons que l'argument c a été placé dans O puisque c'est un «mauvais» défenseur de a (il n'a pas à être avancé par PRO dans la suite). L'argument b est prédécesseur de a, il n'est pas successeur de a, et son seul prédécesseur, *i.e.* c, est dans O. Donc $\perp$ est retourné.

$\perp$ est donc le résultat de l'appel 5.5, et ainsi de l'appel 5.4.

Il n'existe donc pas de ϕ_1-preuve de l'acceptabilité crédule de a.

$\mathcal{SA}_4$ Démarrons l'algorithme :

$$\texttt{CredQA}(R, a). \tag{5.8}$$

$a \notin$ Refl, donc le résultat de cet appel est celui de :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\mathrm{PRO}, a], \{b\}). \tag{5.9}$$

Le seul prédécesseur de a, *i.e.* b, a deux prédécesseurs, c et e, qui ne sont pas dans O. On choisit $x = b$ et $y = c$. Le résultat de la variable *res* est celui de l'appel suivant :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\mathrm{PRO}, a][\mathrm{OPP}, b][\mathrm{PRO}, c], \{b, f\}). \tag{5.10}$$

Le seul argument qui attaque un argument de PRO et qui n'est pas attaqué par PRO est f (f attaque c). f n'a aucun prédécesseur. Ceci signifie que l'argument c, initialement choisi par PRO comme défenseur de a, ne peut être défendu contre son attaquant. C'est donc un «mauvais» défenseur. Par conséquent, $\perp$ est retourné.

$\perp$ est le résultat de la variable *res* de l'appel 5.9. Donc le résultat de l'appel 5.9 est celui de l'appel suivant :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\mathrm{PRO}, a], \{b, c\}). \tag{5.11}$$

L'argument c a été placé dans O puisqu'il est un «mauvais» défenseur de a. Le seul attaquant de a, *i.e.* b, n'a plus qu'un seul prédécesseur qui n'est pas dans O, *i.e.* e. On choisit par exemple $x = b$ et $y = e$. Le résultat de la variable *res* est celui de l'appel suivant :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\mathrm{PRO}, a][\mathrm{OPP}, b][\mathrm{PRO}, e], \{b, c\}). \tag{5.12}$$

Tous les prédécesseurs des arguments avancés par PRO sont successeurs d'arguments avancés par PRO. Par conséquent, le ϕ_1-dialogue $d'_{\mathcal{SA}_4} = [\mathrm{PRO}, a][\mathrm{OPP}, b][\mathrm{PRO}, e]$ est retourné.

$d'_{\mathcal{SA}_4}$ est le résultat de la variable *res* de l'appel 5.11, appel qui retourne donc $d'_{\mathcal{SA}_4}$.

Ce résultat est celui de l'appel 5.9 et donc de l'appel 5.8.

$d'_{\mathcal{SA}_4}$ est une ϕ_1-preuve de l'acceptabilité crédule de a.

$\diamondsuit$

5.2 Acceptabilité sceptique sous la sémantique préférée

Le problème d'acceptabilité sceptique sous la sémantique préférée consiste à déterminer si un argument d'un système d'argumentation donné appartient à toutes les extensions préférées du système.

Nous avons vu en section 2.4 que lorsque le système d'argumentation ne possède qu'*une seule extension préférée*, chercher une preuve de l'acceptabilité sceptique d'un argument revient à chercher une preuve de son acceptabilité crédule. Dans ce cas, il suffit d'utiliser la fonction `CredQA` pour déterminer si un argument est sceptiquement accepté. C'est ce que fait la fonction $\texttt{SceptQA}_{\texttt{uni}}$ de l'algorithme 3 (l'indice «`uni`» souligne le fait que cette fonction n'est correcte et complète que lorsque le système d'argumentation a une extension préférée unique).

Algorithme 3 $\texttt{SceptQA}_{\texttt{uni}}(R, a)$

```
Entrée:
    – une relation binaire $R$,
    – un argument $a$
Sortie:
    une $\phi_1$-preuve pour $a$ si $a$ est sceptiquement accepté,
    $\bot$ sinon
retourner ($\texttt{CredQA}(R, a)$) ;
```

Sachant qu'un argument est non-acceptable sceptiquement s'il n'appartient pas à au moins une extension préférée du système, nous avons montré en section 2.4 qu'une preuve de la non-acceptabilité sceptique d'un argument peut prendre la forme d'une paire composée d'un attaquant de l'argument et d'une preuve de l'acceptabilité crédule de cet attaquant. Cette preuve de la non-acceptabilité sceptique est correcte, mais on ne la sait complète que dans le cas où le système est *cohérent*.

Cette méthode est implémentée par la fonction $\texttt{SceptQA}_{\texttt{coher}}$ de l'algorithme 4 (l'indice «`coher`» souligne le fait que cette fonction n'est complète que lorsque le système est cohérent).

a étant l'argument dont on cherche à prouver l'acceptabilité sceptique, si a s'auto-attaque, il n'est pas sceptiquement accepté et $\texttt{SceptQA}_{\texttt{coher}}$ retourne $\bot$. Si a ne s'auto-attaque pas, $\texttt{SceptQA}_{\texttt{coher}}$ utilise la fonction $\texttt{CredQA}_{\texttt{rec}}$ pour chercher une ϕ_1-preuve pour chaque attaquant x de a qui ne s'auto-attaque pas. Si une telle ϕ_1-preuve d est trouvée, la paire (x, d) est retournée comme preuve de la non-acceptabilité de a. Si aucune ϕ_1-preuve n'est trouvée pour aucun des attaquants de a, alors $\top$ est retourné pour dire que a est sceptiquement accepté. Notons que lorsqu'aucune ϕ_1-preuve de l'acceptabilité d'un attaquant x de a n'existe, alors x n'appartient à aucun ensemble admissible du système d'argumentation. x est donc rejeté des coups possibles de PRO pour tous les dialogues qui vont être explorés par la suite, *i.e.* x est rangé dans O.

Exemple 5.2. Considérons le système d'argumentation $\mathcal{SA}_3$ indiqué dans le tableau 4.1 page 56. Nous avons vu dans l'exemple 4.2 page 56 que l'argument b est crédulement accepté. Pour savoir s'il est sceptiquement accepté, lançons l'algorithme :

$$\texttt{SceptQA}_{\texttt{coher}}(R, b). \tag{5.13}$$

$b \notin$ Refl. b a un seul prédécesseur, l'argument c, donc $X = \{c\}$. $O = \emptyset$. Pour l'élément $x = c$ de X, la fonction va tenter de construire une ϕ_1-preuve, *via* l'appel suivant :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\text{PRO}, c], \{b, f\}). \tag{5.14}$$

Dans le ϕ_1-dialogue $[\text{PRO}, c]$, c attaque le seul argument qui l'attaque, f. Donc ce ϕ_1-dialogue est retourné, c'est une ϕ_1-preuve de l'acceptabilité crédule de c.

$[\text{PRO}, c]$ est le résultat de la variable d de l'appel 5.13. Donc cet appel retourne la paire $(c, [\text{PRO}, c])$ qui montre que b n'est pas sceptiquement accepté.

Algorithme 4 $\texttt{SceptQA}_{\texttt{coher}}(R, a)$

Entrée:
- une relation binaire R,
- un argument a

Sortie:
$\top$ si a est sceptiquement accepté,
$\bot$ si a s'auto-attaque,
une paire (x, d) telle que x attaque a et d est une ϕ_1-preuve pour x sinon

```
Si $a \in \text{Refl}$ Alors
  retourner $(\bot)$ ;
Sinon
  $X \leftarrow R^-(a) \setminus \text{Refl}$ ;
  $O \leftarrow \text{Refl}$ ;
  Tant que $X \neq \emptyset$ Faire
    $x \leftarrow$ un élément de $X$ ;
    $X \leftarrow X \setminus \{x\}$ ;
    $d \leftarrow \texttt{CredQA}_{\texttt{rec}}(R, [\text{PRO}, x], O \cup R^{\pm}(x))$ ;
    Si $d \neq \bot$ Alors
      retourner $((x, d))$ ;
    Sinon
      $O \leftarrow O \cup \{x\}$ ;
    Fin Si
  Fin Tant que
  retourner $(\top)$ ;
Fin Si
```

Le système $\mathcal{SA}_4 = (A, R)$ possède une seule extension préférée et est cohérent. Pour savoir si l'argument a est sceptiquement accepté, il suffit d'utiliser la fonction $\texttt{SceptQA}_{\texttt{uni}}$. Néanmoins, il est également possible d'utiliser la fonction $\texttt{SceptQA}_{\texttt{coher}}$; lançons l'algorithme :

$$\texttt{SceptQA}_{\texttt{coher}}(R, a). \tag{5.15}$$

$a \notin$ Refl. a a un seul prédécesseur, l'argument b, donc $X = \{b\}$. $O = \emptyset$. Pour l'élément $x = c$ de X, la fonction va tenter de construire une ϕ_1-preuve, grâce à l'appel suivant :

$$\texttt{CredQA}_{\texttt{rec}}(R, [\text{PRO}, b], \{a, c, e\}). \tag{5.16}$$

L'argument b a pour prédécesseur l'argument e et e n'a aucun prédécesseur : b ne peut donc être défendu contre e. Il n'existe donc pas de ϕ_1-preuve de l'acceptabilité crédule de b. $\bot$ est retourné par la fonction.

Ce résultat est celui de la variable d de l'appel 5.16. b est rangé dans O, l'ensemble X est vide. Donc $\top$ est retourné par l'appel 5.15. a est sceptiquement accepté. $\Diamond$

L'extension basique étant incluse dans l'intersection des extensions préférées, une preuve de l'acceptabilité d'un argument sous la sémantique basique peut être une preuve de son acceptabilité sceptique sous la sémantique préférée. Cette méthode pour déterminer l'acceptabilité sceptique est correcte, mais elle n'est complète que lorsque le système est relativement basique, c'est-à-dire lorsque l'intersection des extensions préférées et l'extension basique coïncident.

Pour ce cas, il serait intéressant d'établir un algorithme qui détermine si un argument appartient à l'extension basique. Ceci fera l'objet de travaux futurs.

Pour le cas général (celui d'un système d'argumentation non cohérent, non relativement basique et possédant plusieurs extensions préférées), une théorie de la preuve reste à établir.

Chapitre 6

Algorithmes pour la sémantique préférée

Dans ce chapitre, nous allons nous intéresser d'un point de vue algorithmique à certaines sémantiques de Dung et aux problèmes d'acceptabilités associés, sans utiliser la notion de preuve.

Étant donné un système d'argumentation (A, R), nous allons donner des algorithmes pour :

1. énumérer les parties de A ;
2. calculer les ensembles sans-conflit de (A, R) ;
3. calculer les ensembles admissibles de (A, R) ;
4. calculer les extensions préférées de (A, R) ;
5. calculer les extensions stables de (A, R) ;
6. répondre au problème d'acceptabilité crédule sous la sémantique préférée ;
7. répondre au problème d'acceptabilité sceptique sous la sémantique préférée.

Une première version de l'algorithme 4 a été publiée dans [Dou00, DM00], une deuxième version améliorée ainsi que les algorithmes 6 et 7 ont été publiés dans [DM01a], l'algorithme 5 se trouve dans [DM01b].

L'algorithme que nous allons proposer pour répondre au problème d'acceptabilité crédule est différent de celui proposé dans le chapitre précédent ; en effet, ce nouvel algorithme cherche un ensemble admissible contenant l'argument dont l'acceptabilité est à prouver, et renvoie vrai si un tel ensemble existe, faux sinon. Il ne renvoie pas une preuve de l'acceptabilité.

L'algorithme pour répondre au problème d'acceptabilité sceptique va en fait répondre au problème complémentaire (qui consiste à déterminer s'il existe une extension ne contenant pas l'argument de la requête), pour retourner vrai si l'argument est sceptiquement accepté, faux sinon. Cet algorithme ne renvoie pas une preuve de l'acceptabilité sceptique d'un argument, mais il est complet vis-à-vis du problème, contrairement à l'algorithme proposé dans le chapitre précédent.

Les ensembles admissibles sont des sous-ensembles particuliers de l'ensemble A des arguments d'un système d'argumentation (A, R). Pour calculer ces sous-ensembles, une approche naïve consisterait à générer tous les sous-ensembles de A et à déterminer ensuite ceux qui sont admissibles.

Une méthode générale pour générer tous les sous-ensembles de A, appelée *énumération d'ensembles*, consiste à choisir un élément a de A dans le but de générer séparément tous les sous-ensembles de A qui contiennent a et tous ceux qui ne contiennent pas a.

Nous présentons dans un premier temps cette méthode, et nous montrons que la recherche des parties prend la forme d'un arbre binaire dont les feuilles correspondent à tous les sous-ensembles de A.

Tous les ensembles que l'on va vouloir générer étant sans-conflit, nous allons modifier cet algorithme pour qu'il n'énumère que les parties sans-conflit d'un ensemble.

Nous allons ensuite voir sous quelles conditions des feuilles de l'arbre de recherche des parties sans-conflit correspondent à des ensembles admissibles, et sous quelles conditions des branches de cet arbre peuvent être «coupées» quand il est sûr qu'aucun ensemble admissible ne se trouve dans les «descendants» d'un nœud donné.

Ensuite, nous allons voir comment adapter l'algorithme d'énumération des parties sans-conflit pour qu'il calcule une sémantique d'un système d'argumentation. Pour le calcul de la sémantique préférée, nous allons proposer plusieurs heuristiques pour le choix des nœuds de l'arbre de recherche, ainsi que des propriétés de coupure et les conditions sous lesquelles une feuille de l'arbre est une extension préférée. Ces heuristiques peuvent également être utilisées pour optimiser le calcul des ensembles admissibles et pour calculer les extensions stables et l'extension basique.

Nous allons également voir comment adapter l'algorithme d'énumération des parties sans-conflit pour répondre à un problème d'acceptabilité sous la sémantique préférée. Nous utiliserons les heuristiques et les propriétés de coupure précédemment établies, ainsi que de nouvelles propriétés.

Nous comparerons enfin nos travaux avec des travaux existants.

6.1 Énumération des parties d'un ensemble

L'énumération des sous-ensembles d'un ensemble A peut être effectuée en explorant un arbre binaire[1], dont les nœuds sont étiquetés par une partition de A en trois ensembles I, O et U.

Pour un nœud n, la signification des ensembles de la partition (I, O, U) est la suivante :

- I est un ensemble d'éléments qui seront *dans* tout sous-ensemble de A trouvé au niveau d'une feuille descendante de n ;
- O est un ensemble d'éléments qui seront *hors* de tout sous-ensemble de A trouvé au niveau d'une feuille descendante de n ;
- $U = A \setminus (I \cup O)$ est un ensemble d'éléments dont le statut est *non décidé* : ils peuvent se retrouver au niveau d'une feuille descendante de n, et se retrouver hors d'un ensemble au niveau d'une autre feuille descendante de n.

Si $U = \emptyset$, le nœud est une feuille.

La fonction `EnumParties` (algorithme 6) permet d'énumérer toutes les parties d'un ensemble. Elle fait appel à la fonction récursive $\texttt{EnumParties}_{\texttt{rec}}$ (algorithme 5) qui, grâce à une fonction `Select` qui sélectionne «arbitrairement» un élément dans un ensemble, sélectionne à chaque étape un élément x dans U, et se rappelle sur deux nouvelles partitions ; dans l'une, x est rangé dans I, dans l'autre, x est rangé dans O. L'ensemble U est ainsi progressivement vidé.

Algorithme 5 $\texttt{EnumParties}_{\texttt{rec}}(I, O, U)$

Entrée: trois ensembles disjoints I, O et U
Sortie: $\{S \mid I \subseteq S \subseteq (I \cup U)\}$

```
Si $U = \emptyset$ Alors
    retourner $(\{I\})$ ;
Sinon
    $x \leftarrow \texttt{Select}(U)$ ;
    retourner $(\texttt{EnumParties}_{\texttt{rec}}(I \cup \{x\}, O, U \setminus \{x\}) \cup \texttt{EnumParties}_{\texttt{rec}}(I, O \cup \{x\}, U \setminus \{x\}))$ ;
Fin Si
```

Algorithme 6 $\texttt{EnumParties}(A)$

Entrée: un ensemble A
Sortie: $\{S \mid S \subseteq A\}$

```
retourner $(\texttt{EnumParties}_{\texttt{rec}}(\emptyset, \emptyset, A))$ ;
```

---

[1] Un *arbre binaire* est un arbre dont chaque nœud qui n'est pas une feuille, a au plus deux fils.

L'arbre binaire représentant l'exécution de la fonction est tel qu'un nœud n étiqueté par une partition (I, O, U) où $U \neq \emptyset$, après sélection d'un argument x de U, a deux fils n_1 et n_2. La partition associée à n_1 est $(I \cup \{x\}, O, U \setminus \{x\})$, celle associée à n_2 est $(I, O \cup \{x\}, U \setminus \{x\})$. Pour souligner ce qu'il advient de l'argument x, la branche allant de n à n_1 est étiquetée x, celle allant de n à n_2 est étiquetée $\neg x$ (figure 6.1).

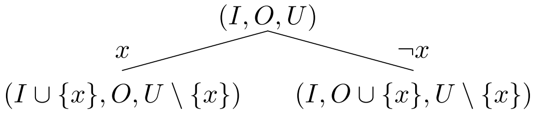


Fig. 6.1 – Schéma d'un nœud de l'arbre binaire d'énumération.

Exemple 6.1. La figure 6.2 donne l'arbre d'énumération de toutes les parties de l'ensemble $A = \{a, b, c\}$ des arguments du système d'argumentation *Zoé l'autruche* (exemple 2.1 page 11). Au niveau d'une feuille, seul l'ensemble I est représenté ; puisque $U = \emptyset$, l'ensemble O est égal à $A \setminus I$. ♢

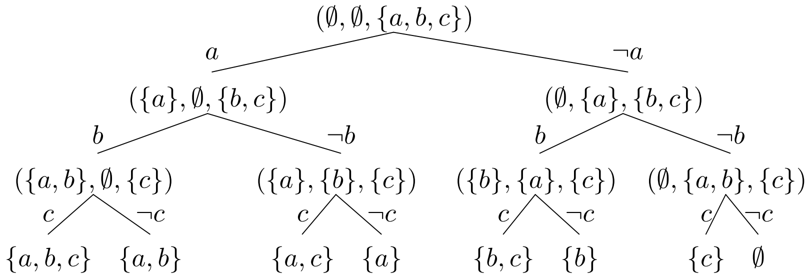


Fig. 6.2 – Arbre d'énumération des parties de $A = \{a, b, c\}$.

Cette technique d'énumération d'ensembles va être adaptée pour énumérer les ensembles sans-conflit.

6.2 Énumération des parties sans-conflit

Étant donné un ensemble A et une relation binaire $R \subseteq A \times A$, nous allons modifier l'algorithme d'énumération d'ensembles pour qu'il retourne seulement les sous-ensembles sans-conflit de A. Les feuilles de l'arbre d'énumération seront alors des ensembles sans-conflit.

Les feuilles descendantes d'un nœud n étiqueté (I, O, U) contiennent I mais ne contiennent aucun élément de O. Si I est sans-conflit, pour que les partitions associées aux fils de n soient telles que I est aussi sans-conflit, nous n'allons générer que les nœuds tels que $R^{\pm}(I) \subseteq O$.

Introduisons quelques définitions et notations qui vont aider à la présentation de nos résultats.

Définition 6.1 (*R*-candidat).
*Un R-*candidat *est un triplet (I, O, U) d'ensembles disjoints tels que $R^{\pm}(I) \subseteq O$.*

Notation 6.1. *Étant donné un R-candidat $C = (I, O, U)$ et un élément x de U, on dénote par :*
- *$C + x$ le triplet $(I \cup \{x\}, O \cup R^{\pm}(x), U \setminus (\{x\} \cup R^{\pm}(x)))$,*
- *$C - x$ le triplet $(I, O \cup \{x\}, U \setminus \{x\})$.*

Le résultat qui suit montre que $C + x$ et $C - x$ sont des R-candidats :

Propriété 6.1. *Étant donné une relation binaire R, un R-candidat $C = (I, O, U)$ et un élément $x \in U$ tel que $x \notin R^+(x)$, $C + x$ et $C - x$ sont des R-candidats.*

Preuve – Dans les preuves, nous allons utiliser les identités logiques suivantes, où A, B, C et D sont des ensembles :

$$\begin{array}{rcl} A \cup (B \setminus A) & = & A \cup B \\ (A \cup B) \cap (C \setminus B) & = & (A \cap C) \setminus B \\ ((A \setminus B) \cup C) \setminus (B \cup D) & = & (A \cup C) \setminus (B \cup D) \end{array}$$

- Prouvons que :

  $$C + x = (I', O', U') = (I \cup \{x\}, O \cup R^\pm(x), U \setminus (\{x\} \cup R^\pm(x)))$$

  est un R-candidat. Notons que puisque $x \notin R^+(x)$, $\{x\} \cap R^\pm(x) = \emptyset$. Prouvons tout d'abord que I', O' et U' sont des ensembles disjoints :

  $$I' \cap U' \;=\; (I \cup \{x\}) \cap (U \setminus (\{x\} \cup R^\pm(x)))$$

  est un ensemble inclus dans :

  $$(I \cup \{x\}) \cap (U \setminus \{x\}) \;=\; (I \cap U) \setminus \{x\} \;=\; \emptyset$$

  car I et U sont des ensembles disjoints.

  $$\begin{array}{rcl} I' \cap O' & = & (I \cup \{x\}) \cap (O \cup R^\pm(x)) \\ & = & (I \cap O) \cup (I \cap R^\pm(x)) \cup (\{x\} \cap O) \cup (\{x\} \cap R^\pm(x)) \\ & = & \emptyset \end{array}$$

  puisque $R^\pm(I) \subseteq O$, et puisque $x \in U$, $R^\pm(I) \cap \{x\} = \emptyset$, qui est équivalent à $I \cap R^\pm(x) = \emptyset$.

  $$\begin{array}{rcl} U' \cap O' & = & (U \setminus (\{x\} \cup R^\pm(x))) \cap (O \cup R^\pm(x)) \\ & = & (U \setminus \{x\}) \cap ((U \setminus R^\pm(x)) \cap (O \cup R^\pm(x))) \\ & = & (U \setminus \{x\}) \cap ((U \cap 0) \setminus R^\pm(x)) \end{array}$$

  Ce dernier ensemble est inclus dans $U \cap O = \emptyset$.

  Il est clair que $I' \cup O' \cup U' = I \cup O \cup U$ puisque U' est obtenu en enlevant des éléments de U qui sont ajoutés à I ou O pour former I' et O'. Pour que $C + x$ soit un R-candidat, il reste à prouver que $R^\pm(I') \subseteq O'$:

  $$\begin{array}{rcl} R^\pm(I') & = & R^\pm(I \cup \{x\}) \\ & = & R^\pm(I) \cup R^\pm(x) \quad \subseteq \quad O \cup R^\pm(x) = O' \end{array}$$

- Prouvons à présent que

  $$C - x = (I'', O'', U'')$$

  est un R-candidat.

  Il est clair que (I'', O'', U'') est un triplet d'ensembles disjoints et que $I'' \cup O'' \cup U'' = I \cup O \cup U$ car :

  - I, O et U sont disjoints,
  - $I'' = I$, et
  - O'' est formé en ajoutant à O un élément x qui est enlevé de U pour obtenir U''.

  De plus, $R^\pm(I'') = R^\pm(I)$ et $R^\pm(I) \subseteq O \subseteq O \cup \{x\} = O''$.

□

La propriété suivante montre que les arguments qui s'auto-attaquent doivent être rangés dans O puisqu'ils sont exclus de tout ensemble sans-conflit :

Propriété 6.2. *Étant donné un système d'argumentation (A, R) :*

$$\{S \subseteq A \mid S \cap R^+(S) = \emptyset\} = \{S \subseteq (A \setminus \mathrm{Refl}(A, R)) \mid S \cap R^+(S) = \emptyset\}.$$

Preuve – Prenons $S \subseteq A$ tel que $S \cap R^+(S) = \emptyset$. Pour tout $x \in \mathrm{Refl}(A, R)$, on a $x \cap R^+(x) \neq \emptyset$, donc $x \notin S$. Par conséquent, $S \cap \mathrm{Refl}(A, R) = \emptyset$ et donc $S \subseteq (A \setminus \mathrm{Refl}(A, R))$.

De plus, comme $(A \setminus \mathrm{Refl}(A, R)) \subseteq A$, tout ensemble $S \subseteq (A \setminus \mathrm{Refl}(A, R))$ tel que $S \cap R^+(S) = \emptyset$ est inclus dans A. □

Étant donné un ensemble A et une relation binaire R, la fonction `EnumPartiesSansconflit` (algorithme 8) permet d'énumérer toutes les parties sans-conflit de A. Elle fait pour cela appel à la fonction $\texttt{EnumPartiesSansconflit}_{\texttt{rec}}$ (algorithme 7).

Cet algorithme va sélectionner, à chaque étape, un argument non décidé x pour générer deux nouveaux R-candidats ; dans l'un d'eux, x sera rangé dans I et ses prédécesseurs et successeurs iront dans O ; dans l'autre, x sera rangé dans O. De cette manière, U va être progressivement vidé.

À l'appel de $\texttt{EnumPartiesSansconflit}_{\texttt{rec}}$ par `EnumPartiesSansconflit`, tous les arguments de (A, R) qui s'auto-attaquent sont rangés dans O (puisqu'ils n'appartiennent à aucun ensemble sans-conflit).

Algorithme 7 $\texttt{EnumPartiesSansconflit}_{\texttt{rec}}(R, C)$

```
Entrée:
  – une relation binaire $R$,
  – un $R$-candidat $C = (I, O, U)$ ;
Sortie: $\{S \mid I \subseteq S \subseteq ((I \cup U) \setminus R^{\pm}(I))\}$
Si $U = \emptyset$ Alors
  retourner $(\{I\})$ ;
Sinon
  $x \leftarrow \texttt{Select}(U)$ ;
  retourner $(\texttt{EnumPartiesSansconflit}_{\texttt{rec}}(R, C + x) \cup \texttt{EnumPartiesSansconflit}_{\texttt{rec}}(R, C - x))$ ;
Fin Si
```

Algorithme 8 $\texttt{EnumPartiesSansconflit}(A, R)$

```
Entrée: un ensemble $A$ d'arguments et une relation binaire $R$ ;
Sortie: $\{S \mid S \subseteq A \text{ et } S \cap R^{+}(S) = \emptyset\}$
retourner $(\texttt{EnumPartiesSansconflit}_{\texttt{rec}}(R, (\emptyset, \text{Refl}(A, R), A \setminus \text{Refl}(A, R))))$ ;
```

Exemple 6.1 (suite). La figure 6.3 donne l'arbre d'énumération de toutes les parties sans-conflit de l'ensemble A des arguments du système d'argumentation *Zoé l'autruche* (indiqué dans l'exemple 2.1 page 11) ; ce système est tel que $A = \{a, b, c\}$ et $R = \{(c, b), (b, a)\}$.

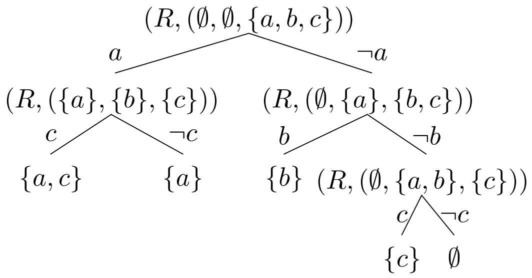


Fig. 6.3 – Énumération des parties sans-conflit de $A = \{a, b, c\}$, avec $R = \{(c, b), (b, a)\}$.

Une autre représentation possible des nœuds de l'arbre de recherche est celle de la figure 6.4 page suivante : à chaque nœud est représenté le système d'argumentation ; les arguments encerclés sont dans I, les arguments barrés sont dans O, les autres arguments sont dans U. ♢

L'algorithme d'énumération des parties sans-conflit va maintenant être adapté pour trouver les ensembles admissibles.

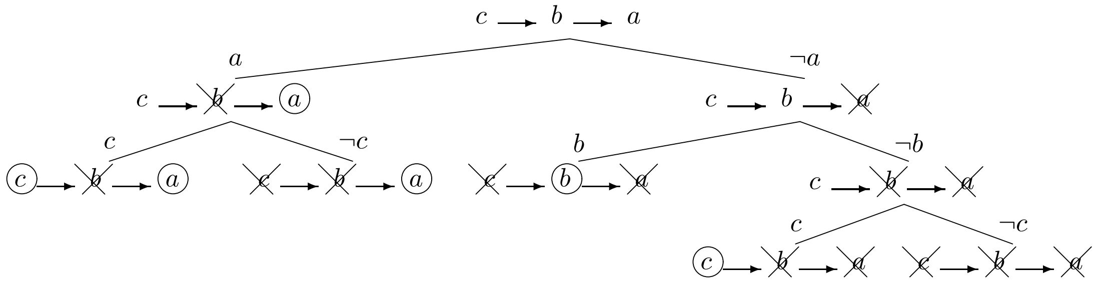


FIG. 6.4 – Énumération des parties sans-conflit de l'ensemble des arguments du système d'argumentation *Zoé l'autruche*.

6.3 Énumération des ensembles admissibles

Étant donné un système d'argumentation (A, R), le résultat suivant montre que l'énumération des ensembles admissibles peut être complète en explorant seulement les nœuds de l'arbre d'énumération des parties sans-conflit qui sont des R-candidats. Avant de donner ce résultat, nous introduisons une notation :

Notation 6.2. *L'ensemble des ensembles admissibles qui apparaissent au niveau des feuilles descendantes d'un nœud étiqueté (I, O, U) est dénoté par :*

$$\mathrm{Adm}^*((I,O,U),R) = \{S \in \mathrm{Adm}(I \cup O \cup U, R) \mid I \subseteq S \subseteq (I \cup U)\}.$$

Propriété 6.3. *Étant donné une relation binaire R, un R-candidat $C = (I, O, U)$ et un élément $x \in U$ tel que $x \notin R^+(x)$,*

$$\mathrm{Adm}^*(C,R) = \mathrm{Adm}^*(C+x,R) \cup \mathrm{Adm}^*(C-x,R).$$

PREUVE – Rappelons que $S \in \mathrm{Adm}(A, R)$ si et seulement si $S \cap R^+(S) = \emptyset$ et $R^-(S) \subseteq R^+(S)$. Posons :

$$\begin{array}{lllll} C+x & = & (I',O',U') & = & (I \cup \{x\}, O \cup R^{\pm}(x), U \setminus (\{x\} \cup R^{\pm}(x))) \\ C-x & = & (I'',O'',U'') & = & (I, O \cup \{x\}, U \setminus \{x\}) \end{array}$$

- Pour prouver que :

  $$\mathrm{Adm}^*(C,R) \supseteq \mathrm{Adm}^*(C+x,R),$$

  prenons $S \in \mathrm{Adm}^*(C+x,R)$. On a :
  - $S \in \mathrm{Adm}(I' \cup O' \cup U', R)$, et comme $I' \cup O' \cup U' = I \cup O \cup U$, $S \in \mathrm{Adm}(I \cup O \cup U, R)$.
  - $I \subseteq I' \subseteq S \subseteq I' \cup U'$ et $I' \cup U' = I \cup \{x\} \cup (U \setminus (\{x\} \cup R^{\pm}(x)))$; cet ensemble est inclus dans $I \cup U$ car $x \in U$.

  Donc, puisque $S \in \mathrm{Adm}(I \cup O \cup U, R)$ et $I \subseteq S \subseteq (I \cup U)$, on a $S \in \mathrm{Adm}^*(C,R)$.

- Pour prouver que

  $$\mathrm{Adm}^*(C,R) \supseteq \mathrm{Adm}^*(C-x,R),$$

  prenons $S \in \mathrm{Adm}^*(C-x,R)$. On a :
  - $S \in \mathrm{Adm}(I'' \cup O'' \cup U'', R)$, et comme $I'' \cup O'' \cup U'' = I \cup O \cup U$, $S \in \mathrm{Adm}(I \cup O \cup U, R)$.
  - $I = I'' \subseteq S$ et $S \subseteq U'' \cup I'' = (U \setminus \{x\}) \cup I \subseteq U \cup I$.

  Donc, puisque $S \in \mathrm{Adm}(I \cup O \cup U, R)$ et $I \subseteq S \subseteq (I \cup U)$, on a $S \in \mathrm{Adm}^*(C,R)$.

- Prouvons que

  $$\mathrm{Adm}^*(C,R) \subseteq \mathrm{Adm}^*(C+x,R) \cup \mathrm{Adm}^*(C-x,R).$$

  Prenons $S \in \mathrm{Adm}^*(C,R)$. Nous allons distinguer deux cas, suivant que x appartient à S ou pas.

- Supposons que $x \in S$. Alors, puisque $I \subseteq S \subseteq I \cup U$, on a $I' \subseteq S$. De plus, $R^+(S) \cap S = \emptyset$, ce qui implique que $R^\pm(x) \cap S = \emptyset$, et donc que $S \subseteq I' \cup U'$: ceci prouve que $S \in \mathrm{Adm}^*(C + x, R)$.
- Supposons que $x \notin S$. Alors on a $I'' = I \subseteq S$, et $I'' \cup U'' = I \cup U \setminus \{x\}$, ce qui implique que $S \subseteq I'' \cup U''$ car $x \notin S$; dans ce cas, $S \in \mathrm{Adm}^*(C - x, R)$.

□

À présent, pour qu'une feuille de l'arbre d'énumération des parties sans-conflit de (A, R) soit un ensemble admissible, il faut que tout élément de I soit défendu par I, en d'autres termes, il faut que tout prédécesseur de I soit également un successeur de I.

Par ailleurs, pour toute partition (I, O, U), tout élément de I doit être défendu par I ou doit être «défendable» : tout prédécesseur de I doit alors avoir des prédécesseurs qui n'ont pas déjà été classés hors des éléments qui peuvent faire partie de I. Si ce n'est pas le cas, il n'y a aucun espoir de trouver un ensemble admissible dans une feuille descendante du nœud courant.

Ces deux conditions sont exprimées par la propriété suivante :

Propriété 6.4. *Étant donné une relation binaire R et un R-candidat $C = (I, O, U)$:*
- *s'il existe $x \in R^-(I) \setminus R^+(I)$ tel que $R^-(x) \subseteq O$, alors $\mathrm{Adm}^*(C, R) = \emptyset$;*
- *sinon, si $R^-(I) \setminus R^+(I) = \emptyset$, alors I est admissible.*

PREUVE – Supposons tout d'abord que $x \in R^-(I) \setminus R^+(I)$ est tel que $R^-(x) \subseteq O$, et supposons que S est tel que $I \subseteq S \subseteq (I \cup U)$.

Alors $S \cap O = \emptyset$, et donc $R^-(x) \cap S = \emptyset$; en d'autres termes, $x \notin R^+(S)$, donc $x \in R^-(S) \setminus R^+(S)$, et par conséquent $R^-(S) \not\subseteq R^+(S)$. Donc $\mathrm{Adm}^*(C, R) = \emptyset$.

Supposons maintenant que $R^-(I) \setminus R^+(I) = \emptyset$, c'est-à-dire que $R^-(I) \subseteq R^+(I)$.

Alors I défend tous ses éléments. De plus, puisque C est un R-candidat, I est sans-conflit. Donc I est admissible. □

Étant donné un ensemble A et une relation binaire R, la fonction `EnumAdm` (algorithme 10 page suivante) permet d'énumérer tous les ensembles admissibles de (A, R). Elle fait pour cela appel à la fonction $\texttt{EnumAdm}_{\texttt{rec}}$ (algorithme 9).

Cette fonction suit le même principe que $\texttt{EnumPartiesSansconflit}_{\texttt{rec}}$ pour sélectionner un argument et générer deux nouveaux R-candidats. Cependant, lorsque $U = \emptyset$, elle vérifie si I défend bien tous ses éléments. De plus, si à un moment donné, l'algorithme trouve qu'aucun ensemble admissible ne peut être trouvé dans une feuille descendante du nœud courant, il arrête l'énumération.

À l'appel de $\texttt{EnumAdm}_{\texttt{rec}}$ par `EnumAdm`, tous les arguments de (A, R) qui s'auto-attaquent sont rangés dans O (puisqu'ils n'appartiennent à aucun ensemble sans-conflit).

Algorithme 9 $\texttt{EnumAdm}_{\texttt{rec}}(R, C)$

```
Entrée:
  – une relation binaire $R$,
  – un $R$-candidat $C = (I, O, U)$ ;
Sortie: $\{S \mid I \subseteq S \subseteq ((I \cup U) \setminus R^\pm(I))$ et $R^-(S) \subseteq R^+(I \cup U)\}$
Si $U = \emptyset$ et $R^-(I) \setminus R^+(I) = \emptyset$ Alors
  retourner $(\{I\})$ ;
Sinon Si $\exists x \in (R^-(I) \setminus R^+(I))$ tel que $R^-(x) \subseteq O$ Alors
  retourner $(\{\})$ ;
Sinon
  $x \leftarrow$ Select$(U)$ ;
  retourner $(\texttt{EnumAdm}_{\texttt{rec}}(R, C + x) \cup \texttt{EnumAdm}_{\texttt{rec}}(R, C - x))$ ;
Fin Si
```

```
Algorithme 10 EnumAdm($A, R$)
Entrée: un ensemble $A$ d'arguments et une relation binaire $R$ ;
Sortie: $\{S \mid S \subseteq A$ et $S \cap R^+(S) = \emptyset$ et $R^-(S) \subseteq R^+(S)\}$
retourner (EnumAdm_rec($R, (\emptyset, \text{Refl}(A, R), A \setminus \text{Refl}(A, R))$)) ;
```

Exemple 6.1 (suite). L'arbre d'énumération des parties sans-conflit du système d'argumentation *Zoé l'autruche* indiqué sur la figure 6.4 page 82 est identique à l'arbre d'énumération des ensembles admissibles. En effet, la condition $\exists x \in (R^-(I) \setminus R^+(I))$ tel que $R^-(x) \subseteq O$ n'est vérifiée qu'au niveau de certaines feuilles (celles qui ne correspondent pas à des ensembles admissibles).

La figure 6.5 indique les feuilles de l'arbre qui correspondent à des ensembles admissibles (feuilles encadrées). ♢

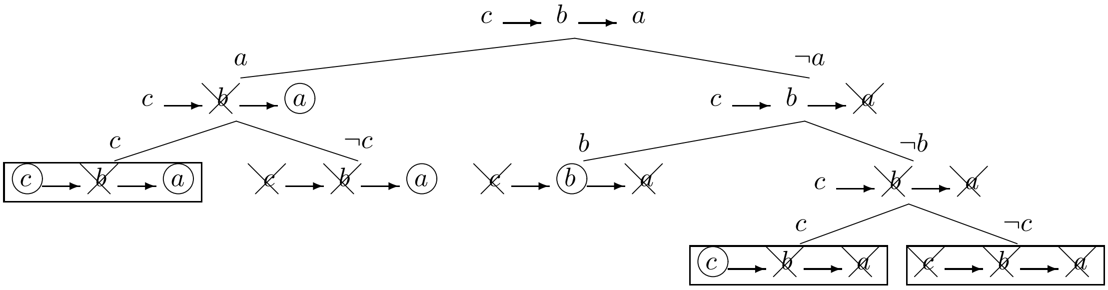


Fig. 6.5 – Énumération des ensembles admissibles du système d'argumentation *Zoé l'autruche*.

Exemple 6.2. L'arbre d'énumération des ensembles admissibles du système d'argumentation de la figure 4.4 page 66 est indiqué sur la figure 6.6 page ci-contre. Les feuilles qui correspondent à des ensembles admissibles sont encadrées.

Cet arbre donne un exemple d'utilisation de la propriété de coupure «si $\exists x \in (R^-(I) \setminus R^+(I))$ tel que $R^-(x) \subseteq O$ alors retourner($\{\}$)». ♢

Nous allons à présent donner un schéma général d'algorithme qui va nous permettre d'énumérer les extensions d'une sémantique donnée.

6.4 Algorithmes d'énumération d'extensions

Parmi les ensembles admissibles, on trouve les extensions préférées et les extensions stables. L'algorithme d'énumération des ensembles admissibles peut être généralisé pour énumérer ces deux types d'extensions. Nous allons proposer dans un premier temps un algorithme général qui va être «instancié» en fonction du problème à traiter.

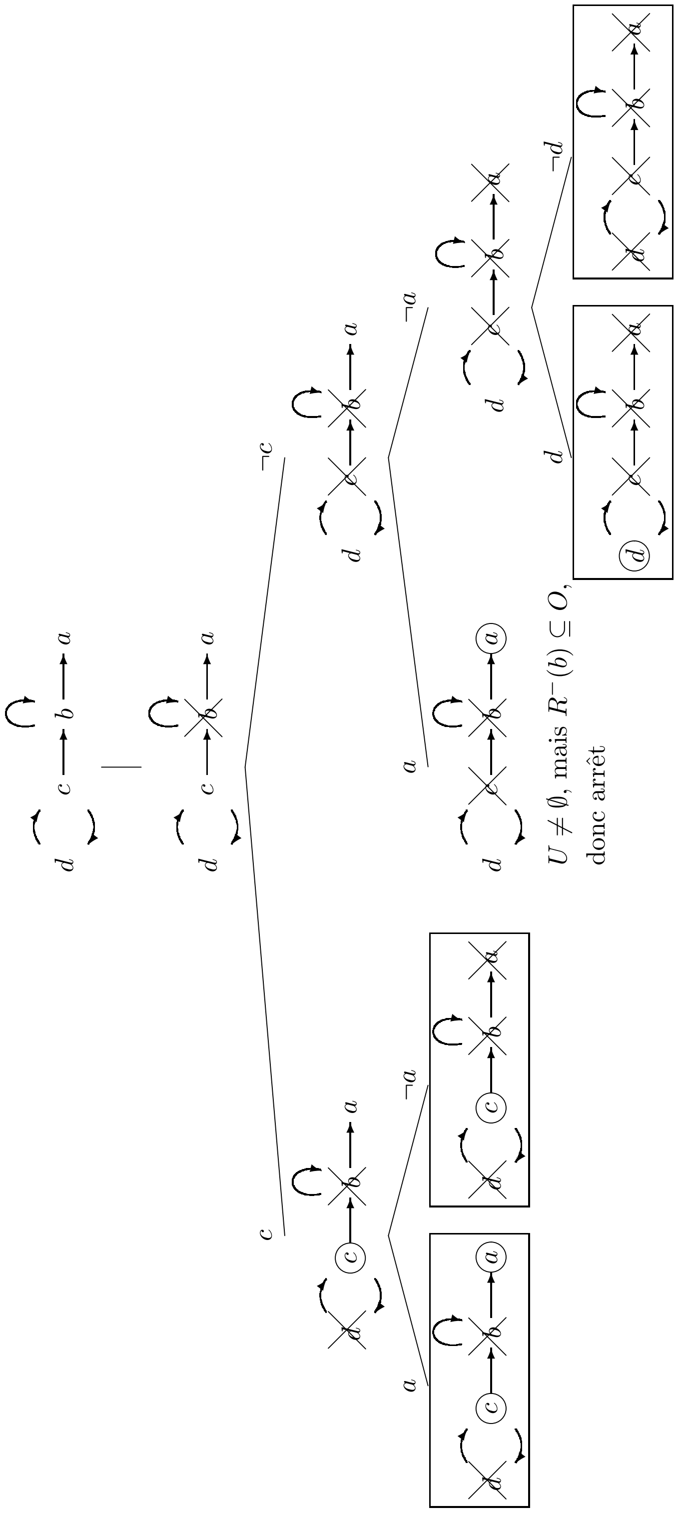
6.4.1 Algorithme général

Étant donné un ensemble A et une relation binaire R, la fonction `EnumExt` (algorithme 12 page 86) permet d'énumérer toutes les extensions d'une sémantique donnée de (A, R). Elle fait pour cela appel à la fonction `EnumExt_rec` (algorithme 11 page 86).

Étant donné une relation binaire R et un R-candidat $C = (I, O, U)$, la fonction récursive `EnumExt_rec` retourne l'ensemble des ensembles de la sémantique.

À chaque étape, la fonction teste si un ensemble de la sémantique peut être trouvé au nœud courant ou dans une feuille descendante du nœud courant :

StopCondVrai(C, R) est vrai si l'ensemble I du nœud courant est un ensemble de la sémantique ;



$U \neq \emptyset$, mais $R^{-}(b) \subseteq O$,
donc arrêt

FIG. 6.6 – Énumération des ensembles admissibles du système d'argumentation de la figure 4.4.

StopCondFaux(C, R) est vrai si aucun ensemble de la sémantique ne peut être trouvé à partir du nœud courant.

S'il n'y a pas de réponse immédiate, alors la fonction s'appelle généralement sur $(R, C + x)$ et sur $(R, C - x)$ pour un élément x sélectionné dans U par la fonction `Select`. Cette fonction `Select` est essentielle, car un bon choix de x peut permettre de n'explorer qu'une branche. Nous proposerons des heuristiques pour le choix d'un argument en fonction des sémantiques à calculer.

Pour cette raison, la fonction `Select`(C, R) retourne un triplet (x, b_I, b_O) où b_I et b_O sont deux booléens tels que :

- b_I est vrai si la branche dans laquelle x est ajouté à I doit être explorée, b_I est faux sinon ;
- b_O est vrai si la branche dans laquelle x est ajouté à O doit être explorée, b_O est faux sinon.

Nous verrons que lorsque b_I et b_O sont faux, il reste seulement à vérifier que $I \cup U$ est une extension de la sémantique.

À l'appel de $\texttt{EnumExt}_{\texttt{rec}}$ par `EnumExt`, tous les arguments de (A, R) qui s'auto-attaquent sont rangés dans O (puisqu'ils n'appartiennent à aucun ensemble sans-conflit, et *a fortiori* à aucune extension, qu'elle soit préférée ou stable).

Algorithme 11 $\texttt{EnumExt}_{\texttt{rec}}(R, C)$

Entrée:
- une relation binaire R,
- un R-candidat $C = (I, O, U)$;

Sortie: $\{S \mid I \subseteq S \subseteq (I \cup U)$ et S satisfait la sémantique$\}$;

```
Si StopCondVrai($C, R$) Alors
  retourner ($\{I\}$) ;
Sinon Si StopCondFaux($C, R$) Alors
  retourner ($\{\}$) ;
Sinon
  $(x, b_I, b_O) \leftarrow$ Select($C, R$) ;
  Si $b_I \wedge b_O$ Alors
    retourner ($\texttt{EnumExt}_{\texttt{rec}}(R, C + x) \cup \texttt{EnumExt}_{\texttt{rec}}(R, C - x)$) ;
  Sinon Si $b_I$ Alors
    retourner ($\texttt{EnumExt}_{\texttt{rec}}(R, C + x)$) ;
  Sinon Si $b_O$ Alors
    retourner ($\texttt{EnumExt}_{\texttt{rec}}(R, C - x)$) ;
  Sinon
    retourner ($\texttt{EnumExt}_{\texttt{rec}}(R, (I \cup U, O, \emptyset))$) ;
  Fin Si
Fin Si
```

Algorithme 12 `EnumExt`(A, R)

Entrée: un ensemble A d'arguments et une relation binaire R ;

Sortie: la sémantique de (A, R) ;

```
retourner ($\texttt{EnumExt}_{\texttt{rec}}(R, (\emptyset, \mathrm{Refl}(A, R), A \setminus \mathrm{Refl}(A, R)))$) ;
```

La fonction `EnumExt` va être utilisée pour calculer les extensions préférées, les extensions stables, ainsi que pour optimiser le calcul des ensembles admissibles.

Pour chacun de ces problèmes, nous donnons les conditions StopCondVrai et StopCondFaux ainsi que l'algorithme de la fonction `Select`. Pour cette dernière, nous ferons appel à deux fonctions :

`Choisir` qui, appliquée à un ensemble non vide (d'arguments ou d'ensembles d'arguments), retourne un élément quelconque de l'ensemble ;

`Cycles` qui, étant donné un ensemble X d'arguments, retourne l'ensemble des ensembles d'arguments qui composent des cycles dans X.

6.4.2 Énumération des extensions préférées

Pour donner les conditions StopCondVrai et StopCondFaux et la fonction de sélection `Select`, nous allons établir quelques propriétés. Nous introduisons tout d'abord la notation suivante :

Notation 6.3. *L'ensemble des extensions préférées qui sont dans les feuilles descendantes du nœud étiqueté* (I, O, U) *est dénoté par :*

$$\text{Pref}^*((I, O, U), R) = \{S \in \text{Pref}(I \cup O \cup U, R) \mid I \subseteq S \subseteq (I \cup U)\}.$$

Complétude de l'énumération des extensions préférées

Le résultat qui suit montre que l'énumération des extensions préférées peut être complète en explorant seulement les nœuds qui sont des R-candidats :

Propriété 6.5. *Étant donné une relation binaire* R*, un* R*-candidat* $C = (I, O, U)$ *et un élément* $x \in U$ *:*

- *si* $x \notin R^+(x)$*, alors*

$$\text{Pref}^*(C, R) = \text{Pref}^*(C + x, R) \cup \text{Pref}^*(C - x, R) \ ;$$

- *si* $x \in R^+(x)$*, alors*

$$\text{Pref}^*(C, R) = \text{Pref}^*(C - x, R).$$

Preuve – Rappelons que $S \in \text{Pref}(A, R)$ si et seulement si

- $S \cap R^+(S) = \emptyset$,
- $R^-(S) \subseteq R^+(S)$, et
- pour tout ensemble non vide $X \subseteq A \setminus S$, $X \cap R^+(S \cup X) \neq \emptyset$ ou $R^-(X) \not\subseteq R^+(S \cup X)$.

Cette dernière condition est exprimée par la condition $\text{Max}(S, A, R)$ suivante :

$$\text{Max}(S, A, R) \equiv \forall X \subseteq A \setminus S, \quad X = \emptyset \quad \vee \quad X \cap R^+(S \cup X) \neq \emptyset \quad \vee \quad R^-(X) \not\subseteq R^+(S \cup X).$$

Rappelons (notation 6.1 page 79) que :

$$\begin{array}{lllll} C + x & = & (I', O', U') & = & (I \cup \{x\}, O \cup R^{\pm}(x), U \setminus (\{x\} \cup R^{\pm}(x))) \\ C - x & = & (I'', O'', U'') & = & (I, O \cup \{x\}, U \setminus \{x\}) \end{array}$$

Cas où $x \notin R^+(x)$:

- Pour prouver que :

$$\text{Pref}^*(C, R) \supseteq \text{Pref}^*(C + x, R),$$

prenons $S \in \text{Pref}^*(C + x, R)$. On a :
  - $R^+(S) \cap S = \emptyset$,
  - $R^-(S) \subseteq R^+(S)$,
  - $\text{Max}(S, I \cup O \cup U, R) \equiv \text{Max}(S, I' \cup O' \cup U', R)$,

De plus, $I \subseteq I' \subseteq S$; $S \subseteq I' \cup U'$ et $I' \cup U' = I \cup \{x\} \cup (U \setminus (\{x\} \cup R^{\pm}(x)))$; cet ensemble est inclus dans $I \cup U$ car $x \in U$.

- Pour prouver que

$$\text{Pref}^*(C, R) \supseteq \text{Pref}^*(C - x, R),$$

prenons $S \in \text{Pref}^*(C - x, R)$. On a :
  - $R^+(S) \cap S = \emptyset$,
  - $R^-(S) \subseteq R^+(S)$, et
  - $\text{Max}(S, I \cup O \cup U, R) \equiv \text{Max}(S, I'' \cup O'' \cup U'', R)$.

De plus, $I = I'' \subseteq S$ et $S \subseteq U'' \cup I'' = (U \setminus \{x\}) \cup I \subseteq U \cup I$.

- Prouvons que
$$\mathrm{Pref}^*(C, R) \subseteq \mathrm{Pref}^*(C + x, R) \cup \mathrm{Pref}^*(C - x, R).$$
Prenons $S \in \mathrm{Pref}^*(C, R)$. Nous allons distinguer deux cas, suivant que x appartient à S ou pas.
  - Supposons que $x \in S$. Alors, puisque $I \subseteq S \subseteq I \cup U$, on a $I' \subseteq S$. De plus, $R^+(S) \cap S = \emptyset$, ce qui implique que $R^\pm(x) \cap S = \emptyset$, et donc que $S \subseteq I' \cup U'$: ceci prouve que $S \in \mathrm{Pref}^*(C + x, R)$.
  - Supposons que $x \notin S$. Alors on a $I'' = I \subseteq S$, et $I'' \cup U'' = I \cup U \setminus \{x\}$, ce qui implique que $S \subseteq I'' \cup U''$ car $x \notin S$; dans ce cas, $S \in \mathrm{Pref}^*(C - x, R)$.

Cas où $x \in R^+(x)$:

- La preuve que $\mathrm{Pref}^*(C, R) \supseteq \mathrm{Pref}^*(C - x, R)$ est toujours valide.
- Pour prouver que
$$\mathrm{Pref}^*(C, R) \subseteq \mathrm{Pref}^*(C - x, R),$$
prenons $S \in \mathrm{Pref}^*(C, R)$. On a :
  - $S \cap R^+(S) = \emptyset$,
  - $R^-(S) \subseteq R^+(S)$,
  - $\mathrm{Max}(S, I \cup O \cup U, R) \equiv \mathrm{Max}(S, I'' \cup O'' \cup U'', R)$.

  De plus, $I'' = I \subseteq S$. Puisque $S \cap R^+(S) = \emptyset$, $x \notin S$, donc on a $I'' = I \subseteq S$, et $I'' \cup U'' = I \cup U - \{x\}$, ce qui implique que $S \subseteq I'' \cup U''$ car $x \notin S$. Par conséquent, $S \in \mathrm{Pref}^*(C - x, R)$.

□

Conditions StopCondVrai **et** StopCondFaux

Lorsque U est vide, pour vérifier que la feuille obtenue est bien une extension préférée, considérons un R-candidat $(I, O, \emptyset)$. I est sans-conflit car $R^\pm(I) \subseteq O$. Il n'est par contre pas garanti que I défend bien tous ses éléments, ni que I est maximal dans le sens qu'aucun sous-ensemble X de O n'est tel que $I \cup X$ est admissible. Nous notons :

$$\mathrm{Max}^*(I, O, U, R) \equiv \forall X \subseteq O, X = \emptyset \vee X \cap R^+(X) \neq \emptyset \vee R^-(X) \not\subseteq R^+(I \cup U \cup X).$$

Propriété 6.6. *Étant donné une relation binaire R et un R-candidat $C = (I, O, U)$ tel que $U = \emptyset$: si $R^-(I) \subseteq R^+(I)$, et si $\mathrm{Max}^*(I, O \setminus R^\pm(I), \emptyset, R)$ est vrai, alors $\mathrm{Pref}^*(C, R) = \{I\}$, sinon $\mathrm{Pref}^*(C, R) = \emptyset$.*

Preuve – Par définition, si $S \in \mathrm{Pref}^*((I, O, \emptyset), R)$ alors $I \subseteq S \subseteq I$ puisque U est vide. Ainsi, le seul élément possible de $\mathrm{Pref}^*((I, O, U), R)$ est I.

On sait que $R^\pm(I) \subseteq O$ car C est un R-candidat. I et O sont des ensembles disjoints. Par conséquent, $R^+(I) \cap I = \emptyset$.

Donc $I \in \mathrm{Pref}^*((I, O, \emptyset), R)$ si et seulement si $R^-(I) \subseteq R^+(I)$ et $\mathrm{Max}(S, I \cup O, R)$ est vrai :

$$\begin{aligned}
&\mathrm{Max}(I, I \cup O, R)\\
&\quad\equiv \quad \forall X \subseteq (I \cup O) \setminus I, X = \emptyset \vee X \cap R^+(I \cup X) \neq \emptyset \vee R^-(X) \not\subseteq R^+(I \cup X)\\
&\quad\equiv \quad \forall X \subseteq O, X = \emptyset \vee X \cap (R^+(I) \cup R^+(X)) \neq \emptyset \vee R^-(X) \not\subseteq R^+(I \cup X))\\
&\quad\equiv \quad \forall X \subseteq O \setminus R^\pm(I), X = \emptyset \vee X \cap R^+(X) \neq \emptyset \vee R^-(X) \not\subseteq R^+(I \cup X))\\
&\quad\equiv \quad \mathrm{Max}^*(I, O \setminus R^\pm(I), \emptyset, R)
\end{aligned}$$

□

Les extensions préférées étant des ensembles admissibles, $\mathrm{Pref}^*(C, R) \subseteq \mathrm{Adm}^*(C, R)$. Par conséquent, la propriété 6.4 page 83 donne une condition sous laquelle aucune extension préférée ne peut être trouvée à partir du nœud courant.

Les conditions StopCondVrai et StopCondFaux sont donc les suivantes :

$$\begin{array}{rcl}\mathrm{StopCondVrai}(C,R) & \equiv & U=\emptyset \text{ et } R^-(I)\subseteq R^+(I) \text{ et } \mathrm{Max}^*(I,(O\setminus R^{\pm}(I),\emptyset,R)) \text{ vrai}\\ \mathrm{StopCondFaux}(C,R) & \equiv & \exists x\in (R^-(I)\setminus R^+(I)) \text{ tel que } R^-(x)\subseteq O\end{array}$$

Heuristiques pour la sélection d'un argument

Nous allons donner des propriétés qui permettent de choisir l'argument sélectionné à un nœud de façon à ce qu'un seul fils du nœud soit généré. Pour cela, nous allons mettre en évidence des conditions qui garantissent qu'un argument donné $x \in U$ est tel que soit toute extension préférée S telle que $I \subseteq S \subseteq (I \cup U)$ contient x, soit aucune de ces extensions ne contient x. Nous définissons les ensembles suivants :

$$\begin{array}{rcl}\mathrm{App}(I,U,R) & = & \{x\in U \mid x\notin (R^{\pm}(I)\cup R^+(U)) \text{ et } R^-(x)\subseteq R^+(I\cup\{x\})\}\\ \mathrm{Nondef}(I,U,R) & = & \{x\in U \mid R^-(x)\not\subseteq R^+(I\cup U)\}\\ \mathrm{Héros}(I,U,R) & = & \{x\in U \mid \exists z\in I, \exists y\in R^-(z), \{x\}=R^-(y)\cap(U\cup I)\}\\ \mathrm{Traitres}(I,O,U,R) & = & \{x\in U \mid \exists y\in R^+(x), \exists z\in O\cap R^+(y),\\ & & \quad z\notin R^+(I\cup U\cup\{z\}), R^-(z)\setminus y\subseteq R^+(I)\}\end{array}$$

$\mathrm{App}(I,U,R)$ est l'ensemble des éléments non décidés :

- qui ne peuvent être en conflit avec aucune extension S qui se trouve dans une feuille descendante du nœud courant (puisque $x \notin R^{\pm}(I) \cup R^+(U)$), et
- qui sont déjà défendus par I, ou qui se défendent eux-mêmes ($R^-(x) \subseteq R^+(I \cup \{x\})$).

Ces éléments seront dans toute extension préférée trouvée dans les feuilles descendantes du nœud courant.

$\mathrm{Nondef}(I,U,R)$ est l'ensemble des éléments non décidés qui ne peuvent plus être défendus car aucun de leurs prédécesseurs ne peut être attaqué ($R^-(x) \not\subseteq R^+(I \cup U)$). Ces éléments ne seront donc dans aucune extension qui se trouve dans une feuille descendante du nœud courant.

Les éléments de $\mathrm{App}(I,U,R)$ seront ajoutés à I, ceux de $\mathrm{Nondef}(I,U,R)$ seront ajoutés à O. Lorsqu'aucune de ces deux conditions n'est vérifiée, on peut avoir à créer deux branches, auquel cas on va devoir ajouter un élément z à I dans une branche, et à O dans l'autre, sans aucune certitude que l'élément z appartienne vraiment à l'ensemble auquel il est ajouté. En fait, la présence de z dans un ensemble ou dans l'autre sera justifiée lorsque le sort d'autres éléments sera décidé. C'est ce que permettent de décider les conditions Héros et Traitres :

$\mathrm{Héros}(I,U,R)$: s'il reste à un argument $z \in I$ un seul défenseur potentiel $x \in U$, alors x doit être ajouté à I ; $\mathrm{Héros}(I,U,R)$ est l'ensemble de ces ultimes défenseurs d'éléments de I.

$\mathrm{Traitres}(I,O,U,R)$: si un argument $z \in O$ n'a aucun attaquant dans I, alors il ne doit pas être défendable par I ; donc si z est déjà défendu contre tous ses attaquants sauf un, dont les attaquants ne sont pas tous dans O, alors ces derniers défenseurs potentiels de z doivent aller dans O. $\mathrm{Traitres}(I,O,U,R)$ est l'ensemble de ces éléments non décidés qui vont dans O pour supporter un autre élément de O.

Propriété 6.7. *Étant donné une relation binaire R, un R-candidat $C = (I,O,U)$ et un argument $x \in U$ tel que $x \notin R^+(x)$, alors :*

1. *si* $x \in \mathrm{App}(I,U,R) \cup \mathrm{Héros}(I,U,R)$*, alors* $\mathrm{Pref}^*(C,R) = \mathrm{Pref}^*(C+x,R)$ *;*
2. *si* $x \in \mathrm{Nondef}(I,U,R) \cup \mathrm{Traitres}(I,O,U,R)$*, alors* $\mathrm{Pref}^*(C,R) = \mathrm{Pref}^*(C-x,R)$.

PREUVE – CAS OÙ $x \in \mathrm{App}(I,U,R) \cup \mathrm{Héros}(I,U,R)$:
Prouvons que

$$\mathrm{Pref}^*(C,R) \subseteq \mathrm{Pref}^*(C+x,R).$$

Étant donné $(I',O',U') = C+x$ et $S \in \mathrm{Pref}^*(C,R)$:

- $R^+(S) \cap S = \emptyset$,
- $R^-(S) \subseteq R^+(S)$ et
- $\mathrm{Max}(I \cup O \cup U, R)$ est vrai.

Pour tout ensemble $X \subseteq ((I' \cup O' \cup U') \setminus S) = (I \cup O \cup U) \setminus S$, on a :

- $X = \emptyset$ ou
- $X \cap R^+(S \cup X) \neq \emptyset$ ou
- $R^-(X) \not\subseteq R^+(S \cup X)$.

Si $x \in \mathrm{App}(I, U, R)$ alors $x \notin R^{\pm}(I) \cup R^+(U)$ et $R^-(x) \subseteq R^+(I \cup \{x\})$.

Montrons que $S \cap R^{\pm}(x) = \emptyset$: puisque $R^-(x) \subseteq O$, et puisque $S \subseteq I \cup U$, $x \notin R^+(S)$ car $(I \cup U) \cap O = \emptyset$. On sait que $x \notin R^-(I \cup U)$, montrons que $x \notin R^-(U) \cap S$. Considérons $y \in R^+(x) \cap U$. Alors, puisque $R^-(x) \cap S = \emptyset$, $R^-(y) \not\subseteq R^+(S)$, et donc $y \notin S$. Ceci prouve que $x \notin R^-(S)$. Puisque $R^{\pm}(S) \subseteq R^{\pm}(I \cup U)$ (car $S \subseteq I \cup U$), et $x \notin R^{\pm}(I \cup U)$, $x \notin R^{\pm}(S)$, et par conséquent $S \cap R^{\pm}(x) = \emptyset$, donc :

$$S \subseteq (I \cup U) \setminus R^{\pm}(x) \subseteq I \cup (U \setminus R^{\pm}(x)) \subseteq I \cup \{x\} \cup (U \setminus R^{\pm}(x)) = I' \cup U'.$$

De plus, considérons $X = \{x\}$:

- puisque $x \notin R^+(I \cup U)$, $x \notin R^+(S \cup X)$, donc $X \cap R^+(S \cup X) = \emptyset$;
- $R^-(X) = R^-(x) \subseteq R^+(I \cup \{x\}) = R^+(I \cup X) \subseteq R^+(S \cup X)$ car $I \subseteq S$.

Ainsi, puisque S est une extension préférée, $X = \{x\} \not\subseteq (I \cup U \cup O) \setminus S$, donc $x \in S$, ce qui prouve que $I' \subseteq S$.

Considérons maintenant le cas où $x \in \mathrm{Héros}(R, I, U)$, c'est-à-dire qu'il existe $z \in I$ et $y \in R^-(z)$ tels que $R^-(y) \setminus O = \{x\}$: S est une extension préférée qui contient I, donc $R^-(z) \subseteq R^+(S)$, et ainsi $y \in R^+(S)$. De plus, puisque $S \cap O = \emptyset$, il faut que $x \in S$. Comme $I \subseteq S$, on conclut que $I' \subseteq S$. Il reste à prouver que $S \subseteq I' \cup U' = (I \cup U) \setminus R^{\pm}(x)$. On sait que $S \subseteq I \cup U$, donc il faut prouver que $S \cap R^{\pm}(x) = \emptyset$; ceci est vrai grâce aux faits que $x \in S$ et $S \cap R^+(S) = \emptyset$.

Cas où $x \in \mathrm{Nondef}(I, U, R) \cup \mathrm{Traitres}(I, O, U, R)$:

Prouvons que

$$\mathrm{Pref}^*(C, R) \subseteq \mathrm{Pref}^*(C - x, R).$$

Étant donné $(I'', O'', U'') = C - x$ et $S \in \mathrm{Pref}^*(C, R)$, alors :

- $R^+(S) \cap S = \emptyset$,
- $R^-(S) \subseteq R^+(S)$ et
- pour tout $X \subseteq (I' \cup O' \cup U') \setminus S = (I \cup O \cup U) \setminus S$, $X \cap R^+(S \cup X) \neq \emptyset$.

Si $x \in \mathrm{Nondef}(R, U, I)$, c'est-à-dire $R^-(x) \not\subseteq R^+(I \cup U)$ alors, puisque $S \subseteq I \cup U$, $R^-(x) \not\subseteq R^+(S)$, et donc $x \notin S$ car $R^-(S) \subseteq R^+(S)$. Par conséquent, $S \subseteq (U \cup I) \setminus \{x\} = (U \setminus \{x\}) \cup I = U' \cup I'$.

Supposons que $x \in \mathrm{Traitres}(I, O, U, R)$. Étant donné $y \in R^+(x)$, $z \in R^+(y) \cap O$ tel que $R^-(z) \subseteq O \setminus \{z\}$ et $R^-(z) \setminus y \subseteq R^+(I)$, considérons $S \in \mathrm{Pref}(R, I, O, U)$. Alors $\{z\} \subseteq A \setminus S$, et $z \notin R^+(S \cup \{z\})$ car $R^-(z) \subseteq O \setminus \{z\}$ et $O \cap S = \emptyset$, donc il faut que $R^-(z) \not\subseteq R^+(S \cup \{z\})$. Cependant, $R^-(z) \setminus y \subseteq R^+(I) \subseteq R^+(S)$, donc il faut que $y \not\subseteq R^+(S)$, c'est-à-dire $R^-(y) \cap S = \emptyset$. Ainsi on doit avoir $x \notin S$. □

Cette proposition donne des conditions sous lesquelles seul un fils d'un nœud donné doit être généré. Ces conditions reposent sur la possibilité de sélectionner un élément non décidé qui satisfait une propriété particulière. Si aucun élément non décidé ne satisfait une de ces propriétés particulières, alors il faut générer les deux fils du nœud.

La question qui se pose est de savoir jusqu'à quand ceci peut être évité ; en effet, on peut s'attendre à ce que l'énumération à partir d'un nœud qui a deux fils soit en moyenne deux fois plus dure qu'à partir d'un nœud qui a un seul fils.

Rappelons-nous qu'un système d'argumentation bien-fondé a une seule extension préférée. Lorsque le système d'argumentation est fini, cela signifie qu'il est sans-cycle. Comme nous nous plaçons dans le cas d'un système d'argumentation fini, il est naturel de vérifier si l'ensemble des éléments non décidés à un nœud donné est sans cycle. Dans ce cas, il n'y aura qu'une seule extension préférée trouvée dans une feuille descendante du nœud. Cette idée a été utilisée dans [DMP97] (voir section 6.6.1) mais n'a pas été formellement prouvée.

Propriété 6.8. *Étant donné une relation binaire R finie et un R-candidat $C = (I, O, U)$ tel que* $\mathrm{Nondef}(I, U, R) = \emptyset$ *et U ne contient aucun cycle, alors*

$$\mathrm{Pref}^*(C, R) = \mathrm{Pref}^*((I \cup U, O, \emptyset), R).$$

Lemme 6.1. *Étant donné :*
- *une relation binaire R finie et*
- *un R-candidat $C = (I, O, U)$ tel que* $\mathrm{Nondef}(I, U, R) = \emptyset$ *et U ne contient aucun cycle,*

alors $R^+(U) \cap U = \emptyset$.

Preuve – Supposons qu'un argument non décidé $x_0 \in U$ ait un prédécesseur dans U, c'est-à-dire qu'il existe $x_1 \in R^-(x_0) \cap U$. Alors, puisque $\mathrm{Nondef}(I, U, R) = \emptyset$, $x_1 \in R^+(I \cup U)$, et puisque $R^+(I) \cap U = \emptyset$, x_1 a un prédécesseur dans U.

On peut itérer le processus et construire une séquence $(x_i)_{i \geq 0}$ d'éléments de U telle que $x_{i+1} R x_i$ pour tout $i \geq 0$. Puisque R est finie, U doit contenir un cycle. Puisque U ne contient aucun cycle, aucun élément de U n'a de prédécesseur dans U, c'est-à-dire $R^+(U) \cap U = \emptyset$. □

Lemme 6.2. *Étant donné :*
- *une relation binaire R finie,*
- *un R-candidat $C = (I, O, U)$ tel que* $\mathrm{Nondef}(I, U, R) = \emptyset$ *et U ne contient aucun cycle, et*
- *une extension préférée* $S \in \mathrm{Pref}^*((I, O, U), R)$*,*

alors $U \subseteq S$.

Preuve – Prouvons que $S \cup U$ est admissible.

Puisque $S \subseteq I \cup U$, et puisque $O \cap (I \cup U) = \emptyset$, $S \cup U$ et O sont des ensembles disjoints. De plus, on sait que $R^+(S) \cap S = R^+(U) \cap U = \emptyset$, et que $R^\pm(I) \cap U = \emptyset$, donc $R^+(S \cup U) \cap (S \cup U) = \emptyset$.

Prouvons à présent que $R^-(S \cup U) \subseteq R^+(S \cup U)$. On sait que $R^-(S) \subseteq R^+(S)$ car S est une extension préférée. De plus, puisque $\mathrm{Nondef}(I, U, R) = \emptyset$, $R^-(U) \subseteq R^+(I \cup U) \subseteq R^+(S \cup U)$ car $I \subseteq S$.

Ceci prouve que $S \cup U$ est admissible et est donc contenu dans une extension préférée S' de $(I \cup O \cup U, R)$ qui contient également S. Mais puisque S est lui-même une extension préférée, il faut que $S' = S$ (car les extensions préférées sont des ensembles maximaux pour l'inclusion) et donc $U \subseteq S$. □

Preuve (propriété 6.8) –
- Prouvons tout d'abord que

$$\mathrm{Pref}^*(C, R) \subseteq \mathrm{Pref}^*((I \cup U, O, \emptyset), R).$$

Considérons $S \in \mathrm{Pref}^*(C, R)$. Alors $S \in \mathrm{Pref}(A, R)$, où $A = I \cup U \cup O$, et $I \subseteq S \subseteq I \cup U$. Mais on sait d'après le lemme 6.2 que $U \subseteq S$. Par conséquent $I \cup U \subseteq S \subseteq (I \cup U) \cup \emptyset$: $S \in \mathrm{Pref}^*((I \cup U, O, \emptyset), R)$.
- Prouvons maintenant que

$$\mathrm{Pref}^*(C, R) \supseteq \mathrm{Pref}^*((I \cup U, O, \emptyset), R).$$

Considérons $S \in \mathrm{Pref}^*((I \cup U, O, \emptyset), R)$. Alors $S \in \mathrm{Pref}(A, R)$ et $I \cup U \subseteq S \subseteq (I \cup U) \cup \emptyset$. Ceci implique que $I \subseteq S \subseteq I \cup U$. Ainsi, $S \in \mathrm{Pref}^*((I, O, U), R)$.

□

Notons également qu'il n'est pas difficile de voir que les composantes maximales connexes du système d'argumentation peuvent être examinées séparément (voir [DM01b]) : pour optimiser l'énumération des extensions préférées, l'appel de `EnumExt` pourrait être découpé en autant d'appels qu'il y a de composantes maximales connexes dans le système d'argumentation.

Fonction de sélection `Select`

La stratégie de la fonction de sélection d'un argument va être de choisir en premier les arguments de l'ensemble Nondef(I, U, R).

Lorsque cet ensemble est vide, si U est sans cycle, alors, d'après la propriété 6.8 page précédente, b_I et b_O sont retournés faux (un argument quelconque de U est sélectionné, mais il ne sera pas utilisé par la suite). Dans l'algorithme général, les arguments de U vont être tous ajoutés à I et il restera à tester si l'ensemble I obtenu est bien une extension préférée.

Si U contient au moins un cycle, `Select` va tenter d'utiliser la propriété qui permet de faire passer un argument dans I de manière justifiée : elle essaie de choisir un argument dans App(I, U, R).

Si cet ensemble est vide, `Select` tente d'utiliser les propriétés qui permettent de justifier la présence d'arguments mis arbitrairement dans I ou O, autrement dit, `Select` va chercher à sélectionner un argument dans Héros(I, U, R) ou Traitres(I, O, U, R).

Si ces deux ensembles sont vides, alors `Select` retourne un argument non décidé qui appartient à un cycle.

L'algorithme 13 est celui de la fonction `Select`.

Algorithme 13 `Select`(C, R)

Entrée:
- une relation binaire R, et
- un R-candidat $C = (I, O, U)$;

Sortie: un triplet (x, b_I, b_O) où x est un argument, b_I et b_O sont deux booléens.

```
Si Nondef$(I, U, R) \neq \emptyset$ Alors
  retourner $((\texttt{Choisir}(\text{Nondef}(I, U, R)), \bot, \top))$ ;
Sinon Si Cycles$(U) = \emptyset$ Alors
  retourner $((\texttt{Choisir}(U), \bot, \bot))$ ;
Sinon Si App$(I, U, R) \neq \emptyset$ Alors
  retourner $((\texttt{Choisir}(\text{App}(I, U, R)), \top, \bot))$ ;
Sinon Si Héros$(I, U, R) \neq \emptyset$ Alors
  retourner $((\texttt{Choisir}(\text{Héros}(I, U, R)), \top, \bot))$ ;
Sinon Si Traitres$(I, U, R) \neq \emptyset$ Alors
  retourner $((\texttt{Choisir}(\text{Traitres}(I, U, R)), \bot, \top))$ ;
Sinon
  retourner $((\texttt{Choisir}(\texttt{Choisir}(\texttt{Cycles}(U))), \top, \top))$ ;
Fin Si
```

Exemple 6.1 (suite). Le système d'argumentation *Zoé l'autruche* possède une seule extension préférée; l'arbre d'énumération (réduit à une branche) qui permet de la calculer est indiqué sur la figure 6.7. ◇

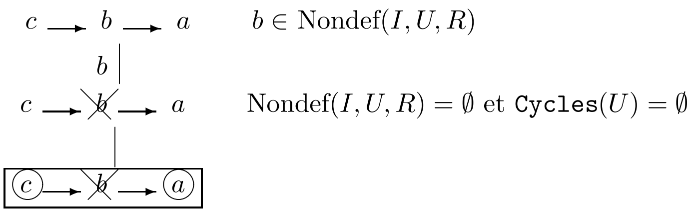


Fig. 6.7 – Arbre d'énumération de la seule extension préférée du système d'argumentation *Zoé l'autruche.*

EXEMPLE 2.1 (SUITE). Le système d'argumentation *Nixon* (où $A = \{a, b\}$ et $R = \{(a, b), (b, a)\}$) possède deux extensions préférées. L'arbre d'énumération qui permet de les trouver est indiqué sur la figure 6.8. ◇

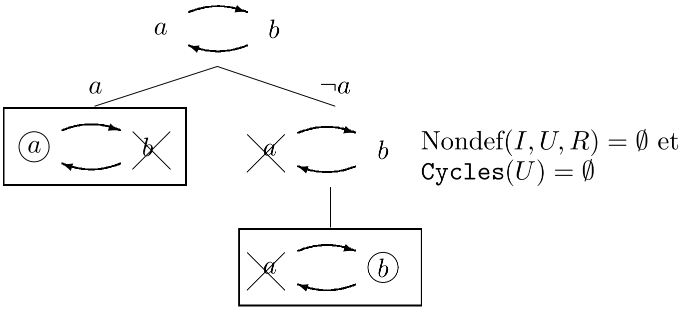


FIG. 6.8 – Arbre d'énumération des extensions préférées du système d'argumentation *Nixon*.

EXEMPLE 6.2 (SUITE). Les extensions préférées du système d'argumentation de la figure 4.4 sont indiquées sur la figure 6.9. ◇

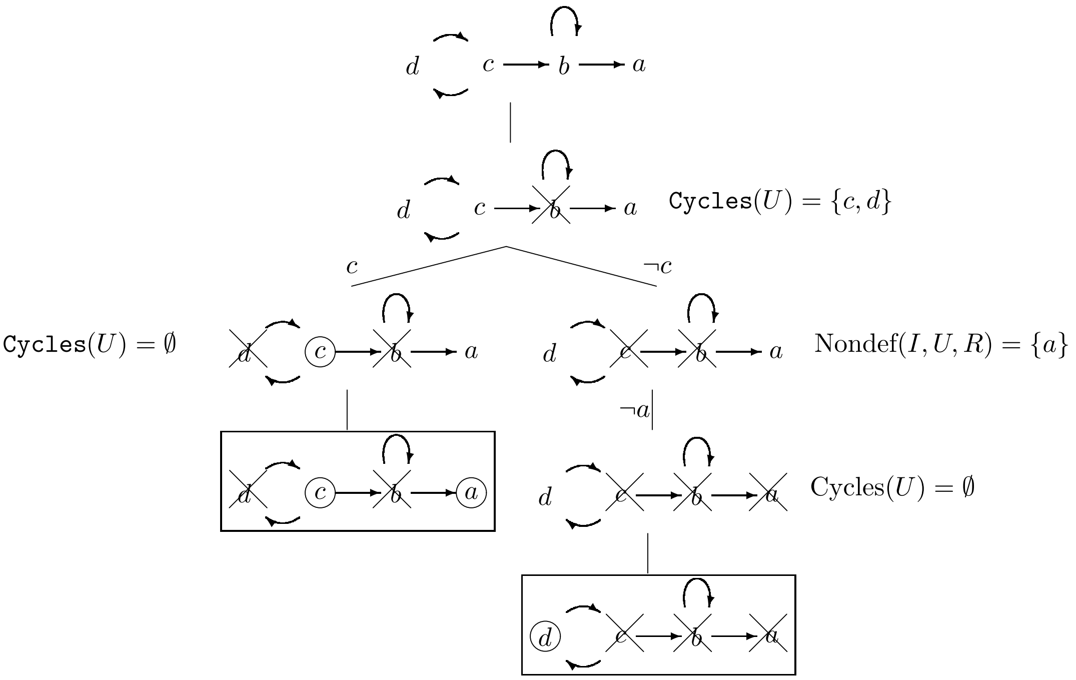


FIG. 6.9 – Énumération des extensions préférées du système d'argumentation de la figure 4.4.

6.4.3 Énumération des extensions stables

Nous allons voir dans cette section deux façons de calculer les extensions stables.

La première utilise la fonction de sélection `Select` du calcul des extensions préférées et des conditions StopCondVrai et StopCondFaux spécifiques au calcul des extensions stables.

La deuxième est celle proposée par [DMP97, Nie95]. Nous verrons que les heuristiques pour la sélection d'un argument qui sont utilisées par [DMP97] s'appuient sur des propriétés plus fortes des extensions stables que celles utilisées dans la première méthode.

Première méthode : calcul à l'aide de la fonction de sélection pour le calcul des extensions préférées

Puisque les extensions stables sont des extensions préférées, l'algorithme de calcul des extensions préférées est complet pour le calcul des extensions stables, mais il n'est pas toujours correct. Pour qu'il soit correct, il faut modifier la condition StopCondVrai.

Lorsque U est vide, pour garantir que I est une extension stable, il suffit de vérifier si tout argument hors de I (*i.e.* dans O) est attaqué par un argument de I ; en d'autres termes, il faut que $O \subseteq R^+(I)$ ou de manière équivalente, que $O \setminus R^+(I) = \emptyset$.

Par ailleurs, la condition StopCondFaux utilisée pour l'énumération des extensions préférées peut être améliorée : si à un moment donné de l'énumération, un argument de O a tous ses prédécesseurs dans O, alors il est sûr que le nœud courant ne peut mener à une extension stable.

Les conditions StopCondVrai et StopCondFaux pour l'énumération des extensions stables sont donc les suivantes :

$$\begin{array}{lcl} \text{StopCondVrai}(C,R) & \equiv & U=\emptyset \text{ et } O\setminus R^+(I)=\emptyset \\ \text{StopCondFaux}(C,R) & \equiv & \exists x\in O_p \text{ tel que } R^-(x)\subseteq O,\ \text{ ou } \exists x\in (O\setminus R^+(I)) \text{ tel que } R^-(x)\subseteq O \end{array}$$

Les heuristiques utilisées pour la sélection d'un argument dans le cas de l'énumération des extensions préférées peuvent toutes être utilisées pour l'énumération des extensions stables, puisque les extensions stables sont des extensions préférées. En considérant la fonction `Select` dédiée au calcul des extensions préférées et les conditions StopCondVrai et StopCondFaux ci-dessus, le calcul de l'extension stable du système d'argumentation de la figure 4.4 page 66 est indiqué sur la figure 6.10.

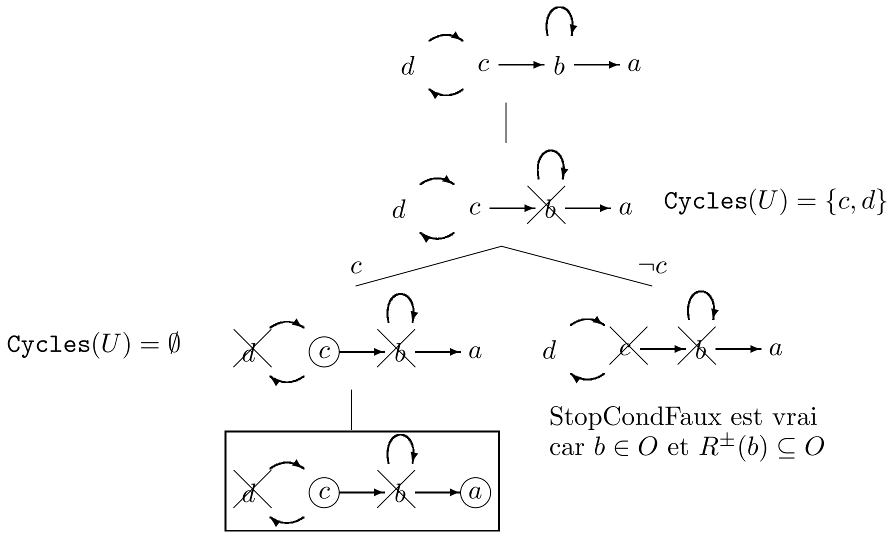


FIG. 6.10 – Énumération des extensions stables du système d'argumentation de la figure 4.4 en utilisant la fonction `Select` définie pour l'énumération des extensions préférées.

Deuxième méthode : calcul selon [DMP97, Nie95]

Les algorithmes de [DMP97, Nie95] permettent de calculer les extensions stables d'une théorie des défauts, *via* une traduction de la théorie des défauts en un graphe orienté ; les noyaux du graphe orienté correspondent aux extensions stables de la théorie. Les extensions stables d'un système d'argumentation étant équivalentes aux noyaux d'un graphe orienté, les extensions stables de la théorie des défauts correspondent aux extensions stables du système d'argumentation de Dung (voir section 3.3).

Les algorithmes d'énumération de [DMP97, Nie95] explorent un équivalent de nos R-candidats, mais ils utilisent des propriétés plus fortes des extensions stables, qui ne peuvent être utilisées pour le calcul des extensions préférées, en particulier :

1. un argument non décidé x peut être ajouté à I dès que tous ses prédécesseurs sont dans O ; cette propriété est propre aux extensions stables, puisque dans le cas des extensions préférées, il faudrait également s'assurer que les prédécesseurs de x ont des prédécesseurs dans I ;
2. si un argument non décidé x est le seul prédécesseur dans $U \cup I$ d'un argument y de O, alors x doit être ajouté à I pour «justifier» la présence de y dans O. Cette propriété est spécifique au calcul des extensions stables, car pour les extensions préférées, un argument rejeté d'une extension n'est pas forcément attaqué par un argument de l'extension.

Ces deux conditions sont exprimées dans la propriété suivante :

Propriété 6.9. *Étant donné* (A, R) *un système d'argumentation,* (I, O, U) *une partition de* A*, un argument* $x \in U$ *et une extension stable* S *de* (A, R) *telle que* $I \subseteq S \subseteq ((I \cup U) \setminus \{x\})$*, alors :*

1. *si* $x \notin R^+(I \cup U)$ *alors* $x \in (A \setminus S) \nsubseteq R^+(S)$ *;*
2. *si* x *est tel qu'il existe* $y \in O$ *tel que* $R^-(y) \cap (I \cup U) = \{x\}$*, alors* $y \in (A \setminus S) \nsubseteq R^+(S)$*.*

Les arguments non décidés qui vérifient la condition 1 de la propriété 6.9, tout comme ceux de l'ensemble $\mathrm{App}(I, U, R)$, peuvent être ajoutés à I de manière «justifiée».

Les arguments qui vérifient la condition 2 de la propriété 6.9 peuvent être comparés aux éléments de $\text{Héros}(I, O, U, R)$. En effet, les éléments de ce dernier ensemble doivent être ajoutés à I car ils sont les seuls à pouvoir justifier la présence d'un argument dans I. Les arguments qui vérifient la condition 2 doivent être ajoutés à I car ils sont les seuls à pouvoir justifier la présence d'un argument dans O.

Notons que tous les arguments qui appartiennent à $\text{Héros}(I, U, R)$ vérifient la condition 2 ; la réciproque n'est pas vraie car tous les arguments qui vérifient la condition 2 ne défendent pas obligatoirement un argument de I.

L'arbre d'énumération qui permet de trouver l'extension stable du système d'argumentation de la figure 4.4 page 66 grâce à l'algorithme de [DMP97] est indiqué sur la figure 6.11 page suivante.

Cet arbre d'énumération contient plus de nœuds que l'arbre d'énumération de la figure 6.10 page ci-contre, obtenu par la première méthode de calcul des extensions stables (utilisant la fonction de sélection pour le calcul des extensions préférées).

La raison à cela est que l'algorithme de [DMP97] ne distingue pas les arguments qui s'auto-attaquent ; ce sont simplement des arguments qui appartiennent à un cycle de longueur 1. Il est donc possible qu'un argument qui s'auto-attaque soit ajouté à I (c'est le cas de l'argument b dans la figure 6.11) : ceci est en contradiction avec la nécessité qu'une extension stable soit sans-conflit.

Par ailleurs, nous verrons en section 6.6.1 que le calcul d'un ensemble coupe-cycle n'est pas toujours nécessaire. Nous pouvons ajouter qu'un tel ensemble ne devrait pas non plus contenir d'argument qui s'auto-attaque. Seuls les cas où les arguments qui s'auto-attaquent sont ajoutés à O devraient être étudiés.

6.5 Algorithmes de réponse aux problèmes d'acceptabilité

6.5.1 Algorithme général

Sous une sémantique donnée, le problème d'acceptabilité crédule consiste à déterminer si un argument a appartient à au moins une extension de la sémantique. Le problème d'acceptabilité sceptique consiste quant à lui à déterminer si a appartient à toutes les extensions de la sémantique.

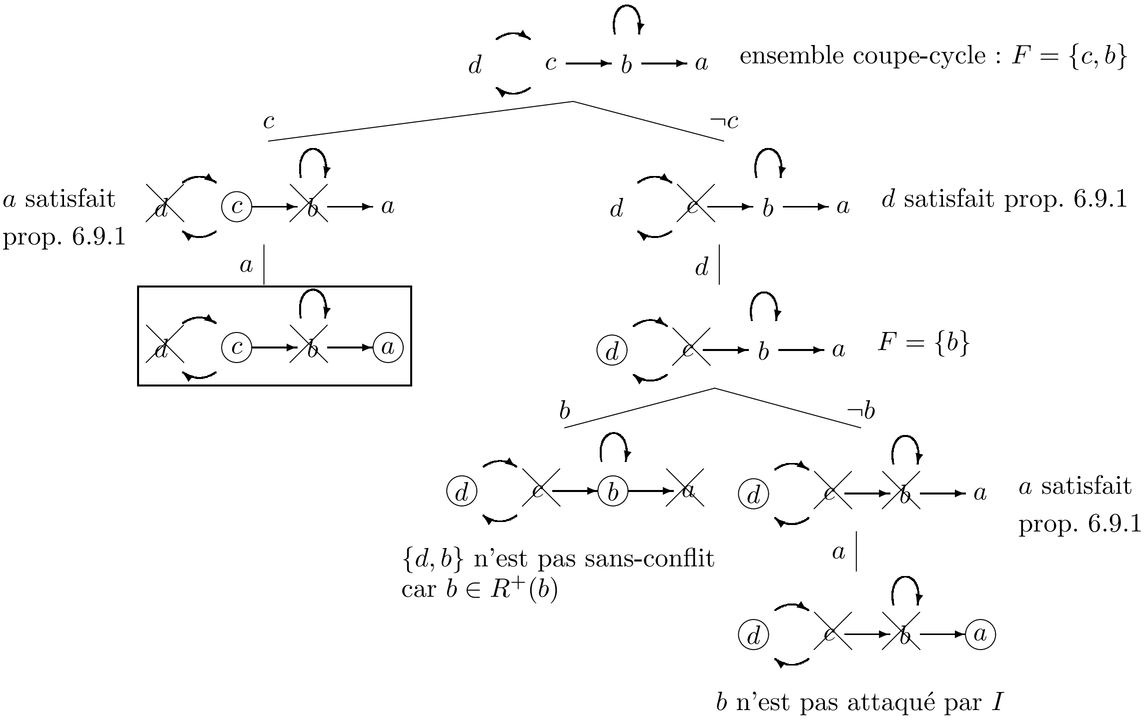


FIG. 6.11 – Énumération des extensions stables en utilisant l'algorithme de [DMP97].

Pour répondre à ce dernier problème, nous allons nous intéresser au problème complémentaire, qui consiste à déterminer s'il existe au moins une extension de la sémantique qui ne contient pas a. Si la réponse au problème complémentaire du problème d'acceptabilité sceptique est *vrai* alors la réponse au problème d'acceptabilité sceptique est *faux*, et vice-versa.

Les algorithmes pour répondre aux problèmes d'acceptabilité crédule et sceptique n'ont donc pas à calculer toutes les extensions, mais seulement à trouver *une* extension qui contient (dans le cas du problème d'acceptabilité crédule) ou ne contient pas (dans le cas du problème d'acceptabilité sceptique) a.

Pour trouver une telle extension, ces deux types d'algorithmes vont faire appel à la fonction $\texttt{RepPb}_{\texttt{rec}}$ (algorithme 14 page suivante).

Étant donné une relation binaire R, un R-candidat $C = (I, O, U)$ et un argument a, $\texttt{RepPb}_{\texttt{rec}}(R, C, a)$ détermine s'il existe une extension de la sémantique qui contient I et qui est incluse dans $I \cup U$.

La fonction $\texttt{RepPb}_{\texttt{rec}}$ commence par tester si une telle extension peut être trouvée au nœud courant ou dans une feuille descendante du nœud courant. Pour cela, elle teste les conditions StopCondVrai et StopCondFaux :

StopCondVrai(C, R) garantit l'existence d'une extension dans une feuille descendante du nœud courant,

StopCondFaux garantit l'absence d'extension dans les feuilles descendantes du nœud courant.

S'il n'y a pas de réponse immédiate, alors un élément x est sélectionné dans U grâce à la fonction `Select`. Cette fonction prend en paramètres la relation R et le R-candidat $C = (I, O, U)$, ainsi que l'argument a (cet argument est nécessaire pour le problème d'acceptabilité sceptique). Comme dans l'algorithme pour l'énumération des extensions (section 6.4.1), `Select` retourne un triplet contenant, non seulement l'argument x sélectionné, mais également deux booléens b_I et b_O tels que : si b_I est vrai, la branche dans laquelle x est ajouté à I doit être explorée ; si b_O est vrai, la branche dans laquelle x est ajouté à O doit être explorée. Si b_I et b_O sont faux, il restera seulement à vérifier que l'ensemble $I \cup U$ est une extension de la sémantique.

Algorithme 14 $\texttt{RepPb}_{\texttt{rec}}(R, C, a)$

Entrée:
- une relation binaire R,
- un R-candidat $C = (I, O, U)$,
- un argument a ;

Sortie:
- $\top$ s'il existe S tel que $I \subseteq S \subseteq (I \cup U)$ et S est une extension qui répond au problème,
- $\bot$ sinon ;

```
1:  Si StopCondVrai($C, R$) Alors
2:    retourner ($\top$) ;
3:  Sinon Si StopCondFaux($C, R$) Alors
4:    retourner ($\bot$) ;
5:  Sinon
6:    $(x, b_I, b_O) \leftarrow \texttt{Select}(R, C, a)$ ;
7:    Si $\neg b_I \wedge \neg b_O$ Alors
8:      retourner ($\texttt{RepPb}_{\texttt{rec}}(R, (I \cup U, O, \emptyset), a)$) ;
9:    Sinon
10:     retourner ($(b_I \wedge \texttt{RepPb}_{\texttt{rec}}(R, C + x, a)) \vee (b_O \wedge \texttt{RepPb}_{\texttt{rec}}(R, C - x, a))$) ;
11:   Fin Si
12: Fin Si
```

L'instruction 10 de l'algorithme 14 ne spécifie pas dans quel ordre doivent être explorées les branches lorsque les deux branches sont à explorer. Pourtant, le choix d'un ordre peut avoir une grande influence sur l'efficacité de l'algorithme, puisque si une réponse positive est trouvée dans la première branche explorée, la deuxième n'a pas à être explorée.

Sous une sémantique donnée, la fonction `RepPbCred` (algorithme 15) permet de répondre au problème d'acceptabilité crédule ; la fonction `RepPbScept` (algorithme 16 page suivante) permet de répondre au problème d'acceptabilité sceptique. Ces deux fonctions prennent en paramètre un ensemble A d'arguments, une relation binaire R et l'argument a de la requête. Elles renvoient *vrai* si a est crédulement (resp. sceptiquement) accepté, *faux* sinon.

Algorithme 15 $\texttt{RepPbCred}(A, R, a)$

Entrée:
- un ensemble A d'arguments,
- une relation binaire R et
- un argument $a \in A$;

Sortie:
- $\top$ si l'argument est crédulement accepté sous la sémantique,
- $\bot$ sinon ;

```
Si $a \in R^+(a)$ Alors
  retourner ($\bot$) ;
Sinon
  retourner ($\texttt{RepPb}_{\texttt{rec}}(R, (\emptyset, \emptyset, A) + a, a)$) ;
Fin Si
```

Un argument qui s'auto-attaque n'appartient à aucune des extensions de Dung, il n'est donc ni crédulement, ni sceptiquement accepté. Pour cette raison, `RepPbCred` et `RepPbScept` commencent par tester si l'argument de la requête s'auto-attaque. Si tel est le cas, elles renvoient *faux*, sinon elles appellent la fonction $\texttt{RepPb}_{\texttt{rec}}$.

Pour répondre au problème d'acceptabilité crédule, `RepPbCred` va chercher s'il existe une extension qui contient a. $\texttt{RepPb}_{\texttt{rec}}$ est donc appelée avec en paramètres R, a et le R-candidat $(\emptyset, \emptyset, A) + a$

Algorithme 16 `RepPbScept`(A, R, a)

```
Entrée:
  – un ensemble $A$ d'arguments,
  – une relation binaire $R$ et
  – un argument $a \in A$ ;
Sortie:
  – $\top$ si l'argument est sceptiquement accepté sous la sémantique,
  – $\bot$ sinon ;
Si $a \in R^+(a)$ Alors
    retourner $(\bot)$ ;
Sinon
    retourner $(\neg\ \mathtt{RepPb_{rec}}(R, (\emptyset, \emptyset, A) - a, a))$ ;
Fin Si
```

(a appartient à I, O contient les successeurs et prédécesseurs de a et U contient les autres arguments de A).

Pour répondre au problème d'acceptabilité sceptique, `RepPbScept` retourne la négation de la réponse au problème complémentaire. Le problème complémentaire consiste à chercher s'il existe une extension qui ne contient pas a. $\mathtt{RepPb_{rec}}$ est donc appelée avec en paramètres R, a et le R-candidat $(\emptyset, \emptyset, A) - a$ (a appartient à O, I est vide et U contient tous les arguments de A sauf a).

Les fonctions `RepPbCred` et `RepPbScept` vont être utilisées dans les sections 6.5.2 et 6.5.3 pour répondre aux problèmes d'acceptabilité crédule et sceptique sous la sémantique préférée.

Dans des travaux futurs, nous envisageons d'utiliser les fonctions `RepPbCred` et `RepPbScept` pour répondre aux problèmes d'acceptabilité crédule et sceptique sous les autres sémantiques de Dung.

6.5.2 Problème d'acceptabilité crédule sous la sémantique préférée

Dans cette section, nous allons montrer comment la fonction `RepPbCred` peut être utilisée pour répondre au problème d'acceptabilité crédule sous la sémantique préférée. Pour cela, nous allons tout d'abord préciser les conditions StopCondVrai et StopCondFaux, puis nous donnerons l'algorithme de la fonction `Select`.

Conditions StopCondVrai **et** StopCondFaux

Comme nous l'avons déjà montré, pour déterminer si un argument a appartient à au moins une extension préférée d'un système d'argumentation, il suffit de trouver un ensemble admissible contenant a (tout ensemble admissible étant contenu dans au moins une extension préférée).

Étant donné un R-candidat (I, O, U), d'après la propriété 6.4 page 83, si $R^-(I) \setminus R^+(I) = \emptyset$ alors I est admissible ; en revanche, si $\exists x \in (R^-(I) \setminus R^+(I))$ tel que $R^-(x) \subseteq O$, alors I n'est pas admissible et ne pourra jamais l'être car tous les défenseurs de l'un de ses éléments sont exclus des éléments qui peuvent être ajoutés à I, autrement dit, l'un des éléments de I ne peut être défendu.

Dans toute la suite, nous allons utiliser la notation suivante :

Notation 6.4. *Étant donné un R-candidat* $C = (I, O, U)$*, l'ensemble* $R^-(I) \setminus R^+(I)$ *est dénoté par* O_p*.*

Les conditions StopCondVrai et StopCondFaux sont donc :

StopCondVrai(C, R) $\equiv$ $O_p = \emptyset$
StopCondFaux(C, R) $\equiv$ $\exists x \in O_p$ tel que $R^-(x) \subseteq O$

Fonction de sélection `Select`

Notre stratégie pour répondre au problème est d'essayer de «vider» l'ensemble O_p. En effet, cet ensemble contient les prédécesseurs de I qui ne sont pas encore successeurs de I, autrement dit il contient les arguments qui attaquent I et qui ne sont pas encore attaqués par I.

Pour vider cet ensemble, la fonction `Select` va chercher à sélectionner des prédécesseurs non décidés des éléments de O_p (autrement dit des défenseurs potentiels des arguments de I attaqués et non encore défendus) pour qu'ils soient ensuite ajoutés à I. `Select` va utiliser pour cela les heuristiques définies dans la section 6.4.2.

Remarquons que si un prédécesseur de O_p s'auto-attaque, il ne peut pas être ajouté à I, puisque I doit rester sans-conflit ; ce prédécesseur sera donc ajouté à O.

Si un prédécesseur de O_p appartient à $\text{Héros}(I, U, R)$ ou $\text{App}(I, U, R)$, alors d'après la propriété 6.7 page 89, il peut être ajouté à I. Notons que $\text{Héros}(I, U, R)$ ne contient que des éléments de $R^-(O_p)$.

Si maintenant il n'existe aucun prédécesseur qui s'auto-attaque ou qui appartient à l'un des deux ensembles précédents, alors un prédécesseur non décidé de O_p est choisi arbitrairement.

L'algorithme 17 est celui de la fonction `Select` spécifique pour répondre au problème d'acceptabilité crédule sous la sémantique préférée ; nous renommons cette fonction `SelectCred`.

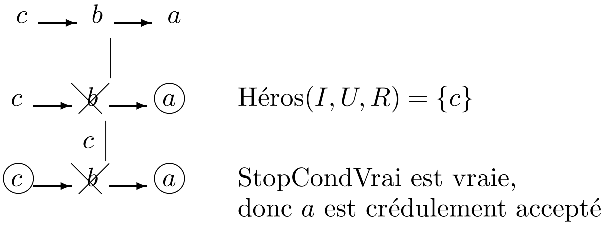
Algorithme 17 `SelectCred`(R, C, a)

Entrée:
- une relation binaire R,
- un R-candidat $C = (I, O, U)$,
- un argument a ;

Sortie: un triplet (x, b_I, b_O) où x est un argument, b_I et b_O sont deux booléens ;

```
Si $\text{Refl}(U \cap R^-(O_p), R) \neq \emptyset$ Alors
   retourner $((\texttt{Choisir}(\text{Refl}(U \cap R^-(O_p), R)), \bot, \top))$ ;
Sinon Si $\text{Héros}(I, U, R) \neq \emptyset$ Alors
   retourner $((\texttt{Choisir}(\text{Héros}(I, U, R)), \top, \bot))$ ;
Sinon Si $\text{App}(I, U, R) \cap R^-(O_p) \neq \emptyset$ Alors
   retourner $((\texttt{Choisir}(\text{App}(I, U, R) \cap R^-(O_p)), \top, \bot))$ ;
Sinon
   retourner $((\texttt{Choisir}(U \cap R^-(O_p)), \top, \top))$ ;
Fin Si
```

Exemple 6.1 (suite). L'arbre d'énumération qui permet de déterminer que l'argument a appartient à une extension préférée du système d'argumentation *Zoé l'autruche* est indiqué sur la figure 6.12. ◇



$\text{Héros}(I, U, R) = \{c\}$

StopCondVrai est vraie,
donc a est crédulement accepté

Fig. 6.12 – Arbre d'énumération pour déterminer si l'argument a du système d'argumentation *Zoé l'autruche* est crédulement accepté sous la sémantique préférée.

6.5.3 Problème d'acceptabilité sceptique sous la sémantique préférée

Dans cette section, nous allons montrer comment la fonction `RepPbScept` peut être utilisée pour répondre au problème d'acceptabilité sceptique sous la sémantique préférée. Pour cela, nous allons tout d'abord préciser les conditions StopCondVrai et StopCondFaux, puis nous donnerons l'algorithme de la fonction `Select`.

Dans toute la suite, pour dénoter l'ensemble des éléments de O qui ne sont ni successeurs ni prédécesseurs de I, nous utiliserons la notation suivante :

Notation 6.5. *Étant donné un R-candidat* $C = (I, O, U)$*, l'ensemble* $O \setminus R^{\pm}(I)$ *est dénoté par* O_u*.*

Conditions StopCondVrai **et** StopCondFaux

À l'appel de $\texttt{RepPb}_{\texttt{rec}}$ par `RepPbScept`, l'argument a de la requête est dans O_u. $\texttt{RepPb}_{\texttt{rec}}$ va donc chercher s'il existe une extension de la sémantique qui ne contient pas a.

À une étape donnée de l'algorithme, il est certain qu'une telle extension existe si :

- l'ensemble I est admissible et a est attaqué par I, ou bien
- l'ensemble I est admissible, $a \in O_u$ (il est rejeté de cet ensemble admissible) et il est certain que I est inclus dans un ensemble maximal admissible, autrement dit la condition $\text{Max}^*(I, O_u, U, R)$ est vérifiée.

La propriété suivante permet de vérifier ces deux conditions :

Propriété 6.10. *Étant donné une relation binaire R, un argument a et un R-candidat* (I, O, U) *tel que* $O_p = \emptyset$ *: si* $a \in R^+(I)$*, ou si* $a \in O_u$ *et* $\text{Max}^*(I, O_u, U, R)$ *est vrai, alors il existe* $S \in \text{Pref}(I \cup U \cup O, R)$ *tel que* $a \notin S$*.*

Preuve – D'après la propriété 6.4, on sait que I est contenu dans au moins une extension préférée S de $(I \cup O \cup U, R)$.

Si $a \in R^+(I) \subseteq R^+(S)$, puisque S est sans-conflit, $a \notin S$.

Sinon, si $a \in O_u$ et $\text{Max}^*(I, O_u, U, R)$ est vrai, alors a ne peut être défendu par S. Prenons $X = \{a\}$; alors :

- soit $X \cap R^+(X) \neq \emptyset$, c'est-à-dire aRa, auquel cas a ne peut être dans aucune extension préférée ;
- soit $R^-(X) \not\subseteq R^+(I \cup U \cup X)$: si $S \subseteq I \cup U$, $R^-(X) \not\subseteq R^+(S)$. Puisque S est une extension préférée, $X \not\subseteq S$, c'est-à-dire $a \notin S$.

□

La condition StopCondVrai est donc :

$$\text{StopCondVrai}(C, R) \quad \equiv \quad O_p = \emptyset \wedge (a \in R^+(I) \vee (a \in O_u \wedge \text{Max}^*(I, O_u, U, R)))$$

En revanche, à une étape donnée, il est certain qu'il n'est pas possible de trouver une extension préférée ne contenant pas a dans une feuille descendante du nœud courant si :

- un prédécesseur de I a tous ses prédécesseurs dans O (propriété 6.4 page 83), ou bien
- U est vide et l'ensemble I n'est pas maximal admissible, autrement dit $\text{Max}^*(I, O_u, \emptyset, R)$ est faux (propriété 6.6 page 88).

La condition StopCondFaux est donc :

$$\text{StopCondFaux}(C, R) \quad \equiv \quad (\exists x \in O_p \text{ tel que } R^-(x) \subseteq O) \vee (U = \emptyset \wedge \neg\text{Max}^*(I, O_u, \emptyset, R))$$

Fonction de sélection `Select`

Pour déterminer s'il existe une extension préférée ne contenant pas l'argument a de la requête, nous allons utiliser plusieurs stratégies.

Tout d'abord, nous allons essayer de construire I de telle sorte que a devienne successeur de I, en vérifiant si les prédécesseurs de a sont dans $\mathrm{App}(I,U,R)$. Si a est successeur de I, on va alors chercher à vider O_p pour essayer de montrer que I est un ensemble admissible ; les arguments devront être sélectionnés de la même manière que pour répondre au problème d'acceptabilité crédule. Pour cette raison, nous utiliserons alors la fonction de sélection d'un argument `SelectCred`.

Si a ne peut être successeur de I, on va chercher à montrer que I est inclus dans un ensemble maximal admissible dont a est rejeté, autrement dit on va essayer de satisfaire la condition $\mathrm{Max}^*(I,O_u,U,R)$. Pour cela, nous allons essayer de faire passer les prédécesseurs de O_u dans O ou dans I en examinant les éléments de $\mathrm{Refl}(U)$, $\mathrm{Traitres}(I,U,R)$, $\mathrm{Nondef}(I,U,R)$ ou $\mathrm{App}(I,U,R)$.

Si aucune des deux stratégies précédentes n'est utilisable, on vérifie si U est sans-cycle, auquel cas, d'après la propriété 6.8 page 91, il reste à vérifier si $I \cup U$ est une extension préférée.

Si U contient des cycles, on essaie de sélectionner un prédécesseur de O_u qui appartient à un cycle. Si un tel élément n'existe pas, on cherche à sélectionner un élément non décidé prédécesseur de O_u. Si un tel élément n'existe pas non plus, on cherche à sélectionner un élément non décidé qui appartient à un cycle.

L'algorithme 18 est celui de la fonction `Select` spécifique au problème d'acceptabilité sceptique sous la sémantique préférée ; nous renommons cette fonction `SelectScept`.

Algorithme 18 `SelectScept`(R,C,a)

```
Entrée:
    – une relation binaire $R$,
    – un $R$-candidat $C = (I,O,U)$,
    – un argument $a$ ;
Sortie: un triplet $(x, b_I, b_O)$ où $x$ est un argument, $b_I$ et $b_O$ sont deux booléens ;
Si $a \in R^+(I)$ Alors
    retourner (SelectCred$(C,R)$) ;
Sinon Si $\mathrm{App}(I,U,R) \cap R^-(a) \neq \emptyset$ Alors
    retourner ((Choisir$(\mathrm{App}(I,U,R) \cap R^-(a)), \top, \bot)$) ;
Sinon Si $\mathrm{Traitres}(I,O,U,R) \neq \emptyset$ Alors
    retourner ((Choisir$(\mathrm{Traitres}(I,O,U,R)), \bot, \top)$) ;
Sinon Si $\mathrm{Refl}(U \cap R^-(O_u), R) \neq \emptyset$ Alors
    retourner ((Choisir$(\mathrm{Refl}(U \cap R^-(O_u), R)), \bot, \top)$) ;
Sinon Si $\mathrm{Nondef}(I,U,R) \cap R^-(O_u) \neq \emptyset$ Alors
    retourner ((Choisir$(\mathrm{Nondef}(I,U,R) \cap R^-(O_u)), \bot, \top)$) ;
Sinon Si $\mathrm{App}(I,U,R) \cap R^-(O_u \setminus \{a\}) \neq \emptyset$ Alors
    retourner ((Choisir$(\mathrm{App}(I,U,R) \cap R^-(O_u \setminus \{a\})), \top, \bot)$) ;
Sinon Si Cycles$(U) = \emptyset$ Alors
    retourner ((Choisir$(U), \bot, \bot)$) ;
Sinon Si $\exists e \in$ Cycles$(U)$ tel que $e \cap R^-(O_u) \neq \emptyset$ Alors
    retourner ((Choisir$(e \cap R^-(O_u)), \top, \top)$) ;
Sinon Si $U \cap R^-(O_u) \neq \emptyset$ Alors
    retourner ((Choisir$(U \cap R^-(O_u)), \top, \top)$) ;
Sinon
    retourner ((Choisir(Choisir(Cycles$(U)$)), $\top, \top)$) ;
Fin Si
```

Notons également qu'il n'est pas difficile de voir que pour déterminer l'acceptabilité sceptique d'un argument, seule la composante maximale connexe à laquelle appartient cet argument devrait être explorée (voir [DM01b]).

Exemple 6.1 (suite). L'arbre d'énumération qui permet de déterminer que l'argument a appartient à toute extension préférée du système d'argumentation *Zoé l'autruche* est indiqué sur la figure 6.13. ♢

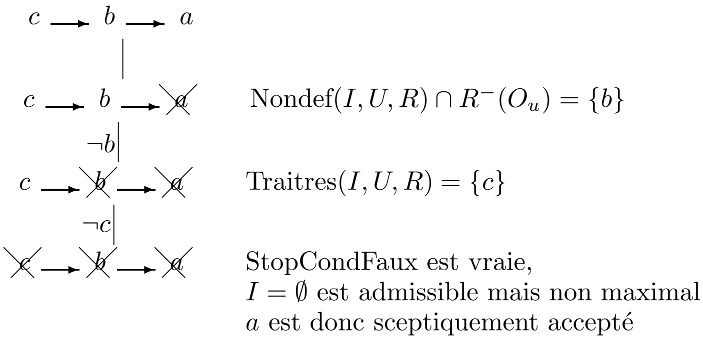


Fig. 6.13 – Arbre d'énumération pour déterminer si l'argument a du système d'argumentation *Zoé l'autruche* est sceptiquement accepté sous la sémantique préférée.

6.6 Travaux voisins

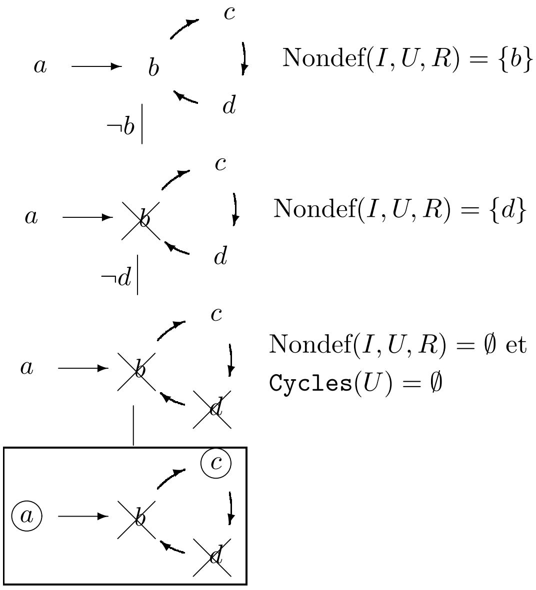
6.6.1 Coupe-cycle

L'idée de choisir un élément qui coupe un cycle a été mise en évidence par [DMP97]. Néanmoins, [DMP97] propose de calculer au tout début de l'algorithme un *ensemble coupe-cycle*, c'est-à-dire un ensemble F de sommets du graphe (A, R) tel que le graphe restreint à $A \setminus F$ ne contient pas de cycle. L'algorithme de [DMP97] choisit ensuite un élément non décidé de F lorsqu'aucune propriété ne permet de n'explorer qu'une branche.

Trouver un ensemble coupe-cycle minimal est connu comme étant un problème NP-complet, bien que des heuristiques puissent être utilisées pour trouver un «petit» ensemble coupe-cycle. Néanmoins, il y a des cas où l'utilisation d'un ensemble coupe-cycle n'est pas utile. Le système d'argumentation (A, R) tel que :

$$A = \{a, b, c, d\}$$
$$R = \{(a, b), (b, c), (c, d), (d, b)\}$$

en est un exemple ; l'arbre d'énumération (réduit à une branche) du calcul de l'extension préférée de ce système est le suivant :



Ce système possède un cycle de longueur impaire, mais il n'a pas été nécessaire de choisir un élément qui coupe le cycle.

Ainsi, il est plus efficace de s'intéresser aux cycles seulement lorsque c'est nécessaire. Nous pourrions apporter une amélioration supplémentaire à notre algorithme en maintenant un ensemble de tous les cycles qui ont été déjà été trouvés jusqu'à un moment donné de l'algorithme, dans le but de ne pas trouver plusieurs fois le même cycle dans des branches différentes.

6.6.2 Calcul des extensions préférées grâce aux algorithmes de calcul des extensions stables

Nous avons vu en section 2.2 la correspondance entre les notions d'extensions stables et d'extensions préférées, et celles de noyaux et semi-noyaux maximaux.

Dimopoulos et Torres montrent dans [DT96] comment un algorithme qui calcule les noyaux d'un graphe peut calculer ses semi-noyaux. Pour cela, ils étendent le graphe d'origine en un graphe équivalent, dont les noyaux permettent de retrouver les semi-noyaux du graphe d'origine.

Définition 6.2 (Graphe semi-noyau-équivalent).
[DT96] Un graphe (V', E') *est le* semi-noyau-équivalent *d'un graphe* (V, E) *si :*
- $V' = V \cup \{v' \mid v \in V\}$, *et*
- $E' = E \cup \{(v, v'), (v', v) \mid v \in V \text{ et } v' \in V' \setminus V\} \cup \{(v_i, v'_j) \mid v_i \in V, v'_j \in V' \text{ et } v_j \in R^-(v_i)\}$.

EXEMPLE 6.3. Considérons le graphe $G = (V, E)$ tel que $V = \{a, b\}$ et $E = \{(b, a)\}$. Le graphe semi-noyau-équivalent de G est $G' = (V', E')$ avec :
- $V' = V \cup \{a', b'\}$ et
- $E' = E \cup \{(a, a'), (a', a), (b, b'), (b', b)\} \cup \{(a, b')\}$.

Ces deux graphes sont indiqués sur la figure 6.14. ◇

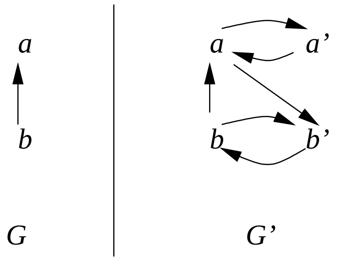


FIG. 6.14 – Un graphe G et son graphe semi-noyau-équivalent G'.

Théorème 6.1. *[DT96] Si* (V', E') *est le semi-noyau-équivalent d'un graphe* (V, E) *alors, si* K *est un noyau de* (V', E'), $K \cap V$ *est un semi-noyau de* (V, E)*. Inversement, si* S *est un semi-noyau de* (V, E) *alors* $S \cup \{v \mid v \in V' \setminus V, v \notin R^+(S)\}$ *est un noyau de* (V', E').

EXEMPLE 6.3 (SUITE). Les noyaux de G' sont $\{a', b\}$ et $\{a', b'\}$, les semi-noyaux de G sont $\{b\}$ et $\emptyset$. ◇

Il serait intéressant de comparer cette méthode avec nos algorithmes.

Troisième partie

Raffinement de la sémantique préférée

Chapitre 7

Sémantique Défense Minimale

Nous nous sommes intéressés dans les chapitres précédents essentiellement à la sémantique préférée de Dung. Cette sémantique possède de nombreuses qualités, mais comme d'autres sémantiques de Dung, elle est critiquable sur plusieurs points (voir section 2.6).

La critique que nous allons corriger ici est le fait que les définitions de l'acceptabilité de Dung ne font aucune distinction entre les arguments, dans le sens où les défenseurs sont tous également désirables : si un argument peut être indépendamment défendu par deux arguments différents, alors on choisit n'importe lequel des deux, ou même les deux s'ils ne s'attaquent pas, pour défendre le premier argument. Cependant, dans certaines applications, une distinction entre plusieurs niveaux de désirabilité est nécessaire.

Supposons par exemple que les arguments «appartiennent» à deux agents : l'un, le partisan, essaie de soutenir un point de vue, alors que l'autre, l'opposant, essaie de le réfuter. Parmi les arguments du partisan, il peut en être certains qui ne doivent être révélés qu'en cas d'absolue nécessité (comme ceux qui contiennent des informations très personnelles ou qui pourraient mettre en danger la sécurité du partisan).

Exemple 7.1 (Le procès de la Mafia). Imaginons un procès où la Mafia est accusée d'avoir commis un crime. Face à elle, la partie civile connaît une personne dont le témoignage apporte des éléments de preuves pour la culpabilité des suspects. La partie civile hésite néanmoins à faire témoigner cette personne, puisqu'alors son identité serait révélée, et la Mafia étant connue pour être très violente contre les personnes qui témoignent contre elle, sa vie serait alors grandement menacée. Dans ce cas, la partie civile va essayer autant que possible de ne pas utiliser les éléments de preuve du témoin pour gagner le procès.

La partie civile peut être vue comme le partisan (elle soutient la thèse que la Mafia est coupable), l'avocat de la Mafia peut être vu comme l'opposant (il va contre la thèse de la partie civile). Parmi les arguments du partisan, il en est de deux catégories : ceux qui peuvent être révélés sans contrainte et les autres, ceux du témoin. ◇

Nous proposons, dans un premier temps, de raffiner le système d'argumentation de Dung en faisant une distinction entre les arguments du partisan et ceux de l'opposant : ceci nous permettra par exemple de garantir, si besoin est, que les arguments de l'opposant ne sont jamais utilisés pour défendre un argument du partisan. Nous distinguons également deux types d'arguments que peut avoir le partisan : les arguments *désirables* ou *non restreints* (*i.e.* dont l'utilisation n'est pas restreinte), et les arguments *optionnels* ou *restreints* (*i.e.* à n'utiliser qu'avec précautions, comme les arguments du témoin dans l'exemple précédent).

L'exemple précédent suggère que les arguments restreints ne doivent être pris en compte que lorsqu'ils défendent directement ou indirectement un argument non restreint. Une nouvelle notion d'acceptabilité est nécessaire pour prendre en compte cette contrainte. La suite de l'exemple suggère encore d'autres contraintes :

Exemple 7.1 (suite). Partie civile et avocat de la Mafia échangent des arguments au cours du procès. Pour gagner le procès, les arguments de la partie civile devront former un ensemble :

- sans-conflit (leur thèse ne doit pas se contredire),
- qui défend tous ses arguments (il faut qu'ils puissent répondre à toute attaque de l'avocat de la Mafia),
- tel que les arguments du témoin ne sont présents dans l'ensemble que parce qu'ils défendent des arguments de la partie civile qui ne pourraient être défendus autrement,
- qui contient un minimum d'arguments du témoin (voire même aucun !).

À ces contraintes peuvent s'en ajouter deux autres :

- La partie civile peut préférer tenter de gagner le procès en utilisant seulement ses propres arguments.
- Pour être la plus convaincante possible, la partie civile peut chercher à avancer un maximum d'arguments.

♢

Étant donné un système d'argumentation raffiné, nous allons définir une nouvelle notion d'acceptabilité (*i.e.* une nouvelle sémantique), qui satisfait les conditions suggérées dans l'exemple précédent. Un ensemble d'arguments acceptables sous la nouvelle sémantique devra :

1. être sans-conflit ;
2. défendre tous ses éléments ;
3. ne contenir que des arguments désirables et optionnels ;
4. être tel que les arguments optionnels défendent tous directement ou indirectement un argument désirable ;
5. contenir un minimum d'arguments optionnels ;
6. contenir un maximum d'arguments désirables.

Nous allons progressivement amener la définition de cette nouvelle sémantique.

Partant des ensembles admissibles (qui satisfont les conditions 1 et 2), nous allons tout d'abord porter notre attention sur ceux qui ne contiennent que des arguments désirables et optionnels.

Parmi ces ensembles, nous allons isoler ceux qui ne contiennent des arguments optionnels que parce qu'ils défendent des arguments désirables ; on dira de ces ensembles qu'ils sont *do-admissibles*[1].

Ensuite, nous allons isoler, parmi les ensembles do-admissibles, ceux qui contiennent un minimum d'arguments optionnels et un maximum d'arguments désirables. Ces ensembles, que nous appellerons les *min-def extensions*, composent la *sémantique défense minimale*. C'est la nouvelle notion d'acceptabilité que nous recherchons.

Notons bien que la distinction entre arguments restreints et non restreints se situe à un *niveau défensif*. D'autres systèmes ont été conçus pour prendre en compte différentes forces pour les arguments dans le but d'aider à la résolution de conflits ; par exemple, dans les systèmes de [AC98] et [PS97], une relation de préférence entre arguments est combinée à la relation de contrariété pour déterminer l'acceptabilité des arguments : pour être une menace sérieuse, un argument doit être préféré à l'argument qu'il attaque. Dans ces systèmes, on ne cherche pas à minimiser le nombre des arguments d'une force donnée, sauf en cas de recherche d'efficacité ou de concision.

Nous allons tout d'abord présenter le système d'argumentation raffiné, en détaillant les deux étapes de la partition de l'ensemble des arguments. Nous amènerons ensuite progressivement la définition de la nouvelle sémantique. Nous présenterons enfin deux autres illustrations du système et de la nouvelle sémantique, puis quelques applications possibles. Ce travail a été publié dans [CDLSM02b].

Dans toute la suite, nous dirons que le système d'argumentation de Dung [Dun95] est le système d'argumentation *classique*.

---

[1] *do-* : *d* pour *d*ésirables et *o* pour *o*ptionnels.

7.1 Partitions de l'ensemble des arguments

Étant donné un système d'argumentation classique $\mathcal{SA} = (A, R)$, nous allons étudier deux cas de partitions :

1. supposons que l'on souhaite focaliser notre attention sur un sous-ensemble F de A ; on va alors isoler l'ensemble F et obtenir une partition de A en deux ensembles : F et $A \setminus F$;
2. supposons que l'on souhaite focaliser notre attention sur un sous-ensemble F de A, dans lequel on distingue deux sortes d'arguments : des arguments dits **désirables** ou non restreints (que l'on désire vivement voir apparaître dans les ensembles d'arguments acceptables de $\mathcal{SA}$) et des arguments dits **optionnels** ou restreints (que l'on veut voir apparaître aussi peu que possible dans les ensembles acceptables de $\mathcal{SA}$) ; on va alors isoler les deux sous-ensembles de F et obtenir une partition de A en trois ensembles : F_d (l'ensemble des arguments désirables de F), F_o (l'ensemble des arguments optionnels de F) et $A \setminus (F_d \cup F_o)$.

La première partition permet d'isoler les arguments du partisan parmi l'ensemble des arguments. La deuxième partition permet de distinguer, parmi les arguments du partisan, ceux qui sont désirables de ceux qui sont optionnels.

Pour pouvoir travailler avec ces partitions, nous allons définir deux nouveaux types de systèmes d'argumentation :

Définition 7.1 (Système `type-1`-partitionné).
Un système d'argumentation `type-1`-partitionné *est un triplet* $\mathcal{SA}_{XY} = (X, Y, R)$ *où* X *et* Y *sont des ensembles d'arguments tels que* $X \cap Y = \emptyset$ *et* R *est une relation binaire sur* $X \cup Y$ *qui représente une notion de contrariété entre arguments (*$R \subseteq ((X \cup Y) \times (X \cup Y))$*).*

À un système d'argumentation `type-1`-partitionné $\mathcal{SA}_{XY} = (X, Y, R)$ correspond le système d'argumentation classique $\mathcal{SA} = (X \cup Y, R)$.

Définition 7.2 (Système `type-2`-partitionné).
Un système d'argumentation `type-2`-partitionné *est un quadruplet* $\mathcal{SA}_{XYZ} = (X, Y, Z, R)$ *où* X*,* Y *et* Z *sont des ensembles d'arguments tels que* $X \cap Y = \emptyset$*,* $X \cap Z = \emptyset$ *et* $Y \cap Z = \emptyset$ *et* R *est une relation binaire sur* $X \cup Y \cup Z$ *qui représente une notion de contrariété entre arguments (*$R \subseteq ((X \cup Y \cup Z) \times (X \cup Y \cup Z))$*).*

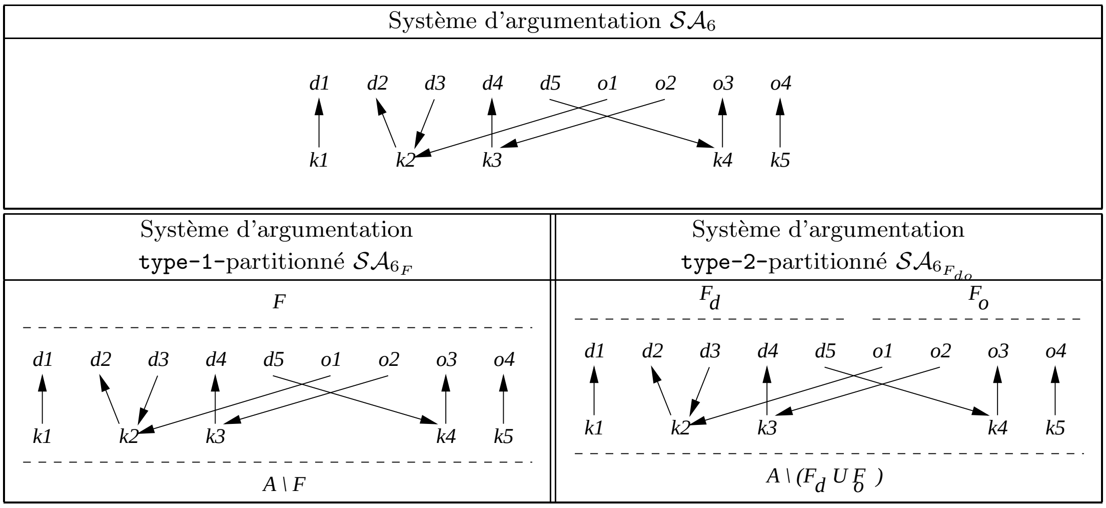
À un système d'argumentation `type-2`-partitionné $\mathcal{SA}_{XYZ} = (X, Y, Z, R)$ correspond le système d'argumentation classique $\mathcal{SA} = (X \cup Y \cup Z, R)$.

Inversement, à un système d'argumentation classique $\mathcal{SA} = (A, R)$ va correspondre un système d'argumentation `type-1-` ou `type-2`-partitionné fonction du ou des ensemble(s) sur le(s)quel(s) on veut se focaliser. Ainsi :

1. si l'on souhaite focaliser notre attention sur un sous-ensemble F de A, alors le système d'argumentation `type-1`-partitionné correspondant à $\mathcal{SA}$ est :
$$\mathcal{SA}_{F(A \setminus F)} = (F, A \setminus F, R)$$
que l'on note pour simplifier $\mathcal{SA}_F$.
2. si l'on souhaite focaliser notre attention sur un sous-ensemble F de A, dans lequel on distingue les arguments désirables et les arguments optionnels, alors le système d'argumentation `type-2`-partitionné correspondant à $\mathcal{SA}$ est :
$$\mathcal{SA}_{F_d F_o (A \setminus (F_d \cup F_o))} = (F_d, F_o, A \setminus (F_d \cup F_o), R)$$
que l'on note pour simplifier $\mathcal{SA}_{F_{do}}$.

Exemple 7.1 (suite). L'ensemble des arguments du système d'argumentation $\mathcal{SA}_6$ représenté dans le tableau 7.1, peut être partitionné de manière à isoler l'ensemble $F = F_d \cup F_o$ où $F_d = \{d_1, d_2, d_3, d_4, d_5\}$ et $F_o = \{o_1, o_2, o_3, o_4\}$. Le système d'argumentation `type-1`-partitionné (resp. `type-2`-partitionné) ainsi obtenu est $\mathcal{SA}_{6_F} = (F, A \setminus F, R)$ (resp. $\mathcal{SA}_{6_{F_{do}}} = (F_d, F_o, A \setminus (F_d \cup F_o), R)$) ; tous deux sont représentés dans le tableau 7.1.

On peut imaginer que ce système d'argumentation $\mathcal{SA}6_{F_{do}}$ représente les arguments et les attaques entre arguments émis au cours du procès contre la Mafia. F est l'ensemble des arguments de la partie civile : F_d sont ceux qui peuvent être révélés ; F_o sont ceux du témoins, ils sont à utiliser avec précaution. $A \setminus F$ est l'ensemble des arguments de l'avocat de la Mafia. ♢



Tab. 7.1 – Systèmes d'argumentation $\mathcal{SA}6$, $\mathcal{SA}6_F$ et $\mathcal{SA}6_{F_{do}}$.

Dans toute la suite :

- les systèmes d'argumentation `type-1`-partitionnés seront notés :
  $$\mathcal{SA}_F = (F, K, R)$$
  et seront tels que F est un ensemble d'arguments sur lequel on souhaite se focaliser ;
- les systèmes d'argumentation `type-2`-partitionnés seront notés :
  $$\mathcal{SA}_{F_{do}} = (F_d, F_o, K, R)$$
  et seront tels que $F_d \cup F_o$ est un ensemble d'arguments sur lequel on souhaite se focaliser, F_d étant l'ensemble des arguments désirables et F_o l'ensemble des arguments optionnels.

À un système d'argumentation `type-2`-partitionné $\mathcal{SA}_{F_{do}} = (F_d, F_o, K, R)$, on fera correspondre le système d'argumentation `type-1`-partitionné $\mathcal{SA}_F = (F, K, R)$ tel que $F = F_d \cup F_o$.

7.2 Ensembles admissibles

Nous étendons la définition de l'admissibilité aux deux nouveaux systèmes d'argumentation : un ensemble d'arguments est admissible dans un système d'argumentation `type-1`-partitionné (resp. `type-2`-partitionné) s'il est admissible dans le système d'argumentation classique associé.

Exemple 7.1 (ensembles admissibles). Les ensembles :

$$\emptyset,\ \{k_1\},\ \{d_2, d_3\},\ \{d_5, o_3\},\ \{k_1, d_2, d_3, d_4, d_5, o_1, o_2, o_3, k_5\}$$

sont des ensembles admissibles du système d'argumentation $\mathcal{SA}6$. Ce sont donc également des ensembles admissibles des systèmes $\mathcal{SA}6_F$ et $\mathcal{SA}6_{F_{do}}$. ♢

Notons que la notion d'extension préférée peut être étendue de la même manière que celle d'ensemble admissible : un ensemble d'arguments est une extension préférée d'un système d'argumentation `type-1`-partitionné (resp. `type-2`-partitionné) s'il est une extension préférée du système d'argumentation classique associé.

Exemple 7.1 (extensions préférées). $\mathcal{SA}6, \mathcal{SA}6_F$ et $\mathcal{SA}6_{F_{do}}$ ont une seule extension préférée :

$$\{k_1, d_2, d_3, d_4, d_5, o_1, o_2, o_3, k_5\}.$$

♢

Pour aller vers la définition de la nouvelle sémantique, nous allons porter notre attention sur les ensembles admissibles inclus dans l'ensemble des arguments désirables ou optionnels.

EXEMPLE 7.1 (ENSEMBLES ADMISSIBLES CONTENANT DES ARGUMENTS DÉSIRABLES OU OPTIONNELS). Les ensembles $\{d_2, d_3\}$ et $\{d_5, o_3\}$ sont des ensembles admissibles inclus dans F. $\diamondsuit$

7.3 Ensembles do-admissibles

Nous raffinons la notion d'admissibilité pour les systèmes d'argumentation `type-2-`partitionnés, dans le but d'accepter les arguments optionnels dans un ensemble admissible seulement si leur présence est justifiée, c'est-à-dire lorsqu'ils défendent au moins un argument désirable de l'ensemble. Ce raffinement est appelé la *do-admissibilité*.

Soit $\mathcal{SA}_{F_{do}} = (F_d, F_o, K, R)$ un système d'argumentation `type-2-`partitionné.

Notation 7.1. *Étant donné un ensemble $S \subseteq F_d \cup F_o$, S_d dénote $S \cap F_d$ et S_o dénote $S \cap F_o$. On dit que S_d est la partie* désirable *(ou* non restreinte*) de S, et que S_o est la partie* optionnelle *(ou* restreinte*) de S.*

Définition 7.3 (Ensemble do-admissible).
Soit $S \subseteq (F_d \cup F_o)$. S est do-admissible *si et seulement si :*

1. *S est admissible et*
2. *$\forall x \in S_o$, $\exists y \in S_d$ tel que x est un défenseur individuel de y.*

EXEMPLE 7.1 (ENSEMBLES DO-ADMISSIBLES). $\mathcal{SA}6_{F_{do}}$ a plusieurs ensembles do-admissibles. Par exemple :

$$\emptyset, \{d_2, d_3\}, \{d_2, d_3, d_4, d_5, o_2\}, \{d_2, d_3, d_4, d_5, o_1, o_2\}.$$

L'ensemble $S = \{d_5, o_3\}$ n'est pas do-admissible car o_3 ne défend aucun argument de la partie désirable de S (qui est $\{d_5\}$). $\diamondsuit$

Dans les ensembles do-admissibles, un argument optionnel peut être utilisé pour défendre un argument désirable, mais cet argument désirable pourra éventuellement être aussi défendu par un argument désirable.

EXEMPLE 7.1 (SUITE). Dans l'ensemble $\{d_2, d_3, d_4, d_5, o_1, o_2\}$, l'argument d_2 est défendu par d_3 ; l'argument o_1 est donc inutile. $\diamondsuit$

Dans les min-def extensions, on restreint l'utilisation des arguments optionnels au cas où ils sont les seuls à pouvoir défendre un argument désirable.

7.4 Min-def extensions

Dans le cadre d'un système d'argumentation `type-2-`partitionné, nous allons sélectionner les ensembles do-admissibles qui contiennent un maximum d'arguments désirables et un minimum d'arguments optionnels. Dans ce but, nous définissons la relation $\prec$ et la sémantique défense minimale.

Soit un système d'argumentation `type-2-`partitionné $\mathcal{SA}_{F_{do}} = (F_d, F_o, K, R)$. Dans toute cette section, on pose $F = F_d \cup F_o$.

7.4.1 Relation $\prec$

Nous allons définir une relation $\prec$ telle que : $S_1 \prec S_2$ si et seulement si S_2 contient strictement plus d'arguments désirables que S_1, ou bien S_1 et S_2 ont les mêmes arguments désirables mais S_2 contient moins d'arguments optionnels que S_1. Formellement :

Définition 7.4 (Relation $\prec$).
Soit S_1 et S_2 deux sous-ensembles de $F_d \cup F_o$. On dit que S_2 est $\prec$-meilleur que S_1 (noté $S_1 \prec S_2$) si et seulement si $S_{1d} \subset S_{2d}$, ou bien $S_{1d} = S_{2d}$ et $S_{2o} \subseteq S_{1o}$.

Propriété 7.1. *La relation $\prec$ est un ordre partiel sur l'ensemble des sous-ensembles de $F = F_d \cup F_o$.*

PREUVE – Il faut vérifier que $\prec$ est (i) réflexive, (ii) antisymétrique et (iii) transitive.

(i) Soit $x \in 2^F$. Il faut vérifier que $x \prec x$ est vrai.

$$x \prec x \quad \Leftrightarrow \quad (x_d \subset x_d) \vee (x_d = x_d \wedge x_o \subseteq x_o)$$

$(x_d \subset x_d)$ est faux, mais $(x_d = x_d \wedge x_o \subseteq x_o)$ est vrai, donc $x \prec x$ est vrai.

(ii) Soit $x, y \in 2^F$. Il faut vérifier que $(x \prec y) \wedge (y \prec x) \Rightarrow (x = y)$.

$$\begin{aligned} x \prec y \quad &\Leftrightarrow \quad x_d \subset y_d \vee (x_d = y_d \wedge y_o \subseteq x_o) \\ y \prec x \quad &\Leftrightarrow \quad y_d \subset x_d \vee (y_d = x_d \wedge x_o \subseteq y_o) \end{aligned}$$

Étant donné A, B, C, D des formules de la logique propositionnelle, on a l'équivalence suivante :

$$((A \vee B) \wedge (C \vee D)) \quad \Leftrightarrow \quad ((A \wedge C) \vee (A \wedge D) \vee (B \wedge C) \vee (B \wedge D))$$

Par conséquent :

$$\begin{aligned} & (x \prec y) \wedge (y \prec x) \\ \Leftrightarrow \quad & ((x_d \subset y_d) \wedge (y_d \subset x_d)) & (7.1) \\ \vee \quad & ((x_d \subset y_d) \wedge (y_d = x_d) \wedge (x_o \subseteq y_o)) & (7.2) \\ \vee \quad & ((y_d \subset x_d) \wedge (x_d = y_d) \wedge (y_o \subseteq x_o)) & (7.3) \\ \vee \quad & ((x_d = y_d) \wedge (y_o \subseteq x_o) \wedge (x_o \subseteq y_o)) \end{aligned}$$

Du fait de l'inclusion stricte, (7.1), (7.2) et (7.3) sont faux. On a donc :

$$\begin{aligned} (x \prec y) \wedge (y \prec x) \quad &\Leftrightarrow \quad (x_d = y_d) \wedge (y_o = x_o) \\ (x \prec y) \wedge (y \prec x) \quad &\Leftrightarrow \quad x = y \end{aligned}$$

(iii) Soit $x, y, z \in 2^F$. Il faut vérifier que $(x \prec y) \wedge (y \prec z) \Rightarrow (x \prec z)$.

$$\begin{aligned} x \prec y \quad &\Leftrightarrow \quad x_d \subset y_d \vee (x_d = y_d \wedge y_o \subseteq x_o) \\ y \prec z \quad &\Leftrightarrow \quad y_d \subset z_d \vee (y_d = z_d \wedge z_o \subseteq y_o) \end{aligned}$$

d'après l'équivalence logique rappelée ci-dessus, on a :

$$\begin{aligned} & (x \prec y) \wedge (y \prec z) \\ \Leftrightarrow \quad & ((x_d \subset y_d) \wedge (y_d \subset z_d)) \\ \vee \quad & ((x_d \subset y_d) \wedge (y_d = z_d) \wedge (z_o \subseteq y_o)) \\ \vee \quad & ((y_d \subset z_d) \wedge (x_d = y_d) \wedge (y_o \subseteq x_o)) \\ \vee \quad & ((x_d = y_d) \wedge (y_d = z_d) \wedge (y_o \subseteq x_o) \wedge (z_o \subseteq y_o)) \\ \Rightarrow \quad & (x_d \subset z_d) & (7.4) \\ \vee \quad & ((x_d \subset z_d) \wedge (z_o \subseteq y_o)) & (7.5) \\ \vee \quad & ((x_d \subset z_d) \wedge (y_o \subseteq x_o)) & (7.6) \\ \vee \quad & ((x_d = z_d) \wedge (z_o \subseteq x_o)) \end{aligned}$$

Étant donné A et B deux formules de la logique propositionnelle, on a l'équivalence suivante : $(A \vee (A \wedge B)) \Leftrightarrow A$. Ainsi, (7.4) $\vee$ (7.5) $\vee$ (7.6) est réduit à (7.4). On a donc :

$$\begin{aligned} (x \prec y) \wedge (y \prec z) &\Rightarrow (x_d \subset z_d) \vee ((x_d = z_d) \wedge (z_o \subseteq x_o)) \\ (x \prec y) \wedge (y \prec z) &\Rightarrow x \prec z. \end{aligned}$$

Comme (i), (ii) et (iii) sont vérifiés, $\prec$ est un ordre partiel sur 2^F. $\square$

7.4.2 Sémantique défense minimale

L'idée de la nouvelle sémantique est d'avoir aussi peu d'arguments optionnels que possible dans un ensemble do-admissible. En d'autres termes, la défense optionnelle doit être minimale. Pour cette raison, nous appelons la nouvelle sémantique la *sémantique défense minimale*, ou, pour simplifier, la *sémantique min-def.* Les ensembles d'arguments acceptables sous cette sémantique sont appelés des *min-def extensions.*

Définition 7.5 (min-def extension).
S est une min-def extension *de* $\mathcal{SA}_{F_{do}}$ *si et seulement si S est* $\prec$*-maximal parmi les ensembles do-admissibles de* $\mathcal{SA}_{F_{do}}$.

On peut noter que les arguments optionnels d'une min-def extension, non seulement défendent un argument désirable x de l'extension, mais en plus sont des *défenseurs essentiels* puisqu'aucun argument désirable ne défend x. Une conséquence de cela est que tout ensemble do-admissible n'est pas nécessairement inclus dans une min-def extension (puisque tous les arguments optionnels d'un ensemble do-admissible ne sont pas des défenseurs essentiels).

EXEMPLE 7.1 (MIN-DEF EXTENSIONS). $\mathcal{SA}6_{F_{do}}$ a une seule min-def extension :

$$S = \{d_2, d_3, d_4, d_5, o_2\}.$$

L'ensemble do-admissible $\{d_2, d_3, d_4, d_5, o_1, o_2\}$ n'est pas une min-def extension car o_1 n'est pas un défenseur essentiel : o_1 défend d_2 contre k_2 mais l'argument désirable d_3 défend lui aussi d_2.

Les min-def extensions correspondent exactement aux ensembles d'arguments que doit avancer la partie civile pour gagner le procès de la Mafia. $\Diamond$

La sémantique min-def peut également être définie par rapport aux ensembles admissibles $\prec$-maximaux. En effet, nous montrons ci-dessous que les ensembles do-admissibles $\prec$-maximaux sont exactement les ensembles admissibles $\prec$-maximaux. Définissons deux notations :

Notation 7.2. *On dénote par* Adm *l'ensemble des parties admissibles de* $\mathcal{SA}_{F_{do}}$ *et par* doAdm *l'ensemble des parties do-admissibles de* $\mathcal{SA}_{F_{do}}$. *Étant donné un ensemble* $E \subseteq 2^{F_d \cup F_o \cup K}$, max-$\prec(E)$ *dénote l'ensemble des éléments de E* $\prec$*-maximaux.*

Propriété 7.2. max-$\prec$ (doAdm) = max-$\prec$ (Adm).

PREUVE – Par définition, tout ensemble do-admissible est un ensemble admissible, c'est-à-dire doAdm $\subseteq$ Adm.

($\supseteq$) Montrons que max-$\prec$ (Adm) $\subseteq$ doAdm. Soit S un élément de max-$\prec$ (Adm), soit G l'ensemble S privé des éléments de S_o qui ne défendent individuellement aucun élément de S_d. Montrons que G est do-admissible, *i.e.* qu'il vérifie les conditions 1 et 2 de la définition 7.3.

1) G est sans-conflit car $G \subseteq S$ et S admissible donc sans-conflit. Soit $x \in G$ et $y \in A$ tels que yRx. Comme $G \subseteq S$, et S admissible, il existe $z \in S$ tel que zRy.
   - Si $x \in G_d$, alors, si $z \in S_d$, on a $z \in G_d$ car $S_d = G_d$. Si $z \in S_o$ alors, comme z défend un élément de S_d, z est par définition dans G_o. Donc $z \in G$.

- Si $x \in G_o$, alors, si $z \in S_d$, on a $z \in G_d$ car $S_d = G_d$. Si $z \in S_o$, alors, comme $x \in G_o$, c'est que x défend individuellement un élément s de S_d, donc, puisque z défend individuellement x, z défend également individuellement $s \in S_d$. Donc, par définition de G, $z \in G_o$. Donc $z \in G$.

Donc G défend tous ses éléments.

2) $\forall x \in G_o$, $\exists y \in G_d$ tel que x défenseur individuel de y est vérifié par définition de G.

G étant do-admissible, il est admissible. Comme $G_d = S_d$ et $G_o \subseteq S_o$, $S \prec G$. Comme $S \in$ max-$\prec$ (Adm), $G = S$, S est do-admissible et donc max-$\prec$ (Adm) $\subseteq$ doAdm.

Comme doAdm $\subseteq$ Adm, tout élément do-admissible de max-$\prec$ (Adm) est dans max-$\prec$ (doAdm). D'après le résultat précédent, on en conclut que max-$\prec$ (Adm) $\subseteq$ max-$\prec$ (doAdm).

($\subseteq$) Soit $S \in$ max-$\prec$ (doAdm) ; S est alors dans Adm. Soit G un ensemble admissible tel que $S \prec G$; alors $\exists Q \in$ max-$\prec$ (Adm) tel que $G \prec Q$ puisque toute chaîne dans (Adm, $\prec$) admet un majorant. Alors $S \prec Q$ car $\prec$ est une relation transitive, et max-$\prec$ (Adm) est inclus dans doAdm, donc, comme $S \in$ max-$\prec$ (doAdm), $S = Q$ et donc $S = G$: cela montre que $S \in$ max-$\prec$ (Adm). □

Un ensemble do-admissible $\prec$-maximal étant, par définition, une min-def extension, cette propriété implique immédiatement le résultat suivant :

Corollaire 7.1. *Soit $S \subseteq F$. S est une min-def extension de $\mathcal{SA}_{F_{do}}$ si et seulement si S est $\prec$-maximal parmi les ensembles admissibles de $\mathcal{SA}_{F_{do}}$.*

7.4.3 Existence

Avant de donner la propriété qui garantit l'existence des min-def extensions, nous introduisons deux notations et nous rappelons quelques définitions sur les ordres partiels, ainsi que le lemme de Kuratowski-Zorn.

Notation 7.3. *Étant donné un système d'argumentation $\mathcal{SA} = (A, R)$, on dénote par* Adm_X *l'ensemble des parties admissibles de $\mathcal{SA}$ incluses dans un ensemble $X \subseteq A$.*

Définition 7.6 (Ordre partiel, ensemble partiellement ordonné).
Soit X un ensemble et R une relation binaire sur X. R est un ordre partiel *si et seulement si R est réflexive, antisymétrique et transitive. Si R est un ordre partiel, le couple (X, R) est un* ensemble partiellement ordonné *(un* p.o.s.*).*

Soit (X, R) un p.o.s. et $Y \subseteq X$. Y est une chaîne *si et seulement si $\forall x, y \in Y$, (xRy) ou (yRx). Soit $m \in X$, m est un* majorant *(resp.* minorant*) de Y si et seulement si $\forall y \in Y$, yRm (resp. mRy). m est* maximal *(resp.* minimal*) si et seulement si $\nexists x \in X$, $x \neq m$ tel que mRx (resp. xRm).*

Lemme 7.1 (Kuratowski-Zorn). *Soit (X, Z) un p.o.s. tel que $X \neq \emptyset$ et toute chaîne de X admet un majorant (resp. minorant). Alors (X, Z) admet un élément maximal (resp. minimal), et $\forall y \in X$, $\exists x$ élément maximal (resp. minimal) tel que yZx (resp. xZy).*

La propriété suivante permet de déduire que tout système d'argumentation `type-2-`partitionné dont l'ensemble des arguments est fini, possède au moins une min-def extension.

Propriété 7.3. *Si $F_d \cup F_o \cup K$ est un ensemble fini, alors pour tout ensemble admissible G de $\mathcal{SA}_{F_{do}}$ inclus dans F, il existe une min-def extension S de $\mathcal{SA}_{F_{do}}$ telle que $G \prec S$.*

PREUVE – Adm_F désigne l'ensemble des ensembles admissibles de $\mathcal{SA}_{F_{do}}$ inclus dans F. D'après la propriété 7.1 page 112, le couple $(\text{Adm}_F, \prec)$ est un p.o.s.. L'ensemble $\emptyset$ étant un ensemble admissible inclus dans F, $\text{Adm}_F \neq \emptyset$. Montrons que toute chaîne de $(\text{Adm}_F, \prec)$ admet un majorant dans Adm_F. Soit

$$X_1 \prec X_2 \prec \ldots \prec X_n \ldots$$

une chaîne de $(\mathrm{Adm}_F, \prec)$: $\forall i, X_i \subseteq F$. Comme on a supposé que l'ensemble des arguments est fini, F_d et F_o sont finis. Donc $\exists i$ tel que $\forall n \geq i$, $(X_n)_d = (X_{n+1})_d$. Soit i_0 le plus petit de ces i. Alors :

$$(X_{i_0-1})_d \subset (X_{i_0})_d = \ldots = (X_n)_d = \ldots.$$

Comme $X_{i_0} \prec X_{i_0+1} \prec \ldots X_n$, on a donc :

$$(X_{i_0})_o \supseteq (X_{i_0+1})_o \supseteq \ldots (X_n)_o \supseteq \ldots.$$

Comme F_o est fini, on ne peut pas avoir une suite infinie strictement décroissante. Il existe donc $j \geq i_0$ tel que $\forall n \geq j, (X_n)_o = (X_{n+1})_o$. Soit j_0 le plus petit de ces j. Alors :

$$(X_{j_0-1})_o \supset (X_{j_0})_o = (X_{j_0+1})_o = \ldots.$$

Posons $M = X_{j_0}$. $M \in \mathrm{Adm}_F$. Montrons que M est un majorant des $(X_i)_{i\geq 1}$. On a $j_0 \geq i_0$. De plus :

$$\begin{array}{cc} X_1 \prec X_{j_0}, X_2 \prec X_{j_0}, \ldots, X_{i_0-1} \prec X_{j_0}, & \textit{puis} \\ (X_{i_0})_d = (X_{j_0})_d, \ldots, (X_{j_0-1})_d = (X_{j_0})_d & \textit{et} \\ (X_{j_0-1})_o \supset (X_{j_0})_o, et (X_{i_0})_o \supseteq \ldots \supseteq (X_{j_0-1})_o. & \end{array}$$

Donc :

$$X_{i_0} \prec X_{j_0}, \ldots, X_{j_0-1} \prec X_{j_0}.$$

Puis, pour $n > j_0$, $(X_n)_d = (X_{j_0})_d$, $(X_n)_o = (X_{j_0})_o$. Donc $X_{j_0} \prec X_n$ et $X_n \prec X_{j_0}$. X_{j_0} est donc bien un majorant des X_i.

Toute chaîne de $(\mathrm{Adm}_F, \prec)$ a donc un majorant dans Adm_F. D'après le lemme de Kuratowski-Zorn, toute partie de Adm_F contient un élément $\prec$-maximal.

Soit maintenant G un ensemble admissible inclus dans F, et soit $\mathcal{G}$ l'ensemble des parties admissibles de F qui sont $\prec$-meilleures que G, alors $\mathcal{G}$ a un élément maximal P : c'est une partie admissible de $\mathcal{G}$, qui est $\prec$-meilleure que G. Pour montrer que c'est une min-def extension, il reste à montrer que P est $\prec$-meilleur que toute autre partie admissible de F. Or, si $T \in \mathrm{Adm}_F$ tel que $P \prec T$, alors $G \prec T$ car $G \prec P$, donc $T \in \mathcal{G}$. Comme P est maximal dans $\mathcal{G}$, $P = T$. □

Si $F_d \cup F_o \cup K$ n'est pas fini, la propriété ci-dessus est fausse. Voici un contre-exemple :

Exemple 7.2. Soit le système d'argumentation `type-2-`partitionné $\mathcal{SA}_7$ tel que $F_d = \{d_i \mid i \geq 1\}$, $F_o = \{o_i \mid i \geq 1\}$, $K = \{k\}$ et $R = \{(k, d_1), (o_i, k), (d_{i+1}, o_i), (o_i, d_{i+1}) \mid i \geq 1\}$.

Le graphe de $\mathcal{SA}_7$ est représenté partiellement sur la figure 7.1.

Soit $Y_i = \{d_j \mid 1 \leq j \leq i\} \cup \{o_i\}$ pour tout $i \geq 1$, et soit $Y = \{Y_i \mid i \geq 1\}$.

Ainsi, les premiers Y_i sont :

$Y_1 = \{d_1, o_1\}$
$Y_2 = \{d_1, d_2, o_2\}$
$Y_3 = \{d_1, d_2, d_3, o_3\} \ldots$

Alors, $(2^{F_d \cup F_o}, \prec)$ étant un p.o.s. :

- Y est une chaîne : pour tout $i \geq 1$, $Y_i \prec Y_{i+1}$ car $Y_{i_d} \subset Y_{i+1_d}$;
- pour tout $i \geq 1$, Y_i est sans-conflit ;
- pour tout $i \geq 1$, Y_i défend tous ses éléments :
  - o_i défend d_1 contre k ;
  - o_i se défend lui-même contre d_{i+1} ;
  - pour tout $1 < j \leq i$, d_j se défend contre o_{j-1}.

Donc Y est une chaîne de parties admissibles de F, qui n'est pas $\prec$-majorée par une partie admissible de F. ♢

Corollaire 7.2. *Soit $\mathcal{SA}_{F_{do}}$ un système d'argumentation* `type-2-`*partitionné tel que $F_d \cup F_o \cup K$ est fini. Alors $\mathcal{SA}_{F_{do}}$ possède au moins une min-def extension.*

Preuve – Conséquence de la propriété 7.3, l'ensemble $\emptyset$ étant toujours admissible. □

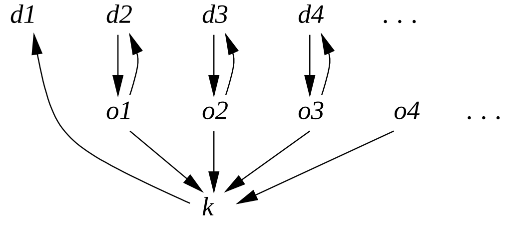


FIG. 7.1 – Système d'argumentation $\mathcal{SA}_7$.

7.4.4 Remarques sur l'inclusion des ensembles do-admissibles dans les min-def extensions

Nous avons vu que tout ensemble do-admissible n'est pas nécessairement inclus dans une min-def extension (voir exemple 7.1 (Min-def extensions) page 113). Ceci est dû au fait que les arguments optionnels d'un ensemble do-admissible ne sont pas tous des défenseurs essentiels. Cependant, lorsque l'ensemble des arguments du système est fini, la partie désirable d'un ensemble do-admissible est toujours incluse dans au moins une min-def extension :

Propriété 7.4. *Si $\mathcal{SA}_{F_{do}}$ est tel que $F_d \cup F_o \cup K$ est fini, la partie désirable d'un ensemble do-admissible de $\mathcal{SA}_{F_{do}}$ est incluse dans au moins une min-def extension de $\mathcal{SA}_{F_{do}}$.*

PREUVE – Soit E un ensemble do-admissible de $\mathcal{SA}_{F_{do}}$. Par définition, E est un ensemble admissible inclus dans F. D'après la propriété 7.3, il existe une min-def extension S de $\mathcal{SA}_{F_{do}}$ telle que $E \prec S$. Donc $E_d \subset S_d$, ou $E_d = S_d$ et $E_o \supsetneq S_o$. Dans les deux cas, E_d est inclus dans S. □

Concernant la partie optionnelle d'un ensemble do-admissible, nous montrons que lorsqu'elle n'est pas «minimale», l'ensemble do-admissible ne peut pas être une min-def extension.

Définition 7.7 (Ensemble do-admissible minimal sur F_o).
Un ensemble do-admissible *S est* minimal sur *F_o si et seulement si il vérifie la condition suivante : $\forall x \in S_o, S \setminus \{x\}$ ne défend pas tous ses éléments.*

Propriété 7.5. *Les ensembles do-admissibles non minimaux sur F_o ne sont pas des min-def extensions de $\mathcal{SA}_{F_{do}}$.*

PREUVE – Soit S un ensemble do-admissible de $\mathcal{SA}_{F_{do}}$ non minimal sur F_o. Alors $\exists x \in S_o$ tel que $S \setminus \{x\}$ défend tous ses éléments. Soit $S' = S \setminus \{x\}$. S' est sans-conflit car inclus dans S qui est sans-conflit, S' défend tous ses éléments par définition, et comme $S'_d = S_d$, $S'_o \subseteq S_o$ et S est do-admissible, tout élément de S'_o défend un élément de S'_d. Donc S' est do-admissible. De plus, comme $S'_d = S_d$ et $S'_o \subseteq S_o$, $S \prec S'$. S ne peut donc être $\prec$-maximal. □

EXEMPLE 7.3. Les ensembles $\{d_1, o_1, o_2\}$ et $\{d_1, o_2, o_3\}$ sont des ensembles do-admissibles minimaux sur F_o du système d'argumentation $\mathcal{SA}_8$ indiqué sur la figure 7.2. Par contre, $S = \{d_1, o_1, o_2, o_3\}$ n'est pas un ensemble do-admissible minimal sur F_o de $\mathcal{SA}_8$ car $S \setminus \{o_1\}$ défend tous ses éléments. S n'est pas une min-def extension. ♢

Il faut bien noter que même un ensemble do-admissible qui est minimal sur F_o n'est pas nécessairement inclus dans une min-def extension. Voici un contre-exemple : $\{d_1, o_1, o_2\}$ est un ensemble do-admissible minimal sur F_o de $\mathcal{SA}_8$, mais sa partie optionnelle n'est pas incluse dans la min-def extension de $\mathcal{SA}_8$ (l'ensemble $\{d_1, d_2, d_3, d_4, o_2, o_3\}$).

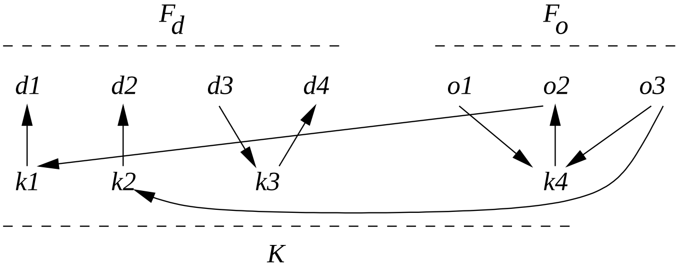


FIG. 7.2 – Système d'argumentation $\mathcal{SA}_8$.

7.5 Synthèse

Le schéma de la figure 7.3 montre les inclusions entre les ensembles d'arguments acceptables sous les différentes sémantiques d'un système d'argumentation `type-2-`partitionné. Ces inclusions découlent des définitions des ensembles admissibles, des extensions préférées, des ensembles do-admissibles et des min-def extensions. Dans le chapitre 8, nous verrons que lorsque $F_o = \emptyset$ et $K = \emptyset$, les extensions préférées coïncident avec les min-def extensions.

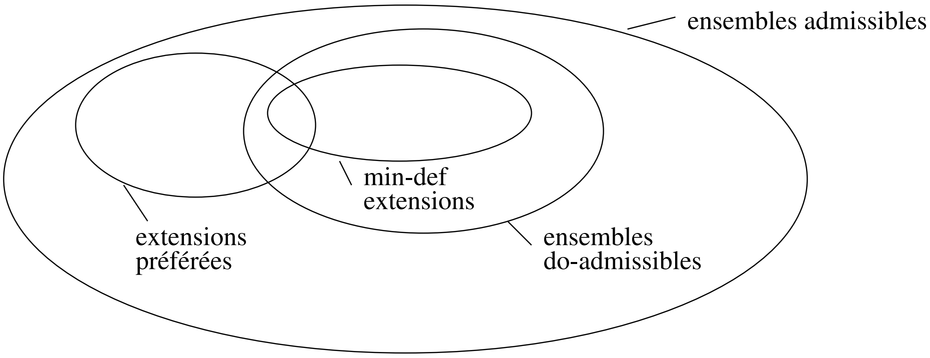


FIG. 7.3 – Relations entre les extensions préférées, les min-def extensions, les ensembles admissibles et les ensembles do-admissibles.

7.6 Autres illustrations et applications

Voici deux autres illustrations du cadre formel défini dans la section précédente. La première, comme l'exemple du procès de la Mafia, est un exemple dans un contexte dialectique ; la deuxième montre que le système peut être appliqué dans d'autres contextes.

EXEMPLE 7.4 (UNE DISCUSSION ENTRE AMIS). Denis et Théo discutent. Denis a un point de vue et donne des arguments pour le soutenir. Denis a une amie, Olivia, qui partage son point de vue. Théo n'est pas d'accord avec le point de vue de Denis et Olivia et donne des arguments qui attaquent leurs arguments. Dans cette discussion, Denis peut être vu comme le *partisan* et Théo comme l'*opposant*. Denis a plusieurs solutions pour défendre ses arguments contre les attaques de Théo :

1. soit Denis utilise seulement ses arguments ;
2. soit Denis utilise ses arguments et ceux d'Olivia, en considérant que les arguments d'Olivia ont la même importance que les siens, *i.e.* il utilise indifféremment ses arguments ou ceux d'Olivia ;

3. soit Denis utilise ses arguments et ceux d'Olivia, mais veut privilégier ses propres arguments : il ne prend en compte les arguments d'Olivia que s'ils lui permettent de défendre certains de ses arguments qu'il ne peut défendre lui-même.

Les différentes manières dont Denis accepte les arguments d'Olivia sont justifiées par le fait que les arguments d'Olivia peuvent contenir, par exemple, des informations très personnelles sur Denis. Par conséquent, on peut imaginer que si Denis ne veut pas réellement que ces informations soient révélées, il accepte les arguments d'Olivia seulement si c'est absolument nécessaire (cas 3), ou il ne les accepte pas du tout (cas 1). Par contre, si Denis est d'accord pour que ces informations soient révélées, il accepte les arguments d'Olivia en ne faisant aucune distinction avec les siens (cas 2).

Les arguments et les attaques entre arguments formalisant cette discussion constituent un système `type-2`-partitionné où les arguments de Denis sont les arguments désirables, ceux d'Olivia sont les arguments optionnels et ceux de Théo sont les arguments restants.

Supposons que les ensembles d'arguments acceptables supportant le point de vue de Denis sont des ensembles sans-conflit, qui défendent tous leurs arguments contre les attaques de Théo et qui contiennent un maximum d'arguments de Denis. Ainsi, dans le cas 1, ces ensembles sont les extensions préférées sur les arguments de Denis ; dans le cas 2, ce sont des extensions préférées sur les arguments de Denis et Olivia ; dans le cas 3, ce sont les min-def extensions. ♢

Dans un système multi-agents, les agents ont généralement besoin de communiquer pour accomplir des tâches ou pour partager des ressources, et pour cela, ils dialoguent. Les travaux de [PSJ98], [Ree98] et [AMP00] mettent en évidence que l'argumentation peut être utilisée comme base pour dialoguer. [AMP00] étudie différents types de dialogues en argumentation, et mentionne les stratégies qu'un agent utilise pour choisir les arguments pendant un dialogue. Ce point doit être étudié en profondeur, mais cet exemple suggère que notre travail pourrait être utilisé comme modèle pour prendre en compte les stratégies. Les arguments désirables d'un agent seraient ceux qui satisfont la stratégie, les arguments optionnels seraient les autres arguments de l'agent.

L'illustration suivante montre que notre travail peut être appliqué à des contextes non dialectiques, à un problème de configuration par exemple.

Exemple 7.5 (Configuration d'un avion militaire). L'armée souhaite configurer ses avions militaires pour qu'ils puissent se défendre contre toute attaque ennemie durant la prochaine mission. Parmi les fonctions qui peuvent être installées sur un avion, certaines sont désirables voire même indispensables, comme les fonctions de vol, de propulsion et de communication ; d'autres sont optionnelles, comme les brouilleurs de radar. Un maximum de fonctions désirables doivent être installées sur l'avion, alors qu'un minimum de fonctions optionnelles doivent être prises en compte, pour des raisons de coût, de vitesse et de poids de l'avion. Les ennemis possèdent des matériels qui peuvent constituer des menaces pour certaines fonctions ; par exemple, leur brouilleur de vol menace la fonction de vol.

Ce problème de configuration peut être aisément modélisé dans un système d'argumentation `type-2`-partitionné : les fonctions font référence aux arguments de la partie F d'une partition de A, les fonctions désirables font référence aux arguments de F_d, les fonctions optionnelles aux arguments de F_o, les menaces aux arguments de $A \setminus F$ et les conflits aux éléments de la relation R.

Une bonne configuration pour un avion est un ensemble C de fonctions tel que :

1. C est sans-conflit ;
2. si une fonction de C est menacée, alors C contient une fonction qui attaque la menace ;
3. C contient un maximum de fonctions souhaitables et un minimum de fonctions optionnelles.

Le parallèle établi avec un système d'argumentation `type-2`-partitionné montre qu'une bonne configuration est un ensemble inclus dans F qui est sans-conflit (1) et qui défend tous ses éléments (2) ; par conséquent, c'est un ensemble admissible inclus dans F. Cet ensemble contient un maximum d'arguments désirables et un minimum d'arguments optionnels (3) ; cet ensemble admissible est donc $\prec$-maximal. Par conséquent, d'après le corollaire 7.1, une bonne configuration est une min-def extension. $\diamondsuit$

Chapitre 8

Méthode de calcul des min-def extensions

Dans ce chapitre, nous allons nous intéresser au calcul de la sémantique défense minimale. Pour effectuer ce calcul, nous allons porter notre attention sur des ensembles do-admissibles particuliers, les extensions préférées incluses dans $F = F_d \cup F_o$.

Nous allons tout d'abord définir la notion d'*extension préférée sur un ensemble*, puis nous allons établir des propriétés qui soulignent ses liens avec les ensembles do-admissibles et les min-def extensions.

À partir de ces propriétés, nous donnerons une méthode générale de calcul des min-def extensions, et nous indiquerons quelques cas particuliers où le calcul est simplifié.

8.1 Extensions préférées sur un ensemble

8.1.1 Définition et propriétés

Soit $\mathcal{SA} = (A, R)$ un système d'argumentation. Étant donné un ensemble $X \subseteq A$, une extension préférée sur X est un ensemble admissible qui doit être maximal pour l'inclusion, non plus sur A, mais sur X. Formellement :

Définition 8.1 (Extension préférée sur un ensemble).
Soit $X \subseteq A$. Une extension préférée de $\mathcal{SA}$ sur X *est un ensemble $S \subseteq X$ tel que S est maximal pour l'inclusion (ensembliste) sur X parmi les ensembles admissibles inclus dans X, i.e. S est admissible et $\forall Y \subseteq (X \setminus S)$ tel que Y est non vide, $Y \cup S$ n'est pas admissible.*

EXEMPLE 7.1 (EXTENSION PRÉFÉRÉE SUR UN ENSEMBLE). Considérons

$$X = \{d_1, d_2, d_3, d_4, d_5, o_1, o_2, o_3, o_4\}$$

(X correspond à l'ensemble F du système $\mathcal{SA}6_F$). $\mathcal{SA}6$, $\mathcal{SA}6_F$ et $\mathcal{SA}6_{F_{do}}$ possèdent une extension préférée sur X :

$$\{d_2, d_3, d_4, d_5, o_1, o_2, o_3\}.$$

♢

À noter :

- Une extension préférée de $\mathcal{SA}$ sur A au sens de la définition 8.1 est une extension préférée de $\mathcal{SA}$ au sens de la définition 2.5 page 13.
- Une extension préférée de $\mathcal{SA}$ sur X n'est pas simplement l'intersection de X et d'une extension préférée de $\mathcal{SA}$. Considérons par exemple le système *Zoé l'autruche* (exemple 2.1 page 11) et l'ensemble $X = \{a\}$; l'extension préférée du système est $\{c, a\}$, son intersection avec X est $\{a\}$ et sa seule extension préférée sur X est $\emptyset$.

- Une extension préférée sur X est également différente d'une extension préférée sur la restriction de $\mathcal{SA}$ à X, que l'on peut définir ainsi :

Définition 8.2 (Restriction d'un système d'argumentation).

- *La restriction d'un système d'argumentation (A, R) à un ensemble $E \subseteq A$ est le système d'argumentation (E, R') avec $R' = \{(x, y) \in R | x \in E \text{ et } y \in E\}$; en d'autres termes, c'est le système d'argumentation correspondant au sous-graphe induit par E.*
- *La restriction d'un système `type-1`-partitionné (X, Y, R) à un ensemble $E \subseteq (X \cup Y)$ est le système $(X \cap E, Y \cap E, R')$ avec $R' = \{(x, y) \in R | x \in E \text{ et } y \in E\}$.*
- *La restriction d'un système `type-2`-partitionné (X, Y, Z, R) à un ensemble $E \subseteq (X \cup Y \cup Z)$ est le système $(X \cap E, Y \cap E, Z \cap E, R')$ avec $R' = \{(x, y) \in R | x \in E \text{ et } y \in E\}$.*

La restriction du système d'argumentation *Zoé l'autruche* à $X = \{a\}$ correspond au système $(X, \emptyset)$; l'extension préférée de ce système est l'ensemble $\{a\}$ alors que la seule extension préférée sur X du système *Zoé l'autruche* est $\emptyset$!

Propriété 8.1. *Soit un système d'argumentation $\mathcal{SA} = (A, R)$ et un ensemble $X \subseteq A$.*

1. *Tout ensemble admissible inclus dans X est inclus dans une extension préférée de $\mathcal{SA}$ sur X.*
2. *Toute extension préférée de $\mathcal{SA}$ sur X est incluse dans une extension préférée de $\mathcal{SA}$.*
3. *Soit $\mathcal{SA}'$ la restriction de $\mathcal{SA}$ à X. Si $\mathcal{SA}'$ est un système d'argumentation bien-fondé, alors $\mathcal{SA}$ possède une et une seule extension préférée sur X.*

PREUVE (PROPRIÉTÉ 8.1) –

1. Soit $\mathcal{Y}$ une chaîne de Adm_X et soit $Z = \bigcup_{Y \in \mathcal{Y}} Y$. Alors $Z \subseteq X$.

   Montrons que Z est sans conflit. Supposons qu'il existe $a, b \in Z$ tels que aRb. Alors il existe Y_a et Y_b dans $\mathcal{Y}$ tels que $a \in Y_a$ et $b \in Y_b$. Si $Y_a \subseteq Y_b$, $a \in Y_b$ et Y_b contient un conflit. Sinon, $Y_b \subseteq Y_a$, $b \in Y_a$ et Y_a contient un conflit. Or Y_a et Y_b sont admissibles. C'est donc que Z ne contient pas de conflit.

   Montrons que Z défend tous ses éléments. Soit $a \in Z$, il existe $Y_a \in \mathcal{Y}$ tel que $a \in Y_a$. Soit $b \in A$ tel que bRa, alors, comme Y_a est admissible, il existe $c \in Y_a$ tel que cRb, et comme $Y_a \subseteq Z$, $c \in Z$. Donc Z défend a.

   Toute chaîne de $(\mathrm{Adm}_X, \subseteq)$ a donc un majorant dans Adm_X. D'après le lemme de Kuratowski-Zorn, toute partie de Adm_X contient un élément $\subseteq$-maximal.

   Soit maintenant S un ensemble admissible inclus dans X, et soit V l'ensemble des parties admissibles de X qui contiennent S, alors V a un élément maximal P : c'est une partie admissible de V, qui contient S. Pour montrer que c'est une extension préférée sur X, il reste à montrer que P n'est strictement inclus dans aucune autre partie admissible de X. Or, si $T \in \mathrm{Adm}_X$ tel que $P \subseteq T$, alors $S \subseteq T$, donc $T \in V$. Comme P est maximal dans V, $P = T$.
2. Par définition, une extension préférée sur X est un ensemble admissible, inclus dans X. Sachant que tout ensemble admissible est inclus dans une extension préférée, donc toute extension préférée sur X est incluse dans une extension préférée.
3. Soit S une extension préférée sur X. Supposons qu'il existe un ensemble admissible $Y \subseteq X$ tel que $Y \not\subseteq S$ (i.e. $Y \cap (X \backslash S) \neq \emptyset$). Comme S est $\subseteq$-maximal sur X et $Y' = Y \cap (X \backslash S) \neq \emptyset$, $S \cup Y'$ n'est pas sans conflit (i) ou ne défend pas tous ses éléments (ii).

   (i) Comme S est une extension préférée, S est sans conflit. Comme $Y' \subseteq Y$ et Y est admissible, Y' est sans conflit. Supposons qu'il existe un conflit entre S et Y'. Ce conflit peut être de deux types :

   (a) soit $\exists s \in S$, $\exists y \in Y'$ tels que sRy.

   i. si $\nexists y' \in Y'$ tel que $y'Rs$, alors, comme $Y' \subseteq Y$ et Y défend tous ses éléments, $\exists y'' \in (Y \setminus Y')$ tel que $y''Rs$. Or, $(Y \setminus Y') \subseteq S$, donc $y'' \in S$ et S ne serait pas alors sans-conflit. Contradiction avec le fait que S est une extension préférée.

    ii. si $\exists y' \in Y'$ tel que $y'Rs$, alors, comme S défend tous ses éléments, $\exists s'' \in S$ tel que $s''Ry'$... On retombe sur un conflit du type 0a. Ceci implique qu'il existerait une séquence infinie $a_0, a_1, \dots, a_n, \dots$ telle que $\forall i, a_i \in X$ et $a_{i+1}Ra_i$. Impossible par hypothèse.

  (b) soit $\exists y \in Y'$, $\exists s \in S$ tels que yRs. Comme S défend tous ses éléments, $\exists s' \in S$ tel que $s'Ry$. On retrouve un conflit du type 0a.

  Il n'existe donc pas de conflit entre S et Y'. $S \cup Y'$ est sans conflit.

(ii) Puisque S est une extension préférée, S défend tous ses éléments. Puisque Y est admissible, $Y' \cup (S \cap Y) = Y$ défend tous ses éléments. Donc $S \cup Y' \cup (S \cap Y) = S \cup Y'$ défend tous ses éléments.

Puisque (i) et (ii) sont faux, l'hypothèse d'un ensemble admissible Y non inclus dans S est impossible. S étant un ensemble admissible maximal sur X, S est la seule extension préférée sur X.

□

La sémantique préférée sur un ensemble est plus particulièrement appropriée à un système d'argumentation `type-1`-partitionné. En effet, nous focalisons notre attention sur un sous-ensemble X d'arguments de A, autrement dit, nous travaillons implicitement sur le système $\mathcal{SA}_X = (X, A \setminus X, R)$. Une extension préférée sur X du système $\mathcal{SA}_X$ se définit donc comme une extension préférée sur X du système classique correspondant (A, R).

Plus généralement, on peut dire que, étant donné un système d'argumentation `type-1`-partitionné (resp. `type-2`-partitionné) $\mathcal{SA}_F = (F, K, R)$ (resp. $\mathcal{SA}_{F_{do}} = (F_d, F_o, K, R)$), les extensions préférées sur un ensemble $X \subseteq (F \cup K)$ (resp. sur $X \subseteq (F_d \cup F_o \cup K)$) sont les extensions préférées sur X du système classique correspondant.

8.1.2 Lien avec les ensembles do-admissibles

La propriété suivante fait le lien entre les extensions préférées sur F_d et les ensembles do-admissibles :

Propriété 8.2. *Étant donné $\mathcal{SA}_{F_{do}}$ un système `type-2`-partitionné, toute extension préférée de $\mathcal{SA}_{F_{do}}$ sur F_d est un ensemble do-admissible.*

> PREUVE – Soit X une extension préférée sur F_d. Par définition, X est un ensemble admissible inclus dans F_d. Il vérifie donc la condition 1 de la définition 7.3. De plus, comme $X \cap F_o = \emptyset$, la condition 2 est vérifiée. X est donc un ensemble do-admissible. □

Par contre, toute extension préférée de $\mathcal{SA}_{F_{do}}$ sur $F_d \cup F_o$ n'est pas forcément do-admissible. Voici un contre-exemple : l'ensemble $S = \{d_2, d_3, d_4, d_5, o_1, o_2, o_3\}$ est l'extension préférée sur $F_d \cup F_o$ de $\mathcal{SA}6_{F_{do}}$ mais n'est pas un ensemble do-admissible car o_3 ne défend aucun argument désirable de S.

8.1.3 Lien avec les min-def extensions

Nous allons établir deux propriétés importantes qui permettent de faire le lien entre les extensions préférées sur F et les min-def extensions. Grâce à ces propriétés, nous donnerons dans le chapitre suivant une méthode de calcul des min-def extensions.

Considérons $\mathcal{SA}_{F_{do}} = (F_d, f_o, K, R)$ un système d'argumentation `type-2`-partitionné et $F = F_d \cup F_o$. La propriété ci-dessous permet de déduire que toute min-def extension de $\mathcal{SA}_{F_{do}}$ est $\prec$-meilleure qu'une extension préférée de $\mathcal{SA}_{F_{do}}$ sur F.

Propriété 8.3. *Soit $G \subseteq F$. Si G est un ensemble admissible $\prec$-maximal de $\mathcal{SA}_{F_{do}}$ alors il existe E extension préférée de $\mathcal{SA}_{F_{do}}$ sur F telle que $E_d = G_d$ et $E_o \supseteq G_o$ et donc $E \prec G$.*

La preuve de cette propriété repose sur le lemme suivant :

Lemme 8.1. *Si* $G \in \text{max-}\prec(\text{Adm})$ *et si* E *est admissible et contient* G*, alors* $G_d = E_d$*.*

Preuve – Puisque $G \subseteq E$, $G_d \subseteq E_d$. Si $G_d \subset E_d$ alors $G \prec E$ et par $\prec$-maximalité de G, $E = G$. Donc $G_d = E_d$. □

Preuve (propriété 8.3) – On peut trouver E admissible maximal pour l'inclusion qui contient G d'après le point 1 de la propriété 8.1 page 122, et d'après le lemme 8.1, $G_d = E_d$. Puisque de plus $G_o \subseteq E_o$, on a $E \prec G$. □

Propriété 8.4. *Si* S *est une min-def extension de* $\mathcal{SA}_{F_{do}}$*, alors il existe* E *extension préférée de* $\mathcal{SA}_{F_{do}}$ *sur* F *telle que* $E_d = S_d$ *et* $E_o \supseteq S_o$ *et donc* $E \prec S$*.*

Preuve – Conséquence du corollaire 7.1 et de la propriété 8.3. □

La propriété suivante permet de déduire que la partie désirable d'une extension préférée sur F est incluse dans au moins une min-def extension :

Propriété 8.5. *Soit* $F_d \cup F_o \cup K$ *fini et soit* E *une extension préférée sur* F*. Alors il existe* S *min-def extension telle que* $E_d \subseteq S_d$*.*

Preuve – La preuve repose sur le corollaire 7.1 et la propriété 7.3. □

Corollaire 8.1. *Toute extension préférée sur* F_d *est incluse dans une min-def extension.*

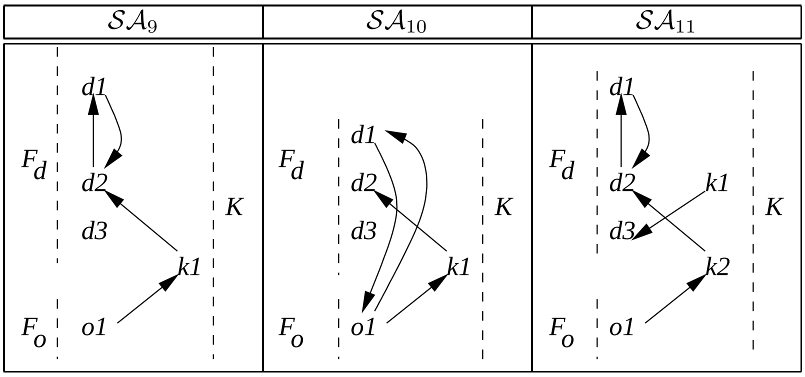
Preuve – Conséquence de la propriété 8.5 puisqu'une extension préférée E sur F_d est telle que $E = E_d$. □

On peut noter que l'extension préférée sur F de $\mathcal{SA}6_{F_{do}}$ de l'exemple page 121 est telle que sa partie désirable $\{d_2, d_3, d_4, o_1\}$ est égale à la partie désirable de la min-def extension de $\mathcal{SA}6_{F_{do}}$ et que sa partie optionnelle $\{o_1, o_2, o_3\}$ contient la partie optionnelle $\{o_2\}$ de la min-def extension de $\mathcal{SA}6_{F_{do}}$.

Nous pouvons faire deux remarques sur l'inclusion des extensions préférées sur F_d dans les min-def extensions :

- Si le système d'argumentation réduit à F n'est pas bien-fondé, toute min-def extension de $\mathcal{SA}_{F_{do}}$ ne contient pas nécessairement une extension préférée de $\mathcal{SA}_{F_{do}}$ sur F_d.
  Les systèmes d'argumentation $\mathcal{SA}_9$ et $\mathcal{SA}_{10}$ indiqués dans le tableau 8.1 page ci-contre en sont deux contre-exemples. Ces deux systèmes possèdent une seule extension préférée sur F_d : $\{d_1, d_3\}$, et deux min-def extensions : $S_1 = \{d_1, d_3\}$ et $S_2 = \{d_2, d_3, o1\}$. Dans aucun des deux systèmes, la min-def extension S_2 ne contient l'extension préférée sur F_d.
- Toute min-def extension de $\mathcal{SA}_{F_{do}}$ ne contient pas nécessairement une extension préférée sur la restriction de $\mathcal{SA}_{F_{do}}$ à F_d.
  Le système d'argumentation $\mathcal{SA}_{11}$ indiqué dans le tableau 8.1 page suivante en est un contre-exemple, puisqu'il possède deux min-def extensions : $\{d_1\}$ et $\{d_2, o_1\}$. La restriction de $\mathcal{SA}_{11}$ à F_d est le système $\mathcal{SA}'_{11} = (F_d, \emptyset, \emptyset, R')$ tel que $R' = \{(d_1, d_2), (d_2, d_1)\}$. Ce système possède deux extensions préférées : $\{d_1, d_3\}$ et $\{d_2, d_3\}$. Ces deux extensions ne sont pas incluses dans les min-def extensions de $\mathcal{SA}_{11}$.

On peut noter également qu'il a été établi dans [CDLSM02a] que toute min-def extension S contient une extension préférée sur S_d.



Tab. 8.1 – Systèmes d'argumentation $\mathcal{SA}_9$, $\mathcal{SA}_{10}$ et $\mathcal{SA}_{11}$.

8.2 Calcul des min-def extensions

D'après la propriété 8.4 page ci-contre, il est clair que chaque min-def extension peut être obtenue à partir d'une des extensions préférées sur F, en minimisant sa partie optionnelle. Il résulte de la propriété 8.5 page précédente qu'il est suffisant de considérer les extensions préférées sur F dont la partie désirable est maximale pour l'inclusion ensembliste.

L'algorithme de calcul des extensions préférées peut donc être adapté pour calculer les min-def extensions en utilisant les propriétés supplémentaires suivantes : nous nous intéressons aux sous-ensembles de F qui sont des extensions préférées sur F et dont la partie désirable est maximale pour l'inclusion ensembliste.

Nous proposons donc une méthode en deux étapes pour calculer les min-def extensions :

Algorithme 19 Calcul des min-def extensions

Entrée: un système d'argumentation `type-2-`partitionné $\mathcal{SA}_{F_{do}}$ tel que $F_d \cup F_o \cup K$ est fini
Sortie: les min-def extensions de $\mathcal{SA}_{F_{do}}$

```
1: Calculer les extensions préférées $X_1 \ldots X_p$ de $\mathcal{SA}_{F_{do}}$ sur $F_d \cup F_o$ telles que $\forall i$, $1 \leq i \leq p$, $X_{i_d}$ est $\subseteq$-maximal ;
2: À partir de chacun des $X_i$ précédents, calculer les ensembles admissibles en enlevant autant d'éléments de $X_{i_o}$ que possible.
```

Pour réaliser l'étape 1, une idée est d'adapter l'algorithme pour le calcul des extensions préférées que nous avons établi dans le chapitre 6 pour le calcul des extensions préférées sur un ensemble. Des ensembles retournés par l'algorithme d'énumération des extensions préférées, il restera à supprimer ceux qui ne seraient pas maximaux sur X_{i_d}.

À l'étape 2, on enlèvera en particulier les défenseurs non essentiels.

Cette méthode est correcte et complète, c'est-à-dire qu'à l'issue de l'étape 2, on n'obtient que des min-def extensions et on les obtient toutes.

Propriété 8.6. *L'algorithme 19 de calcul des min-def extensions est correct.*

Preuve – Soit $X_1, \ldots, X_p$ les extensions préférées sur F telles que X_{i_d} est $\subseteq$-maximal.

Soit S un ensemble admissible obtenu à partir de l'un des X_i, $1 \leq i \leq p$, en minimisant sa partie optionnelle. On a $S_d = X_{i_d}$, $S_o \subseteq X_{i_o}$ et $\nexists S'$ ensemble admissible tel que $S'_d = X_{i_d}$ et $S'_o \subset S_o$.

Montrons que S est une min-def extension. S étant par définition un ensemble admissible, d'après le corollaire 7.1, il suffit de montrer que S est $\prec$-maximal parmi les ensembles admissibles de $\mathcal{SA}_{F_{do}}$.

Supposons que S ne soit pas $\prec$-maximal. D'après la propriété 7.3, il existe S' min-def extension de $\mathcal{SA}_{F_{do}}$ telle que $S \prec S'$, *i.e.* telle que : (i) $S_d = S'_d$ et $S'_o \subset S_o$, ou (ii) $S'_d \subset S_d$.

Si (i), alors S' est un ensemble admissible tel que $S'_d = S_d = X_{i_d}$ et $S'_o \subset S_o$, ce qui est impossible par définition de S.

S' étant une min-def extension, d'après la propriété 8.4, il existe E extension préférée sur F telle que $E_d = S'_d$ et $S'_o \subseteq E_o$. Or, si (ii), $X_{i_d} = S_d$ et $S_d \subset S'_d$: contradiction avec le fait que X_i est une extension préférée sur F avec une partie désirable maximale.

Puisque (i) et (ii) sont impossibles, c'est que S est $\prec$-maximale. La correction est prouvée. □

Propriété 8.7. *L'algorithme 19 de calcul des min-def extensions est complet.*

Preuve – Soit S une min-def extension. Montrons que la méthode 19 retourne un ensemble T tel que $T = S$.

D'après la propriété 8.4, il existe E extension préférée sur F telle que $E_d = S_d$ et $S_o \subseteq E_o$.

Supposons que E ne soit pas l'un des X_i, $1 \leq i \leq p$, c'est-à-dire que E_d ne soit pas $\subseteq$-maximal. Il existe alors E' extension préférée sur F telle que $E_d \subset E'_d$. D'après la propriété 8.5, il existe donc S' min-def extension telle que $E'_d \subseteq S'_d$. Donc $E_d \subset S'_d$. Or $E_d = S_d$, donc $S_d \subset S'_d$ et ainsi $S \prec S'$: contradiction avec le fait que S est $\prec$-maximal.

Par conséquent, $\exists i, 1 \leq i \leq p$, tel que $E = X_i$. Donc $S_d = X_{i_d}$ et $S_o \subseteq X_{i_o}$. Montrons que S et obtenu à partir de X_i en enlevant autant d'éléments de X_{i_o} que possible.

Supposons qu'il existe T ensemble admissible tel que $T_d = X_{i_d}$ et $T_o \subset S_o$. Alors $T_d = S_d$ et $T_o \subset S_o$, et donc $S \prec T$: contradiction avec le fait que S est $\prec$-maximal.

S est donc bien un ensemble obtenu à partir de X_i en minimisant X_{i_o}. La complétude est prouvée. □

Exemple 8.1. Calculons les min-def extensions du système d'argumentation $\mathcal{SA}_{11}$ indiqué dans le tableau 8.1 page précédente.

1. Les extensions préférées sur F de $\mathcal{SA}_{11}$ sont :
   - $X_1 = \{d_2, o_1\}$
   - $X_2 = \{d_1, o_1\}$
2. Minimisation de X_{1_o} et X_{2_o} :
   - o_1 est un défenseur essentiel de d_2, il doit donc rester dans X_{1_o}. X_1 est une min-def extension de $\mathcal{SA}_{11}$.
   - o_1 ne défend aucun élément de X_{2_o}, il doit donc être enlevé de X_{2_d}. L'ensemble $X'_2 = \{d_1\}$ est la deuxième min-def extension de $\mathcal{SA}_{11}$.

◇

8.3 Cas particuliers

Dans plusieurs cas, le calcul des min-def extensions est simplifié :

1. Si le système d'argumentation réduit à F est bien-fondé :
   il n'existe qu'une extension préférée X sur F (point 3 de la propriété 8.1 page 122). L'étape 1 est donc réduite au calcul de cette extension préférée.

   Exemple 7.1 (suite). Le système $\mathcal{SA}_{6_{F_{do}}}$ réduit à F est bien-fondé et ne possède qu'une seule min-def extension.

   L'étape 1 du calcul de la min-def extension est réduite au calcul de l'extension préférée sur F, soit $X = \{d_2, d_3, d_4, d_5, o_1, o_2, o_3\}$.

   La partie optionnelle de X est $X_o = \{o_1, o_2, o_3\}$. Minimisons cet ensemble (étape 2). L'argument o_3 ne défend aucun argument de la partie désirable de X, donc on enlève o_3 de X. Tous les autres arguments de X_o sont des défenseurs essentiels, on les garde donc tous dans X.

   L'étape 2 est terminée. $\mathcal{SA}_{6_{F_{do}}}$ possède une min-def extension : $\{d_2, d_3, d_4, d_5, o_1, o_2\}$. ◇

2. Si $F_o = \emptyset$:
   D'après la propriété ci-dessous, les extensions préférées sur F sont les min-def extensions. Le calcul des min-def extensions est donc réduit à l'étape 1.

   Propriété 8.8. *Soit $S \subseteq F$. Si $F_o = \emptyset$, S est une min-def extension de $\mathcal{SA}_{F_{do}}$ si et seulement si S est une extension préférée de $\mathcal{SA}_{F_{do}}$ sur F_d.*

   Preuve –
   - Soit S une min-def extension. Par définition, et puisque $F_o = \emptyset$, S est un ensemble admissible inclus dans F_d. De plus, comme S est $\prec$-maximal et $F_o = \emptyset$, S est $\subseteq$-maximal sur F_d. Donc S est une extension préférée sur F_d.
   - Soit S une extension préférée sur F_d. D'après la propriété 8.2, S est un ensemble do-admissible. S est $\subseteq$-maximal sur F_d, et comme $S_o = \emptyset$, S est $\prec$-maximal. S est donc une min-def extension.

   □

3. Si $F_o = \emptyset$ et $K = \emptyset$:
   Les min-def extensions de $\mathcal{SA}_{F_{do}}$ correspondent aux extensions préférées de $\mathcal{SA}_{F_{do}}$ (puisque l'ensemble des arguments est réduit à F_d). Il suffit alors d'utiliser l'algorithme de calcul des extensions préférées pour calculer les min-def extensions.

Conclusion et perspectives

Le travail de ce mémoire est basé sur le système d'argumentation de Dung [Dun95]. Dans ce système, nous avons plus particulièrement porté notre attention sur la sémantique préférée de Dung [Dun95].

Le cadre *unificateur* du système de Dung nous a permis de définir des algorithmes très généraux qui s'appliquent non seulement à la sémantique préférée, mais aussi à d'autres sémantiques définies aussi bien pour l'argumentation que pour d'autres mécanismes utilisés pour formaliser le raisonnement de sens commun. Le système de Dung s'est par ailleurs avéré suffisamment *souple* pour permettre la définition d'une nouvelle sémantique qui prend en compte la notion d'argument restreint.

Nous faisons ici un bilan des apports de cette thèse, du point de vue théories de la preuve et algorithmes tout d'abord, puis du point de vue raffinement. Nous indiquons également des perspectives de recherche futures.

Théories de la preuve et algorithmes

Nous avons proposé deux théories de la preuve pour le problème d'acceptabilité crédule sous la sémantique préférée en utilisant le modèle du cadre formel de [JV99a]. Nous avons également traduit dans le modèle de [JV99a] les théories de la preuve pour le problème d'acceptabilité sous la sémantique basique proposées par [AC02] et [PS97]. Pour le problème d'acceptabilité sceptique sous la sémantique préférée, nous avons montré comment les théories de la preuve précédemment établies peuvent être utilisées.

Nous avons ensuite établi un algorithme de calcul d'une des deux théories de la preuve pour le problème d'acceptabilité crédule sous la sémantique préférée, ainsi qu'un algorithme pour le problème d'acceptabilité sceptique qui utilise l'algorithme précédent ; cet algorithme est correct, mais il n'est pas complet.

Nous avons également établi des algorithmes qui répondent aux problèmes d'acceptabilité crédule et sceptique sous la sémantique préférée en se contentant de répondre *vrai* ou *faux* selon que l'argument de la requête est accepté ou non, sans retourner de preuve. L'algorithme pour le problème d'acceptabilité sceptique est cette fois-ci non seulement correct, mais aussi complet.

Le tableau suivant résume les apports de cette thèse vis-à-vis des problèmes d'acceptabilité crédule et sceptique sous la sémantique préférée ; le symbole • (resp. ◦) dénote que la théorie ou l'algorithme proposé est correct et complet (resp. correct mais non complet).

Sémantique préférée	Problème d'acceptabilité	
	crédule	sceptique
Théorie de la preuve	•	◦
Algorithme de calcul de preuves	•	◦
Algorithme de réponse *vrai* ou *faux*	•	•

Concernant le calcul des extensions sous une sémantique donnée d'un système d'argumentation, nous avons tout d'abord établi un algorithme de calcul des ensembles sans-conflit, puis des ensembles admissibles d'un système d'argumentation. À partir de ces algorithmes, nous avons proposé un algorithme très général qui, instancié, permet de calculer les extensions préférées et les extensions stables d'un système d'argumentation. L'algorithme pour le calcul des extensions préférées est un des premiers du genre, alors que celui pour le calcul des extensions stables s'ajoute à une liste d'algorithmes existants ; la caractéristique principale de l'algorithme que nous proposons est d'utiliser les heuristiques de sélection d'un argument proposées pour le calcul des extensions préférées.

Pour les algorithmes de calcul des extensions préférées et de réponse aux problèmes d'acceptabilité crédule et sceptique présentés dans le chapitre 6, une version préliminaire (celle présentée dans [DM01b]) a été implémentée en langage C et testée avec succès sur des exemples simples.

Pour répondre au problème d'acceptabilité sceptique sous la sémantique préférée, un algorithme correct et complet qui utilise une des théories de la preuve pour le problème d'acceptabilité crédule est en cours de validation ([CDM02]).

À court terme, nous envisageons d'implémenter la version finale de ces algorithmes, ainsi que les algorithmes pour le calcul de preuves dialectiques présentés dans ce mémoire. Nous envisageons également de faire une étude de complexité des algorithmes établis.

À moyen terme, nous souhaitons établir un algorithme de calcul des stratégies gagnantes. En effet, les ϕ_2-preuves pour le problème d'acceptabilité crédule sous la sémantique préférée, ainsi que les ϕ_{db}-preuves pour le problème d'acceptabilité sous la sémantique basique (selon [AC02] et [PS97]), sont des formes particulières de stratégies gagnantes, et il n'existe pour le moment aucun algorithme qui les calcule.

À long terme, nous envisageons d'appliquer ces algorithmes dans le cadre de la modélisation de l'interaction dans les systèmes multi-agents.

Raffinement de la sémantique préférée

Dans le système d'argumentation défini par Dung [Dun95], lorsqu'un argument donné est attaqué, les arguments succeptibles de le défendre sont tous également désirables. Or, nous avons mis en évidence des cas où une distinction entre différents types d'arguments est nécessaire.

Pour cette raison, en utilisant une partition de l'ensemble des arguments, nous avons proposé deux raffinements du système d'argumentation de Dung (les systèmes `type-1-` et `type-2-`partitionnés). Le but de cette partition est de privilégier un ensemble d'arguments et, de plus, de faire une distinction entre arguments désirables (ou non restreints) et arguments optionnels (ou restreints). Dans ce contexte, nous avons défini deux nouvelles sémantiques et les ensembles acceptables qui leur sont associés :

- les *ensembles do-admissibles* sont des ensembles admissibles d'arguments dans lesquels la présence des arguments optionnels est justifiée (un argument optionnel appartient à un ensemble do-admissible seulement s'il défend un argument désirable) ;
- les *min-def extensions* sont des ensembles do-admissibles qui contiennent un maximum d'arguments désirables et un minimum d'arguments optionnels.

Nous avons montré différentes propriétés qui permettent de faire le lien entre ces nouvelles sémantiques, l'admissibilité et la sémantique préférée, ce qui nous a permis d'établir dans le chapitre 8 une méthode de calcul des min-def extensions.

Le tableau suivant fait une synthèse sur différentes sémantiques que nous avons étudiées dans ce mémoire ; nous indiquons pour quel type de système d'argumentation elles sont définies et plus particulièrement appropriées ($*$), et si nous avons établi un algorithme ($\bullet$) ou une méthode permettant de les calculer ($\circ$).

Sémantique	Système d'argumentation		
	classique	`type-1`-partitionné	`type-2`-partitionné
admissible	$*\bullet$		
do-admissible			$*$
préférée	$*\bullet$		
préférée sur un ensemble		$*$	
défense minimale			$*\circ$

À moyen terme, nous souhaitons donner un algorithme de calcul des min-def extensions basé sur la méthode décrite dans le chapitre 8. Pour cela, nous établirons tout d'abord un algorithme de calcul des extensions préférées sur un ensemble à partir de l'algorithme de calcul des extensions préférées.

Nous souhaitons également établir des théories de la preuve dialectiques pour déterminer l'appartenance d'un argument à une ou à toute extension préférée (problèmes d'acceptabilité crédule et sceptique sous la sémantique min-def).

À long terme, nous envisageons une extension de de la sémantique défense minimale au cas où l'ensemble des arguments optionnels serait stratifié. Ceci est motivé par plusieurs exemples :

- Nous avons vu que les arguments optionnels d'un agent peuvent être ceux qui contiennent des informations personnelles. Ces informations peuvent être plus ou moins privées, et on préférera révéler en priorité les arguments qui contiennent les informations les moins privées.
- Parmi les fonctions optionnelles qui peuvent être installées sur un avion, certaines peuvent être plus coûteuses que d'autres ; on préférera choisir en priorité les moins coûteuses.

Par conséquent, nous voulons étendre un système d'argumentation `type-2`-partitionné avec une relation de priorité sur les arguments optionnels. Nous intégrerons cette relation de priorité dans la définition de la relation $\prec$ et dans la définition de la sémantique min-def, dans le but de choisir les arguments optionnels qui ont la plus grande priorité (*i.e.* ceux qui contiennent les fonctions les moins coûteuses ou les informations les moins personnelles) pour défendre les arguments désirables.

Autres perspectives de recherche

Systèmes d'argumentation dynamiques Dans tout ce mémoire, nous sommes partis de l'hypothèse que l'ensemble des arguments et des éléments de la relation de contrariété du système d'argumentation de Dung étaient connus *a priori*. Or, dans bien des applications, cela n'est pas le cas (dans le cadre de dialogues entre agents par exemple). On parle alors de systèmes d'argumentation *dynamiques*.

Des travaux récents ont cherché à définir des cadres formels de théories de la preuve pour de tels systèmes ([Pra01] par exemple). L'adaptation de nos théories de la preuve et algorithmes pour ces systèmes dynamiques est à étudier.

Lien avec la logique classique Nous souhaitons faire un lien entre la sémantique préférée pour le système d'argumentation de Dung et la logique classique. Pour cela, nous avons dans l'idée d'associer un ensemble de clauses à un système d'argumentation, de telle sorte que les modèles maximaux de l'ensemble de clauses permettent de retrouver les extensions préférées du système.

Un lien similaire a déjà été fait par Ben-Eliyahu et Dechter dans [BED96] entre le raisonnement par défaut et la logique classique. Pour saisir la sémantique stable dans le système d'argumentation de Dung, Dunne et Bench-Capon suggèrent dans [DBC02] d'associer à un système d'argumentation (A, R) une base de clauses telle que :

- les atomes présents dans les clauses sont les éléments de A ;
- si $(x, y) \in R$ alors $\neg x \vee \neg y$ est une clause de la base (ce type de clause permet de saisir la condition de sans-conflit des extensions stables) ;

- pour tout $x \in A$, $x \vee \bigvee_{y:(y,x)\in R} y$ est une clause de la base (ce type de clause permet de garantir que tout argument soit appartient à l'extension stable, soit est attaqué par un argument qui lui appartient).

Les modèles maximaux de la base de clauses correspondent aux extensions stables du système d'argumentation : si un atome a la valeur *vrai* (resp. *faux*) dans un modèle, alors l'argument correspondant à l'atome appartient (resp. n'appartient pas) à l'extension stable correspondante au modèle.

Nous souhaitons de la même manière définir une base de clauses dont les modèles maximaux permettront de retrouver les extensions préférées du système d'argumentation correspondant.

L'intérêt de cette modélisation est qu'elle nous permettra d'utiliser les algorithmes existants pour le calcul de modèles maximaux et le problème de satisfiabilité d'une base de clauses, pour améliorer nos algorithmes.

Bibliographie

[AA97] Aleven (V.) et Ashley (K.D.). – Evaluating a Learning Environment for Case-Based Argumentation Skills. *In : Proceedings of the Sixth International Conference on Artificial Intelligence and Law, ICAIL-97.* pp. 170–179. – New York, 1997.

[AC98] Amgoud (L.) et Cayrol (C.). – On the acceptability of arguments in preference-based argumentation. *In : 14th Conference on Uncertainty in Artificial Intelligence (UAI'98), Madison, Wisconsin.* pp. 1–7. – San Francisco, California, juillet 1998.

[AC02] Amgoud (L.) et Cayrol (C.). – A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, vol. 34, 2002, pp. 197–215.

[Amg99] Amgoud (L.). – *Contribution à l'intégration des préférences dans le raisonnement argumentatif.* – Thèse de doctorat, Université Paul Sabatier, Toulouse, juillet 1999.

[AMP00] Amgoud (L.), Maudet (N.) et Parsons (S.). – Modelling dialogues using argumentation. *In : Fourth International Conference on MultiAgent Systems (ICMAS'2000), Boston, MA, USA*, pp. 31–38.

[BC98] Bench-Capon (T.J.M.). – Specification and Implementation of Toulmin Dialogue Game. *In : Proceedings of JURIX 98*, pp. 5–20.

[BDKT97] Bondarenko (A.), Dung (P.M.), Kowalski (R.A.) et Toni (F.). – An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, vol. 93, 1997, pp. 63–101.

[BED96] Ben-Eliyahu (R.) et Dechter (R.). – Default reasoning using classical logic. *Artificial Intelligence*, vol. 84, 1996, pp. 113–150.

[Ber73] Berge (C.). – *Graphs and Hypergraphs.* – North-Holland, North-Holland Mathematical Library, 1973.

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

[CDLSM02a] Cayrol (C.), Doutre (S.), Lagasquie-Schiex (M.C.) et Mengin (J.). – *Défense minimale : un raffinement de la sémantique préférée pour les systèmes d'argumentation.* – Rapport de recherche n° 2002-03-R, France, Institut de Recherche en Informatique de Toulouse (IRIT), février 2002.

[CDLSM02b] Cayrol (C.), Doutre (S.), Lagasquie-Shiex (M.-C.) et Mengin (J.). – "Minimal defence" : a refinement of the preferred semantics for argumentation frameworks. *In : Proc. 9th International Workshop on Non-Monotonic Reasoning (NMR'2002)*, éd. par Benferhat (S.) et Giunchiglia (E.), pp. 408–415. – Toulouse, France, avril 2002.

[CDM01a] Cayrol (C.), Doutre (S.) et Mengin (J.). – *Dialectical Proof Theories for the Credulous Preferred Semantics of Argumentation Frameworks.* – Rapport technique n° 2001-09-R, IRIT, mai 2001.

[CDM01b] Cayrol (C.), Doutre (S.) et Mengin (J.). – Dialectical Proof Theories for the Credulous Preferred Semantics of Argumentation Frameworks. *In : ECSQARU 2001.* pp. 668–679. – Springer-Verlag.

[CDM02] Cayrol (C.), Doutre (S.) et Mengin (J.). – On Decision Problems related to the preferred semantics for argumentation frameworks. *Journal of Logic and Computation*, 2002. – À paraître.

[CL91] Chaty (G.) et Levy (F.). – *Default logic and kernel in digraph.* – Rapport technique n° 91-9, Avenue J.B. Clément, 93430 Villetaneuse, LIPN, Université Paris Nord, 1991.

[CLS02] Cayrol (C.) et Lagasquie-Schiex (M.-C.). – Gradual handling of contradiction in argumentation frameworks. *In : Proc. of the Ninth International Conference, Information Processing and Management of Uncertainty in Knowledge-based Systems(IPMU 2002)*, pp. 83–90. – Annecy, juillet 2002.

[DBC01a] Dunne (P.E.) et Bench-Capon (T.J.M.). – *Coherence in Finite Argument Systems.* – Rapport technique n° ULCS-01-006, University of Liverpool, Dept. of Computer Science, décembre 2001.

[DBC01b] Dunne (P.E.) et Bench-Capon (T.J.M.). – *Complexity and Combinatorial Properties of Argument Systems.* – Rapport technique, University of Liverpool, Dept. of Computer Science, juillet 2001.

[DBC01c] Dunne (P.E.) et Bench-Capon (T.J.M.). – *Two Party Immediate Response Disputes : Properties and Efficiency.* – Rapport technique n° ULCS-01-005, University of Liverpool, Dept. of Computer Science, octobre 2001.

[DBC02] Dunne (P.E.) et Bench-Capon (T.J.M.). – Coherence in Finite Argument Systems. *Artificial Intelligence*, vol. 141, n° 1–2, octobre 2002, pp. 187–203.

[DKT96] Dung (P.M.), Kowalski (R.A.) et Toni (F.). – Synthesis of Proof Procedures for Default Reasoning. *In : Proceedings of the International Workshop on Logic Program Synthesis and Transformation (LOPSTR'96).*

[DM94] Dimopoulos (Y.) et Magirou (V.). – A graph-theoretic approach to default logic. *Information and Computation*, vol. 112, n° 2, août 1994, pp. 239–256.

[DM00] Doutre (S.) et Mengin (J.). – An algorithm that computes the preferred extensions of argumentation frameworks. *In : ECAI'2000, Third International Workshop on Computational Dialectics (CD'2000)*, pp. 55–62.

[DM01a] Doutre (S.) et Mengin (J.). – Preferred Extensions of Argumentation Frameworks : Computation and Query Answering. *In : IJCAR 2001*, éd. par Goré (R.), Leitsch (A.) et Nipkow (T.). pp. 272–288. – Springer-Verlag.

[DM01b] Doutre (S.) et Mengin (J.). – *Preferred Extensions of Argumentation Frameworks : Computation and Query Answering.* – Rapport technique n° 2001-03-R, IRIT, mars 2001.

[DMP97] Dimopoulos (Y.), Magirou (V.) et Papadimitriou (C.H.). – On kernels, Defaults and Even Graphs. *Annals of Mathematics and AI*, 1997, pp. 1–12.

[DMT01] Dung (P. M.), Mancarella (P.) et Toni (F.). – *Argumentation-based proof procedures for credulous and sceptical non-monotonic reasoning.* – Rapport technique, Department of Computing, Imperial College, London, 2001.

[DNT99] Dimopoulos (Y.), Nebel (B.) et Toni (F.). – Preferred Arguments are Harder to Compute than Stable Extensions. *In : Proc. IJCAI'99*, pp. 36–41.

[DNT00] Dimopoulos (Y.), Nebel (B.) et Toni (F.). – Finding Admissible and Preferred Arguments Can be Very Hard. *In : Proc. KR'200*, pp. 53–61.

[DNT02] Dimopoulos (Y.), Nebel (B.) et Toni (F.). – On the Computational Complexity of Assumption-based Argumentation for Default Reasoning. *Artificial Intelligence*, vol. 141, n° 1–2, octobre 2002, pp. 57–78.

[Dou99] Doutre (S.). – *Prise en compte de Préférences Conditionnelles dans les Systèmes d'Argumentation.* – Rapport de DEA, 1999.

[Dou00] Doutre (S.). – Un algorithme de calcul des extensions préférées dans les systèmes d'argumentation. *In : Actes des Rencontres Jeunes Chercheurs en Intelligence Artificielle (RJCIA'2000)*, éd. par Ayel (M.) et Fouet (J.-M.), pp. 103–112.

[DT96] Dimopoulos (Y.) et Torres (A.). – Graph theoretical structures in logic programs and default theories. *Theoretical Computer Science*, vol. 170, 1996, pp. 209–244.

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

[EGH95] Elvang-Goransson (M.) et Hunter (A.). – Argumentative logics : Reasoning with classically inconsistent information. *Data and Knowledge Engineering*, vol. 6, 1995, pp. 125–145.

[EGKF93] Elvang-Goransson (M.), Krause (P.) et Fox (J.). – Acceptability of arguments as 'logical uncertainty'. *In : Proc. ECSQARU'93*, pp. 85–90.

[EK89] Eshghi (K.) et Kowalski (R.A.). – Abduction compared with negation as failure. *In : Proc. ICLP'89.* – MIT Press.

[GCJ00] Grasso (F.), Cawsey (A.) et Jones (R.). – Dialectical Argumentation to Solve Conflicts in Advice Giving : a Case Study in the Promotion of Healthy Nutrition. *International Journal of Human-Computer Studies*, vol. 53, n° 6, 2000, pp. 1077–1115.

[GJ79] Garey (M.R.) et Johnson (D.S.). – *Computers and Intractability : a Guide to the Theory of NP-completeness.* – New York, W.H. Freeman and Company, 1979.

[GK97] Gordon (T.F.) et Karacapilidis (N.). – The Zeno Argumentation Framework. *In : Proc. ICAIL'97.* pp. 10–18. – Melbourne, Australia, 1997.

[GL88] Gelfond (M.) et Lifschitz (V.). – The stable model semantics for logic programming. *In : Proc. ICLP'88.* – MIT Press.

[JV96] Jakobovits (H.) et Vermeir (D.). – Contradiction in Argumentation Frameworks. *In : Proc. of the Conference on Information Processing and Management of Uncertainty (IPMU96)*, pp. 821–826.

[JV99a] Jakobovits (H.) et Vermeir (D.). – Dialectic Semantics for Argumentation Frameworks. *In : Proc. ICAIL'99.* pp. 53–62. – ACM Press.

[JV99b] Jakobovits (H.) et Vermeir (D.). – Robust Semantics for Argumentation Frameworks. *Journal of Logic and Computation*, vol. 9, n° 2, 1999, pp. 215–261.

[KAEGF95] Krause (P.), Ambler (S.), Elvang-Goransson (M.) et Fox (J.). – A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, vol. 11, n° 1, 1995, pp. 113–131.

[KMD94] Kakas (A.C.), Mancarella (P.) et Dung (P.M.). – The acceptability semantics for logic programs. *In : Proc. of the 11th International Conference on Logic Programming*, éd. par Van Hentenrijck (P.). pp. 504–519. – MIT Press.

[KT99] Kakas (A.C.) et Toni (F.). – Computing Argumentation in Logic Programming. *Journal of Logic and Computation*, vol. 9, n° 4, août 1999, pp. 515–562.

[LS89] Lin (F.) et Shoham (Y.). – Argument systems. A uniform basis for nonmonotonic reasoning. *In : Proc. First International Conference on Principles of Knowledge Representation and Reasoning.* pp. 245–255. – Morgan Kaufmann.

[McC80] McCarthy (John). – Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, vol. 13, 1980, pp. 27–39.

[McD82] McDermott (D.). – Nonmonotonic logic II : non-monotonic modal theories. *JACM*, vol. 29, n° 1, 1982.

[MM99] Maudet (N.) et Moore (D.). – Dialogue games for computer-supported collaborative argumentation. *In : Proc. of the Workshop on Computer-Supported Collaborative Argumentation for Learning Communities.* – Stanford, 1999.

[Moo85] Moore (R.). – Semantical considerations on non-monotonic logic. *Artificial Intelligence*, vol. 25, 1985, pp. 75–94.

[Nie95] Niemelä (I.). – Towards Efficient Default Reasoning. *In : Proceedings IJCAI'95*, pp. 312–318.

[Pol87] Pollock (J.L.). – Defeasible reasoning. *Cognitive Science*, vol. 11, 1987, pp. 481–518.

[Pol94] Pollock (J.L.). – Justification and defeat. *Artificial Intelligence*, vol. 67, 1994, pp. 377–407.

[Pol95] Pollock (J.L.). – *Cognitive Carpentry. A Blueprint for How to Build a Person.* – Cambridge, MIT Press, 1995.

[Poo88] Poole (D.). – A logical framework for default reasoning. *Artificial Intelligence*, vol. 36, 1988, pp. 27–47.

[Pra01] Prakken (H.). – Relating protocols for dynamic dispute with logics for defeasible argumentation. *Synthese*, vol. 127, 2001, pp. 187–219.

[Prz91] Przymusinski (T.). – Semantics of disjuntive logic programs and deductive databases. *In : Proceedings of the Conference on Deductive Object-Oriented Databases (DOOD'91)*, pp. 85–107.

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

[PSJ98] Parsons (S.), Sierra (C.) et Jennings (N. R.). – Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, vol. 8, n° 3, 1998, pp. 261–292.

[PV02] Prakken (H.) et Vreeswijk (G.). – *Handbook of Philosophical Logic*, chap. Logics for Defeasible Argumentation, pp. 219–318. – Dordrecht/Boston/London, Kluwer Academic Publishers, 2002, second édition.

[Ree98] Reed (C.). – Dialogue frames in agent communication. *In : Proceedings of the 3rd International Conference on Multi Agent Systems*, pp. 246–253.

[Rei80] Reiter (R.). – A logic for default reasoning. *Artificial Intelligence*, vol. 13, 1980, pp. 81–132.

[SL92] Simari (G.R.) et Loui (R.P.). – A mathematical treatment of defeasible argumentation and its implementation. *Artificial Intelligence*, vol. 53, 1992, pp. 125–157.

[SWCP95] Suthers (D.), Weiner (A.), Connelly (J.) et Paolucci (M.). – Belvedere : Engaging students in critical discussion of science and public policy issues. *In : AI-Ed 95, the 7th World Conference on Artificial Intelligence in Education*, pp. 266–273. – Washington DC, août 1995.

[SZ90] Saccà (D.) et Zaniolo (C.). – Stable models and non-determinism in logic programs with negation. *In : Proc. 9th Symposium on Principles of Database Systems*, pp. 205–217.

[Ver99] Verheij (H.B.). – Automated Argument Assistance for Lawyers. *In : Proc. of the Seventh International Conference on Artificial Intelligence and Law (ICAIL'99).* pp. 43–52. – New York, 1999.

[VGRS88] Van Gelder (A.), Ross (K.) et Schlipf (J.S.). – Unfounded sets and well-founded semantics for general logic programs. *In : Proc. 7th Symposium on Principles of Database Systems*, pp. 221–230.

[VP00] Vreeswijk (G.A.W.) et Prakken (H.). – Credulous and Sceptical Argument Games for Preferred Semantics. *In : Proc. JELIA'2000.* pp. 239–253. – LNAI 1919.

[Vre97] Vreeswijk (G.A.W.). – Abtract argumentation systems. *Artificial Intelligence*, vol. 90, 1997, pp. 225–279.

[WK95] Walton (D.N.) et Krabbe (E.C.W.). – *Commitment in dialogue : basic concepts of interpersonal reasoning.* – State University of New York Press, 1995.

Index

TITLE : On the preferred semantics for argumentation frameworks

---

ABSTRACT :

Argumentation is a mode of reasoning which consists in building and evaluating arguments, generally conflicting. The argumentation framework of Dung constitutes an adequate formal setting to study this mode of reasoning. Its abstract structure makes it possible to unify many other approaches suggested for argumentation and various formalisms used to model non-monotonic reasoning. Given a set of arguments and a binary relation between arguments representing a notion of attack, arguments are evaluated in order to distinguish the most acceptable. Among the semantics given to acceptability, we pay attention in this thesis to a simple and outstanding semantics, the preferred semantics introduced by Dung.

We propose sound and complete algorithms to compute the preferred semantics and to answer the credulous and the sceptical decision problems associated with this semantics. We also propose two dialectical proof theories for the credulous decision problem on the model of the formal framework of Jakobovits and Vermeir, as well as an algorithm to compute proofs in one of them, and we show how to use this algorithm to answer in particular cases of argumentation frameworks the sceptical decision problem.

We are also interested in a refinement of the preferred semantics allowing to take into account the fact that the use of certain arguments can have to be restricted to the case where they are the only ones to make acceptable other arguments. Given a partition of the argumentation framework of Dung authorizing a distinction between several types of arguments, we propose a new semantics called minimal defense.

KEYWORDS : argumentation, non-monotonic reasoning, dialectical proof theory, graph theory, algorithm.

AUTEUR : Sylvie DOUTRE

TITRE : Autour de la sémantique préférée des systèmes d'argumentation

DIRECTRICE DE THÈSE : Claudette CAYROL

DATE ET LIEU DE SOUTENANCE : 12 décembre 2002, IRIT (Université Toulouse III)

DISCIPLINE ADMINISTRATIVE : Informatique

RÉSUMÉ :

L'argumentation est un mode de raisonnement qui consiste à construire et à évaluer des arguments, le plus souvent conflictuels. Le système d'argumentation de Dung constitue un cadre formel adéquat pour étudier ce mode de raisonnement. Sa structure abstraite permet d'unifier de nombreuses autres approches proposées pour l'argumentation et pour divers formalismes permettant de modéliser le raisonnement non-monotone. À partir d'un ensemble d'arguments et d'une relation binaire entre arguments représentant une notion de contrariété, les arguments sont évalués afin de distinguer les plus acceptables. Parmi les sémantiques données à l'acceptabilité, nous portons notre attention dans cette thèse à une sémantique à la fois simple et remarquable, la sémantique préférée introduite par Dung.

Nous proposons des algorithmes corrects et complets pour calculer la sémantique préférée et pour répondre aux problèmes de décision crédule et sceptique associés à cette sémantique. Nous proposons également deux théories de la preuve dialectiques pour le problème de décision crédule sur le modèle du cadre formel de Jakobovits et Vermeir ainsi qu'un algorithme permettant de calculer des preuves dans l'une d'elles, et nous montrons comment utiliser ces preuves pour répondre dans des cas particuliers de systèmes d'argumentation au problème de décision sceptique.

Nous nous intéressons également à un raffinement de la sémantique préférée permettant de prendre en compte le fait que l'utilisation de certains arguments peut devoir être restreinte au cas où ils sont les seuls à rendre acceptables d'autres arguments. À partir d'une partition du système de Dung autorisant une distinction entre plusieurs types d'arguments, une nouvelle sémantique, appelée défense minimale, est proposée.

MOTS-CLÉS : argumentation, raisonnement non-monotone, théorie de la preuve dialectique, théorie des graphes, algorithmes.

INTITULÉ ET ADRESSE DU LABORATOIRE :

Institut de Recherche en Informatique de Toulouse
UMR 5505 CNRS – INP – UPS
Université Paul Sabatier
118, route de Narbonne
31062 Toulouse cedex 4