Chapitre 3
Représentation compacte

Nous avons introduit dans le chapitre 1 une modélisation formelle des principaux éléments du problème de partage. Si nous avons défini les concepts de base en termes mathématiques, en revanche, nous ne nous sommes pas préoccupé de la manière dont une instance réelle devait être représentée. Ce problème est crucial notamment en ce qui concerne la représentation des préférences des agents : une description explicite des préférences de chaque agent sur un espace d'alternatives $\mathscr{E}$ requiert soit la donnée de $|\mathscr{E}|^2$ comparaisons entre alternatives si l'on s'intéresse à des préférences ordinales, soit la donnée de $|\mathscr{E}|$ utilités si l'on s'intéresse à des préférences cardinales. Dans de nombreux problèmes réels, une telle description explicite est rédhibitoire, à cause de l'aspect combinatoire de l'espace des alternatives.

Il convient de préciser ce que l'on entend par «combinatoire». Comme nous allons le préciser dans ce chapitre, un espace combinatoire désignera un espace défini par un produit cartésien de n domaines finis de taille m : un tel espace est de taille m^n. Avoir affaire à ce genre d'espaces d'alternatives est courant en théorie de la décision : c'est concrètement le cas lorsque chaque alternative est définie par l'attribution d'une valeur particulière à un certain nombre de variables de décision, ce qui est précisément le cas en particulier dans les problèmes de partage de biens indivisibles, comme nous allons le voir.

Illustrons le phénomène d'explosion combinatoire sur deux exemples. L'exemple 3.2 nous servira de base pour illustrer certains langages de représentation compacte de préférences introduits dans ce chapitre.

Exemple 3.1 (Le repas) L'exemple du repas est un exemple classique illustrant le phénomène d'explosion combinatoire [Lang, 2004, 2006]. Considérons un agent devant exprimer ses préférences au sujet d'un repas qui se compose d'un apéritif, d'une entrée, d'un plat principal, d'un fromage, d'un dessert et d'un vin. Considérons pour simplifier qu'il y a 6 choix possibles pour chaque composante du repas (parmi lesquels un choix «vide» si l'agent désire ne rien prendre pour une composante donnée du repas). L'expression des préférences de l'agent ne pose aucun problème si celles-ci sont indépendante sur chacune des composantes du repas (par exemple si le choix sur le plat principal n'influe pas sur le choix sur la boisson) : dans ce cas, les préférences de l'agent peuvent être exprimées sous la forme de 6 structures de préférences indépendantes, ce qui ne fait au final que 36 «informations» à éliciter, si toutefois les préordres sont totaux. Cependant, face à un tel choix, un agent voudra certainement exprimer de manière naturelle des dépendances entre les variables : «si le plat principal est un poisson, je préfère un vin blanc à un vin rouge, sinon je préfère un vin rouge à un vin blanc», «s'il y a du fromage, je ne prends pas de dessert» ou encore «je n'aime pas le vin blanc sec en apéritif, mais s'il y a du foie gras en entrée je ferai une exception». Dans ce cas, l'agent devra exprimer une relation de préférence sur l'ensemble des repas différents possibles, soit

$6^6 = 46\,656$ alternatives.

Exemple 3.2 (Héritage) Tous les problèmes de partage impliquant des biens indivisibles font apparaître de manière naturelle une structure combinatoire dès qu'il y a des dépendances préférentielles entre les objets, c'est-à-dire si certains objets sont complémentaires (avoir un ensemble de deux objets apporte une plus-value à l'agent par rapport à l'obtention des deux objets séparés) ou substituables (l'obtention de deux objets ensemble n'apporte rien à l'agent, car l'un de ces objets suffit à sa satisfaction).

Considérons par exemple un problème concernant le partage d'un héritage entre plusieurs successeurs[1]. L'héritage contient des biens immobiliers (maison, terres cultivables, parcelles boisées, ...), des véhicules (automobile, tracteur, tondeuse à gazon, ...), et un certain nombre de biens mobiliers. Si le nombre de biens à partager est m, il y a donc 2^m partages possibles (pour $m = 10$, $2^m = 1\,024$).

Tout comme dans l'exemple du restaurant, ce nombre d'alternatives est dissuasif pour l'élicitation exhaustive des préférences, sauf si les objets sont «additivement indépendants» pour les agents, c'est-à-dire si la valeur attribuée à un couple d'objets (x, y) est égale à la somme des valeurs attribuées à chaque objet séparément. Dans ce cas, il suffit d'attribuer une utilité à chaque objet. Cependant, dans la pratique, il existe des dépendances préférentielles entre les objets : par exemple, un agent qui hérite des terres cultivables mais pas de la maison n'aura que faire de la tondeuse à gazon, mais il voudra à tout prix hériter du tracteur. Un agent qui hérite d'une cuisinière électrique ne voudra peut-être pas d'un lave-vaisselle en plus s'il habite un appartement minuscule, alors que le lave-vaisselle seul l'aurait intéressé.

Cette explosion combinatoire pose deux problèmes majeurs :

- ▷ celui de l'implantation logicielle des instances réelles, la taille de l'espace requis dépassant très rapidement la taille de l'espace mémoire disponible sur toute machine raisonnable ;
- ▷ celui de la représentation explicite de l'espace des alternatives et de l'élicitation des préférences sur cet espace, sachant que seul un être surnaturel (et doté d'une espérance de vie hors du commun) aura la faculté d'énumérer l'ensemble des alternatives ou des couples d'alternatives entre lesquels il a le choix.

Afin de pallier ces deux problèmes liés à l'explosion combinatoire, on introduit classiquement des *langages de représentation compacte*. Le rôle de ces langages est simplement de permettre une description concise d'un espace combinatoire ou d'un ensemble de structures de préférences (ordinales ou cardinales) sur un tel espace combinatoire.

Ce chapitre est organisé comme suit. Nous allons dans un premier temps nous intéresser aux moyens de représenter l'espace des alternatives en spécifiant les contraintes de manière compacte, puis nous tenterons de dresser un aperçu aussi exhaustif que possible des langages de représentation compacte qui ont été étudiés dans le domaine de l'expression des préférences, ce qui nous donnera l'occasion de discuter de la pertinence de ces langages pour le domaine des problèmes de partage. Nous nous intéresserons enfin à la représentation compacte des problèmes de partage, abordée sous deux angles différents : celui de critères purement ordinaux comme la Pareto-efficacité et l'absence d'envie appliqués à des préférences dichotomiques représentées sous forme logique, et celui de critères numériques fondés sur le *welfarisme* cardinal et une représentation logique des préférences et des contraintes.

[1] Comme nous l'avons vu au chapitre 1, il s'agit d'un problème classique dans la littérature économique sur les problèmes de partage.

3.1 Représentation compacte de l'espace des alternatives

3.1.1 Cadre formel

3.1.1.1 Espace d'alternatives combinatoires

La première question concernant la problématique de l'explosion combinatoire est liée à la représentation de l'espace des alternatives admissibles lui-même, qui est un espace combinatoire. Cependant, nous nous devons avant toute chose de préciser formellement ce que l'on entend par «espace combinatoire».

Comme nous l'avons fait remarquer en début de chapitre, l'aspect «combinatoire» d'un espace d'alternatives provient du fait qu'il est défini comme un produit cartésien de domaines finis. Nous pouvons donc définir de manière informelle un espace combinatoire comme étant un sous-ensemble en bijection avec un produit cartésien de domaines finis : dans ce contexte, chaque alternative admissible de l'espace combinatoire en question correspond à un tuple du produit cartésien des domaines. Nous prenons cependant le parti d'adopter une définition légèrement différente, et de centrer notre étude des espaces combinatoires sur la notion de *variable de décision* et d'*instanciation* :

Définition 3.1 (Variables, domaines, instanciation) *Soient* $\mathscr{X} = \{\mathbf{x_i}, \ldots, \mathbf{x_n}\}$ *un ensemble fini de* variable de décision*, et* $\mathcal{D}$ *une application de domaine qui à chaque variable* $\mathbf{x_i}$ *associe un ensemble fini* $\mathscr{D}_{\mathbf{x_i}}$ *appelé le* domaine *de* $\mathbf{x_i}$*. On appelle* instanciation *sur* $(\mathscr{X}, \mathcal{D})$ *toute application* $v_{\mathscr{X},\mathcal{D}}$ *qui à toute variable* $\mathbf{x_i} \in \mathscr{X}$ *associe une valeur* $v_{\mathscr{X}}(\mathbf{x_i})$ *de son domaine* $\mathscr{D}_{\mathbf{x_i}} = \mathcal{D}(\mathbf{x_i})$*.*

Étant donnés un ensemble de variables $\mathscr{X}$ *et l'ensemble de leurs domaines défini par une application* $\mathcal{D}$*, on notera* $Inst(\mathscr{X}, \mathcal{D})$ *l'ensemble des instanciations sur* $(\mathscr{X}, \mathcal{D})$*.*

Par la suite, nous ferons quelques simplifications de notations. Nous noterons $Inst(\mathscr{X})$ au lieu de $Inst(\mathscr{X}, \mathcal{D})$ car $\mathcal{D}$ sera toujours sous-entendu. De plus, nous noterons $v_{\mathscr{X}}$, voire v au lieu de $v_{\mathscr{X},\mathcal{D}}$, lorsque ces notations ne prêtent pas à confusion. On pourra utiliser la notation $(\mathbf{x_1} : v(\mathbf{x_1}), \ldots, \mathbf{x_n} : v(\mathbf{x_n}))$ pour désigner l'ensemble des couples (variable, valeur) correspondant à l'instanciation v, ou, lorsqu'aucune confusion n'est possible, la notation simple $(v(\mathbf{x_1}), \ldots, v(\mathbf{x_n}))$.

L'ensemble $Inst(\mathscr{X})$ des instanciations sur $\mathscr{X}$ est bien en bijection avec un produit cartésien de domaines finis, $\mathscr{D}_{\mathbf{x_1}} \times \cdots \times \mathscr{D}_{\mathbf{x_n}}$; il correspond donc à l'idée intuitive d'espace combinatoire que nous avons proposée ci-avant. C'est donc la définition que nous choisirons pour la notion d'*espace d'alternatives combinatoires* :

Définition 3.2 (Espace d'alternatives combinatoire) *Un* espace d'alternatives combinatoire *fondé sur un ensemble de variables* $\mathscr{X} = \{\mathbf{x_i}, \ldots, \mathbf{x_n}\}$ *et une application de domaine* $\mathcal{D}$ *est un ensemble* $\mathscr{A}_{\mathscr{X},\mathcal{D}} \subseteq Inst(\mathscr{X}, \mathcal{D})$*. Chaque élément de* $\mathscr{A}_{\mathscr{X},\mathcal{D}}$ *est une* alternative admissible *de l'espace.*

Tout comme pour la définition d'une instanciation, nous pourrons par la suite omettre $\mathcal{D}$ et $\mathscr{X}$ dans l'emploi de la notation $\mathscr{A}_{\mathscr{X},\mathcal{D}}$. Cette définition d'un espace d'alternatives combinatoire est intuitive et commode : toute alternative correspond à la «décision» d'affecter une valeur particulière à chaque variable. Dans l'exemple 3.1, l'ensemble des alternatives admissibles est un espace d'alternatives combinatoire fondé sur les variables $\{\mathbf{apéritif}, \mathbf{entrée}, \mathbf{plat}, \mathbf{fromage}, \mathbf{dessert}, \mathbf{vin}\}$. Les domaines de ces variables correspondent aux choix possibles, et une alternative est une application qui à chacune des variables associe une valeur de son domaine, comme par exemple l'application (**apéritif** : Macvin du Jura, **entrée** : Salade comtoise, **plat** : Saucisse de Morteau, **fromage** : Comté, **dessert** : Salade de fruits rouges, **vin** : Château l'Étoile).

Introduisons quelques notations et termes de vocabulaire supplémentaires qui nous seront utiles pour la suite. Si $\mathscr{A}$ est un espace d'alternatives combinatoires fondé sur les variables $\mathscr{X}$, alors toute instanciation v de $\mathscr{A}$ sera appelée instanciation *complète* (en plus d'être nommée alternative admissible). Une instanciation $v_{\mathscr{S}}$ ne portant que sur un sous-ensemble $\mathscr{S} \subsetneq \mathscr{X}$ sera dite *partielle*. De plus, $v_{\downarrow \mathscr{S}}$ désignera la restriction (ou projection) de l'instanciation v aux variables de $\mathscr{S}$. Étant données deux instanciations v_x et v_y sur deux sous-ensembles disjoints $\mathscr{S}_x$ et $\mathscr{S}_y$ de variables, nous noterons $\langle v_x, v_y \rangle$ l'instanciation qui attribue à toutes les variables $\mathbf{x} \in \mathscr{S}_x$ les valeurs $v_x(\mathbf{x})$, et à toutes les variables $\mathbf{y} \in \mathscr{S}_y$ les valeurs $v_y(\mathbf{y})$.

3.1.1.2 Application au partage : espace des allocations

Revenons sur le problème de partage de biens indivisibles tels qu'il a été défini au chapitre 1. Dans ce problème, l'espace des alternatives est défini comme étant l'ensemble des partages admissibles, un partage étant défini comme un ensemble de parts. Si $\mathscr{N}$ est l'ensemble des n agents et $\mathscr{O}$ l'ensemble des p objets, une alternative est donc un ensemble de n sous-ensembles de $\mathscr{O}$: c'est un élément de $\wp(\mathscr{O})^n$. L'espace des alternatives admissibles est un ensemble d'alternatives, donc un élément de $\wp(\wp(\mathscr{O})^n)$.

Il s'agit bien d'un espace combinatoire tel que nous l'avons défini. Pour s'en convaincre, nous allons exhiber un ensemble $\mathscr{X}$ de variables de décision et montrer que $\wp(\wp(\mathscr{O})^n)$ est en bijection avec l'ensemble $Inst(\mathscr{X})$. La proposition suivante est immédiate :

Proposition 3.1 (Espace des allocations) *Soient $\mathscr{N}$ un ensemble d'agents et $\mathscr{O}$ un ensemble d'objets, et soit $Alloc_{\mathscr{O},\mathscr{N}}$ l'ensemble de variables binaires $\{\mathbf{alloc}(\mathbf{o},\mathbf{i}) \mid o \in \mathscr{O}$ et $i \in \mathscr{N}\}$. Alors $\wp(\wp(\mathscr{O})^n)$ est en bijection avec $Inst(Alloc_{\mathscr{O},\mathscr{N}})$.*

> **Démonstration** Soit h l'application qui à toute instanciation $v \in Inst(Alloc_{\mathscr{O},\mathscr{N}})$ associe le partage $h(v)$ tel que $\forall i \in \mathscr{N}$, $\pi_i = \{o \in \mathscr{O} \mid v(\mathbf{alloc}(\mathbf{o},\mathbf{i})) = 1\}$. Le partage $h(v)$ est très clairement bien défini et unique. En outre, pour tout partage $\overrightarrow{\pi}$, l'instanciation $v_{\overrightarrow{\pi}}$ telle que $\forall \mathbf{alloc}(\mathbf{o},\mathbf{i}) \in Alloc_{\mathscr{O},\mathscr{N}}$, $v_{\overrightarrow{\pi}}(\mathbf{alloc}(\mathbf{o},\mathbf{i})) = 1$ si et seulement si $o \in \pi_i$ est bien définie, unique, et on peut vérifier facilement que $h(v_{\overrightarrow{\pi}}) = \overrightarrow{\pi}$. h est donc bijective. ▲

Cette proposition implique donc que tout partage peut être représenté par une instanciation v sur l'ensemble des variables $Alloc_{\mathscr{O},\mathscr{N}}$: chaque variable $\mathbf{alloc}(\mathbf{o},\mathbf{i})$ telle que $v(\mathbf{alloc}(\mathbf{o},\mathbf{i})) = 1$ correspond à l'attribution de l'objet o à l'agent i. Ainsi, tout ensemble de partages admissibles correspond à un ensemble d'instanciations. La bijection h nous fournira donc un moyen commode pour définir l'espace des alternatives admissibles.

3.1.1.3 La représentation compacte

Nous allons nous intéresser maintenant à la question de la représentation compacte. Cette question compacte concerne ici la manière de spécifier le sous-ensemble des alternatives (ou instanciations complètes) admissibles. Bien entendu, lorsque l'on parle de «description concise», que ce soit pour l'espace des alternatives ou pour la structure de préférences, il ne s'agit pas de décrire de manière compacte tous les espaces combinatoires ou toutes les structures de préférences possibles. Une telle description compacte serait impossible pour des raisons simples liées à la théorie de l'information : il y a strictement moins de 2^t formules du langage de moins de t bits. La taille caractéristique des espaces combinatoires dont nous allons parler ici est exponentielle en m^n (la taille de l'ensemble des ensembles de tuples de n éléments de domaines de taille m est 2^{m^n}). Cela prouve que certains

éléments de l'espace combinatoire devront être codés par une formule de taille exponentielle en n. Ainsi, le terme de «représentation compacte» désigne l'expression concise de l'ensemble des éléments *intéressants*, ou *réalistes* de l'espace combinatoire. Naturellement, il est difficile de mettre une définition formelle derrière les termes «intéressant» ou «réaliste», dont la définition est relativement empirique : la plupart des langages d'expression compacte d'espaces combinatoires exploitent les régularités observées liées à ce que serait une description informelle et intuitive de cet espace. Ainsi par exemple, les langages à base de logique sont fondés sur le fait que la description d'un espace combinatoire par un ensemble de relations logiques entre les variables de cet espace est intuitive. Les langages à base d'enchères combinatoires fondés sur les lots (voir section 3.2.5) parient sur le fait que les agents enchérisseurs expriment naturellement leurs préférences sous la forme de mises sur des lots et de relations entre ces mises.

3.1.2 Réseaux de contraintes

Le paradigme dominant dans le domaine de la représentation d'espaces d'alternatives admissibles combinatoires est le paradigme des réseaux de contraintes [Montanari, 1974; Mackworth, 1977a]. Il est fondé sur la spécification d'un ensemble d'alternatives admissibles sous la forme de contraintes, d'une manière similaire à notre approche introduite au chapitre 1 pour définir l'espace des partages admissibles (voir la définition 1.3) :

Définition 3.3 (Réseau de contraintes) *Un réseau de contraintes est un triplet $(\mathscr{X}, \mathcal{D}, \mathscr{C})$, où :*
- ▷ $\mathscr{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_p}\}$ *est un ensemble de variables ;*
- ▷ $\mathcal{D}$ *est une application qui à toute variable* $\mathbf{x_i}$ *associe un domaine fini* $\mathscr{D}_{\mathbf{x_i}}$ *;*
- ▷ $\mathscr{C}$ *est un ensemble de contraintes, avec, pour tout* $C \in \mathscr{C}$ *:*
  - $\mathscr{X}(C) \subseteq \mathscr{X}$ *est un ensemble de variables appelé le scope de la contrainte,*
  - $\mathscr{R}(C) \subseteq Inst(\mathscr{X}(C))$ *est l'ensemble des instanciations autorisées par la contrainte.*

Une contrainte C impliquant k variables sera dite k-aire, k étant appelé l'arité de C. Si $k = 1$, la contrainte est unaire et si $k = 2$, la contrainte est dite binaire. Dans ce cadre, l'espace des alternatives admissibles est défini comme étant l'ensemble des solutions du réseau de contraintes, c'est-à-dire l'ensemble des instanciations vérifiant toutes les contraintes :

Définition 3.4 (Instanciation cohérente, solution) *Soit C une contrainte. Une instanciation v sur un ensemble de variables $\mathscr{S} \supseteq \mathscr{X}(C)$* satisfait *la contrainte C si $v_{\downarrow \mathscr{X}(C)} \in \mathscr{R}(C)$. Sinon, v* viole *C.*

Soit $(\mathscr{X}, \mathcal{D}, \mathscr{C})$ un réseau de contraintes. Une instanciation v sur $\mathscr{X}$ est cohérente *(ou* consistante*) si elle satisfait toutes les contraintes $c \in \mathscr{C}$. Elle est dite* incohérente *(ou* inconsistante*) sinon.*

Une solution *est une instanciation complète cohérente. Un réseau de contraintes est dit* consistant *s'il possède au moins une solution. L'ensemble des solutions d'un réseau de contraintes $(\mathscr{X}, \mathcal{D}, \mathscr{C})$ sera noté $sol(\mathscr{X}, \mathcal{D}, \mathscr{C})$.*

Si l'on fait la supposition que pour toute contrainte C et toute instanciation v on est capable de vérifier en temps polynomial que v est autorisée par C, ce qui est une hypothèse raisonnable dans la plupart des cas, alors le problème de déterminer si un réseau de contraintes est consistant, appelé problème de satisfaction de contraintes (et sur lequel nous reviendrons au chapitre 5), est NP-complet.

Notons que la définition introduite ne précise pas explicitement comment sont spécifiés les tuples relatifs aux contraintes. S'ils sont représentés de manière explicite (c'est-à-dire par des tables associées à chaque contrainte), l'économie d'espace réalisée en utilisant ce cadre de représentation

résulte du gain obtenu en décomposant la contrainte globale décrivant l'espace admissible en sous-contraintes d'arité inférieure : si le réseau comporte e contraintes d'arité maximale $r < n$, portant sur des variables dont le domaine est de taille maximale d, le coût spatial de la représentation est de ed^r, ce qui est potentiellement très inférieur à la taille de la représentation explicite d^n. Bien entendu, le gain est nul si le réseau de contraintes comporte une contrainte d'arité n.

Parmi les réseaux de contraintes, ceux qui ne comportent que des contraintes binaires ont été particulièrement étudiés dans la littérature. Il y a plusieurs raisons à cela. Tout d'abord, de nombreuses contraintes issues de problèmes réels s'expriment de manière relativement naturelle sous la forme de contraintes binaires, ce qui fait des réseaux de contraintes binaires un cadre relativement expressif. De plus, le gain obtenu en terme d'espace est relativement conséquent, ce qui est moins le cas si l'on introduit des contraintes d'arité supérieure. Cela s'explique enfin par le fait qu'il existe des algorithmes de résolution génériques très efficaces, fondés sur des mécanismes de filtrage associés aux contraintes binaires [de Givry *et al.*, 2005; Cooper et Schiex, 2004; Larrosa et Schiex, 2004; Mackworth, 1977a; Bessière et Régin, 2001]. En outre, comme le fait remarquer [Bessière, 2006], la plupart des concepts nouveaux sont beaucoup plus faciles à expliquer et à présenter si les contraintes sont binaires. On peut citer enfin un certain nombre de travaux relatifs à la traduction de contraintes non binaires en contraintes binaires (voir par exemple [Smith, 2006, page 391]).

On peut aussi, par commodité ou par économie d'espace si l'on a affaire à des contraintes non binaires, définir les contraintes *en intension* (contrairement à la forme explicite, dite *en extension*), c'est-à-dire sous la forme : $\{(\alpha_1, \ldots, \alpha_k) \mid f(\alpha_1, \ldots, \alpha_k) = 1\}$, où f est une fonction Turing-calculable à valeurs dans $\mathbb{B} = \{0, 1\}$. Dans ce cas, à l'instar des langages de représentation logique que nous allons introduire plus loin, la complexité spatiale est reportée sur la complexité temporelle liée au calcul de l'ensemble de tuples tels que $f(\alpha_1, \ldots, \alpha_k) = 1$. De nombreux travaux concernant la spécification et le développement d'algorithmes liés aux contraintes k-aires définies en intension (aussi appelées «contraintes globales») ont été effectués dans le domaine de la programmation par contraintes, qui est devenu l'un des domaines historiques de l'intelligence artificielle.

Nous reviendrons sur ces notions de réseau de contraintes et algorithmes de résolution et de filtrage dans le chapitre 5 consacré à la programmation par contraintes et au calcul de solutions leximin-optimales.

3.1.3 Variables de décision binaires

3.1.3.1 Représentation logique

Parmi les espaces d'alternatives combinatoires, ceux qui sont fondés sur des variables de décision binaires (c'est-à-dire dont le domaine de valeurs est $\mathbb{B} = \{0, 1\}$) ont une importance particulière, et ce pour plusieurs raisons. La première raison est que la binarité des variables permet l'utilisation de langages fondés sur la logique propositionnelle, qui est un langage puissant, relativement intuitif, et liée au développement historique de l'intelligence artificielle et plus précisément de la représentation des connaissances. La deuxième raison est que de nombreux problèmes réels se modélisent de manière naturelle à l'aide de variables binaires, et ce en particulier dans le domaine du partage, comme nous l'avons vu.

Introduisons avant de poursuivre quelques notions de logique propositionnelle et quelques notations qui nous seront utiles pour la suite.

Définition 3.5 (Langage propositionnel) *Soit Var un ensemble de variables propositionnelles. Le langage $\mathcal{L}_{Var}$ est le langage propositionnel fondé sur les variables propositionnelles de Var, les symboles de constantes $\bot$ et $\top$, et les connecteurs logiques $\neg$, $\vee$, et $\wedge$, c'est-à-dire tel que :*

▷ $\forall \mathbf{x} \in v$, $\mathbf{x}$ *est une formule de* $\mathcal{L}_{Var}$;
▷ $\bot$ *et* $\top$ *sont des formules de* $\mathcal{L}_{Var}$;
▷ $\forall \varphi_1$ *et* φ_2 *formules de* $\mathcal{L}_{Var}$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$ *et* $\neg\varphi_1$ *sont des formules de* $\mathcal{L}_{Var}$.

Nous noterons de plus $\mathcal{L}^+_{Var}$ le langage $\mathcal{L}^+_{Var}$ restreint aux formules ne comportant pas de symbole $\neg$. Notons que nous n'introduisons pas les symboles d'implication $\rightarrow$ et d'équivalence $\leftrightarrow$ dans la définition du langage $\mathcal{L}_{Var}$, pour des raisons liées à la définition de $\mathcal{L}^+_{Var}$. Étant donnée une formule φ de $\mathcal{L}^+_{Var}$, nous noterons $Var(\varphi)$ l'ensemble des variables propositionnelles apparaissant dans φ.

Les définitions suivantes sont classiques en logique propositionnelle. Un *littéral* l désigne une variable propositionnelle $\mathbf{x}$ ou sa négation $\neg\mathbf{x}$. Une *clause* est une disjonction de littéraux $l_1 \vee \cdots \vee l_n$, un *cube* est une conjonction de littéraux $l_1 \wedge \cdots \wedge l_n$, une formule est en *forme normale négative* (NNF) si le symbole $\neg$ n'apparaît que devant une variable propositionnelle, elle est en *forme normale conjonctive* (CNF) si elle s'écrit comme une conjonction de clauses $Cl_1 \wedge \cdots \wedge Cl_n$, et elle est en *forme normale disjonctive* (DNF) si elle s'écrit comme une disjonction de cubes $Cu_1 \vee \cdots \vee Cu_n$.

Nous définissons de plus les notions d'interprétation et de modèle, correspondant aux notions d'instanciation et d'instanciation consistante sur des variables binaires :

Définition 3.6 (Interprétation, modèle) *Soit* $\mathcal{L}_{Var}$ *le langage propositionnel fondé sur l'ensemble de variables Var. Une* interprétation *sur Var est une instanciation* v *de* $Inst(Var)$*, c'est-à-dire une fonction de Var dans* $\mathbb{B}$.

Un modèle *de* φ *est une interprétation qui rend vraie la formule* φ *au sens de la théorie des modèles classique en logique propositionnelle. Si* v *est un modèle de* φ*, nous noterons* $v \vDash \varphi$*, et l'ensemble des modèles de* φ *sera noté* $Mod(\varphi)$.

Puisqu'à toute formule logique φ correspond un ensemble de modèles, donc un ensemble d'instanciations sur les variables propositionnelles du langage, nous pouvons donc représenter un espace d'alternatives combinatoire fondé sur des variables binaires par une formule logique :

Définition 3.7 (Représentation logique d'un espace d'alternatives) *Soit un espace combinatoire* $\mathscr{A}_{Var}$ *fondé sur des variables binaires* $Var = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$. *Une* représentation logique de l'espace $\mathscr{A}_{Var}$ *est une formule propositionnelle* $\varphi \in \mathcal{L}_{Var}$ *telle que* $Mod(\varphi) = \mathscr{A}_{Var}$.

Il convient de commenter quelque peu cette définition. Tout d'abord, nous pouvons remarquer que la formule représentant un ensemble de sous-ensembles d'objets n'est pas unique[2]. Pis encore, sa taille peut varier énormément : considérons par exemple la formule en forme normale conjonctive $(\mathbf{x_0^1} \vee \mathbf{x_1^1}) \wedge \cdots \wedge (\mathbf{x_0^n} \vee \mathbf{x_1^n})$. La formule en forme normale disjonctive qui lui correspond est $(\mathbf{x_0^1} \wedge \mathbf{x_0^2} \wedge \cdots \wedge \mathbf{x_0^{n-1}} \wedge \mathbf{x_0^n}) \vee (\mathbf{x_0^1} \wedge \mathbf{x_0^2} \wedge \cdots \wedge \mathbf{x_0^{n-1}} \wedge \mathbf{x_1^n}) \vee \cdots \vee (\mathbf{x_1^1} \wedge \mathbf{x_1^2} \wedge \cdots \wedge \mathbf{x_1^{n-1}} \wedge \mathbf{x_0^n}) \vee (\mathbf{x_1^1} \wedge \mathbf{x_1^2} \wedge \cdots \wedge \mathbf{x_1^{n-1}} \wedge \mathbf{x_1^n})$. Alors que la première formule a une taille de l'ordre de n, la seconde a une taille de l'ordre de $n2^n$. Il convient donc de bien choisir la formule logique pour obtenir un gain en terme d'espace.

Ensuite, il est intéressant de remarquer ici que, tout comme pour les réseaux de contraintes introduits ci-avant, le gain potentiel en espace apporté par l'expression des contraintes sous forme logique a une contrepartie en terme de temps de calcul, puisque le problème de recherche de modèle d'une formule logique quelconque (ou en forme normale conjonctive) est NP-complet d'après le théorème de Cook (voir [Cook, 1971], et le problème 19 en annexe A).

[2] Elle l'est cependant à l'équivalence logique près.

3.1.3.2 Application à l'espace des allocations

Comme nous l'avons vu, la modélisation à base de variables binaires est particulièrement bien adaptée au problème de partage, car une alternative peut être spécifiée de manière intuitive par une instanciation d'un ensemble de variables binaires $\mathbf{alloc}(\mathbf{o}, \mathbf{i})$. Tout ensemble de partages peut donc être spécifié par une formule logique du langage propositionnel suivant :

Définition 3.8 (Langage $\mathcal{L}_{\mathscr{O}}^{\mathbf{alloc}}$) *$\mathcal{L}_{\mathscr{O}}^{alloc}$ est le langage propositionnel défini par :*
- ▷ *l'ensemble $Alloc_{\mathscr{O},\mathscr{N}}$ des variables propositionnelles $\mathbf{alloc}(\mathbf{o}, \mathbf{i})$ en bijection avec l'ensemble des couples objet-agent $\mathscr{O}\times\mathscr{N}$,*
- ▷ *l'ensemble des connecteurs $\{\neg, \vee, \wedge\}$.*

En d'autres termes, le langage $\mathcal{L}_{\mathscr{O}}^{alloc}$ correspond au langage $\mathcal{L}_{Alloc_{\mathscr{O},\mathscr{N}}}$. L'ensemble des partages admissibles (ou l'ensemble des contraintes d'admissibilité) pourra donc être spécifié de manière compacte grâce à la logique propositionnelle, par un ensemble de formules sur le langage

3.1.3.3 Représentation binaire de variables n-aires

Une autre raison de l'intérêt de la modélisation d'espaces combinatoires à l'aide de variables binaires est que l'on est capable de représenter n'importe quel type d'espace combinatoire à l'aide de variables binaires, moyennant l'introduction de contraintes logiques. Il suffit pour cela d'introduire un symbole propositionnel $\mathbf{egal}(\mathbf{x_i}, \alpha)$ pour chaque couple $(\mathbf{x_i}, \alpha)$, où $\mathbf{x_i}$ est l'une des variables initiales et α une des valeurs de son domaine : une telle variable propositionnelle correspond simplement à l'affectation de la valeur α à la variable initiale $\mathbf{x_i}$. Il faut cependant assurer que les variables $\mathbf{x_i}$ ne prennent qu'une et une seule valeur parmi les α_i de leur domaine. Cela peut être spécifié à l'aide d'une contrainte logique :

$$\psi = \bigwedge_{i=1}^{n} \left(\bigvee_{\alpha \in \mathscr{D}_{\mathbf{x_i}}} \mathbf{egal}(\mathbf{x_i}, \alpha) \wedge \bigwedge_{(\alpha,\alpha') \in \mathscr{D}^2_{\mathbf{x_i}}, \alpha \neq \alpha'} \neg\mathbf{egal}(\mathbf{x_i}, \alpha) \vee \neg\mathbf{egal}(\mathbf{x_i}, \alpha') \right).$$

L'espace des alternatives admissibles peut donc être spécifié, à l'aide des symboles $(\mathbf{x_i}, \alpha)$, et d'une formule logique $\varphi = \varphi' \wedge \psi$. Cette traduction implique l'introduction de $b \times k$ variables propositionnelles, si initialement le problème était modélisé sous la forme de n variables de domaines de taille k, ainsi que l'introduction d'une formule ψ comportant de l'ordre de $n \times k^2$ variables.

On trouve quelques références concernant la traduction de variables n-aires en variables binaires [Smith, 2006, page 393], question qui a été étudiée principalement dans le contexte de la traduction du problème de satisfaction de contraintes dans le cadre du problème [SAT] [Walsh, 2000]. Bien entendu, le passage de variables n-aires aux variables binaires n'est pas sans effet en terme d'efficacité sur la résolution du problème initial.

3.1.3.4 Représentation logique et compilation de connaissances

Le succès de la logique propositionnelle dans des domaines tels que la représentation des connaissances ou l'expression compacte de domaines combinatoires a fait naturellement émerger la question de la complexité liée aux tâches de raisonnement à base de logique. Cette question est cruciale, car le fait de pouvoir exprimer des connaissances ou un espace combinatoire de manière compacte n'est pas très utile si l'on s'avère incapable de raisonner sur cette information, à cause de la trop grande

complexité liée à l'expression compacte. De cette constatation est né le domaine relativement récent de la *compilation de connaissances* [Darwiche et Marquis, 2002].

La compilation de connaissances est centrée sur la notion de traduction d'une formule logique dans un langage cible, l'objectif étant de reporter une partie de la complexité de la tâche de raisonnement sur la tâche de traduction (qui peut être exécutée hors-ligne). Les trois aspects fondamentaux du langage cible d'expression de la formule logique sont les suivants : (1) sa compacité vis-à-vis du langage propositionnel initial, (2) l'ensemble des requêtes de raisonnement qui peuvent être exécutées en temps polynomial sur ce langage, et (3) l'ensemble des transformations qui peuvent être appliquées en temps polynomial sur une formule de ce langage.

Nous allons présenter succinctement quelques langages cibles dédiés à la compilation de connaissances (pour un aperçu détaillé on pourra consulter [Darwiche et Marquis, 2002], dont cette sous-section est largement inspirée). Ces langages sont tous des fragments de la logique propositionnelle, et plus précisément, la plupart des travaux sur la compilation font l'hypothèse que ces fragments sont aussi expressifs que la logique propositionnelle elle-même (c'est-à-dire que toute formule logique peut être compilée), et que ce sont des sous-ensembles du langage NNF (*Negation Normal Form*).

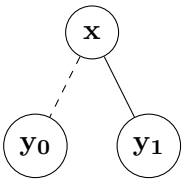
Les premiers langages considérés sont fondés sur la notion de *platitude* d'une formule du langage NNF : une formule est considérée comme plate si elle ne contient que deux niveaux d'opérateurs imbriqués. Cette propriété fournit la première grande famille de langages cibles de compilation, qui contient entre autres les f-NNF (ensemble des formules NNF plates), les CNF et les DNF.

La seconde famille de langages cibles de compilation n'impose aucune restriction sur le degré d'imbrication des opérateurs logiques, mais est fondée sur un ensemble de propriétés liées à l'agencement des littéraux et des opérateurs dans la formule. Les trois propriétés suivantes fournissent une première classe de langages :

- ▷ la *décomposabilité* : une formule φ en forme normale négative satisfait cette propriété si pour toute conjonction $\varphi_1 \wedge \varphi_2$ apparaissant dans φ, $Var(\varphi_1) \cap Var(\varphi_2) = \varnothing$ (les deux sous-formules ne partagent aucune variable) ;
- ▷ le *déterminisme* : une formule φ en forme normale négative satisfait cette propriété si pour toute disjonction $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2 \models \bot$ (les deux sous-formules sont contradictoires) ;
- ▷ la *régularité* (*smoothness*) : une formule φ en forme normale négative satisfait cette propriété si pour toute disjonction $\varphi_1 \vee \varphi_2$ $Var(\varphi_1) = Var(\varphi_2)$

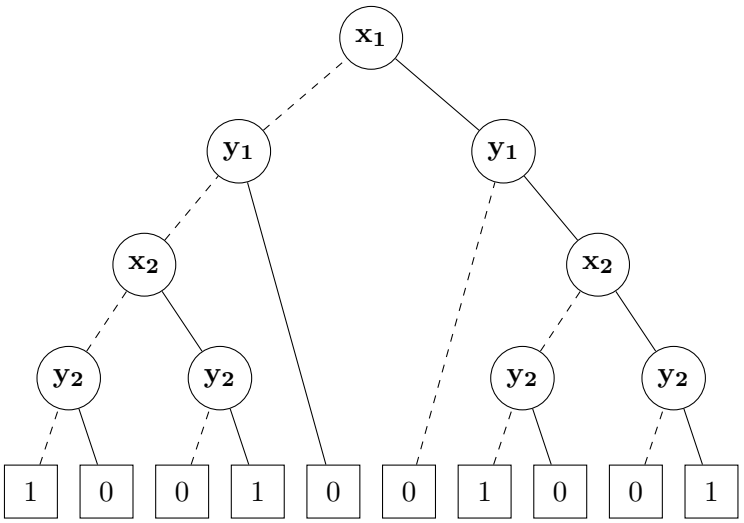
Les propriétés de décomposabilité, déterminisme et régularité définissent respectivement les classes des formules DNNF, d-NNF et s-NNF. L'ensemble des formules en forme normale négative vérifiant les propriétés de décomposabilité et déterminisme (resp. décomposabilité, déterminisme et régularité) constitue la classe d-DNNF (resp. sd-DNNF).

Nous allons terminer cette courte revue des langages de représentation de connaissances par l'un des langages les plus utilisés dans ce domaine : celui des *diagrammes de décision binaires* (ou BDD pour *Binary Decision Diagrams*). Ce langage, introduit dans [Bryant, 1986], est, tout comme les langages précédents, un sous-ensemble du langage NNF, et plus précisément il s'agit d'un sous-ensemble du langage DNNF. Il s'appuie sur une représentation logique fondée sur un opérateur particulier, l'opérateur *if-then-else* $:\mathbf{x} \,?\, \mathbf{y_1} : \mathbf{y_0} = (\mathbf{x} \wedge \mathbf{y_1}) \vee (\neg\mathbf{x} \wedge \mathbf{y_0})$. Cet opérateur se représente graphiquement par un arbre enraciné en $\mathbf{x}$ et possédant deux feuilles $\mathbf{y_0}$ et $\mathbf{y_1}$. Un arc appelé *low-edge* (en pointillés) relie la racine à la feuille correspondant à la partie négative de la formule $\mathbf{y_0}$, et un arc appelé *high-edge* (en trait plein) relie la racine à la feuille correspondant à la partie positive de la formule $\mathbf{y_1}$:

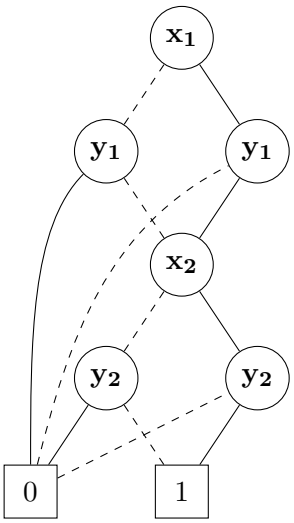


Toute formule logique quelconque peut être traduite en forme normale *if-then-else* (INF), c'est-à-dire en une formule logique ne comportant que des variables propositionnelles, les constantes $\bot$ et $\top$, et des connecteurs *if-then-else* dont le premier opérande est une variable. Cette traduction s'effectue par une *expansion de Shannon*, c'est-à-dire par une succession de traductions élémentaires $\varphi \rightsquigarrow \mathbf{x} \,?\, \varphi(\mathbf{x} \leftarrow \top) : \varphi(\mathbf{x} \leftarrow \bot)$, où $\mathbf{x}$ est une variable propositionnelle de φ, et où $\varphi(\mathbf{x} \leftarrow c)$ désigne la formule φ dans laquelle on a remplacé $\mathbf{x}$ par la constante propositionnelle c. Formellement, cela revient à construire un arbre de recherche (arbre de décision) associé à la formule logique, c'est-à-dire à instancier dans un certain ordre les variables propositionnelles de φ.

Exemple 3.3 Nous allons illustrer à l'aide de l'exemple 3.2 comment une contrainte sur l'ensemble des partages possibles peut être exprimée sous la forme d'un arbre de décision, puis sous la forme d'un BDD. Considérons par exemple que pour des raisons logistiques, le tracteur (tr) et les terres cultivables (ch) forment un lot indivisible, ainsi que la tondeuse (to) et la maison (m). Une telle contrainte peut être exprimée simplement par une formule $\psi = \bigwedge_{i=1}^{n} \psi_i$, avec pour tout agent i : $\psi = (\mathbf{alloc}(\mathbf{tr}, \mathbf{i}) \leftrightarrow \mathbf{alloc}(\mathbf{ch}, \mathbf{i})) \wedge (\mathbf{alloc}(\mathbf{to}, \mathbf{i}) \leftrightarrow \mathbf{alloc}(\mathbf{m}, \mathbf{i}))$. La formule ψ peut donc être éclatée en n contraintes différentes s'exprimant chacune par une formule du type $\varphi = (\mathbf{x_1} \leftrightarrow \mathbf{y_1}) \wedge (\mathbf{x_2} \leftrightarrow \mathbf{y_2})$. Cette formule peut être représentée par un arbre de décision comme celui de la figure 3.1(a) qui est fondé sur l'ordre $\mathbf{x_1}$, $\mathbf{y_1}$, $\mathbf{x_2}$, $\mathbf{y_2}$ sur les variables logiques.



(a) Un arbre de décision pour $(\mathbf{x_1} \leftrightarrow \mathbf{y_1}) \wedge (\mathbf{x_2} \leftrightarrow \mathbf{y_2})$



(b) Un BDD pour $(\mathbf{x_1} \leftrightarrow \mathbf{y_1}) \wedge (\mathbf{x_2} \leftrightarrow \mathbf{y_2})$ ordonné par $\mathbf{x_1} < \mathbf{y_1} < \mathbf{x_2} < \mathbf{y_2}$.

Figure 3.1 — Un arbre de décision et le diagramme de décision binaire réduit associé pour la formule $(\mathbf{x_1} \leftrightarrow \mathbf{y_1}) \wedge (\mathbf{x_2} \leftrightarrow \mathbf{y_2})$

Les diagrammes de décision binaires sont fondés sur l'arbre de décision associé à une formule et à un certain ordre des variables. La construction du BDD se fait par deux types d'opérations de réduction :

- ▷ en identifiant tous les sous-arbres isomorphes de l'arbre de décision initial (et en conséquence aussi tous les nœuds terminaux 0 et 1) ;
- ▷ en supprimant tous les nœuds ayant le même successeur pour ses deux arcs sortants *low-* et *high-edge*.

Ainsi, par exemple, dans l'arbre de la figure 3.1(a), les deux sous-arbres de racine $\mathbf{x_2}$ sont isomorphes, et peuvent donc être réduits à un seul sous arbre. On aboutit donc par ces opérations de réduction à un graphe acyclique dirigé possédant une racine et deux nœuds terminaux 0 et 1, et dont tous les nœuds possèdent deux arcs sortants. Si de plus le parcours des nœuds le long de tous les chemins se fait toujours dans le même ordre (comme c'est le cas dans le BDD de la figure 3.1(b)), le BDD est dit *ordonné* (c'est un OBDD). Si enfin le BDD ne peut plus être réduit par les opérations décrites ci-avant, le BDD est dit *réduit* (R(O)BDD).

L'intérêt principal de cette représentation, outre le gain potentiellement conséquent en terme d'espace utilisé, est fondé sur la propriété d'unicité du ROBDD : étant donné un ordre sur les variables, il n'existe qu'un seul ROBDD associé à une formule logique donnée. Cette propriété permet de tester en temps constant si une formule est insatisfiable ou s'il s'agit d'une tautologie : les ROBDD représentant ces deux formules sont des graphes ne comportant qu'un seul nœud, respectivement 0 et 1.

Pour avoir un aperçu plus détaillé des diagrammes de décision binaires, on pourra consulter l'article [Bryant, 1986] considéré comme fondateur du formalisme des BDD, ou un tutoriel proposé dans [Andersen, 1999].

3.2 Représentation compacte de préférences

3.2.1 Cadre formel

3.2.1.1 Langage de représentation compacte de préférences

Comme nous l'avons fait remarquer dans l'introduction de ce chapitre, la structure combinatoire de l'espace des alternatives rend impossible, outre la description explicite de l'ensemble des tuples formant l'espace des alternatives admissibles comme nous l'avons vu ci-avant, mais aussi et surtout la description explicite intégrale de la structure de préférences, c'est-à-dire la spécification de l'utilité de chaque alternative, ou de la relation de préférence sur tous les couples d'alternatives. La description de la structure de préférence nécessite donc l'introduction d'un *langage de représentation compacte de préférences*.

Avant d'introduire cette notion de manière formelle, nous avons besoin de quelques définitions supplémentaires. La notion de langage de représentation compacte de préférences est fondée sur un *langage formel*. Par langage formel, nous entendons de manière générale tout langage récursivement énumérable, c'est-à-dire engendré par une grammaire (pour une introduction aux langages formels, on pourra consulter l'ouvrage [Alliot *et al.*, 2002]). Cependant, la plupart des langages de représentation compacte de préférences seront fondés sur des langages classiques tels que la logique propositionnelle ou l'arithmétique de $\mathbb{N}$, ou bien seront définis par un ensemble. Un élément m d'un tel langage formel sera appelé un *mot*, ou une *formule* si l'on est dans le cas de la logique propositionnelle.

Équipés de ces notions, nous pouvons désormais introduire la définition d'un *langage de représentation compacte de préférences* :

Définition 3.9 (Langage de représentation de préférences) *Un* langage de représentation

de préférences ordinales *(resp.* cardinales*)* $\mathcal{R}$ *est un couple* $(\mathcal{L}_\mathcal{R}, \mathcal{I}_\mathcal{R})$*, où :*

- ▷ $\mathcal{L}_\mathcal{R}$ *associe à tout ensemble de variables* $\mathscr{X}$ *un langage formel* $\mathcal{L}_\mathcal{R}(\mathscr{X})$ *;*
- ▷ $\mathcal{I}_\mathcal{R}$ *associe à tout ensemble de variables* $\mathscr{X}$ *une fonction de* $\mathcal{L}_\mathcal{R}(\mathscr{X})$ *dans l'ensemble des structures de préférence ordinales (resp. cardinales) sur* $Inst(\mathscr{X})$*.*

Pour tout mot m *de* $\mathcal{L}_\mathcal{R}(\mathscr{X})$ *nous noterons* $\succeq^m_\mathcal{R}$ *(resp.* $u^m_\mathcal{R}$*) la structure de préférence ordinale (resp. la fonction d'utilité)* $\mathcal{I}_\mathcal{R}(m)$ *sur* $Inst(\mathscr{X})$*.*

En d'autres termes, un langage de représentation de préférences est fondé sur deux composantes :

- ▷ un langage de représentation, dont la forme dépend de l'espace $Inst(\mathscr{X})$ à représenter,
- ▷ une fonction dont le rôle est de traduire n'importe quel mot (ou formule) du langage formel en la structure de préférence sur $Inst(\mathscr{X})$ à laquelle correspond ce mot.

Notons que nous considérons que l'expression des préférences se fait sur l'ensemble des alternatives $Inst(\mathscr{X})$, que celles-ci soient admissibles ou non. Cela nous permet de définir la notion de langage de représentation de préférences de manière indépendante à la définition des alternatives admissibles.

Exemple 3.4 Nous pouvons considérer par exemple un langage de représentation compacte de préférences $\mathcal{R}_{simple}$ relativement simple (nous abrégeons $\mathcal{L}_{\mathcal{R}_{simple}}$ en $\mathcal{L}_{simple}$, et $\mathcal{I}_{\mathcal{R}_{simple}}$ en $\mathcal{I}_{simple}$) défini comme suit :

- ▷ $\mathcal{L}_{simple}(\mathscr{X}) = \{(\mathscr{S}, v) \mid \mathscr{S} \subseteq \mathscr{X} \text{ et } v \in Inst(\mathscr{S})\}$;
- ▷ $\mathcal{I}_{simple}$ est défini comme suit : pour tout couple $(\mathscr{S}, v)$ tel que $\mathscr{S} \subseteq \mathscr{X}$ et $v \in Inst(\mathscr{S})$ et pour toute instanciation $v' \in Inst(\mathscr{X})$,
  - $u(v') = 1$ si $v'_{\downarrow\mathscr{S}} = v$;
  - $u(v') = 0$ sinon.

En d'autres termes, dans ce langage assez frustre, les préférences d'un agent sont représentées par une instanciation particulière sur un sous-ensemble $\mathscr{S}$ de variables. L'utilité de l'agent sera de 1 pour toute instanciation qui concorde avec ses préférences exprimées sur le sous-ensemble $\mathscr{S}$, et de 0 sinon. Ainsi, sur l'exemple 3.1 du repas, l'agent peut exprimer le fait qu'il veuille du vin rouge et de la salade en entrée. Tout repas qui vérifie ces préférences lui conférera une utilité égale à 1.

3.2.1.2 Application au partage : espace combinatoire d'objets

Intéressons-nous au cas particulier de l'expression des préférences dans le problème de partage de biens indivisibles. Nous avons introduit dans le chapitre 1 l'hypothèse de non exogénéité des préférences (voir section 1.2.2). Cela signifie donc concrètement que les préférences sont exprimées sur l'ensemble des parts possibles, autrement dit $\wp(\mathscr{O})$. Il s'agit encore une fois d'un espace combinatoire, qui peut être représenté à l'aide d'un ensemble de variables binaires :

Proposition 3.2 (Espace d'objets) *Soit* $\mathscr{O}$ *un ensemble d'objets, et soit* $Var(\mathscr{O})$ *l'ensemble de variables binaires* $\{\mathbf{o} \mid o \in \mathscr{O}\}$*. Alors* $\wp(\mathscr{O})$ *est en bijection avec* $Inst(Var(\mathscr{O}))$ *(appelé espace d'objets associé à* $\mathscr{O}$*).*

La preuve de ce résultat est évidente : à tout sous-ensemble $\pi \subseteq \mathscr{O}$ (nous employons la notation π afin de rester cohérent avec la définition des parts des agents utilisée dans le reste du document) est associée une interprétation de $\mathcal{L}_{Var(\mathscr{O})}$. Nous introduisons à cet effet la fonction γ, définie comme suit : $\gamma(\pi)$ est l'interprétation telle que $\gamma(\pi)(\mathbf{o}) = 1$ si et seulement si $o \in \pi$. Cette fonction est clairement bijective, donc à toute alternative v de l'espace $Inst(Var(\mathscr{O}))$ correspond un ensemble d'objets $\pi = \gamma^{-1}(v)$.

Nous pouvons remarquer que puisque l'on est capable de représenter tout sous-ensemble de $\mathscr{O}$ par une instanciation sur un ensemble de variables binaires, nous pouvons donc représenter toute

part π par une formule logique :

$$\varphi(\pi) = \bigwedge_{o \in \pi} \mathbf{o} \wedge \bigwedge_{o \notin \pi} \neg \mathbf{o}.$$

Ainsi, tout ensemble de sous-ensembles de $\mathscr{O}$ peut être représenté par une formule équivalente à la disjonction des formules $\varphi(\pi)$, pour tout π faisant partie de l'ensemble. En conséquence, à tout ensemble de sous-ensembles de $\mathscr{O}$ — c'est-à-dire en termes de partage à tout ensemble de parts — correspond une formule logique φ (unique à l'équivalence logique près) dont l'ensemble des modèles correspond à l'ensemble des instanciations admissibles. Réciproquement, toute formule logique φ sur $\mathcal{L}_{Var(\mathscr{O})}$ correspond à un ensemble de parts défini par $\gamma^{-1}(Mod(\varphi))$.

Par la suite, nous emploierons quelques abus de notations pour simplifier la présentation des différentes notions introduites. La première simplification sera d'identifier les objets de $\mathscr{O}$ avec les variables propositionnelles de $Var(\mathscr{O})$ (il nous arrivera d'employer la notation mathématique simple o lorsque l'on parle d'un élément en tant qu'objet, et la notation mathématique à fonte grasse $\mathbf{o}$ lorsque l'on désignera la variable propositionnelle correspondant à o) : ainsi, nous noterons $\mathcal{L}_{\mathscr{O}}$ et $\mathcal{L}^+_{\mathscr{O}}$ les langages respectifs $\mathcal{L}_{Var(\mathscr{O})}$ et $\mathcal{L}^+_{Var(\mathscr{O})}$. De plus, nous identifierons, lorsque cela ne prête pas à confusion, l'ensemble $Inst(\mathscr{O})$ des instanciations (interprétations) sur $\mathscr{O}$ à l'ensemble $\wp(\mathscr{O})$ des sous-ensembles de $\mathscr{O}$, à l'aide de la bijection γ introduite ci-avant. Ainsi, une part $\pi \in \wp(\mathscr{O})$ sera identifiée par abus de notation à l'interprétation $\gamma(\pi) \in Inst(\mathscr{O})$. En particulier, nous emploierons la notation $\pi \vDash \varphi$ pour $\gamma(\pi) \vDash \varphi$.

Exemple 3.5 Considérons le langage logique bâti sur l'ensemble $\mathscr{O} = \{o_1, o_2, o_3, o_4, o_5\}$. L'interprétation $v = (\mathbf{o_1} : 0, \mathbf{o_2} : 1, \mathbf{o_3} : 1, \mathbf{o_4} : 0, \mathbf{o_5} : 0)$ correspond au sous-ensemble $\{o_2, o_3\}$. Il s'agit par exemple d'un modèle de la formule $\varphi = (\mathbf{o_2} \wedge \mathbf{o_3}) \vee (\mathbf{o_1} \wedge \mathbf{o_4} \wedge \mathbf{o_5})$, décrivant l'ensemble d'alternatives suivant : $\{o_2, o_3\}$, $\{o_1, o_2, o_3\}$, $\{o_1, o_4, o_5\}$, $\{o_2, o_3, o_4\}$, $\{o_2, o_3, o_5\}$, $\{o_1, o_2, o_3, o_4\}$, $\{o_1, o_2, o_3, o_5\}$, $\{o_1, o_2, o_4, o_5\}$, $\{o_2, o_3, o_4, o_5\}$, $\{o_1, o_2, o_3, o_4, o_5\}$. On a par exemple $\{o_2, o_3, o_5\} \vDash \varphi$.

3.2.1.3 La représentation compacte de préférences

L'étude des langages de représentation compacte de préférences est un domaine traditionnel de l'intelligence artificielle, et plus particulièrement des communautés *représentation de la connaissance et raisonnement* (KR) et *incertitude en intelligence artificielle* (UAI), simplement parce que la problématique de l'explosion combinatoire liée à l'expression de connaissances ou de préférences apparaît de manière naturelle lorsque l'on s'intéresse à la formalisation et à la simulation logicielle du raisonnement humain. Ces communautés ont introduit de nombreux langages de représentation compacte qui sont tous fondés sur l'exploitation de la structure particulière des préférences humaines. Certains de ces langages sont fondés sur la logique propositionnelle (voir la sous-section 3.2.2), partant du principe que les connaissances humaines se représentent de manière assez intuitive et efficace à l'aide de la logique. D'autres (voir entre autres les sous-section 3.2.3.2 et paragraphe 3.2.4.2) sont des modèles graphiques exploitant des propriétés locales sur l'expression des préférences.

Il est souvent difficile de juger précisément de la pertinence d'un langage de représentation compacte, et surtout de son adéquation au problème traité. Certains auteurs proposent néanmoins un certain nombre de critères plus ou moins formels (voir [Chevaleyre *et al.*, 2006a; Coste-Marquis *et al.*, 2004]) pour juger de la pertinence d'un langage $\mathcal{R}$:

- ▷ *Élicitation* : Est-il difficile d'éliciter les préférences, c'est-à-dire de construire une formule du langage $\mathcal{L}_{\mathcal{R}}$ à partir de préférences réelles ?
- ▷ *Pertinence cognitive* : Le langage $\mathcal{R}$ est-il proche de la manière dont les humains expriment leurs préférences ?

- ▷ *Puissance expressive* : Le langage $\mathcal{R}$ peut-il exprimer tous les préordres possibles, toutes les fonctions d'utilité possibles, etc.
- ▷ *Compacité relative* : Étant donnés deux langages $\mathcal{R}$ et $\mathcal{R}'$, existe-t-il pour tout élément m de $\mathcal{L}_{\mathcal{R}}$ un élément m' de $\mathcal{L}_{\mathcal{R}'}$ dont la taille est une fonction polynomiale de celle de m, et telle que $\mathcal{I}_{\mathcal{R}}(m) = \mathcal{I}_{\mathcal{R}'}(m)$?
- ▷ *Complexité computationnelle* : Est-il algorithmiquement difficile de comparer deux alternatives, de déterminer si une alternative est non-dominée, ou encore de déterminer s'il existe une alternative non dominée, vis-à-vis d'un préordre ou d'une fonction d'utilité représentés par une formule du langage $\mathcal{R}$?

Les deux premiers critères cités sont très clairement liés à la manière dont l'être humain exprime naturellement des préférences. Le problème de l'élicitation se pose en particulier dans le cas où les préférences sont numériques : s'il est facile pour un humain d'énoncer une préférence entre une bicyclette rouge et une bicyclette bleue, en revanche, il semble difficile d'affirmer que l'on préfère la bicyclette rouge à hauteur de 8.4 par rapport à la bicyclette bleue.

La puissance expressive est un critère important : il est fortement souhaitable en général que le langage puisse exprimer, si ce n'est n'importe quelle fonction d'utilité, au moins n'importe quel préordre. Pour la plupart des langages classiques présentés ci-après, ce sera le cas.

Les deux derniers critères ont respectivement trait à la complexité spatiale et temporelle des langages de représentation. Si la quantité de mémoire utilisée pour représenter un préordre est exponentielle, alors le gain espéré de la représentation compacte est perdu. De même, les langages pour lesquels la comparaison de deux alternatives est trop complexe risquent de s'avérer inutilisables en pratique pour la recherche d'une alternative optimale.

Une étude des processus d'élicitation de préférences peut être trouvée dans [Chen et Pu, 2004] ; cette question est étudiée de même dans [Sandholm et Boutilier, 2006], dans le cadre particulier des enchères combinatoires. La question de la pertinence cognitive est plus difficile à élucider et n'a été que rarement étudiée à notre connaissance dans la littérature. En revanche, les questions concernant la compacité relative et l'expressivité ont été largement étudiées dans la communauté de l'intelligence artificielle et du choix social, soit à propos de préférences ordinales [Coste-Marquis *et al.*, 2004], soit pour des préférences logiques pondérées [Chevaleyre *et al.*, 2006b], soit pour les langages de lots dans la communauté des enchères combinatoires [Boutilier et Hoos, 2001; Sandholm, 2002; Chevaleyre *et al.*, 2004; Nisan, 2006]. La complexité liée à l'expression compacte de préférences commence à être étudiée dans les mêmes communautés. On peut citer notamment [Lang, 2004] qui étudie l'impact des langages de représentation à base de logique sur la complexité du vote combinatoire, [Lipton *et al.*, 2004; Bouveret et Lang, 2005] qui s'intéressent à la complexité liée à l'absence d'envie dans le partage de ressources (nous y consacrerons une partie du chapitre 4), ou encore [Nisan, 2006; Lehmann *et al.*, 2006] qui traitent dans une moindre mesure de complexité liée à la représentation compacte de problèmes d'enchères combinatoires.

3.2.2 Modélisation à base de logique

De nombreux langages de représentation de préférences sont fondés sur la logique propositionnelle. Comme nous l'avons déjà fait remarquer, les raisons de ce succès sont diverses : nous pouvons mettre en avant le lien historiques entre la logique propositionnelle et le développement de l'intelligence artificielle, ou encore remarquer qu'il s'agit d'un langage très expressif et relativement proche de la manière dont les humains expriment naturellement leurs préférences. On pourra trouver par exemple dans [Lang, 2004, 2006] des vues d'ensemble des langages de représentation de préférences à base de logique.

Afin de ne pas perdre de vue le problème de partage et de rester cohérents avec le reste du manuscrit, nous définirons tous les langages de ce chapitre en supposant que nous avons affaire à un espace combinatoire fondé sur un ensemble d'objets $\mathscr{O}$.

3.2.2.1 Préférences dichotomiques

La plupart des langages de représentation logique de préférences sont fondés sur la notion de *but* : dans ce contexte, un but est simplement une formule logique traduisant un désir ou une préférence élémentaire. La manière la plus simple de représenter des préférences en logique propositionnelle est de ne spécifier qu'un seul but sous la forme d'une formule propositionnelle de $\mathcal{L}_{\mathscr{O}}$.

Définition 3.10 (Représentation logique d'une structure dichotomique) *Le langage de représentation logique dichotomique $\mathcal{R}_{dicho}$ est défini par :*

- ▷ $\mathcal{L}_{dicho}(\mathscr{O}) = \mathcal{L}_{\mathscr{O}}$;
- ▷ *étant donnée une formule propositionnelle φ de $\mathcal{L}_{\mathscr{O}}$, $\mathcal{I}_{dicho}(\mathscr{O})(\varphi)$ est la structure de préférence dichotomique définie par $\mathscr{G} = Mod(\varphi)$.*

En d'autres termes, une alternative (c'est-à-dire une interprétation de $\mathcal{L}_{\mathscr{O}}$) est une bonne alternative si et seulement si c'est un modèle de φ. Par exemple, si $\mathscr{O} = \{o_1, o_2, o_3\}$ et $\varphi = (\mathbf{o_1} \wedge \mathbf{o_2} \wedge \neg\mathbf{o_3}) \vee (\neg\mathbf{o_1} \wedge \mathbf{o_2} \wedge \mathbf{o_3})$, alors $\mathscr{G} = \{\{o_1, o_2\}, \{o_2, o_3\}\}$, et donc par exemple $\{o_2, o_3\}$ est préféré à $\{o_1, o_2, o_3\}$.

Il est clair que ce langage est capable d'exprimer toutes les structures de préférence dichotomiques, car tous les ensembles de sous-ensembles de $\mathscr{O}$ peuvent être représentés par une formule propositionnelle.

3.2.2.2 Préférences ordinales

Base de buts simple Le langage de représentation dichotomique étant relativement fruste, il peut être raffiné en spécifiant un ensemble de buts $GB = \{G_1, \ldots, G_n\}$ (GB pour *Goal Base*). Nous noterons $GB_{\mathscr{O}}$ une base de buts dont les formules sont dans $\mathcal{L}_{\mathscr{O}}$, et $\mathcal{GB}_{\mathscr{O}}$ le langage correspondant à l'ensemble des bases de buts sur $\mathscr{O}$. Étant donné un sous-ensemble π de $\mathscr{O}$, nous noterons de plus :

- ▷ $sat(\pi, GB_{\mathscr{O}}) = \{G_i \mid \pi \vDash G_i\}$;
- ▷ $nonsat(\pi, GB_{\mathscr{O}}) = \{G_i \mid \pi \nvDash G_i\}$

Étant donnée une base de buts, la manière la plus basique de construire une structure de préférences ordinale est de considérer la relation d'inclusion entre les ensembles de buts satisfaits :

Définition 3.11 (Langage $\mathcal{R}_{Pareto}$) *Le langage de représentation de préférences $\mathcal{R}_{Pareto}$ est défini par :*

- ▷ $\mathcal{L}_{Pareto}(\mathscr{O}) = \mathcal{GB}_{\mathscr{O}}$;
- ▷ *étant donnée une base de buts $GB_{\mathscr{O}}$, la structure de préférences ordinale $\succeq_{Pareto}^{GB_{\mathscr{O}}}$ est telle que :*

$$\forall(\pi, \pi') \in (\wp(\mathscr{O}))^2,\ \pi \succeq_{Pareto}^{GB_{\mathscr{O}}} \pi' \Leftrightarrow sat(\pi, GB_{\mathscr{O}}) \supseteq sat(\pi', GB_{\mathscr{O}}).$$

Exemple 3.6 Considérons par exemple un espace d'alternatives fondé sur les objets $\mathscr{O} = \{o_1, o_2, o_3, o_4, o_5\}$, et la base de buts d'un agent $GB_{\mathscr{O}} = \{\mathbf{o_1} \vee (\mathbf{o_2} \wedge \mathbf{o_3}), \mathbf{o_4}, \mathbf{o_4} \vee \mathbf{o_5}\}$. Alors, si $\pi = \{o_1, o_4\}$ et $\pi' = \{o_2, o_3, o_5\}$ nous avons $\pi \succ_{Pareto}^{GB_{\mathscr{O}}} \pi'$, car $sat(\pi, GB_{\mathscr{O}}) = GB_{\mathscr{O}}$ et $sat(\pi', GB_{\mathscr{O}}) = \{\mathbf{o_1} \vee (\mathbf{o_2} \wedge \mathbf{o_3}), \mathbf{o_4} \vee \mathbf{o_5}\}$.

Base de buts stratifiée Le langage $\mathcal{R}_{Pareto}$ est capable de représenter tous les préordres sur l'ensemble des alternatives. Il est possible de raffiner ce langage quelque peu basique en introduisant une relation de priorité sur les formules représentant les buts :

Définition 3.12 (Base de buts stratifiée) *Une base de buts stratifiée sur $\mathscr{O}$ est un vecteur $GB^{strat}_{\mathscr{O}} = (GB_{\mathscr{O},1}, \ldots, GB_{\mathscr{O},n})$, où $GB_{\mathscr{O},i}$ est un multiensemble[3] de formules logiques de $\mathcal{L}_{\mathscr{O}}$.*

Nous noterons comme pour les bases de buts non stratifiées $\mathcal{GB}^{strat}_{\mathscr{O}}$ l'ensemble des bases de buts stratifiées sur $\mathcal{L}_{\mathscr{O}}$.

Cette notion de buts stratifiés est à la base d'un certain nombre de langages s'appuyant sur les buts et leur priorité pour définir une relation de préférence.

Définition 3.13 (Langages à base de buts stratifiés) *Les langages de représentation de préférences à base de buts stratifiés $\mathcal{R}_{best-out}$, $\mathcal{R}_{discrimin}$ et $\mathcal{R}_{leximin}$ sont tous fondés sur un langage $\mathcal{GB}^{strat}_{\mathscr{O}}$.*

Étant donnée une base de buts stratifiée $GB^{strat}_{\mathscr{O}} = (GB_{\mathscr{O},1}, \ldots, GB_{\mathscr{O},n})$ sur $\mathcal{L}_{\mathscr{O}}$, les structures de préférences induites par ces langages sont définies comme suit :

- ▷ Langage best-out *: $\pi \succeq^{GB^{strat}_{\mathscr{O}}}_{bestout} \pi'$ si et seulement si* $\min\{i \mid nonsat(\pi, GB_{\mathscr{O},i}) \neq \varnothing\} \geq \min\{i \mid nonsat(\pi', GB_{\mathscr{O},i}) \neq \varnothing\}$.
- ▷ Langage discrimin *: $\pi \succ^{GB^{strat}_{\mathscr{O}}}_{discrimin} \pi'$ si et seulement si $\exists i \leq n$ tel que $sat(\pi, GB_{\mathscr{O},i}) \supset sat(\pi', GB_{\mathscr{O},i})$ et $\forall j < i$, $sat(\pi, GB_{\mathscr{O},j}) = sat(\pi', GB_{\mathscr{O},j})$. $\pi \sim^{GB^{strat}_{\mathscr{O}}}_{discrimin} \pi'$ si et seulement si $\forall i \leq n$, $sat(\pi, GB_{\mathscr{O},i}) = sat(\pi', GB_{\mathscr{O},i})$.*
- ▷ Langage leximin *: $\pi \succ^{GB^{strat}_{\mathscr{O}}}_{discrimin} \pi'$ si et seulement si $\exists i \leq n$ tel que $|sat(\pi, GB_{\mathscr{O},i})| > |sat(\pi', GB_{\mathscr{O},i})|$ et $\forall j < i$, $|sat(\pi, GB_{\mathscr{O},j})| = |sat(\pi', GB_{\mathscr{O},j})|$. $\pi \sim^{GB^{strat}_{\mathscr{O}}}_{discrimin} \pi'$ si et seulement si $\forall i \leq n$, $|sat(\pi, GB_{\mathscr{O},i})| = |sat(\pi', GB_{\mathscr{O},i})|$.*

En d'autres termes, le langage *best-out* compare les alternatives en considérant le niveau de priorité du but non satisfait le plus prioritaire. Ce langage souffre, à l'instar de la fonction d'utilité collective min (voir chapitre 1) d'un «effet de noyade» : le langage est incapable de discriminer deux alternatives dont le plus haut niveau de priorité i du but non satisfait est identique, et inhibe l'effet de toutes les formules de niveau $j > i$ (ainsi que les autres formules de niveau i). Les critères *discrimin* et *leximin* raffinent l'ordre *best-out* sans pâtir de l'effet de noyade, en comparant par ordre de priorité les ensembles *sat* respectivement selon un critère d'inclusion et de cardinalité.

Notons que les langages $\mathcal{R}_{best-out}$ et $\mathcal{R}_{leximin}$ sont capables de générer l'ensemble des structures de préférence ordinales complètes, et que le langage $\mathcal{R}_{discrimin}$ peut engendrer l'ensemble des structures de préférence.

Exemple 3.7 Considérons toujours un espace d'alternatives fondé sur les objets $\mathscr{O} = \{o_1, o_2, o_3, o_4, o_5\}$, et une base de buts stratifiée formée des multiensembles : $GB_{\mathscr{O},1} = \{\varphi_1 = \mathbf{o_1} \vee (\mathbf{o_2} \wedge \mathbf{o_3}), \varphi_2 = \mathbf{o_4}\}$, $GB_{\mathscr{O},2} = \{\varphi_3 = \mathbf{o_5}, \varphi_4 = \neg\mathbf{o_5}, \varphi_5 = \mathbf{o_2} \wedge \mathbf{o_3}\}$ et $GB_{\mathscr{O},3} = \{\varphi_6 = \mathbf{o_4} \wedge \mathbf{o_5}\}$. Considérons les deux alternatives $\pi = \{o_1, o_4, o_5\}$ et $\pi' = \{o_2, o_3, o_4\}$. Nous avons :

	$GB_{\mathscr{O},1}$	$GB_{\mathscr{O},2}$	$GB_{\mathscr{O},3}$
$sat(\pi, .)$	φ_1, φ_2	φ_3	φ_6
$sat(\pi', .)$	φ_1, φ_2	φ_4, φ_5	$\varnothing$

[3] Un multiensemble est un ensemble dans lequel les éléments peuvent être dupliqués : $\{1, 2, 5, 2, 2\}$.

Cela nous donne les relations suivantes : $\pi \sim^{GB^{strat}_{\mathscr{O}}}_{bestout} \pi'$, π et π' sont incomparables selon $\succeq^{GB^{strat}_{\mathscr{O}}}_{discrimin}$, et $\pi' \succ^{GB^{strat}_{\mathscr{O}}}_{leximin} \pi$.

3.2.2.3 Préférences cardinales

Base de buts simple L'équivalent cardinal du langage $\mathcal{R}_{Pareto}$, fondé sur l'inclusion entre buts satisfaits, s'appuie sur le cardinal des ensembles de buts satisfaits :

Définition 3.14 (Langage $\mathcal{R}_{Card}$) *Le langage de représentation de préférences $\mathcal{R}_{Card}$ est défini par :*

- ▷ $\mathcal{L}_{Card}(\mathscr{O}) = \mathcal{GB}_{\mathscr{O}}$;
- ▷ *étant donnée une base de buts $GB_{\mathscr{O}}$, la structure de préférences ordinale $u^{GB_{\mathscr{O}}}_{Card}$ est telle que :*

$$\forall \pi \in \wp(\mathscr{O}),\ u(\pi) = |sat(\pi, GB_{\mathscr{O}})|.$$

Ce langage est capable de représenter l'ensemble des fonctions d'utilité à valeurs dans $\mathbb{N}$, d'une manière relativement basique.

L'une des extensions classiques de ce langage est fondée sur la notion de distance entre alternatives :

Définition 3.15 (pseudo-distance) *Une* pseudo-distance *est une fonction $d : \wp(\mathscr{O}) \times \wp(\mathscr{O}) \to \mathbb{N}$ telle que :*

- ▷ $\forall \pi, \pi',\ d(\pi, \pi') = 0 \Leftrightarrow \pi = \pi'$;
- ▷ $\forall \pi, \pi',\ d(\pi, \pi') = d(\pi', \pi)$.

Par exemple, la distance de Hamming est définie comme le nombre de symbole propositionnels dont la valeur diffère entre π et π'. De manière plus basique, la distance binaire est définie comme étant égale à 1 si les deux alternatives diffèrent, et 0 sinon.

Les langages de représentation logique à base de distances sont une extension directe du langage $\mathcal{R}_{Card}$. Dans ce dernier langage, on essayait de satisfaire le maximum de formules. Dans les langages à base de distances, on s'intéresse non plus à la satisfaction des formules de la base de buts, mais à la «distance» entre l'alternative considérée et ces formules (plus une alternative est «loin» des buts, moins elle est intéressante).

Définition 3.16 (Langage $\mathcal{R}_d$) *Le langage de représentation de préférences $\mathcal{R}_d$ associé à la fonction de pseudo-distance d est défini par :*

- ▷ $\mathcal{L}_d(\mathscr{O}) = \mathcal{GB}_{\mathscr{O}}$;
- ▷ *étant donnée une base de buts $GB_{\mathscr{O}} = \{G_1, \ldots, G_p\}$, la structure de préférences ordinale $u^{GB_{\mathscr{O}}}_d$ est telle que :*

$$\forall \pi \in \wp(\mathscr{O}),\ u(\pi) = f(d(\pi, G_1), \ldots, d(\pi, G_p)),$$

où f est une fonction de $\mathbb{N}$ dans $\mathscr{V}$.

Notons que si d est la distance binaire et que f est égal à $(d_1, \ldots, d_p) \mapsto p - \sum_{i=1}^{p} d_i$, ce langage est complètement équivalent au langage $\mathcal{R}_{Card}$, puisqu'il associe à chaque alternative la même utilité que celle associée à la même alternative par $\mathcal{R}_d$.

Buts pondérés À l'instar de l'introduction de priorités pour les langages ordinaux fondés sur les bases de buts, il est possible de raffiner le langage $\mathcal{R}_{Card}$ en introduisant des poids sur les formules, assortis de fonctions d'agrégation permettant de calculer l'utilité d'une alternative à partir des formules satisfaites.

Définition 3.17 (Base de buts pondérée) *Une* base de buts pondérée *sur $\mathscr{O}$ est un ensemble $GB_{\mathscr{O}}^{weighted}$ de couples $\delta = (\varphi(\delta), w(\delta))$, où $\varphi(\delta)$ est une formule logique de $\mathcal{L}_{\mathscr{O}}$ et $w(\delta)$ une valeur dans un espace de valuation totalement ordonné $\langle \mathscr{V}, \succeq \rangle$.*

Nous noterons comme à l'accoutumée $\mathcal{GB}_{\mathscr{O}}^{weighted}$ l'ensemble des bases de buts pondérées sur $\mathcal{L}_{\mathscr{O}}$.

Définition 3.18 (Langage des buts pondérés) *Le langage de représentation de préférences $\mathcal{R}_{weighted}$ est défini par :*

- ▷ $\mathcal{L}_{weighted}(\mathscr{O}) = \mathcal{GB}_{\mathscr{O}}^{weighted}$;
- ▷ *la structure de préférences cardinale induite $u_{weighted}^{GB_{\mathscr{O}}^{weighted}}$ définie par :*

$$\forall \pi \in \wp(\mathscr{O}),\ u(\pi) = f_1(f_2(\{w(\delta) \mid \pi \vDash \varphi(\delta)\}), f_3(\{w(\delta) \mid \pi \nvDash \varphi(\delta)\})),$$

où f_1, f_2 et f_3 sont trois fonctions sur $\mathscr{V}$.

Le calcul de l'utilité d'une alternative est donc calculé par agrégation séparée des poids des formules satisfaites et des poids des formules insatisfaites. Dans le langage de représentation compacte dédié au partage que nous introduirons à la toute fin de ce chapitre, la représentation des préférences des agents est fondée sur un langage à base de buts pondérés, mais dans lequel on ne prend en compte que les formules satisfaites (en d'autres termes, $f_1 = (x, y) \mapsto x$).

Ce langage de représentation est capable d'exprimer toutes les fonctions d'utilité à valeurs dans $\mathscr{V}$. La puissance expressive des restrictions (clauses, cubes, poids positifs uniquement,...) des langages à base de logique pondérée dans lesquels on ne considère que les formules satisfaites a été étudiée en détail dans [Chevaleyre *et al.*, 2006b].

Exemple 3.8 Considérons encore un espace d'alternatives fondé sur les objets $\mathscr{O} = \{o_1, o_2, o_3, o_4, o_5\}$, et la base de buts pondérée suivante : $GB_{\mathscr{O}}^{weighted} = \{\delta_1 = (\mathbf{o_1} \vee (\mathbf{o_2} \wedge \mathbf{o_3}), 5), \delta_2 = (\mathbf{o_4}, 2), \delta_3 = (\mathbf{o_5}, 2), \delta_4 = (\neg \mathbf{o_5}, 3), \delta_5 = (\mathbf{o_2} \wedge \mathbf{o_3}, 10)\}$. Nous considérons deux langages de buts pondérés : le premier est fondé sur les fonctions $+$, $+$ et $-$, et le second sur les fonctions $+$, max et $-\max$. Nous considérons les deux alternatives $\pi = \{o_1, o_4, o_5\}$ et $\pi' = \{o_2, o_3, o_4\}$, et nous notons pour chaque alternative et chaque formule pondérée le poids de la formule barré si elle n'est pas satisfaite, et non barré si elle est satisfaite :

	δ_1	δ_2	δ_3	δ_4	δ_5	$u(+,+,+)$	$u(+, \max, -\max)$
π	5	2	2	~~3~~	~~10~~	$5+2+2-3-10=-4$	$\max\{5,2,2\} - \max\{3,10\} = -5$
π'	5	2	~~2~~	3	10	$5+2-2+3+10=18$	$\max\{5,2,3,10\} - \max\{2\} = 8$

Langages de compilation de connaissances Concluons cette sous-section en notant l'existence de cadres de représentation compacte de fonctions d'utilités, qui sont issus de la communauté de la compilation des connaissances :

- ▷ Les *diagrammes de décision algébriques* (ADD), permettant de représenter des fonctions pseudo-booléennes, c'est-à-dire des fonctions de $\mathbb{B}$ dans $\mathbb{N}$: le principe de représentation est assez similaire aux BDD, mais les nœuds terminaux contiennent des valeurs numériques. Pour plus de précisions on pourra se référer à [Bahar *et al.*, 1993].

▷ Les *VNNF*, qui généralisent le cadre des langages NNF au cas de fonction non exclusivement booléennes, dont la généricité permet de représenter un grand nombre de cadres de représentation compacte [Fargier et Marquis, 2007], et dont l'approche n'est pas sans rappeler celle du cadre PFU [Pralet, 2006].

3.2.3 Préférences *ceteris paribus* et CP-*nets*

3.2.3.1 Préférences *ceteris paribus*

Dans les langages de représentation de préférences à base de logique propositionnelle introduits ci-avant, les préférences sur des alternatives étaient exprimées en termes de *buts* à atteindre (ou éventuellement d'anti-buts si les pondérations étaient négatives). Si la notion de préférence entre des formules logiques pouvait y apparaître (sous la forme de priorités, ou d'utilités associées aux buts), aucune sémantique spécifique n'y était pour autant associée. L'idée des préférences *ceteris paribus* ou conditionnelles que nous allons introduire maintenant est de donner un sens précis à la notion de préférence entre formules logiques. Nous allons expliquer de façon semi-formelle de quelle manière les structures de préférence sur des alternatives peuvent être construites à partir de relations de préférence informelle sur des formules logiques.

Les logiques des préférences du type *ceteris paribus* introduites dans [Von Wright, 1963] sont fondées sur la constatation suivante : les individus expriment souvent des préférences se référant à des ensembles d'options, et non à des alternatives isolées. En terme de formules logiques, cela signifie que les préférences des agents ne sont pas représentables par des formules complètes, mais par des formules partielles. L'interprétation de la relation de préférence sur ces formules pose alors plusieurs problèmes.

▷ Que signifie exactement l'expression «la formule φ est préférée à la formule ψ» pour un agent, si φ et ψ ne sont pas mutuellement exclusives ?

▷ Comment interpréter la relation de préférence $\geq$ sur les formules en terme de relation de préférence $\succeq$ sur les alternatives ?

Répondre à la première question revient à lever des ambiguïtés liées à la non-exclusion des formules, en restreignant son application aux cas où les deux formules ne sont pas satisfaites en même temps. Considérons par exemple la préférence suivante : «je préfère le tracteur à la tondeuse à gazon» (dans le problème de l'héritage), noté tracteur$\triangleright$tondeuse. Rien ne s'oppose à ce que l'agent ne reçoive le tracteur et la tondeuse à gazon. La traduction intuitive est bien entendu la suivante : tracteur $\wedge$ $\neg$tondeuse $>$ tondeuse $\wedge$ $\neg$tracteur, c'est-à-dire que l'agent préfère avoir le tracteur sans la tondeuse que la tondeuse sans le tracteur. Certains cas sont légèrement plus compliqués, car une formule peut être la conséquence logique de l'autre (par exemple «je préfère avoir la tondeuse et le tracteur que la tondeuse seule» : tondeuse$\wedge$tracteur$\triangleright$tondeuse). Bien entendu, cette préférence doit être comprise comme tondeuse $\wedge$ tracteur $\triangleright$ tondeuse $\wedge$ $\neg$tracteur. En résumé, la traduction d'une préférence $\varphi \triangleright \psi$ sur des formules non mutuellement exclusives s'interprète comme $\varphi \setminus \psi > \psi \setminus \varphi$, avec $\varphi \setminus \psi = \varphi$ si $\varphi \wedge \neg\psi$ est inconsistant, et $\varphi \wedge \neg\psi$ sinon.

Il est possible d'enrichir quelque peu la relation $>$ en tenant compte de contextes logiques. De tels contextes permettent d'exprimer des préférences du type «si j'hérite des terres cultivables, alors je préfère le tracteur à la tondeuse», qui ne sont valables que dans un contexte donné. On notera $\lambda : \varphi \triangleright \psi$, qui exprime $\lambda \wedge \varphi \triangleright \lambda \wedge \psi$. Il reste maintenant à interpréter le sens de la relation $\geq$ sur les formules logiques en terme de structure de préférence ordinale sur les alternatives. Si les alternatives concernées par les formules φ et ψ sont uniques, en d'autres termes si les formules sont complètes, l'interprétation de $\geq$ ne pose aucun problème. En revanche, si ce n'est pas le cas, il faut lever les ambiguïtés liées à la relation $\geq$. Par exemple, si l'agent dit «je préfère le tracteur à la tondeuse

à gazon», il ne veut certainement pas dire « quoi qu'il arrive, je préfère le tracteur à la tondeuse à gazon ». Ainsi, si on lui laisse le choix entre hériter du tracteur seul et toucher l'intégralité de l'héritage (tondeuse y compris) sauf le tracteur, il y a fort à parier que la préférence exprimée sur le tracteur et la tondeuse ne tiendra plus.

On peut traiter ce problème en interprétant les formules « *ceteris paribus* », c'est-à-dire «toutes autres choses étant égales». Formellement, étant donnée une relation de préférence $\succeq$ sur les alternatives, et une préférence élémentaire $\varphi > \psi$ sur deux formules logiques mutuellement exclusives, $\succeq$ satisfait la relation $>$ *ceteris-paribus* si et seulement si, pour tout couple d'alternatives (π, π'), on a $\pi \succeq \pi'$ si et seulement si :

▷ $\pi \vDash \varphi$ (donc $\pi \nvDash \psi$),

▷ et $\forall \mathbf{o} \in Var(\mathscr{O}) \setminus (Var(\varphi) \cup Var(\psi))$, $\pi \vDash \mathbf{o} \Leftrightarrow \pi' \vDash \mathbf{o}$.

En d'autres termes, une alternative est préférée à une autre relativement à une préférence $\varphi > \psi$ si et seulement si elle vérifie φ et ces deux alternatives sont identiques pour toutes les variables qui ne concernent ni φ ni ψ. Notons qu'il existe d'autres interprétations possibles pour la relation $\varphi > \psi$ que celle introduite ici. Nous ne les détaillerons pas.

On peut ainsi définir la notion de relation *ceteris paribus* induite par un ensemble de préférences GB_{CP} élémentaires du type $\varphi > \psi$ ou $\varphi \sim \psi$ comme étant l'intersection de toutes les relations $\succeq$ sur les alternatives qui satisfont les relations $>$ et $\sim$ *ceteris paribus*. Il peut ne pas exister de telle structure de préférence, s'il y a un cycle impliquant au moins une préférence stricte dans la base de buts (du type $\{\mathbf{a} > \neg\mathbf{a}, \neg\mathbf{a} > \mathbf{a}\}$). Dans ce cas la base de buts est dite *incohérente*.

Nous résumons dans le tableau ci-dessous les trois étapes de construction d'une relation de préférence *ceteris-paribus* à partir de préférences exprimées sur les formules :

Expression humaine des préférences	Clarification du flou lié à la non contradiction entre formules	Interprétation *ceteris paribus* de la relation $>$
Relation de préférence sur les formules logiques	Relation de préférence sur des formules logiques mutuellement exclusives	Relation de préférence sur les alternatives
$\varphi \triangleright \psi$	$\varphi \setminus \psi > \psi \setminus \varphi$	$\pi \geq \pi'$ ssi $\pi \vDash \varphi \setminus \psi$ et $\forall o \in \mathscr{O} \setminus (Var(\psi) \cup Var(\psi))$, $\pi(o) = \pi'(o)$.

3.2.3.2 CP-*nets*

Un langage de représentation graphique de préférences ordinales fondé sur des préférences *ceteris paribus* particulières a été introduit récemment dans la communauté de l'intelligence artificielle : celui des CP-*nets* [Domshlak, 2002; Boutilier *et al.*, 2004a,b]. Ce langage est à la fois moins général que le langage des préférences *ceteris paribus*, car il se restreint à une interprétation particulière de la préférence $\triangleright$, et à la fois plus général car il ne se limite pas à des préférences sur des variables binaires. La notion fondamentale sur laquelle est construit ce langage est la notion d'*indépendance préférentielle* entre variables :

Définition 3.19 (Indépendance préférentielle) *Soient $\{\mathscr{S}_x, \mathscr{S}_y, \mathscr{S}_z\}$ une partition de l'ensemble des variables de décision $\mathscr{X}$ et $\succeq$ une structure de préférence ordinale sur l'espace combinatoire engendré par $\mathscr{X}$. Alors $\mathscr{S}_x$ est* préférentiellement indépendant *de $\mathscr{S}_y$ étant donné $\mathscr{S}_z$ pour la relation $\succeq$ si et seulement si pour toute paire d'instanciations (v_x, v'_x) sur $\mathscr{S}_x$, toute paire*

d'instanciations (v_y, v'_y) *sur* $\mathscr{S}_y$ *et toute instanciation* v_z *sur* $\mathscr{S}_z$ *:*

$$\langle v_x, v_y, v_z\rangle \succeq \langle v'_x, v_y, v_z\rangle \text{ si et seulement si } \langle v_x, v'_y, v_z\rangle \succeq \langle v'_x, v'_y, v_z\rangle.$$

Nous rappelons que la notation $\langle v_x, v_y, v_z\rangle$ désigne l'interprétation identique à v_x sur les variables de $\mathscr{S}_x$, identique à v_y sur les variables de $\mathscr{S}_y$, et identique à v_z sur les variables de $\mathscr{S}_z$. Dans cette définition, $\mathscr{S}_z$ est interprété comme étant le contexte de la préférence (correspondant au λ introduit pour les préférences *ceteris paribus*), $\mathscr{S}_x$ est l'ensemble des variables sur lesquelles porte la préférence, et $\mathscr{S}_y$ est l'ensemble des variables dont la préférence «ne dépend pas», c'est-à-dire l'ensemble des «toutes autres choses» dans les préférences *ceteris paribus*.

Le langage des CP-*nets* est un langage de représentation graphique s'appuyant sur la notion d'indépendance préférentielle, où l'ensemble $\mathscr{S}_x$ est restreint à un singleton. Les dépendances préférentielles entre variables y sont représentées par un graphe dirigé. On pourra noter l'analogie entre ce formalisme et celui des réseaux bayésiens [Pearl, 1988].

Définition 3.20 (CP-*net*) *Un* CP-*net sur un espace combinatoire fondé sur les variables de décision* $\mathscr{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ *est un couple* $(\mathcal{G}, \mathscr{C})$*, où* $\mathcal{G}$ *est un graphe dirigé dont l'ensemble de sommets est isomorphe à* $\mathscr{X}$*, et* $\mathscr{C}$ *est un ensemble de tables de préférences conditionnelles :* $\mathscr{C} = \{C(\mathbf{x_i}) \mid \mathbf{x_i} \in \mathscr{X}\}$.

Pour tout $\mathbf{x_i} \in \mathscr{X}$ $Pa(\mathbf{x_i})$*, on note l'ensemble des variables dont les nœuds dans* $\mathcal{G}$ *sont les parents du nœud de* $\mathbf{x_i}$*. Chaque préférence conditionnelle* $C(\mathbf{x_i})$ *associe à toute instanciation* v *de* $\{\mathbf{x_j} \in Pa(\mathbf{x_i})\}$ *une relation de préférence stricte* $\succ_v^{\mathbf{x_i}}$ *sur* $\mathscr{D}_{\mathbf{x_i}}$.

Exemple 3.9 Considérons à nouveau l'exemple de l'héritage dans lequel un agent doit exprimer ses préférences sur cinq «objets» : la maison (m), les terres cultivables (ch), le tracteur (tr) et la tondeuse à gazon (to). De manière informelle, voici quelles sont les préférences de l'agent : «le tracteur ne m'intéresse pas, sauf si j'obtiens les terres cultivables», «dans ce cas-là, je ne veux pas m'encombrer à la fois d'un tracteur et d'une tondeuse à gazon : le tracteur suffira», et «la tondeuse ne m'intéresse pas en soit, sauf si j'obtiens la maison et pas le tracteur, auquel cas il faudra bien que je coupe mon herbe».

Ces préférences peuvent se traduire sous la forme d'un CP-*net* impliquant des variables binaires, présenté dans la figure 3.2. La relation de préférence induite par ce CP-*net* est présentée dans la figure 3.3.

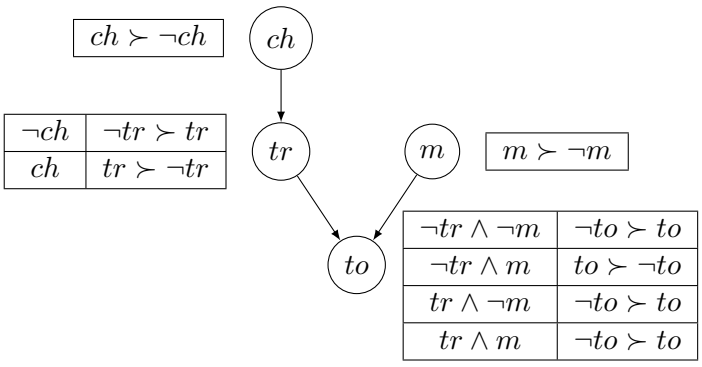


Figure 3.2 — Le CP-*net* associé aux préférences de l'exemple 3.9.

La sémantique d'un CP-*net* (c'est-à-dire la fonction $\mathcal{I}$ qui associe à un CP-*net* la structure de préférence qui lui correspond) est fondée sur la notion de relation d'ordre cohérente avec les tables :

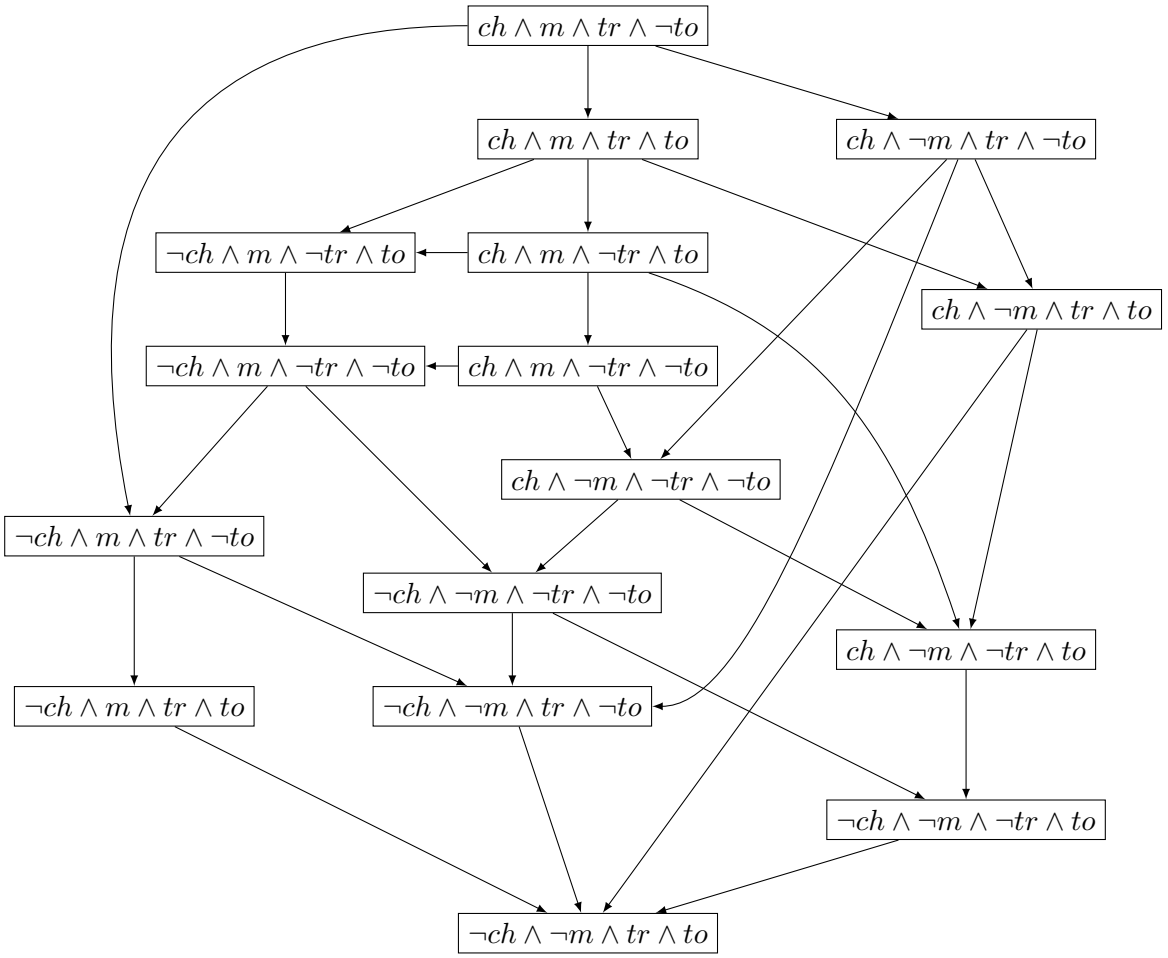


Figure 3.3 — Les préférences induites par le CP-*net* de la figure 3.2 associé aux préférences de l'exemple 3.9. La signification des arcs est la même que dans la figure 1.1 page 21 : «est préféré à».

Définition 3.21 (CP-*net* satisfiable) *Soit $(\mathcal{G}, \mathscr{C})$ un* CP-*net sur un espace combinatoire fondé sur les variables de décision $\mathscr{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$, et soit $\succ$ une relation de préordre strict sur l'ensemble des instanciations possibles des variables de $\mathscr{X}$.*

La relation $\succ$ satisfait $\succ_v^{\mathbf{x_i}}$*, avec v une instanciation de $Pa(\mathbf{x_i})$ si et seulement si pour toute instanciation v_y de $\mathscr{X} \setminus \left(\{\mathbf{x_i}\} \cup \bigcup_{\mathbf{x_j} \in Pa(\mathbf{x_i})} \{\mathbf{x_j}\}\right)$, et toute paire (v_x, v'_x) d'instanciations de $\mathbf{x_i}$ on a $\langle v_y, v_x, v\rangle \succ \langle v_y, v'_x, v\rangle$ lorsque $v_x(\mathbf{x_i}) \succ_v^{\mathbf{x_i}} v'_x(\mathbf{x_i})$. La relation* $\succ$ satisfait *la table $C(\mathbf{x_i})$ si et seulement si elle satisfait $\succ_v^{\mathbf{x_i}}$ pour toute instanciation v de $Pa(\mathbf{x_i})$.*

Un CP-net est satisfiable *si et seulement s'il y a au moins une relation $\succ$ qui le satisfait.*

Cette définition spécifie comment les tables de préférences conditionnelles doivent être interprétées dans le graphe : pour chaque variable $\mathbf{x}$, le CP-*net* spécifie une relation de préférence portant sur son domaine, et dépendant de la valeur des parents de $\mathbf{x}$ dans le graphe. Une relation de préférence sur $\mathbf{x}$ et $Pa(\mathbf{x})$ est «compatible» avec le CP-*net* si elle respecte la relation de préférences sur $\mathscr{D}_\mathbf{x}$ pour toute instanciation des parents de $\mathbf{x}$.

De manière générale, il peut exister plusieurs ordres qui satisfont un CP-*net* donné. La structure de préférence induite par un CP-*net* est donc définie comme l'intersection de tous ces ordres, c'est-à-dire :

Définition 3.22 (Structure de préférence induite par un CP-*net*) *Étant donné un CP-net $(\mathcal{G}, \mathscr{C})$, la structure de préférence ordinale induite par $(\mathcal{G}, \mathscr{C})$ est telle que :*

$$v \succ_{CP-net}^{(\mathcal{G},\mathscr{C})} v' \text{ si et seulement si } v \succ v' \text{ dans toute relation } \succ \text{ satisfaisant } (\mathcal{G}, \mathscr{C}).$$

La plupart des travaux sur les CP-*nets* font l'hypothèse d'acyclicité du graphe. On peut montrer [Boutilier *et al.*, 2004a] que dans ce cas précis (acyclicité du CP-*net*), le réseau est toujours satisfiable. En outre, l'acyclicité du graphe implique des propriétés intéressantes sur la complexité des tâches associées au raisonnement sur la relation de préférence : ainsi, la comparaison d'alternatives et la recherche d'une alternative non dominée sont computationnellement raisonnables [Boutilier *et al.*, 2004b].

En revanche, le gain computationnel obtenu par ce formalisme graphique a une contrepartie en terme d'expressivité, puisque les CP-*nets* ne sont pas capables de représenter tous les préordres possibles.

Certains travaux récents s'intéressent à l'extension de ce cadre de représentation de préférences. Ainsi par exemple, [Brafman et Domshlak, 2002] introduit le cadre des TCP-*nets*, qui étendent le cadre des CP-*nets* en introduisant des importances relatives entre variables. Ce cadre est fondé sur un modèle graphique, tout comme les CP-*nets*, mais dont les arcs peuvent être de trois types différents :

- ▷ arcs de type dépendance préférentielle, dirigés et dont la sémantique est similaire à celle des arcs des CP-*nets* ;
- ▷ arcs de relation d'importance entre variables, dirigés et exprimant le fait qu'une variable est plus importante qu'une autre ;
- ▷ arcs d'importance conditionnelle, non dirigés et exprimant une relation d'importance entre variables, conditionnée par d'autres variables.

D'autres travaux se sont intéressés à la transposition des CP-*nets* dans le domaine des préférences quantitatives. Les UCP-*nets*, introduits dans [Boutilier *et al.*, 2001], sont fondés sur le même modèle graphique que les CP-*nets*, mais les tables de préférence conditionnelle contiennent des utilités au lieu de contenir des relations de préférence. Le calcul de l'utilité d'une alternative complète se fait en utilisant la notion d'indépendance additive généralisée (GAI), que nous allons introduire dans la prochaine sous-section.

3.2.3.3 Application au problème de partage

On peut légitimement s'interroger sur la pertinence des langages à base de préférences conditionnelle, et plus précisément du langage des CP-*nets* dans le cadre du partage. Nous avons montré dans l'exemple 3.9 de quelle manière les CP-*nets* peuvent être utilisés pour la représentation compacte de préférences sur des objets. Cependant, l'exemple est-il vraiment réaliste ?

La principale limitation des CP-*nets* est la restriction de la portée des tables conditionnelles à une variable unique. Cette restriction est valable dans le cas de problèmes tels que celui du repas (exemple 3.1) : un agent peut naturellement exprimer le fait qu'étant donné le choix du plat, il préfère un vin rouge à un vin blanc. Cependant, dans le cadres des problèmes de partage, l'expression d'une table de préférences conditionnelles revient à se prononcer sur l'attribution d'un objet ou non. Ce n'est pas très réaliste. Même si un agent ne désire pas un objet o, il n'exprimera probablement pas intuitivement cette préférence par $\neg o \succ o$ (en outre, dans la plupart des problèmes, l'agent peut tout simplement mettre de côté ou vendre un objet qu'il ne désire pas ; il n'y a donc pas réellement dans ce cas-là de préférence «négative»). En revanche, il sera peut-être plus enclin à exprimer le fait qu'il préfère avoir un objet o' que l'objet o.

Une telle préférence $o' \succ o$ n'est pas représentable dans le cadre strict des CP-*nets*. Cependant, cette idée d'importance relative entre la satisfaction des variables est à la base du cadre des TCP-*nets* [Brafman et Domshlak, 2002] que nous avons brièvement évoqué. Cette approche semble particulièrement intéressante pour le problème de partage. On pourrait ainsi envisager de construire un nouveau cadre de représentation compacte de préférences dédié aux problèmes d'allocations, et fondés sur les relations d'importance relative et d'importance relative conditionnelle introduites dans les TCP-*nets*. Cette ébauche de cadre semble *a priori* prometteuse.

3.2.4 Préférences additives généralisées

Si les langages de représentation de préférences introduits jusqu'ici tirent parti de la logique propositionnelle pour représenter des relations de manière compacte, les langages que nous allons introduire dans cette section utilisent des indépendances additives entre variables dans le même but[4]. L'exploitation de ces indépendances additives pour la représentation compacte de préférences a conduit aux langages k-additifs pour le partage de ressources (donc dédiés aux espaces combinatoires d'objets), et aux GAI-nets pour les espaces d'alternatives combinatoires fondés sur des variables non binaires. Nous supposerons dans toute cette sous-section (sauf spécification contraire explicite) que l'espace de valuation des utilités $\mathscr{V}$ est numérique ($\mathbb{R}$ ou $\mathbb{N}$ par exemple).

3.2.4.1 Langages de lots k-additifs

La manière la plus simple d'exprimer des utilités sur un espace d'objets sans tomber dans le piège combinatoire est de supposer une *indépendance additive* entre les objets : en d'autres termes, on suppose que $\forall(\pi_1, \pi_2) \in \wp(\mathscr{O}) \times \wp(\mathscr{O})$, $u(\pi_1 \cup \pi_2) = u(\pi_1) + u(\pi_2) - u(\pi_1 \cap \pi_2)$. De telles fonctions d'utilité sont dites *modulaires* [Chevaleyre *et al.*, 2005b; Rosenschein et Zlotkin, 1994], et peuvent être représentées uniquement par l'attribution d'une utilité à l'ensemble vide et à tous les singletons d'objets. En effet, on peut vérifier aisément le fait que, pour une fonction d'utilité modulaire u :

$$u(\pi) = u(\varnothing) + \sum_{o \in \pi} u(\{o\}).$$

Si de plus $u(\varnothing) = 0$ (ce que l'on suppose habituellement), la fonction d'utilité est dite *additive*.

Si l'hypothèse de modularité de la fonction d'utilité permet d'éviter l'explosion combinatoire, en revanche, elle exclue toute possibilité d'expression de dépendances préférentielles du type complémentarités ou substituabilités entre les variables. Afin de pallier ce défaut, la classe des langages k-additifs autorise l'expression d'utilités sur des ensembles d'objets, mais en limitant la taille maximale k des lots sur lesquels exprimer les utilités élémentaires, et en conservant l'interprétation additive de l'utilité :

Définition 3.23 (Langage k-additif) *Un langage d'expression de préférences k-additif est défini par :*

- ▷ *$\mathcal{L}_{k-add}$: un élément de $\mathcal{L}_{k-add}(\mathscr{O})$ est un ensemble $\mathscr{M}$ de couples (π, α_π), où π est un sous-ensemble de $\mathscr{O}$ de taille au plus k, et α_π un nombre de l'espace de valuation $\mathscr{V}$;*
- ▷ *$\mathcal{I}_{k-add}$: la fonction d'utilité $u^{\mathscr{M}}_{k-add}$ est définie par*

$$u^{\mathscr{M}}_{k-add}(\pi) = \sum_{\pi' \subseteq \pi} \alpha_{\pi'} \text{ (avec } \alpha_{\pi'} = 0 \text{ si } \pi' \text{ n'apparaît pas dans } \mathscr{M}).$$

[4] En cela, ils se rapprochent, dans un contexte légèrement différent, des formalismes fondés sur les réseaux de contrainte.

Toute fonction d'utilité exprimée sous forme explicite (*bundle form*, dans la terminologie des problèmes d'allocation de ressources), c'est-à-dire associant à chaque sous-ensemble de $\mathscr{O}$ une utilité, est exprimable dans un langage k-additif. Il suffit, pour représenter une fonction d'utilité u sous forme explicite, de calculer les coefficients α de manière inductive [Chevaleyre *et al.*, 2004] :

- ▷ $\alpha^u_{\varnothing} = u(\varnothing)$;
- ▷ $\alpha^u_{\pi} = u(\pi) - \sum_{\pi' \subsetneq \pi} \alpha^u_{\pi'}$ pour tout $\pi \subseteq \mathscr{O}$ avec $\pi \neq \varnothing$.

Les coefficients α^u_{π} peuvent s'exprimer sous la forme équivalente (non inductive) suivante :

$$\alpha^u_{\pi} = \sum_{\pi' \subsetneq \pi} (-1)^{|\pi \setminus \pi'|} u(\pi').$$

La fonction qui à toute fonction d'utilité u associe la fonction ensembliste $\pi \mapsto \alpha^u_{\pi}$ est appelée *transformation de Möbius*. Cette transformation, qui associe à toute fonction d'utilité sa forme k-additive, est étudiée dans le domaine des mesures floues [Grabisch, 1997] afin de réduire la complexité de représentation des mesures discrètes.

Une fonction d'utilité quelconque u est dite k-additive si sa transformée de Möbius (unique) est telle que pour tout ensemble π de taille supérieure strictement à k, on a $\alpha^u_{\pi} = 0$. Cette notion de transformation de Möbius met en évidence le fait qu'un langage k-additif n'est complètement expressif que si $k = |\mathscr{O}|$, car un tel langage n'est capable d'exprimer que les fonctions dont la transformée de Möbius est telle que $\alpha^u_{\pi} = 0$ pour tout π de taille strictement supérieure à k. À titre d'exemple, on pourra vérifier que la fonction d'utilité telle que $u(\mathscr{O}) = 1$ et $u(\pi) = 0$ pour tout $\pi \subsetneq \mathscr{O}$ n'est pas exprimable dans un langage k-additif tel que $k < |\mathscr{O}|$.

On pourra remarquer de plus qu'un langage k-additif n'est ni plus compact ni moins compact que la représentation explicite : si la représentation de certaines fonctions d'utilités requiert exponentiellement moins d'espace dans un langage k-additif que ne le requiert la représentation explicite, d'autres fonctions d'utilités exigent un espace exponentiellement plus important dans un langage k-additif que pour la représentation explicite. Les langages k-additifs ne sont donc adaptés que pour la représentation de certains types de fonctions d'utilité qui font apparaître de manière naturelle une structure additive.

Exemple 3.10 Il est possible d'exprimer des notions de complémentarité et de substituabilité entre les objets dans un langage k-additif. Revenons sur l'exemple de l'héritage, et les préférences exprimées par l'agent dans l'exemple 3.9. L'agent doit pouvoir être capable d'exprimer une complémentarité entre les terres cultivables et le tracteur, entre la maison et la tondeuse, mais en revanche une substituabilité entre le tracteur et la tondeuse (avoir les deux n'intéresse pas vraiment l'agent). L'agent commence par exprimer ses utilités sur les objets individuels, par exemple : $\alpha_{\{ch\}} = 10$, $\alpha_{\{tr\}} = 2$, $\alpha_{\{to\}} = 1$ et $\alpha_{\{m\}} = 15$. Puis il peut exprimer des complémentarités ou des substituabilités entre les objets, en donnant des utilités à des groupes d'objets : $\alpha_{\{tr,ch\}} = 8$, $\alpha_{\{tr,to\}} = -8$, $\alpha_{\{tr,m\}} = 7$, $\alpha_{\{to,m\}} = 10$.

On voit apparaître de manière naturelle les complémentarités : $\alpha_{\{to,m\}} > \alpha_{\{m\}} + \alpha_{\{to\}}$, $\alpha_{\{tr,ch\}} > \alpha_{\{tr\}} + \alpha_{\{ch\}}$. La substituabilité entre la tondeuse et le tracteur est exprimée par un poids négatif sur ce lot.

3.2.4.2 Indépendance additive généralisée, GAI-nets et CSP valués

GAI-*nets* La notion d'indépendance k-additive pour les langages impliquant des variables binaires a une correspondance pour des langages sur des domaines combinatoires impliquant des variables non binaires. La notion d'*indépendance additive généralisée* (GAI) a été appliquée de manière

explicite à la représentation des utilités dans [Fishburn, 1970] et plus récemment dans [Bacchus et Grove, 1995]. L'idée est de transposer la notion d'indépendance probabiliste à la base du formalisme des réseaux bayésiens [Pearl, 1988] à la représentation des fonctions d'utilité.

Définition 3.24 (GAI-décomposition) *Soit $\mathscr{X} = \{\mathbf{x_i}, \ldots, \mathbf{x_n}\}$ un ensemble de variables. Soient $\mathscr{Z}_1, \ldots, \mathscr{Z}_k$ k sous-ensembles de $[\![1, n]\!]$ tels que $\bigcup_{i=1}^{k} \mathscr{Z}_i = [\![1, n]\!]$.*

Une fonction d'utilité sur $Inst(\mathscr{X})$ GAI-décomposable *selon $\{\mathscr{Z}_1, \ldots, \mathscr{Z}_k\}$ est définie par la donnée de fonctions $u_i : Inst(\mathscr{Z}_i) \mapsto \mathscr{V}$, telles que :*

$$u(v) = \sum_{i=1}^{k} u_i(v_{\downarrow \mathscr{Z}_i}) \text{ pour toute instanciation } v.$$

Notons que la notion de GAI-décomposition est plus précise que la notion de k-additivité sur les lots. La k-additivité signifie simplement que l'on est capable de représenter une fonction d'utilité sur un ensemble de lots en ne spécifiant que les poids que pour des lots de taille inférieure ou égale à k. La notion de GAI-décomposition est plus forte : une fonction d'utilité GAI-décomposable est construite précisément à partir de sa décomposition en sous-fonctions.

Un modèle graphique est associé aux GAI-décompositions : le modèle des GAI-*nets* [Gonzales et Perny, 2004].

Définition 3.25 (GAI-*net*) *Soient $\mathscr{X} = \{\mathbf{x_i}, \ldots, \mathbf{x_n}\}$ un ensemble de variables, et u une fonction d'utilité sur $Inst(\mathscr{X})$ GAI-décomposable selon $\{\mathscr{Z}_1, \ldots, \mathscr{Z}_k\}$. Un* GAI-*net représentant u est un graphe $\mathcal{G} = (V, E)$ tel que :*

- ▷ *$V = \{\mathscr{Z}_1, \ldots, \mathscr{Z}_k\}$;*
- ▷ *pour tout arc $(\mathscr{Z}_i, \mathscr{Z}_j) \in E$, $\mathscr{Z}_i \cap \mathscr{Z}_j \neq \varnothing$;*
- ▷ *pour tout $i \in [\![1, n]\!]$, le sous-graphe $\mathcal{G}_i = (V_i, E_i)$ induit par les sommets de $\mathcal{G}$ contenant i, défini par $V_i = \{\mathscr{Z}_j \mid i \in \mathscr{Z}_i\}$ et $E_i = \{(\mathscr{Z}_j, \mathscr{Z}_{j'}) \mid (\mathscr{Z}_j, \mathscr{Z}_{j'}) \in V_i^2 \text{ et } (\mathscr{Z}_j, \mathscr{Z}_{j'}) \in E\}$, est un sous-arbre connexe de $\mathcal{G}$ (propriété d'intersection courante).*

Les nœuds de $\mathcal{G}$ sont appelés des cliques *(ou clusters), et chaque arête $(\mathscr{Z}_i, \mathscr{Z}_j)$ est appelée* séparateur *et étiquetée avec $\mathscr{Z}_i \cap \mathscr{Z}_j$.*

Cette notion de GAI-*net* est similaire à celle de *graphe de jonction* dans les réseaux bayésiens. Dans le cas général le GAI-*net* correspondant à une fonction d'utilité GAI-décomposable n'est pas acyclique, ce qui rend les tâches liés au raisonnement sur les préférences (élicitation, agrégation de préférences...) plus difficiles. C'est pourquoi la plupart des travaux sur le sujet se restreignent à des GAI-*trees*, c'est-à-dire des GAI-*nets* acycliques.

Notons toutefois que tout GAI-*net* non acyclique peut être transformé en GAI-*tree*, comme le signale [Gonzales et Perny, 2004]. L'opération, correspondant à la transformation d'un graphe de jonction en arbre de jonction [Jensen *et al.*, 1994], procède en plusieurs étapes :

- ▷ la construction du graphe $\mathcal{G}_{dep}$ de dépendance entre variables, dont l'ensemble des sommets correspond aux variables de $\mathscr{X}$, et $(\mathbf{x_i}, \mathbf{x_j})$ est une arête de $\mathcal{G}_{dep}$ si et seulement si $\exists \mathscr{Z}_l$ tel que $\{i, j\} \subset \mathscr{Z}_l$;
- ▷ la triangulation du graphe $\mathcal{G}_{dep}$ [Robertson et Seymour, 1986], par la recherche d'un ordre d'élimination optimal (éventuellement de façon heuristique, car il s'agit d'un problème NP-difficile) ;
- ▷ la construction des nouvelles cliques à partir du graphe triangulé.

On pourra aussi noter l'existence d'un algorithme de construction d'un arbre de jonction qui ne fait pas appel à la triangulation du graphe de dépendances, mais à des recherches successives de coupes minimales [Koster, 1999]. Enfin, nous pouvons citer les travaux [Jégou et Terrioux, 2003;

de Givry *et al.*, 2006] issus de la communauté des problèmes de satisfaction de contraintes, qui sont très proches des travaux sur les GAI-*nets* en ceci qu'ils exploitent une décomposition arborescente du graphe de contraintes associé à un réseau de contraintes pour faciliter la résolution du problème. La notion de décomposition arborescente est formellement identique à la notion de transformation d'un graphe de jonction (ou d'un GAI-*net*) en arbre de jonction (ou GAI-*tree*).

Exemple 3.11 Si le modèle des GAI-*nets* est un formalisme dédié aux variables non exclusivement binaires, il peut très bien s'appliquer en particulier aux variables binaires, et donc aux problèmes de partage. Revenons sur le problème de l'héritage, l'agent ayant les mêmes préférences que dans l'exemple 3.9. Ces préférences sont naturellement GAI-décomposables, exactement de la même manière qu'elles sont exprimables dans un langage 2-additif. Les utilités associées aux objets et aux 2-lots de l'exemple 3.10 fournissent directement les tables correspondant aux relations GAI (unaires et binaires) :

ch	0	1
u_1	0	10

m	0	1
u_2	0	15

tr	0	1
u_3	0	2

to	0	1
u_4	0	1

ch	0	0	1	1
tr	0	1	0	1
u_5	0	0	0	8

tr	0	0	1	1
to	0	1	0	1
u_6	0	0	0	-8

m	0	0	1	1
tr	0	1	0	1
u_7	0	0	0	7

m	0	0	1	1
to	0	1	0	1
u_8	0	0	0	10

Ces préférences produisent par exemple le GAI-*net* présenté sur la figure 3.4 (les nœuds circulaires correspondent aux clusters, et les nœuds rectangulaires sont les séparateurs entre les clusters). Ce GAI-*net* peut être transformé en le GAI-*tree* présenté sur la même figure, par triangulation et calcul des cliques maximales.

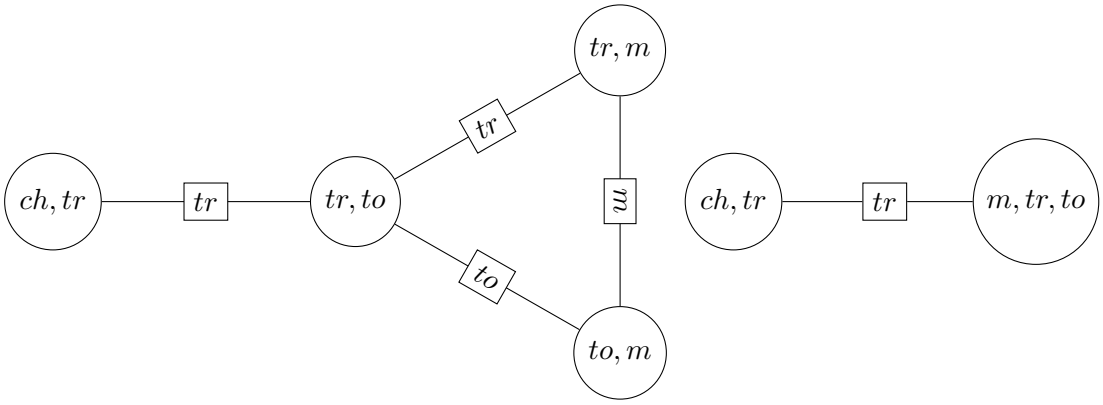


Figure 3.4 — Le GAI-*net* et son GAI-*tree* correspondant pour l'exemple 3.11.

Pour clore cette brève introduction aux GAI-*nets*, nous pouvons noter l'existence de travaux récents portant sur la transcription de structures de préférences exprimées dans un langage compact ordinal en fonctions d'utilités additives généralisées. Nous pouvons citer par exemple [Brafman *et al.*, 2004], qui introduit un certain nombre de résultats sur la transcriptions de TCP-*nets* sous la forme numérique additive généralisée.

CSP valués La notion d'indépendance additive généralisée et l'outil graphique des GAI-*nets* ont de nombreuses similitudes avec un cadre de représentation de problèmes combinatoires que nous avons déjà évoqué : celui des problèmes de satisfaction de contraintes. En effet, comme nous l'avons

signalé précédemment, le fait d'exprimer des contraintes portant sur des sous-ensembles de variables correspond à une décomposition de l'espace combinatoire en plusieurs entités indépendantes (dans le sens où la satisfaction d'une contrainte particulière sur un sous-ensemble n'a aucune influence sur la satisfaction d'une autre contrainte si cette autre contrainte ne partage aucune variable avec la première).

Jusqu'à peu, l'idée de préférence était complètement absente du cadre des problèmes de satisfaction de contraintes, mais l'extension de ce cadre aux contraintes valuées a comblé ce manque, en permettant l'expression de préférences sur la satisfaction des différentes contraintes. Cette extension a mené à l'introduction de deux cadres différents [Bistarelli *et al.*, 1999] : les *Semiring*-CSP [Bistarelli *et al.*, 1995, 1997], et les CSP valués (VCSP) [Schiex *et al.*, 1995, 1997]. Nous présentons très rapidement le deuxième cadre.

Définition 3.26 (Réseau de contraintes valué) *Un* réseau de contraintes valué *est un quintuplet* $(\mathscr{X}, \mathcal{D}, \mathscr{C}, \mathscr{V}, \varphi)$*, où :*

- ▷ $(\mathscr{X}, \mathcal{D}, \mathscr{C})$ *est un réseau de contraintes classique ;*
- ▷ $\langle \mathscr{V}, \geq, \oplus \rangle$ *est une structure de valuation ;*
- ▷ φ *est une fonction de valuation fondée sur un ensemble de fonctions* $\{\varphi_c \mid c \in \mathscr{C}\}$ *telles que* $\varphi_c : Inst(\mathscr{X}(c)) \setminus \mathscr{R}(c) \rightarrow \mathscr{V}$

La notion de réseau de contraintes classique a été introduite dans la définition 3.3 en début de chapitre, et la structure de valuation est identique à celle introduite dans la définition 1.12 du chapitre 1. Le rôle des fonctions φ_c est d'associer à chaque instanciation violant la contrainte c un coût dans $\mathscr{V}$, représentant en quelque sorte la «gravité» du non-respect de la contrainte (l'élément $\top$ de la structure de valuation correspond à une instanciation interdite, donc à une contrainte qu'il faut absolument respecter). On voit donc ici que même si la structure de valuation est identique à celle introduite pour les structures de préférence, la sémantique qui lui est associée est ici toutefois légèrement différente, car on ne parle plus d'utilités (qu'il faut maximiser donc), mais de coûts de violation de contraintes (coût à minimiser). Cette légère différence est toutefois peu importante en pratique, et il suffit d'assimiler le coût à une «désutilité» associée à la violation d'une contrainte.

La valuation val d'une instanciation v est calculée par agrégation des valuations de toutes les contraintes violées par v :

$$val(v) = \bigoplus_{\substack{c \text{ tel que} \\ v_{\downarrow \mathscr{X}(c)} \in Inst(\mathscr{X}(c)) \setminus \mathscr{R}(c)}} \varphi_c(v_{\downarrow \mathscr{X}(c)}).$$

Bien entendu, on ne peut s'empêcher de remarquer la similitude formelle entre ce cadre et celui des utilités additives généralisées, bien que ces deux formalismes soient utilisés dans des contextes différents : satisfaction de contraintes pour les réseaux de contraintes valués (donc recherche de solutions minimisant le coût des contraintes violées), et élicitation ou agrégation de fonctions d'utilités pour les GAI-*nets*. Les réseaux de contraintes valués ont, tout comme les utilités GAI-décomposables, un formalisme graphique associé, fondé sur la notion de graphe de contraintes (ou hypergraphe de contraintes) et correspondant au graphe de dépendances entre variables dans les GAI-*nets*. De plus, comme nous l'avons fait remarquer, l'intérêt pour la décomposition arborescente (le calcul d'un arbre de jonction) est le même dans les graphes de contraintes que dans les GAI-*nets*.

Notons pour clore cette section sur les CSP valués que des travaux récents s'intéressent à la définition d'un cadre de modélisation et de résolution beaucoup plus générique que celui des CSP valués, qui intègre à la fois la notion de faisabilité d'une solution, la notion de plausibilité (faisant référence au concept de croyance), et enfin d'utilité. Le détail de ces travaux apparaît dans la thèse [Pralet, 2006].

3.2.5 Langages d'enchères combinatoires

Les enchères combinatoires sont un autre exemple de domaine d'étude récent dans lequel le besoin de représentation compacte de préférences est de première importance, ce qui le situe à l'interface entre l'économie, la représentation des connaissances, la complexité et l'algorithmique. Comme indiqué dans la description de ce problème (voir l'application 5 dans l'introduction), les enchères combinatoires se distinguent des enchères classiques par le fait que les acheteurs ont la possibilité de miser sur des lots d'objets, en plus de miser sur des objets indépendants. Cela transforme donc le problème initial, où les préférences des agents étaient représentés par des mises sur des objets individuels (donc au plus m valeurs à exprimer), en un problème par nature combinatoire, puisque l'espace des alternatives sur lequel les agents doivent miser (en d'autres termes exprimer leurs préférences) est un espace d'objets, donc de taille 2^m.

Le problème central des enchères combinatoires est le *Winner Determination Problem* :

Problème 1: Winner Determination Problem

INSTANCE : Un ensemble d'agents $\mathscr{N}$, un ensemble d'objets $\mathscr{O}$, un ensemble de fonctions d'utilité $(f_1, \ldots, f_n)$ exprimées sous forme de mises dans un langage d'enchères combinatoires, et un entier K.

QUESTION : Existe-t-il un partage $\overrightarrow{\pi}$ des objets qui satisfait la contrainte de préemption et tel que $\sum_{i=1}^{n} f_i(\pi_i) \geq K$?

Bien entendu, nous avons adapté la définition de ce problème au formalisme que nous avons introduit jusqu'ici, et on pourra trouver dans la littérature un ensemble de formulations légèrement différentes de celle-ci. Nous nous devons d'ajouter quelques précisions sur cette définition. Par «langage d'enchères combinatoires», nous entendons tout langage qui peut être utilisé par les agents pour exprimer des mises sur un ensemble de lots (autrement dit, tout langage numérique sur $\wp(\mathscr{O})$), tels que ceux que nous allons introduire dans cette sous-section. Comme nous allons le voir, les utilités individuelles représentent le prix que les agents sont prêts à payer pour un ensemble d'objets donné. Maximiser la somme des utilités individuelles revient donc à maximiser le revenu du commissaire-priseur (ce qui est un point de vue résolument utilitariste classique).

Nous allons présenter brièvement le paradigme dominant en matière de langages d'enchères combinatoires [Nisan, 2006]. Ces langages sont utilisés de manière explicite ou implicite dans la plupart des travaux portant sur le *Winner Determination Problem* [Sandholm, 1999; Fujishima *et al.*, 1999]. On pourra trouver une classification générale des langages d'enchères combinatoires dans [Nisan, 2000].

3.2.5.1 Langages OR et XOR

La plupart des langages permettant d'exprimer des préférences sur des espaces combinatoires d'objets que nous avons introduits ci-avant étaient fondés sur les objets comme «unité de base». Ainsi par exemple l'un des plus simples de ces langages, le langage additif, s'appuie sur l'ensemble des valuations attribués aux singletons d'objets. L'utilité d'un lot non réduit à un singleton se déduit ensuite par calcul à partir de ces valuations.

Dans le domaine des enchères combinatoires, le point de vue est légèrement différent puisque la plupart des langages que l'on peut trouver dans ce domaine sont centrés sur la notion de lot. En d'autres termes, ces langages sont fondés sur le principe que les acheteurs formulent un ensemble de mises[5] correspondant à des lots d'objets, mais n'ont que faire des objets individuels contenus

[5]Nous traduisons par «mise» le terme «*bid*» classique dans les enchères combinatoires, et par «lot» le terme

dans les lots. Un autre aspect des langages dédiés aux enchères combinatoires est que l'espace de valuation correspond en général à l'ensemble des prix que les agents associent aux lots d'objets : en d'autres termes, l'utilité d'un agent pour un lot donné est le prix que cet agent est disposé à payer pour ce lot[6]. Nous supposerons donc dans notre présentation des langages d'enchères combinatoires que l'espace de valuation est $\mathbb{R}^+$.

Le langage le plus basique des enchères combinatoires est le langage des mises atomiques : chaque agent n'a le droit de soumettre au commissaire-priseur qu'une seule mise, c'est-à-dire la donnée d'un seul couple (lot, valuation).

Définition 3.27 (Langage des mises atomiques) *Le langage de représentation de préférences $\mathcal{R}_{at.bids}$ est défini par :*

- ▷ $\mathcal{L}_{at.bids}(\mathscr{O}) = \wp(\mathscr{O}) \times \mathbb{R}^+$;
- ▷ *pour toute mise (π, w), la structure de préférences cardinale induite $u_{at.bids}^{(\pi,w)}$ est définie par :*

$$\forall \pi' \in \wp(\mathscr{O}),\ u(\pi') = \begin{cases} w & si\ \pi' \supseteq \pi \\ 0 & sinon. \end{cases}$$

Le langage des mises atomiques, aussi appelé *single-minded bids language* dans [Lehmann *et al.*, 1999], est assez fruste et est très clairement incapable d'exprimer même une fonction d'utilité additive.

Dans les langages d'enchères combinatoires classiques, on ne limite plus l'expression des préférences à une seule mise atomique par agent, mais on suppose que chaque agent est capable d'en exprimer un nombre arbitraire. Les différentes manières d'interpréter cette multiplicité de mises atomiques conduisent à des langages différents. Le premier d'entre eux est le langage OR. Dans ce langage, la valeur d'une part est calculée en cherchant l'ensemble de lots mutuellement exclusifs contenus dans cette part, qui maximise la somme des prix associés.

Définition 3.28 (Langage OR) *Le langage de représentation de préférences $\mathcal{R}_{OR}$ est défini de la manière suivante :*

- ▷ *un élément de $\mathcal{L}_{OR}$ est un ensemble fini $\mathscr{M}_{OR}$ de mises atomiques, noté $(\pi_1, w_1)\ OR\ \ldots\ OR\ (\pi_p, w_p)$;*
- ▷ *pour tout ensemble de mises $\mathscr{M}_{OR} = (\pi_1, w_1)\ OR\ \ldots\ OR\ (\pi_p, w_p)$, la structure de préférences cardinale induite $u_{OR}^{\mathscr{M}_{OR}}$ est définie par :*

$$\forall \pi \in \wp(\mathscr{O}),\ u(\pi) = \max_{\mathscr{W}\ partition\ de\ \pi} \sum_{\pi' \in \mathscr{W}} w(\pi'),$$

$$avec\ w(\pi') = \begin{cases} w_k & si\ \exists (\pi_k, w_k) \in \mathscr{M}_{OR}\ tel\ que\ \pi' = \pi_k \\ 0 & sinon. \end{cases}$$

L'idée sous-jacente à la notion de mise OR est qu'un agent désire acquérir n'importe quel nombre de lots parmi les mises atomiques de la mise OR, et ce pour la somme des prix respectifs de ces mises. On remarquera qu'une mise OR correspond exactement à un ensemble de mises atomiques soumises par des enchérisseurs différents.

Ce langage de représentation de préférences est beaucoup plus expressif que celui des mises atomiques, puisqu'il est capable de représenter n'importe quelle fonction d'utilité u qui ne comporte pas de substituabilités (c'est-à-dire telle que $u(\pi \cup \pi') \geq u(\pi) + u(\pi')$).

«*bundle*».

[6] Remarquons toutefois que cet aspect n'est pas particulier aux enchères combinatoires, puisque de manière générale le prix qu'un agent est prêt à payer pour une alternative est un bon indicateur de son utilité pour cette alternative.

En revanche, ce gain d'expressivité se paie au prix fort, puisque le calcul de la valuation d'un lot quelconque est NP-complet. Cependant, ce saut de complexité n'est pas un très gros problème en pratique, car il est noyé dans le *Winner Determination Problem* global sans augmenter sa complexité.

Un autre langage est classique dans le domaine des enchères combinatoires : le langage XOR [Sandholm, 1999, 2002]. Contrairement au langage OR, il est complètement expressif (dans la limite de préférences monotones uniquement), et permet en particulier d'exprimer des substituabilités entre objets. Le langage XOR se distingue du langage OR en ceci que les lots sont tous mutuellement exclusifs, même s'ils ne possèdent aucun objet en commun. La signification sous-jacente à ce langage est qu'un agent ne désire qu'un et un seul lot parmi ceux de ses mises.

Définition 3.29 (Langage XOR) *Le langage de représentation de préférences $\mathcal{R}_{XOR}$ est défini de la manière suivante :*

- ▷ *un élément de $\mathcal{L}_{XOR}$ est un ensemble fini $\mathscr{M}_{XOR}$ de mises atomiques, noté (π_1, w_1) XOR ... XOR (π_p, w_p) ;*
- ▷ *pour tout ensemble de mises $\mathscr{M}_{XOR} = (\pi_1, w_1)$ XOR ... XOR (π_p, w_p), la structure de préférences cardinale induite $u_{XOR}^{\mathscr{M}_{XOR}}$ est définie par :*

$$\forall \pi \in \wp(\mathscr{O}), u(\pi) = \max_{\substack{(\pi_k, w_k) \in \mathscr{M}_{XOR} \\ \pi_k \subseteq \pi}} w_k.$$

Notons que si ce langage est pleinement expressif, en revanche il est incapable d'exprimer de manière compacte toutes les fonctions d'utilité représentables succinctement dans le langage OR. Ainsi, par exemple, une simple fonction d'utilité additive sur m objets, dont la représentation nécessite une mise OR de taille m, ne peut pas être représentée par une mise XOR de taille inférieure à 2^m.

3.2.5.2 Combiner les langages OR et XOR

La combinaison des langages OR et XOR est une extension naturelle des travaux sur la représentation compacte de mises. Cette combinaison permet d'allier l'expressivité du langage XOR à la compacité du langage OR. Le premier des langages fondé sur ce principe est le langage OR-of-XOR [Sandholm, 1999] :

Définition 3.30 (Langage OR-of-XOR) *Le langage de représentation de préférences $\mathcal{R}_{OoX}$ est défini de la manière suivante :*

- ▷ *un élément de $\mathcal{L}_{OoX}$ est un ensemble fini $\mathscr{M}_{OoX}$ de mises XOR : $\mathscr{M}_{OoX} = \mathscr{M}_{XOR}^1$ OR ... OR $\mathscr{M}_{XOR}^p$, avec, pour tout k, $\mathscr{M}_{XOR}^k = (\pi_1^k, w(\pi_1^k))$ XOR ... XOR $(\pi_{p_k}^k, w(\pi_{p_k}^k))$;*
- ▷ *pour tout ensemble de mises $\mathscr{M}_{OoX}$, la structure de préférences cardinale induite $u_{OoX}^{\mathscr{M}_{OoX}}$ est définie par :*

$$\forall \pi \in \wp(\mathscr{O}),\ u(\pi) = \max_{(\pi_1, \ldots, \pi_p) \in \mathcal{P}} \sum_{k=1}^{n} w(\pi_k),$$

avec $\mathcal{P}$ l'ensemble des séquences de lots $(\pi_1, \ldots, \pi_p)$ tels que

- *$\forall i$, $\exists j$ tel que $\pi_i = \pi_j^i$ (autrement dit π_i apparaît dans la $i^{ème}$ mise XOR) ou $\pi_i = \varnothing$,*
- *$\forall i \neq j$, $\pi_i \cap \pi_j = \varnothing$.*

En d'autres termes, un enchérisseur qui exprime une mise OR-of-XOR ne désire qu'au plus un lot par mise XOR, les lots sélectionnés pour chaque mise XOR étant mutuellement exclusifs. Le langage OR-of-XOR généralise à la fois le langage OR et le langage XOR.

Exemple 3.12 L'agent concerné par le problème de l'héritage (problème 3.2) peut exprimer de manière naturelle ses préférences sous la forme OR-of-XOR. Il peut exprimer notamment le fait qu'il ne veuille qu'un seul bien immobilier, et qu'un seul bien parmi la tondeuse et le tracteur : $((\{ch\}, 5\,000 €)\ XOR\ (\{m\}, 60\,000 €))\ OR\ ((\{to\}, 500 €)\ XOR\ (\{tr\}, 3\,000 €))$. Par exemple, si l'agent obtient les terres cultivables (ch) et le tracteur (tr), il évaluera ce lot à 8 000 €. S'il obtient en plus la maison, le lot sera évalué à 63 000 € (dans ce cas, les terres cultivables ne comptent plus car elles sont éliminées par la maison dans la mise XOR correspondante).

Si l'agent voulait exprimer ces mêmes préférences dans le langage XOR, il lui faudrait une mise de taille 8 :

$$\begin{array}{ccccccc} (\{ch\}, 5\,000 €) & XOR & (\{m\}, 60\,000 €) & XOR & (\{to\}, 500 €) & XOR \\ (\{tr\}, 3\,000 €) & XOR & (\{ch, to\}, 5\,500 €) & XOR & (\{ch, tr\}, 8\,000 €) & XOR \\ (\{m, to\}, 60\,500 €) & XOR & (\{m, tr\}, 63\,000 €). & & & \end{array}$$

La combinaison duale des langages OR et XOR existe aussi, sous la forme du langage XOR-of-OR. Bien que moins utilisé dans le domaine des enchères combinatoires, en raison de son aspect légèrement moins intuitif que le langage OR-of-XOR, notons tout de même qu'il permet de représenter de manière plus succincte que ce dernier langage un ensemble de fonctions d'utilité telles que les fonctions monochromatiques (de manière informelle, les fonctions d'utilité telles que, chaque objet ayant une couleur bien définie, l'agent ne désire que des objets de couleur identique).

Nous citerons enfin deux autres extensions des langages fondés sur des combinaisons des langages OR et XOR. Le langage OR/XOR est fondé sur une combinaison arbitraire de mises formées à partir de ces deux opérateurs. Formellement, une formule du langage OR/XOR est une mise atomique ou formule du type $\mathscr{M}_1\ OR\ \mathscr{M}_2$ ou $\mathscr{M}_1\ XOR\ \mathscr{M}_2$, avec $\mathscr{M}_1$ et $\mathscr{M}_2$ des formules du langage OR/XOR. Les langages OR, XOR, OR-of-XOR et XOR-of-OR sont des cas particuliers de ce langage générique.

Un dernier langage d'importance dans le domaine des enchères combinatoires [Fujishima *et al.*, 1999; Nisan, 2000] est le langage OR*. Ce langage est fondé sur le langage OR, avec lequel il partage la même syntaxe, mais il permet de simuler des mises mutuellement exclusives par l'introduction d'objets factices. Le rôle de ces objets est simplement d'empêcher l'attribution simultanée de deux lots, et ainsi de simuler des mises XOR. Par exemple, si un agent désire placer la mise $(\pi_1, w_1)\ XOR\ (\pi_2, w_2)$, il peut introduire un objet factice d («d» pour *dummy*) et exprimer la mise précédente par une mise OR : $(\pi_1 \cup \{d\}, w_1)\ OR\ (\pi_2 \cup \{d\}, w_2)$. L'intérêt principal de ce langage, et la raison de son succès dans le domaine des enchères combinatoires est qu'il permet de simuler des mises du type XOR, tout en gardant une syntaxe identique à celle du langage OR, permettant ainsi à une instance du *Winner Determination Problem* exprimée dans le langage OR* d'être traitée par un solveur dédié aux instances exprimées dans le langage OR. Notons que ce langage est celui qui est utilisé par le générateur CATS [Leyton-Brown *et al.*, 2000], devenu une référence en terme de génération d'instances du problème d'enchères combinatoires.

3.2.5.3 Langages logiques pour les enchères combinatoires

Si le paradigme dominant dans le domaine de la représentation de préférences dans les enchères combinatoires est fondé sur les langages OR et XOR et leurs combinaisons, d'autres travaux récents du domaine s'appuient sur des langages à base de logique propositionnelle pondérée [Boutilier et Hoos, 2000, 2001]. Ces langages logiques ressemblent à la fois aux langages que nous avons introduits plus tôt dans ce chapitre s'appuyant sur la logique propositionnelle et aux langages de lots introduits dans cette section. En effet, ils permettent d'exprimer à la fois des combinaisons logiques pondérées d'objets simples et des combinaisons logiques de lots, cumulant les avantages de ces deux approches.

Décrivons rapidement le langage logique introduit et décrit en détails dans [Boutilier et Hoos, 2001]. Ce langage étend le langage logique à base de buts pondérés introduit dans la définition 3.18 en ceci qu'il permet d'attribuer des valuations à des sous-formules, et non uniquement à des formules entières. En d'autres termes, si, dans le langage à base de buts pondérés, les préférences d'un agent étaient représentées par un ensemble de formules logiques pondérées $(\varphi_1, w_1), \ldots, (\varphi_p, w_p)$, dans ce nouveau langage, les préférences d'un agent sont représentées par une formule d'un langage $\mathcal{L}_{GWL}$ («*GWL*» pour *Generalized Weighted Logic*) dont la syntaxe est la suivante :

- ▷ pour tout $o \in \mathscr{O}$ et $w \in \mathbb{R}^+$, $\langle o, w\rangle$ est une formule de $\mathcal{L}_{GWL}$;
- ▷ pour toutes formules b_1 et b_2 de $\mathcal{L}_{GWL}$ et tout $w \in \mathbb{R}^+$, $\langle b_1 \wedge b_2, w\rangle$ et $\langle b_1 \vee b_2, w\rangle$ sont des formules de $\mathcal{L}_{GWL}$ ([Boutilier et Hoos, 2001] introduit de plus un opérateur logique ou-exclusif que nous n'introduisons pas ici pour simplifier).

L'utilité d'un lot est calculée par agrégation des poids de toutes les sous-formules satisfaites. Ainsi, par exemple et de manière informelle, l'utilité d'un lot π vis-à-vis d'une formule du type $\langle b_1 \wedge b_2, w\rangle$ est calculée en agrégeant :

- ▷ l'utilité de π vis-à-vis de b_1 ;
- ▷ l'utilité de π vis-à-vis de b_2 ;
- ▷ w si π «satisfait» $b_1 \wedge b_2$.

Comme nous l'avons fait remarquer ci-avant, ce langage cumule les avantages de l'approche fondée sur les objets et de l'approche fondée sur les lots, en autorisant la combinaison logique de formules pondérées (qui peuvent s'apparenter à des mises). Cette expressivité a probablement une contrepartie en terme de difficulté de résolution du *Winner Determination Problem*, bien que cette difficulté ne semble pas encore avoir été réellement mise en évidence de manière empirique. L'article [Boutilier et Hoos, 2001] s'oriente en tout cas vers un algorithme de résolution fondé sur la recherche locale stochastique, plutôt que sur une approche exacte comme dans l'approche traditionnelle des enchères combinatoires.

3.2.6 Conclusion sur les langages de représentation de préférences

Nous avons tenté de présenter une vue d'ensemble de l'ensemble des langages d'expression compacte classiques de la littérature sur la représentation des préférences. Nous avons distingué plusieurs types de langages :

- ▷ les langages à base de logique, très expressifs et intuitifs, et relativement adaptés dans le cadre du partage ;
- ▷ les langages à base de préférences *ceteris paribus*, qui offrent une alternative aux langages logiques relativement élégante et intéressante d'un point de vue cognitif, mais qui ne sont pas toujours très pertinents dans le cadre des problèmes de partage ;
- ▷ les langages fondés sur l'indépendance additive généralisée, dont les langages de lots k-additifs sont étudiés dans le cadre du partage ;
- ▷ les langages d'enchères combinatoires, centrés sur l'expression de mises sur des lots.

Nous avons résumé l'ensemble de ces langages dans le tableau 3.1.

3.3 Représentation compacte des problèmes de partage équitable

Nous avons jusqu'ici dressé un aperçu des principaux langages de représentation compacte d'espace combinatoires et de préférences sur ces espaces, et nous avons brièvement discuté de la pertinence de ces langages dans le cadre des problèmes de partage. Nous allons dans cette section nous appuyer sur ces langages pour définir deux langages de représentation compacte de problèmes de partage équitable.

Langage	syntaxe	type	var. binaires
Langages logiques			
Dichotomique	$\mathcal{L}_{\mathscr{O}}$	dichotomique	oui
Pareto	$\mathcal{GB}_{\mathscr{O}}$	ordinale	oui
Leximin, discrimin, *best-out*	$\mathcal{GB}_{\mathscr{O}}^{strat}$	ordinale	oui
Card	$\mathcal{GB}_{\mathscr{O}}$	cardinale	oui
Distances	$\mathcal{GB}_{\mathscr{O}}$	cardinale	oui
Logique pondérée	$\mathcal{GB}_{\mathscr{O}}^{weighted}$	cardinale	oui
Préférences *ceteris paribus*			
Ceteris Paribus	relations entre formules $\lambda : \varphi \triangleright \psi$	ordinale	oui
CP-*nets*	Graphe dirigé, tables de préférences CP	ordinale	non
TCP-*nets*	Graphe dirigé possédant 3 types d'arcs, tables de préférences CP	ordinale	non
UCP-*nets*	Graphe dirigé, tables de préférences CP avec des utilités	cardinale	non
Additivité généralisée			
Additif	Utilités $u(o)$	cardinale	oui
k-additif	Utilités $u(\pi)$, pour $\lvert\pi\rvert \leq k$	cardinale	oui
GAI-*net*	Graphe, tables de valuations GAI	cardinale	non
CSP valué	Spécification de contraintes valuées	cardinale	non
Langages de lots pour les enchères combinatoires			
OR	$(\pi_1, w_1)\ OR\ (\pi_2, w_2)$	cardinale	oui
XOR	$(\pi_1, w_1)\ XOR\ (\pi_2, w_2)$	cardinale	oui
OR-of-XOR, XOR-of-OR, XOR/OR	Combinaisons de mises OR ou XOR	cardinale	oui
Logique pondérée généralisée	Combinaisons logiques pondérées de sous-formules logiques pondérées	cardinale	oui

Tableau 3.1 — Liste des langages de représentation de préférences introduits dans ce chapitre.

Comme nous l'avons signalé dans l'introduction du manuscrit, les problèmes de partage équitable n'ont été que rarement — et très récemment — étudiés sous l'angle de la représentation compacte (et de la complexité). Le besoin de représentation compacte dans les problèmes de partage naît du dilemme suivant, qui a été formulé par plusieurs théoriciens du choix social. (a) Soit on autorise les agents à exprimer n'importe quelle relation de préférence sur l'ensemble de tous les sous-ensembles d'objets, et l'on a à faire face à une représentation exponentiellement large. C'est le point de vue de [Herreiner et Puppe, 2002], qui suppose que les agents ont des préférences linéaires (un ordre strict) sur l'ensemble $\wp(\mathscr{O})$, et propose des procédures relativement intéressantes pour traiter avec ce genre de préférences. Cependant, comme nous l'avons vu dans l'introduction, l'explosion combinatoire liée à l'expression des préférences rend de telles procédures inutilisables en pratique. (b) La deuxième solution est de limiter de manière drastique l'ensemble des préférences exprimables, en supposant typiquement l'indépendance additive entre les objets. C'est la voie suivie par [Brams *et al.*, 2003] et [Demko et Hill, 1998]. Cependant, comme nous l'avons vu dans ce chapitre, il est possible de concilier les deux approches : c'est l'objectif de la représentation compacte.

Nous allons nous intéresser à la représentation compacte de deux problèmes de partage équitable particuliers, correspondant aux deux visions de l'équité présentées dans le chapitre 1. Le premier de ces problèmes est lié à la propriété d'absence d'envie. L'absence d'envie est une propriété très intéressante, mais n'est cependant pas suffisante pour assurer la qualité du partage : il faut un critère d'efficacité. Nous nous intéresserons donc en particulier au problème de recherche d'un partage Pareto-efficace et sans envie. Nous aborderons ce problème sous l'angle des préférences les plus simples qui soient, c'est-à-dire les préférences dichotomiques représentées sous forme logique, et en présence de la contrainte de préemption uniquement. Nous verrons que dans ce cadre, l'absence d'envie et la Pareto-efficacité s'expriment comme des propriétés logiques simples.

Le second problème auquel nous allons nous intéresser est le problème de maximisation d'une fonction d'utilité collective, lorsque les agents ont des préférences numériques exprimées sous forme logique. Nous fournirons un cadre de représentation compacte générique pour ce problème, cadre dont l'intérêt sera d'être assez expressif pour représenter un large spectre de problèmes de partage de biens indivisibles.

3.3.1 Pareto-efficacité et absence d'envie en présence de préférences dichotomiques : représentation logique

Nous allons donc nous intéresser dans un premier temps à la structure de préférences dichotomique (voir définition 1.10). Comme nous l'avons vu, une structure de préférences dichotomique est une structure de préférences ordinale dégénérée, pour laquelle il n'existe que deux niveaux de satisfaction pour les alternatives : un ensemble de «bonnes» alternatives (noté $\mathscr{G}$), et un ensemble de «mauvaises» alternatives. Toute bonne alternative est strictement meilleure qu'une mauvaise alternative ; mais les bonnes alternatives sont indifférentes entre elles, de même que les mauvaises entre elles.

Comme nous l'avons vu dans la section 3.2.2.1, il existe une manière évidente d'exprimer une structure de préférences dichotomique de manière compacte sur un ensemble de parts d'un ensemble d'objets $\mathscr{O}$: il suffit d'introduire une formule propositionnelle φ sur le langage $\mathcal{L}_{\mathscr{O}}$, tel que $Mod(\varphi) = \mathscr{G}$.

Exemple 3.13 $\mathscr{O} = \{o_1, o_2, o_3\}$ et $\mathscr{G} = \{\{o_1, o_2\}, \{o_2, o_3\}\}$. On peut remarquer que la relation de préférences $\succeq$ correspondant à cet ensemble $\mathscr{G}$ n'est pas monotone. Alors $\varphi = (\mathbf{o_1} \wedge \mathbf{o_2} \wedge \neg\mathbf{o_3}) \vee (\neg\mathbf{o_1} \wedge \mathbf{o_2} \wedge \mathbf{o_3})$ représente $\succeq$, de même que $\varphi' = \mathbf{o_2} \wedge ((\mathbf{o_1} \wedge \neg\mathbf{o_3}) \vee (\neg\mathbf{o_1} \wedge \mathbf{o_3}))$, qui est logiquement équivalente à φ.

On peut légitimement se demander quel intérêt on peut porter à un langage de représentation de préférences aussi frustre. Tout d'abord, l'hypothèse de dichotomie des préférences est raisonnable dans un certain nombre de problèmes, comme par exemple les problèmes pour lesquels les agents ne désirent qu'un seul objet ou ensemble d'objets. Cependant, l'intérêt principal de cette structure de préférences est «computationnel». Sa relative simplicité permet d'exprimer la plupart des propriétés ayant trait au partage comme des propriétés logiques, et de se ramener ainsi à des grands problèmes classiques de la logique propositionnelle, comme nous allons le voir. Cependant, malgré la simplicité de cette structure de préférences et de son langage de représentation compacte associé, nous allons voir que ce langage concentre toute la complexité du problème de partage liée à la représentation compacte, à l'efficacité et à l'absence d'envie : en d'autres termes, il n'est pas plus facile de résoudre un problème de partage avec des préférences dichotomiques qu'avec des préférences plus générales munies de leur langage de représentation compacte raisonnable. La structure de préférences dichotomique et son langage logique associé constituent donc un point d'entrée idéal pour l'étude de la complexité de l'existence d'un problème de partage efficace et sans envie.

Introduisons un résultat facile à obtenir mais utile pour la suite :

Proposition 3.3 *Soit $\succeq_i$ une structure de préférences dichotomique sur $\wp(\mathscr{O})$. Alors les affirmations suivantes sont équivalentes :*

1. *$\succeq_i$ est monotone ;*
2. *$\mathscr{G}_i$ est supérieurement clos, c'est-à-dire que $\mathscr{S} \in \mathscr{G}_i$ et $\mathscr{S}' \supseteq \mathscr{S}$ implique que $\mathscr{S}' \in \mathscr{G}_i$.*
3. *$\succeq_i$ peut être représentée dans $\mathcal{R}_{dicho}$ par une formule propositionnelle positive.*

Démonstration (1) $\Rightarrow$ (2). Supposons que $\succeq_i$ est monotone ; soient $\mathscr{S} \in \mathscr{G}_i$ et $\mathscr{S}' \supseteq \mathscr{S}$. Alors nous avons forcément $\mathscr{S}' \succeq_i \mathscr{S}$, et donc $\mathscr{S}' \in \mathscr{G}_i$ (puisque $\mathscr{S} \in \mathscr{G}_i$).

(2) $\Rightarrow$ (3). Supposons que $\mathscr{G}_i$ est supérieurement clos, et considérons l'ensemble $\min_{\subseteq}(\mathscr{G}_i)$ de tous les ensembles de $\mathscr{G}_i$ minimaux pour l'inclusion. Alors la formule $\varphi_i = \bigvee_{\mathscr{S} \in \min_{\subseteq}(\mathscr{G}_i)} \left(\bigwedge_{o \in \mathscr{S}} \mathbf{o}\right)$ représente $\succeq_i$ pour les raisons suivantes. Pour tout $\mathscr{S}' \in \mathscr{G}_i$, il existe un ensemble $\mathscr{S} \in \min_{\subseteq}(\mathscr{G}_i)$ tel que $\mathscr{S} \subseteq \mathscr{S}'$. Donc la conjonction correspondante dans φ_i est satisfaite, et donc φ_i est satisfaite. Réciproquement, pour tout ensemble $\mathscr{S}' \notin \mathscr{G}_i$, il n'existe aucun $\mathscr{S} \subseteq \min_{\subseteq}(\mathscr{G}_i)$ tel que $\mathscr{S} \in \mathscr{S}'$. En conséquence, aucun des cubes de φ_i n'est satisfait, et donc la formule φ_i est insatisfaite. De plus, φ_i est clairement une formule propositionnelle positive.

(3) $\Rightarrow$ (1). Supposons que $\succeq_i$ peut être représentée par une formule propositionnelle positive φ_i, et soient $\mathscr{S}$ et $\mathscr{S}'$ deux ensembles d'objets tels que $\mathscr{S} \subseteq \mathscr{S}'$. Si $\mathscr{S} \notin \mathscr{G}_i$, alors très clairement $\mathscr{S}' \succeq_i \mathscr{S}$. Si $\mathscr{S} \in \mathscr{G}_i$, alors $\mathscr{S} \in Mod(\varphi_i)$. Puisque φ_i est positive, alors on a aussi $\mathscr{S}' \in Mod(\varphi_i)$. En conséquence, $\mathscr{S}' \in \mathscr{G}_i$, et donc au final $\mathscr{S}' \succeq_i \mathscr{S}$. ▲

La donnée des agents, des objets, et des formules logiques représentant les préférences des agents suffit donc à définir une instance du problème de partage équitable de biens indivisibles avec préférences dichotomiques. Nous supposerons donc à partir de maintenant que les instances $\mathcal{P}$ de ce problème sont représentées sous la forme $(\mathscr{N}, \mathscr{O}, (\varphi_1, \ldots, \varphi_n))$ au lieu de spécifier $\mathscr{N}$, $\mathscr{O}$ et l'ensemble des relations de préférence $(\succeq_1, \ldots, \succeq_n)$.

Soit $\mathcal{P} = (\mathscr{N}, \mathscr{O}, (\varphi_1, \ldots, \varphi_n))$ un problème d'allocation avec préférences dichotomiques. Les formules φ_i impliquent toutes les mêmes variables propositionnelles sur $\mathscr{O}$. Dans la suite de la construction du modèle, nous allons avoir besoin de différentier les variables entre les formules. Nous allons donc transposer les préférences des agents dans le langage $\mathcal{L}_{\mathscr{O}}^{alloc}$. Plus concrètement, pour chaque $i \in \mathscr{N}$, nous notons φ_i^* la formule φ_i dans laquelle on a remplacé chaque variable $\mathbf{o}$ par le symbole $\mathbf{alloc(o, i)}$.

Exemple 3.14 Considérons le problème de partage suivant : $\mathscr{N} = \{1,2,3\}$, $\mathscr{O} = \{o_1, o_2, o_3\}$, $\varphi_1 = \mathbf{o_1} \vee (\mathbf{o_2} \wedge \mathbf{o_3})$, $\varphi_2 = \mathbf{o_1}$ et $\varphi_3 = \mathbf{o_1} \vee \mathbf{o_2}$. Toutes les formules sont positives, et donc les préférences sont monotones. Par exemple, $\mathscr{G}_1$ est l'ensemble composé de tous les sur-ensembles de $\{\mathbf{o_1}\}$ et tous les sur-ensembles de $\{\mathbf{o_2}, \mathbf{o_3}\}$. Nous avons alors :

$$\varphi_1^* = \mathbf{alloc}(\mathbf{o_1}, \mathbf{1}) \vee (\mathbf{alloc}(\mathbf{o_2}, \mathbf{1}) \wedge \mathbf{alloc}(\mathbf{o_3}, \mathbf{1}));$$
$$\varphi_2^* = \mathbf{alloc}(\mathbf{o_2}, \mathbf{2});$$
$$\varphi_3^* = \mathbf{alloc}(\mathbf{o_1}, \mathbf{3}) \vee \mathbf{alloc}(\mathbf{o_2}, \mathbf{3}).$$

Comme nous l'avons précisé en tout début de section, nous nous limitons pour ce modèle aux problèmes de partage sans autre contrainte que la contrainte de préemption. Cette contrainte peut bien entendu être représentée sous forme logique, à l'aide de la formule suivante : $\Gamma_{\mathcal{P}} = \bigwedge_{o \in \mathscr{O}} \bigwedge_{i \neq j} \neg(\mathbf{alloc}(\mathbf{o}, \mathbf{i}) \wedge \mathbf{alloc}(\mathbf{o}, \mathbf{j}))$. En d'autres termes, les seuls partages admissibles sont ceux qui correspondent à un modèle de $Alloc_{\mathscr{O},\mathscr{N}}$ qui satisfait *au plus* un des $\mathbf{alloc}(\mathbf{o}, \mathbf{i})$ pour tout $o \in \mathscr{O}$. Nous pouvons donc associer à toute interprétation $M \in Mod(\Gamma_{\mathcal{P}})$ un partage $\overrightarrow{\pi}$ défini simplement par $\pi_i = \{\mathbf{x} \mid M \vDash \mathbf{x_i}\}$. Cette transformation est clairement bijective, et donc nous pouvons associer à tout partage $\overrightarrow{\pi}$ une interprétation de $Mod(\Gamma_{\mathcal{P}})$, que nous noterons $F(\overrightarrow{\pi})$.

Exemple 3.14.a Reprenons le problème de partage présenté dans l'exemple 3.14. Par souci de place, nous abrégerons, dans cet exemple et dans le suivant uniquement, $\mathbf{alloc}(\mathbf{o_i}, \mathbf{j})$ en $\mathbf{o_i^j}$. Nous avons :

$$\begin{aligned}\Gamma_{\mathcal{P}} = \quad & \neg(\mathbf{o_1^1} \wedge \mathbf{o_1^2}) \wedge \neg(\mathbf{o_1^1} \wedge \mathbf{o_1^3}) \wedge \neg(\mathbf{o_1^2} \wedge \mathbf{o_1^3}) \\ \wedge \quad & \neg(\mathbf{o_2^1} \wedge \mathbf{o_2^2}) \wedge \neg(\mathbf{o_2^1} \wedge \mathbf{o_2^3}) \wedge \neg(\mathbf{o_2^2} \wedge \mathbf{o_2^3}) \\ \wedge \quad & \neg(\mathbf{o_3^1} \wedge \mathbf{o_3^2}) \wedge \neg(\mathbf{o_3^1} \wedge \mathbf{o_3^3}) \wedge \neg(\mathbf{o_3^2} \wedge \mathbf{o_3^3}).\end{aligned}$$

L'interprétation M telle que M n'instancie que $\mathbf{o_1^1}$, $\mathbf{o_2^3}$ et $\mathbf{o_3^1}$ à vrai est très clairement un modèle de $\Gamma_{\mathcal{P}}$. Ce modèle correspond à l'allocation $F^{-1}(M) = \overrightarrow{\pi}$, avec $\pi_1 = \{\mathbf{o_1}, \mathbf{o_3}\}$, $\pi_2 = \varnothing$ et $\pi_3 = \{\mathbf{o_2}\}$.

3.3.1.1 Absence d'envie

Nous allons maintenant montrer comment le problème de recherche d'un partage sans envie peut être transformé en un problème de recherche de modèle dans une formule propositionnelle. Soit $\varphi^*_{j|i}$ la formule obtenue à partir de φ_i^* en substituant chaque symbole $\mathbf{alloc}(\mathbf{o}, \mathbf{i})$ dans φ_i^* par le symbole $\mathbf{alloc}(\mathbf{o}, \mathbf{j})$: par exemple, si $\varphi_1^* = \mathbf{alloc}(\mathbf{o_1}, \mathbf{1}) \wedge (\mathbf{alloc}(\mathbf{o_2}, \mathbf{1}) \vee \mathbf{alloc}(\mathbf{o_3}, \mathbf{1}))$ alors $\varphi^*_{2|1} = \mathbf{alloc}(\mathbf{o1}, \mathbf{2}) \wedge (\mathbf{alloc}(\mathbf{o_2}, \mathbf{2}) \vee \mathbf{alloc}(\mathbf{o_3}, \mathbf{2}))$. Remarquons que l'on a bien entendu $\varphi^*_{i|i} = \varphi_i^*$.

Nous introduisons le lemme suivant, qui est évident mais très utile :

Lemme 1 *Pour tout (i,j), $\pi_j \in \mathscr{G}_i$ si et seulement si $F(\overrightarrow{\pi}) \vDash \varphi^*_{j|i}$.*

En particulier, lorsque $i = j$ on a $\pi_i \in \mathscr{G}_i$ si et seulement si $F(\overrightarrow{\pi}) \vDash \varphi_i^*$.

Démonstration Par définition de F, $\pi_j \in \mathscr{G}_i$ si et seulement si $\{o \mid F(\overrightarrow{\pi}) \vDash \mathbf{alloc}(\mathbf{o}, \mathbf{j})\} \in \mathscr{G}_i$, c'est-à-dire $\{o \mid F(\overrightarrow{\pi}) \vDash \mathbf{alloc}(\mathbf{o}, \mathbf{j})\} \vDash \varphi_i$. Cette dernière relation est équivalente à $\{\mathbf{alloc}(\mathbf{o}, \mathbf{i}) \mid F(\overrightarrow{\pi}) \vDash \mathbf{alloc}(\mathbf{o}, \mathbf{j})\} \vDash \varphi_i^*$, et enfin à $\{\mathbf{alloc}(\mathbf{o}, \mathbf{j}) \mid F(\overrightarrow{\pi}) \vDash \mathbf{alloc}(\mathbf{o}, \mathbf{j})\} \vDash \varphi^*_{j|i}$ par définition de $\varphi^*_{j|i}$. Nous pouvons donc en déduire le résultat. ▲

À l'aide de ce lemme, nous pouvons maintenant exprimer la propriété d'absence d'envie comme une propriété de satisfiabilité d'une formule logique :

Proposition 3.4 *Soit $\mathcal{P} = (\mathcal{N}, \mathscr{O}, (\varphi_1, \ldots, \varphi_n))$ une instance du problème de partage avec préférences dichotomiques sous forme propositionnelle, et soient $\varphi^*_{j|i}$ la formule et F la bijection définis ci-avant. Soit :*

$$\Lambda_{\mathcal{P}} = \bigwedge_{i=1,\ldots,n} \left[\varphi^*_i \vee \left(\bigwedge_{j \neq i} \neg \varphi^*_{j|i} \right) \right].$$

Alors $\overrightarrow{\pi}$ est sans envie si et seulement si $F(\overrightarrow{\pi}) \vDash \Lambda_{\mathcal{P}}$.

Démonstration Soit $\overrightarrow{\pi}$ une allocation. $\overrightarrow{\pi}$ n'est pas sans envie si et seulement s'il existe une paire d'agents (i, j), avec $i \neq j$, telle que $\pi_j \succ_i \pi_i$, c'est-à-dire $\pi_j \in \mathscr{G}_i$ et $\pi_i \notin \mathscr{G}_i$, ce qui est à son tour équivalent à $F(\overrightarrow{\pi}) \vDash \varphi^*_{j|i}$ et $F(\overrightarrow{\pi}) \nvDash \varphi^*_i$ d'après le lemme 1. Donc π est sans envie si et seulement si $F(\overrightarrow{\pi}) \vDash \Lambda_{\mathcal{P}}$. ▲

La recherche d'allocations sans envie peut donc se ramener à un problème de recherche de modèles : $\{F^{-1}(M) \mid M \vDash \Gamma_{\mathcal{P}} \wedge \Lambda_{\mathcal{P}}\}$ est l'ensemble des allocations sans envie pour $\mathcal{P}$. Remarquons bien que $\Gamma_{\mathcal{P}} \wedge \Lambda_{\mathcal{P}}$ a une taille polynomiale (précisément, quadratique) en la taille des données d'entrée. Cette remarque est importante pour les résultats de complexité du chapitre 4.

Comme nous l'avons déjà fait remarquer, rechercher un partage sans envie (sans requérir aucune autre propriété sur le partage) n'est pas très intéressant, car une telle allocation existe toujours : il suffit de considérer le partage qui donne une part vide à tous les agents. Cependant :

- ▷ le problème de déterminer s'il existe une allocation sans envie vérifiant une propriété donnée exprimable par une formule logique de taille polynomiale (par exemple la propriété de complétude du partage) peut se réduire à un problème de satisfiabilité ;
- ▷ le problème de recherche (resp. de décompte) de tous les partages sans envie se ramène à un problème de recherche (resp. de décompte) de tous les modèles de la formule $\Lambda_{\mathcal{P}} \wedge \Gamma_{\mathcal{P}}$.

Exemple 3.14.b Revenons sur le problème de l'exemple 3.14. Nous avons :

$$\begin{array}{rl} \Lambda_{\mathcal{P}} = & [(\mathbf{o_1^1} \vee (\mathbf{o_2^1} \wedge \mathbf{o_3^1})) \vee (\neg(\mathbf{o_1^2} \vee (\mathbf{o_2^2} \wedge \mathbf{o_3^2})) \wedge \neg(\mathbf{o_1^3} \vee (\mathbf{o_2^3} \wedge \mathbf{o_3^3})))] \\ \wedge & [\mathbf{o_1^2} \vee (\neg \mathbf{o_1^1} \wedge \neg \mathbf{o_1^3})] \\ \wedge & [(\mathbf{o_1^3} \vee \mathbf{o_2^3}) \vee (\neg(\mathbf{o_1^1} \vee \mathbf{o_2^1}) \wedge \neg(\mathbf{o_1^2} \vee \mathbf{o_2^2}))] \end{array}$$

$$Mod(\Gamma_{\mathcal{P}} \wedge \Lambda_{\mathcal{P}}) = \{\{\mathbf{o_3^1}\}, \{\mathbf{o_3^1}, \mathbf{o_2^3}\}, \{\mathbf{o_3^2}, \mathbf{o_2^3}\}, \{\mathbf{o_3^2}\}, \{\mathbf{o_2^3}\}, \{\mathbf{o_3^3}\}, \varnothing\}.$$

Il y a donc 7 allocations sans envie, qui sont les suivantes : $(o_3, -, -)$, $(o_3, -, o_2)$, $(-, o_3, o_2)$, $(-, o_3, -)$, $(-, -, o_2)$, $(-, -, o_3)$ et $(-, -, -)$. Notons qu'aucune d'entre elles n'est sans envie.

3.3.1.2 Partages efficaces

De manière similaire à la propriété d'absence d'envie, la propriété de Pareto-efficacité peut être traduite en une propriété logique. L'expression logique de cette propriété requiert la définition de la notion de sous-ensemble maximal β-consistant d'un ensemble de variables.

Définition 3.31 *Soient $\Delta = \{\alpha_1, \ldots, \alpha_m\}$ un ensemble de formules et β une formule. $\mathscr{S} \subseteq \Delta$ est un* sous-ensemble maximal β-consistant de Δ *si et seulement si (a) $\bigwedge \mathscr{S} \wedge \beta$ est consistant et (b) il n'existe aucun ensemble $\mathscr{S}'$ tel que $\mathscr{S} \subset \mathscr{S}' \subseteq \Delta$ et $\bigwedge \mathscr{S}' \wedge \beta$ est consistant (satisfiable). Soit $MaxCons(\Delta, \beta)$ l'ensemble de tous les sous-ensembles maximaux β-consistants de Δ. Nous noterons de plus $MaxCons(\Delta)$ l'ensemble de tous les sous-ensembles maximaux-consistants de Δ, c'est-à-dire l'ensemble $MaxCons(\Delta, \top)$.*

Proposition 3.5 *Soit $\mathcal{P} = (\mathcal{N}, \mathcal{O}, (\varphi_1, \ldots, \varphi_n))$ un problème de partage. Soit $\Phi_\mathcal{P} = \{\varphi_1^*, \ldots, \varphi_n^*\}$. Alors $\overrightarrow{\pi}$ est Pareto-efficace pour $\mathcal{P}$ si et seulement si $\{\varphi_i^* \mid F(\overrightarrow{\pi}) \models \varphi_i^*\}$ est un sous-ensemble maximal $\Gamma_\mathcal{P}$-consistant de $\Phi_\mathcal{P}$.*

Démonstration Soit $\overrightarrow{\pi}$ une allocation. Soit $Sat(\overrightarrow{\pi})$ l'ensemble des agents satisfaits par $\overrightarrow{\pi}$, c'est-à-dire, d'après le lemme 1, l'ensemble $\{i \mid F(\overrightarrow{\pi}) \models \varphi_i^*\}$. Pour tout $i \in Sat(\overrightarrow{\pi})$, $F(\overrightarrow{\pi}) \models \varphi_i^*$ par définition de $Sat(\overrightarrow{\pi})$, et $F(\overrightarrow{\pi}) \models \Gamma_\mathcal{P}$ par définition de $F(\overrightarrow{\pi})$. Donc $F(\overrightarrow{\pi}) \models \Gamma_\mathcal{P} \wedge \bigwedge\{\varphi_i^* \mid i \in Sat(\overrightarrow{\pi})\} = \Gamma_\mathcal{P} \wedge \bigwedge\{\varphi_i^* \mid F(\overrightarrow{\pi}) \models \varphi_i^*\}$. En conséquence $\bigwedge\{\varphi_i^* \mid F(\overrightarrow{\pi}) \models \varphi_i^*\} \wedge \Gamma_\mathcal{P}$ est consistant.

Par définition, $\overrightarrow{\pi}$ est Pareto-dominé si et seulement s'il existe une allocation $\overrightarrow{\pi}'$ telle que $Sat(\overrightarrow{\pi}') \supsetneq Sat(\overrightarrow{\pi})$. En conséquence, si $\overrightarrow{\pi}$ est Pareto-dominé, il existe un sous-ensemble consistant $\mathcal{S} \subseteq \Phi_\mathcal{P}$ (correspondant à $\{\varphi_i^* \mid F(\overrightarrow{\pi}') \models \varphi_i^*\}$) tel que $\{\varphi_i^* \mid F(\overrightarrow{\pi}) \models \varphi_i^*\} \subset \mathcal{S}$. De plus, puisque $\overrightarrow{\pi}'$ est un partage, $\bigwedge \mathcal{S} \wedge \Gamma_\mathcal{P}$ est consistant.

Réciproquement, soit $\mathcal{S} \subseteq \Phi_\mathcal{P}$ tel que $\{\varphi_i^* \mid F(\overrightarrow{\pi}) \models \varphi_i^*\} \subset \mathcal{S}$ et $\bigwedge \mathcal{S} \wedge \Gamma_\mathcal{P}$ est consistant, et soit M un modèle de $\bigwedge \mathcal{S} \wedge \Gamma_\mathcal{P}$. Par définition, $\overrightarrow{\pi}' = F^{-1}(M)$ est une allocation bien définie (car M est un modèle de $\Phi_\mathcal{P}$), et $\overrightarrow{\pi}'(i) \in \mathcal{G}_i$ pour tout $\varphi_i^* \in \mathcal{S}$. Puisque $\{\varphi_i^* \mid F(\overrightarrow{\pi}) \models \varphi_i^*\} \subset \mathcal{S}$, on a $Sat(\overrightarrow{\pi}') \supsetneq Sat(\overrightarrow{\pi})$. Donc $\overrightarrow{\pi}$ est Pareto-dominé. ▲

Ce résultat simple suggère que les partages efficaces peuvent être calculés à partir de l'ensemble de formules logiques $\Phi_\mathcal{P}$ du problème, plus précisément en calculant les sous-ensembles maximaux $\Gamma_\mathcal{P}$-consistants de $\Phi_\mathcal{P}$; nous les appellerons $\{\mathcal{S}_1, \ldots, \mathcal{S}_q\}$. Pour chaque $\mathcal{S}_i$, soit $M_i = Mod(\bigwedge \mathcal{S}_i \wedge \Gamma_\mathcal{P})$ et soit $M = \cup_{i=1}^q M_i$. Alors $F^{-1}(M)$ est l'ensemble de toutes les allocations efficaces pour $\Phi_\mathcal{P}$. On peut remarquer qu'il y a en général un nombre exponentiel de sous-ensembles maximaux $\Gamma_\mathcal{P}$-consistants de $\Phi_\mathcal{P}$ (et donc en conséquence un nombre exponentiel de partages efficaces). Cette affirmation peut être néanmoins tempérée, car (a) il y a en pratique de nombreux cas pour lesquels le nombre de sous-ensembles maximaux consistants est faible ; (b) on ne cherche généralement pas *toutes* les allocations efficaces ; si l'on n'en cherche qu'une seule, elle peut être trouvée par calcul d'un sous-ensemble maximal $\Gamma_\mathcal{P}$-consistant de $\Phi_\mathcal{P}$.

Exemple 3.14.c Dans l'exemple 3.14, les sous-ensembles maximaux $\Gamma_\mathcal{P}$-consistants de $\Phi_\mathcal{P}$ sont $\mathcal{S}_1 = \{\varphi_1^*, \varphi_2^*\}$, $\mathcal{S}_2 = \{\varphi_1^*, \varphi_3^*\}$ et $\mathcal{S}_3 = \{\varphi_2^*, \varphi_3^*\}$. $\bigwedge \mathcal{S}_1 \wedge \Gamma_\mathcal{P}$ n'a qu'un seul modèle : $\{\mathbf{alloc(o_2, 1)}, \mathbf{alloc(o_3, 1)}, \mathbf{alloc(o_1, 2)}\}$. $\bigwedge \mathcal{S}_2 \wedge \Gamma_\mathcal{P}$ a deux modèles : $\{\mathbf{a_1}, \mathbf{alloc(o_2, 3)}\}$ et $\{\mathbf{alloc(o_2, 1)}, \mathbf{alloc(o_3, 1)}, \mathbf{alloc(o_1, 3)}\}$. $\bigwedge \mathcal{S}_3 \wedge \Gamma_\mathcal{P}$ a un seul modèle : $\{\mathbf{alloc(o_1, 2)}, \mathbf{alloc(o_2, 3)}\}$. En conséquence, les quatre allocations efficaces pour $\mathcal{P}$ sont $(\{o_2, o_3\}, \{o_1\}, -)$, $(\{o_1\}, -, \{o_2\})$, $(\{o_2, o_3\}, -, \{o_1\})$ et $(-, \{o_1\}, \{o_2\})$. Aucune d'entre elles n'est sans envie.

3.3.1.3 Partages efficaces et sans-envie

Rassemblons maintenant toutes les notions introduites. Puisque les partages sans envie correspondent aux modèles de $\Lambda_\mathcal{P}$ et que les allocations efficaces correspondent aux modèles des sous-ensembles maximaux $\Gamma_\mathcal{P}$-consistant de $\Phi_\mathcal{P}$, la propriété l'existence d'un partage efficace et sans-envie (EEF pour *Efficient and Envy-Free*) est équivalente à la condition suivante : *il existe un sous-ensemble $\mathcal{S}$ maximal $\Gamma_\mathcal{P}$-consistant de $\Phi_\mathcal{P}$ tel que $\bigwedge \mathcal{S} \wedge \Gamma_\mathcal{P} \wedge \Lambda_\mathcal{P}$ est consistant.* Dans ce cas, les modèles de cette dernière formule correspondent aux partages EEF. De manière intéressante, il s'agit d'une instance d'un problème connu dans le domaine du raisonnement non-monotone :

Définition 3.32 *Une* théorie des défauts supernormale[7] *est un couple $D = \langle \beta, \Delta \rangle$ avec $\Delta = \{\alpha_1, \ldots, \alpha_m\}$, tel que $\alpha_1, \ldots, \alpha_m$ et β sont des formules propositionnelles. Une formule proposi-*

[7] Les défauts «supernormaux» sont aussi appelés «défauts normaux sans prérequis» (voir par exemple [Reiter, 1980])

tionnelle ψ est une conséquence sceptique de D, *ce qui est noté* $D \mid\!\sim^{\forall} \psi$, *si et seulement si pour tout* $\mathscr{S} \in MaxCons(\Delta, \beta)$ *on a* $\bigwedge \mathscr{S} \wedge \beta \vDash \psi$.

Une théorie des défauts est un cadre logique dédié à la modélisation de tâches de raisonnement du type «lois générales à exceptions». De manière informelle, l'ensemble des défauts supernormaux correspond à un ensemble de formules logiques, que nous tenons pour vraies si rien ne nous en empêche. Le β de la théorie est une formule logique qui est appelée un «fait», tenu pour vrai. Sa présence nous amène à nous interroger sur l'ensemble des formules de notre base de croyance Δ qui sont cohérentes avec le fait. Les sous-ensembles maximaux β-consistants de Δ sont les sous-ensembles maximaux dans notre base de croyance que le fait ne «remet pas en cause». La question de l'inférence sceptique correspond donc à déterminer si l'on peut déduire une formule à partir du fait β et en changeant le moins possible l'ensemble de nos croyances.

Le lien avec l'absence d'envie et l'efficacité dans les problèmes de partage n'est pas évident. Et pourtant, nous avons la proposition suivante :

Proposition 3.6 *Soit* $\mathcal{P} = (\mathscr{N}, \mathscr{O}, (\varphi_1, \ldots, \varphi_n))$ *une instance du problème de partage équitable avec préférences dichotomiques. Soit* $D_P = \langle \Gamma_{\mathcal{P}}, \Phi_{\mathcal{P}} \rangle$. *Alors il existe un partage efficace et sans envie pour* $\mathcal{P}$ *si et seulement si* $D \not\mid\!\sim^{\forall} \neg\Lambda_{\mathcal{P}}$.

> **Démonstration** Soit $\mathcal{P} = (\mathscr{N}, \mathscr{O}, (\varphi_1, \ldots, \varphi_n))$ une instance du problème de partage équitable avec préférences dichotomiques. Soit $\overrightarrow{\pi}$ une allocation Pareto-efficace et sans envie, et $\mathscr{S} = \{\varphi_i^* \mid F(\overrightarrow{\pi}) \vDash \varphi_i^*\}$. Alors $F(\overrightarrow{\pi}) \vDash \Lambda_{\mathcal{P}}$ d'après la proposition 3.4. Nous avons aussi $F(\overrightarrow{\pi}) \vDash \mathscr{S}$ par définition de $\mathscr{S}$, et donc $F(\overrightarrow{\pi}) \vDash \bigwedge \mathscr{S} \wedge \Lambda_{\mathcal{P}}$, ce qui prouve que $\bigwedge \mathscr{S} \wedge \Lambda_{\mathcal{P}}$, ou en d'autres termes, que $\bigwedge \mathscr{S} \nvDash \neg\Lambda_{\mathcal{P}}$. De plus, $\mathscr{S}$ est un sous-ensemble maximal $\Gamma_{\mathcal{P}}$-consistant de $\Phi_{\mathcal{P}}$ d'après la proposition 3.5. Ainsi $\mathscr{S} \in MaxCons(\Phi_{\mathcal{P}}, \Gamma_{\mathcal{P}})$, et $\bigwedge \mathscr{S} \wedge \Gamma_{\mathcal{P}} \nvDash \neg\Lambda_{\mathcal{P}}$, ce qui implique $\langle \Gamma_{\mathcal{P}}, \Phi_{\mathcal{P}} \rangle \not\mid\!\sim^{\forall} \neg\Lambda_{\mathcal{P}}$ par définition de $\not\mid\!\sim^{\forall}$.
>
> Réciproquement, supposons que $\langle \Gamma_{\mathcal{P}}, \Phi_{\mathcal{P}} \rangle \not\mid\!\sim^{\forall} \neg\Lambda_{\mathcal{P}}$. Alors il existe un ensemble $\mathscr{S} \in MaxCons(\Phi_{\mathcal{P}}, \Gamma_{\mathcal{P}})$ tel que $\bigwedge \mathscr{S} \wedge \Gamma_{\mathcal{P}} \wedge \Lambda_{\mathcal{P}}$ a un modèle M. D'après la proposition 3.5, $F^{-1}(M)$ est un partage Pareto-efficace (puisque M est un modèle de $\bigwedge \mathscr{S} \wedge \Gamma_{\mathcal{P}}$), et d'après la proposition 3.4, $F^{-1}(M)$ est sans envie (puisque M est un modèle de $\Lambda_{\mathcal{P}}$). ▲

Ce lien plutôt inattendu avec le raisonnement non monotone a de nombreuses implications. Tout d'abord, les allocations EEF correspondent aux modèles de $\bigwedge \mathscr{S} \wedge \Gamma_{\mathcal{P}} \wedge \Lambda_{\mathcal{P}}$ avec $\mathscr{S} \in MaxCons(\Phi_{\mathcal{P}}, \Gamma_{\mathcal{P}})$. Cependant, l'ensemble $MaxCons(\Phi_{\mathcal{P}}, \Gamma_{\mathcal{P}})$ peut être exponentiellement large, ce qui nous dissuade de commencer par calculer les allocations efficaces et de filtrer ensuite celles qui ne sont pas sans envie, et plaide plutôt en faveur du calcul des allocations EEF en une seule passe, en utilisant des algorithmes issus du raisonnement à base de défauts. Ainsi, le partage équitable peut éventuellement bénéficier du travail algorithmique sur la logique des défauts et sur des domaines connexes tels que la révision des croyances. De plus, les critères alternatifs pour la sélection des extensions en raisonnement à base de défauts (tels que la cardinalité, les poids ou les priorités) correspondent à une alternative à la propriété d'efficacité dans les problèmes de partage.

3.3.1.4 Conclusion sur le problème avec préférences dichotomiques

Nous avons donc introduit dans cette sous-section un langage de représentation compacte dédié à la représentation compacte d'un problème de partage de biens indivisibles avec des préférences dichotomiques. Ce langage est fondé sur la définition d'une instance comme un ensemble d'agents, un ensemble d'objets et un ensemble de formules logiques. Dans ce cadre, les propriétés d'absence d'envie et de Pareto-efficacité s'expriment de manière logique, ce qui nous a permis de dresser un

parallèle intéressant entre le problème d'existence d'un partage Pareto-efficace et sans envie et le problème d'inférence sceptique en raisonnement des défauts. Ce parallèle sera à la base de l'analyse de la complexité de ce problème, que nous effectuerons au chapitre 4.

3.3.2 Un langage générique pour le problème de partage de biens indivisibles

Le précédent langage que nous avons introduit était dédié à la représentation compacte de propriétés ordinales du partage telles que la Pareto-efficacité et l'absence d'envie. Les préférences dichotomiques se prêtaient bien à cette modélisation : d'une part ce sont les structures de préférence ordinales les plus basiques, d'autres part, elles se prêtent bien à la représentation compacte par le biais de la logique propositionnelle, et enfin, comme nous le verrons au chapitre suivant, elles permettent d'aborder de manière simple le problème de la complexité liée à la représentation compacte, à l'absence d'envie et à la Pareto-efficacité.

Nous allons maintenant nous intéresser à la définition d'un langage générique pour représenter les problèmes de partage de biens indivisibles. L'objectif de ce langage est d'être intuitif et expressif. Nous allons nous fonder pour sa définition sur l'ensemble des notions introduites au chapitre 1, notamment sur la définition générique d'une instance du problème de partage de biens indivisibles (voir la définition 1.37).

Le langage de représentation compacte que nous allons introduire ici est fondé sur :
- ▷ une représentation logique des contraintes d'admissibilité ;
- ▷ des agents ayant des préférences numériques exprimées sous la forme de formules pondérées ;
- ▷ une modélisation *welfariste* du bien-être social, c'est-à-dire à l'aide d'une fonction d'utilité collective.

Ce langage de représentation a été décrit en détails dans [Fargier *et al.*, 2004a; Bouveret *et al.*, 2005a,b]. Il est fondé sur les mêmes éléments de base que dans la définition 1.37, c'est-à-dire :
- ▷ un ensemble fini d'agents $\mathscr{N} = \{1, \ldots, n\}$;
- ▷ un ensemble fini d'objets $\mathscr{O}$, de taille p .

De même, on définit toujours un partage comme un vecteur $\overrightarrow{\pi} = (\pi_1, \ldots, \pi_n)$, où $\pi_i \subseteq \mathscr{O}$ est la part de l'agent i.

3.3.2.1 Contraintes

Dans le chapitre 1, nous avons supposé que la spécification des alternatives admissibles se faisait à l'aide d'un ensemble de contraintes, définies — dans la définition 1.3 — comme un ensemble de tuples de $\mathscr{O}^n$. Nous avons vu dans ce chapitre qu'il existait un moyen simple de représenter un tel ensemble, par le biais de la logique propositionnelle et du langage $\mathcal{L}_{\mathscr{O}}^{alloc}$. Nous spécifierons donc dans notre langage l'ensemble des partages admissibles à l'aide d'un ensemble de contraintes exprimées sous forme logique :

Définition 3.33 (Contrainte, partage admissible) *Une contrainte est une formule de $L_{\mathscr{O}}^{alloc}$.*

L'ensemble des partages admissibles *vis-à-vis d'un ensemble de contraintes $\mathscr{C}$ exprimées comme des formules de $L_{\mathscr{O}}^{alloc}$ est l'ensemble $\mathscr{A} = \{\overrightarrow{\pi} \mid \forall C \in \mathscr{C},\ h^{-1}(\overrightarrow{\pi}) \vDash C\}$, avec h la bijection introduite dans la preuve de la proposition 3.1 page 84.*

Ce langage permet de représenter de manière compacte la plupart des contraintes usuelles :
- ▷ La contrainte de préemption globale s'exprime comme un ensemble de $n \times (n-1) \times p$ contraintes (ou comme une contrainte unique, comme nous l'avons vu dans le langage dédié aux préférences dichotomiques) : $\forall (i, j, o) \in \mathscr{N}^2 \times \mathscr{O}$ tels que $i \neq j$, $\neg(\mathbf{alloc(o, i)} \wedge \mathbf{alloc(o, j)})$.

- ▷ Une contrainte d'exclusion sur un ensemble d'objets $\mathscr{S}_{excl}$ s'exprime comme un ensemble de n contraintes : $\forall i \in \mathscr{N}, \neg \bigwedge_{o \in \mathscr{S}_{excl}} \mathbf{alloc(o, i)}$.
- ▷ Plus généralement, on peut exprimer facilement des contraintes de dépendance entre les objets à l'aide de formules logiques générales. Par exemple, $\neg\mathbf{alloc(o_1, i)} \vee (\mathbf{alloc(o_1, i)} \wedge (\mathbf{alloc(o_2, i)} \vee \mathbf{alloc(o_3, i)}))$ exprime le fait que si un objet o_1 est attribué à l'agent i, alors cet agent doit recevoir l'un des objets o_2 et o_3.

S'il est facile d'exprimer la plupart des contraintes usuelles dans ce langage de représentation, il est en revanche difficile (et, de manière générale, impossible) de représenter des contraintes de volume de manière compacte dans ce langage. De telles contraintes pourraient bien entendu être représentées par un ensemble de contraintes d'exclusion, mais une telle traduction peut nécessiter un nombre de contraintes exponentiel en p. Lors de notre analyse des classes de complexité dans le chapitre 4, nous nous autoriserons ponctuellement l'extension de ce langage de représentation de contraintes permettant l'expression de contraintes de volume. Nous introduirons la même extension dans l'implantation informatique du modèle, présentée dans le chapitre 6.

Ce langage d'expression de contraintes amène une autre remarque, liée à la nature des contraintes elle-même. Nous avons défini la notion de contrainte comme un sous-ensemble de l'ensemble des partages admissibles, autorisant ainsi la définition de contraintes s'appliquant spécifiquement à des agents. Il est naturel de s'interroger sur la pertinence de cette définition, alors que la plupart des contraintes naturelles s'appliquent de manière homogène à tous les agents (par exemple, des contraintes physiques ou légales interdisant l'attribution simultanée de deux objets au même agent s'appliquent probablement à tous les agents), et donc pourraient être exprimées de manière plus naturelle et plus compacte sur le langage $\mathcal{L}_{\mathscr{O}}$, et non sur l'ensemble des partages possibles, langage pour lequel l'expression de telles contraintes nécessite l'introduction de n contraintes (ou d'une contrainte impliquant une conjonction de taille n) identiques au numéro agent près (voir par exemple l'expression d'une contrainte d'exclusion ci-dessus). Nous avons cependant fait le choix d'une plus grande expressivité en autorisant l'expression de contraintes s'appliquant spécifiquement à certains agents, sachant que ce choix, s'il implique une représentation légèrement moins compacte que si nous avions exprimé des contraintes sur l'ensemble des lots, n'entraîne pas une augmentation exponentielle de la taille de la représentation, et plus généralement, n'entraîne pas d'augmentation de la complexité du problème de partage.

3.3.2.2 Demandes pondérées et préférences individuelles

L'espace des alternatives étant défini, il nous faut maintenant spécifier la manière dont les agents expriment leurs préférences. Dans la modélisation que nous avons choisie, qui s'inspire du *welfarisme*, les préférences des agents sont représentées par des fonctions d'utilité. Il reste à préciser comment sont construites ces fonctions d'utilité.

Parmi les langages de représentation de préférences présentés ci-avant, le langage $\mathcal{R}_{weighted}$ à base de buts pondérés (définition 3.18) fournit un bon compromis entre la concision, la puissance d'expression, et l'utilisation intuitive. Nous considérerons donc que les préférences des agents sont exprimées sous la forme d'une base de buts pondérée sur le langage propositionnel $\mathcal{L}_{\mathscr{O}}$ fondé sur l'ensemble d'objets. Nous limiterons cependant le langage à l'agrégation des demandes satisfaites, et nous ne prendrons donc pas en compte les demandes non satisfaites.

Définition 3.34 (Demande pondérée) *Une* demande pondérée *est un couple* $\delta = (\varphi(\delta), w(\delta))$, *avec :*

- ▷ $\varphi(\delta)$ *une formule de* $L_{\mathscr{O}}$,
- ▷ $w(\delta) \in \mathscr{V}_{ind}$, *où* $\langle \mathscr{V}_{ind}, \succeq_{ind}, \oplus \rangle$ *est un espace de valuation ordonné par* $\succeq_{ind}$, *et dont la loi* $\oplus$ *respecte toutes les conditions de la définition 1.12.*

L'ensemble des demandes pondérées de l'agent i est noté Δ_i.

Par la suite, nous ferons l'hypothèse de *monotonie* des préférences individuelles (voir définition 1.13 à la page 26), vis-à-vis de la loi $\oplus$ et de l'ordre $\succeq_{ind}$, c'est-à-dire telles que $\forall(\pi, \pi')$, $\pi \subseteq \pi' \Rightarrow u(\pi) \preceq u(\pi')$. Concrètement cela revient à restreindre les ensembles Δ_i aux demandes δ telles que $\varphi(\delta) \in \mathcal{L}^+_{\mathscr{O}}$ et $w(\delta)$ est un poids positif, c'est-à-dire tel que $\forall a \in \mathscr{V}_{ind}$, $a \oplus w(\delta) \succeq_{ind} a$.

Tout comme dans le langage de la définition 3.18, le calcul de l'utilité d'un agent est effectué par agrégation des poids des demandes satisfaites. Les définitions suivantes décrivent de manière plus précise le processus de calcul de l'utilité individuelle d'un agent à partir de la part qu'il reçoit.

Définition 3.35 (Valeur de satisfaction d'une demande) *Soient π un ensemble d'objets et δ une demande pondérée. On dira que δ est* satisfaite *par π si et seulement si $\pi \vDash \varphi(\delta)$. La* valeur de satisfaction *de δ vis-à-vis de π est définie comme suit :*

$$\sigma(\delta, \pi) = \begin{cases} w(\delta) & \text{si } \delta \text{ est satisfaite par } \pi, \\ \bot & \text{sinon.} \end{cases}$$

Définition 3.36 (Utilité individuelle) *Étant donnés un agent i, son ensemble de demandes pondérées Δ_i, et sa part π_i, l'*utilité individuelle *de l'agent i est définie comme suit :*

$$u_i(\pi_i) = \bigoplus_{\delta \in \Delta_i} \sigma(\delta, \pi_i).$$

La loi $\oplus$ sera appelé fonction d'agrégation individuelle.

Commentons cette définition. Premièrement, elle implique que la satisfaction d'un agent ne dépend que de ce qu'il reçoit. Nous pouvons donc écrire $u_i(\overrightarrow{\pi}) = u_i(\pi_i)$. La croissance des demandes vis-à-vis de la loi $\oplus$ signifie que le fait de satisfaire une demande ne peut jamais avoir un impact négatif (sur la satisfaction d'un agent), la commutativité de la loi $\oplus$ implique que l'ordre dans lequel sont faites des demandes n'est pas significatif, et l'associativité de cette même loi (due au fait qu'il s'agit d'une loi interne binaire) permet de calculer l'utilité d'un agent de manière simple, et ce même quand l'ensemble des demandes est énorme.

Le choix le plus évident pour la fonction d'agrégation $\oplus$ qui apparaît dans la définition 3.36 est $\oplus = +$; cependant, il peut y avoir des raisons, dans certains cas, de choisir $\oplus = \max$. Premièrement, $\oplus = \max$ est appropriée dans les cas où les agents ne veulent pas plus d'un objet. Bien entendu, de telles fonctions d'utilité pourraient être exprimées en utilisant $\oplus = +$, mais leur expression serait bien moins compacte. De plus, certains problèmes ont une complexité moindre avec $\oplus = \max$ qu'avec $\oplus = +$, comme nous allons le voir au chapitre 4. Deuxièmement, dans certains cas, les agents ne veulent pas (ou ne peuvent pas) exprimer leurs préférences numériquement, et en conséquence sont plus enclins à exprimer des listes de priorité. Dans ce dernier cas cependant, il serait peut-être plus judicieux d'utiliser un espace $\mathscr{V}_{ind}$ vectoriel, et d'utiliser un préordre leximax pour comparer les vecteurs des poids des formules satisfaites.

3.3.2.3 Utilité collective

Tout comme dans le modèle introduit au chapitre 1 fondé sur la théorie de l'utilitarisme, la comparaison des partages se fait sur la base des profils d'utilité et de l'utilité collective déduite de ces profils d'utilité.

Définition 3.37 (Utilité collective) *Soient $\mathcal{N}$ un ensemble d'agents et $\overrightarrow{\pi}$ un partage. L'utilité collective des agents vis-à-vis du partage $\overrightarrow{\pi}$ est définie comme suit :*

$$u_c(\overrightarrow{\pi}) = g(u_1(\overrightarrow{\pi}), \ldots, u_n(\overrightarrow{\pi})),$$

avec g une fonction de $\mathcal{V}_{ind}^n$ dans $\mathcal{V}$, espace de valuation ordonné par $\succeq$, non décroissante par rapport à chaque composante, c'est-à-dire telle que :

$$\forall (u_1, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_n) \in \mathcal{V}_{ind}^n \text{ et } \forall u_i' \succeq_{ind} u_i,$$
$$g(u_1, \ldots, u_{i-1}, u_i', u_{i+1}, \ldots, u_n) \succeq g(u_1, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_n).$$

Notons que cette définition générique autorise toutes les fonctions d'utilité à valeurs réelles introduites dans le chapitre 1, mais permet aussi de représenter de manière explicite des ordres de bien-être social tels que l'ordre leximin, en choisissant $\mathcal{V} = \mathcal{V}_{ind}^n$, $\succeq = \succeq_{leximin}$, et $g = Id$. Notons de plus que la généricité de la fonction g autorise en particulier toutes les fonctions d'utilité collective à droits exogènes inégaux, introduites au chapitre 2.

L'utilité collective associée à un partage admissible $\overrightarrow{\pi}$ est donc calculée par deux agrégations successives : agrégation des poids des demandes individuelles en utilités individuelles, puis agrégation des utilités individuelles en utilités collectives :

$$\left.\begin{array}{ccccccc} \sigma(\delta_{1,1}, \pi_1) & \oplus & \ldots & \oplus & \sigma(\delta_{1,m_1}, \pi_1) & \rightsquigarrow & u_1(\pi_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \sigma(\delta_{n,1}, \pi_n) & \oplus & \ldots & \oplus & \sigma(\delta_{n,m_n}, \pi_n) & \rightsquigarrow & u_n(\pi_n) \end{array}\right\} \rightsquigarrow u_c(\overrightarrow{\pi}) = g(u_1(\pi_1), \ldots, u_n(\pi_n)),$$

avec bien entendu m_i désignant le nombre de demandes de l'agent i.

3.3.2.4 Problème de partage de biens indivisibles générique

Nous avons maintenant tous les éléments nécessaires à la définition d'une instance du problème de partage de biens indivisibles générique dans notre langage de représentation compacte :

Définition 3.38 (Instance du problème de partage de biens indivisibles générique) *Une instance du problème de partage de biens indivisibles générique est un tuple $(\mathcal{N}, \mathcal{O}, (\Delta_1, \ldots, \Delta_n), \mathcal{C}, \langle \mathcal{V}_{ind}, \succeq_{ind}, \oplus \rangle, \langle \mathcal{V}, \succeq \rangle, g)$, où :*

- ▷ *$\mathcal{N} = \{1, \ldots, n\}$ est un ensemble fini d'agents ;*
- ▷ *$\mathcal{O}$ un ensemble fini d'objets ;*
- ▷ *$\Delta_i \subset L_{\mathcal{O}}^{+} \times \mathcal{V}_{ind}$ est un ensemble fini de demandes pondérées pour chaque agent i ;*
- ▷ *$\mathcal{C} \subset L_{\mathcal{O}}^{alloc}$ est un ensemble de contraintes ;*
- ▷ *$\langle \mathcal{V}_{ind}, \succeq_{ind}, \oplus \rangle$ est la structure de valuation individuelle, vérifiant les conditions de la définition 1.12, avec $\oplus$ la fonction d'agrégation individuelle ;*
- ▷ *$\langle \mathcal{V}, \succeq \rangle$ est la structure de valuation collective, avec g la fonction d'utilité collective.*

Définition 3.39 (Solution du problème de partage) *Une solution du problème de partage est un partage admissible $\overrightarrow{\pi}$.*

L'utilité collective associée à une solution $\overrightarrow{\pi}$ du problème de partage est calculée par double agrégation des poids des demandes et des utilités individuelles :

$$u_c(\overrightarrow{\pi}) = g\left(\bigoplus_{\delta \in \Delta_1} \sigma(\delta, \pi_1), \ldots, \bigoplus_{\delta \in \Delta_n} \sigma(\delta, \pi_n)\right).$$

3.3.2.5 Traduction des problèmes d'enchères combinatoires

Il est naturel de se demander dans quelle mesure il est possible de représenter de manière concise dans notre cadre générique les problèmes de partage classiques que sont les problèmes d'enchères combinatoires, représentés de manière compacte à l'aide des langages de lots (OR, XOR et autres langages dérivés). En fait, il est possible d'utiliser notre cadre générique sans perte de concision pour représenter des problèmes typiques des enchères combinatoires, ce qui rend ce modèle encore plus intéressant :

Proposition 3.7 *Toute instance du Winner Determination Problem dans les enchères combinatoires, pour laquelle les préférences sont exprimées à l'aide du langage OR ou XOR peut être traduite en une instance équivalente de notre problème de partage de biens indivisibles de taille polynomiale en la taille de l'instance initiale.*

Par «équivalente», nous entendons le fait que toute solution optimale initiale du *Winner Determination Problem* est une solution optimale (maximisant l'utilité collective) de notre problème de partage, et vice-versa.

Démonstration Soit $\mathcal{P}_{OR} = (\mathscr{N}_{OR}, \mathscr{O}_{OR}, (\Delta_{1,OR}, \ldots, \Delta_{n,OR}))$ une instance du *Winner Determination Problem* dont les mises sont exprimées dans le langage OR. Pour chaque mise atomique (π_i^j, w_i^j) dans $\Delta_{i,OR}$, nous introduisons un objet supplémentaire α_i^j qui nous servira à empêcher qu'un agent ne reçoive à la fois l'utilité de deux mises dont les lots correspondants ont une intersection non vide.

L'instance $\mathcal{P}_{OR}$ est traduite en une instance $\mathcal{P}_{PBI} = (\mathscr{N}, \mathscr{O}, (\Delta_1, \ldots, \Delta_n), \mathscr{C}, \langle \mathscr{V}_{ind}, \succeq_{ind}, \oplus \rangle, \langle \mathscr{V}, \succeq, \rangle, g)$ du problème de partage de biens indivisibles, où :

- $\mathscr{N} = \mathscr{N}_{OR}$;
- $\mathscr{O} = \mathscr{O}_{OR} \cup \bigcup_{i \in \mathscr{N},\ (\pi_i^j, w_i^j) \in \Delta_{i,OR}} \alpha_i^j$;
- $\forall i,\ \Delta_i = \{\langle \alpha_i^j \wedge \bigwedge_{o_k \in \pi_i^j} \mathbf{o_k}, w_i^j \rangle \mid (\pi_i^j, w_i^j) \in \Delta_{i,OR}\}$
- $\mathscr{C} = \{\neg \mathbf{alloc}(\alpha_{\mathbf{i}}^{\mathbf{j_1}}, \mathbf{i}) \wedge \mathbf{alloc}(\alpha_{\mathbf{i}}^{\mathbf{j_2}}, \mathbf{i}) \mid i \in \mathscr{N},\ (\pi_i^{j_1}, w_i^{j_1}) \in \Delta_i,\ (\pi_i^{j_2}, w_i^{j_2}) \in \Delta_i \text{ et } \pi_i^{j_1} \cap \pi_i^{j_2} \neq \varnothing\} \cup C_{preempt}$;
- $\langle \mathscr{V}_{ind}, \succeq_{ind}, \oplus \rangle = \langle \mathbb{R}^+, \succeq, + \rangle$;
- $\langle \mathscr{V}, \succeq, \rangle = \langle \mathbb{R}^+, \succeq \rangle$;
- $g : (u_1, \ldots, u_n) \mapsto \sum_{i=1}^n u_i$.

Il est clair que $\mathcal{P}_{PBI}$ peut être obtenue à partir de $\mathcal{P}_{OR}$ en temps polynomial. Nous avons maintenant à prouver que toutes les solutions optimales de $\mathcal{P}_{OR}$ correspondent à des solutions optimales de $\mathcal{P}_{PBI}$ et vice-versa.

Soit $\overrightarrow{\pi}_{OR}$ une solution optimale du *Winner Determination Problem* associé à $\mathcal{P}_{OR}$, et pour tout $i \in \mathscr{N}$, soit $\mathscr{W}_i$ une partition de $\pi_{i,OR}$ qui satisfait $\mathscr{W}_i = \underset{\mathscr{W} \text{ partition de } \pi_{i,OR}}{\operatorname{argmax}} \sum_{\pi_i^j \in \mathscr{W}} w_i^j$ (en d'autres termes, la partition $\mathscr{W}_i$ sélectionne l'ensemble des mises de l'agent i dont les valeurs vont contribuer à l'utilité de cet agent). Alors l'allocation $\overrightarrow{\pi}$ de $\mathcal{P}_{PBI}$ telle que $\forall i$, $\pi_i = \pi_{i,OR} \cup \bigcup_{\pi_i^j \in \mathscr{W}_i} \alpha_i^j$ est admissible. En effet, puisque les $\mathscr{W}_i$ sont des partitions, $\overrightarrow{\pi}$ satisfait les contraintes d'exclusion de $\mathscr{C}$, et bien sûr aussi la contrainte de préemption. De plus, on peut vérifier aisément que pour tout agent i, $u_i(\overrightarrow{\pi}) = u_i(\overrightarrow{\pi}_{OR})$, et donc $u_c(\overrightarrow{\pi}) = u_c(\overrightarrow{\pi}_{OR})$ par définition des opérateurs d'agrégation de $\mathcal{P}_{PBI}$.

Maintenant supposons qu'il existe une solution $\overrightarrow{\pi}'$ de $\mathcal{P}_{PBI}$ telle que $u_c(\overrightarrow{\pi}') > u_c(\overrightarrow{\pi})$. Soit alors $\overrightarrow{\pi}'_{OR}$ l'allocation de $\mathcal{P}_{OR}$ telle que toute part $\pi'_{i,OR}$ d'un agent i correspond

à la part π'_i restreinte à $\mathscr{O}_{OR}$. Pour chaque agent i, l'ensemble $\mathscr{W}'_i = \bigcup_{j \text{ tel que } \alpha_i^j \in \pi'_i} \pi_i^j$ est une partition de π'_i. Donc l'utilité individuelle de i dans $\mathcal{P}_{OR}$ est supérieure ou égale à $\sum_{\pi_i^j \in \mathscr{W}'_i} w_i^j = u'_i$, l'utilité individuelle de l'agent i dans $\mathcal{P}_{PBI}$. Nous avons donc $u_c(\overrightarrow{\pi}'_{OR}) \succeq u_c(\overrightarrow{\pi}')$, donc $u_c(\overrightarrow{\pi}'_{OR}) > u_c(\overrightarrow{\pi})$ (avec l'hypothèse de départ que $u_c(\overrightarrow{\pi}') > u_c(\overrightarrow{\pi})$), et donc finalement $u_c(\overrightarrow{\pi}'_{OR}) > u_c(\overrightarrow{\pi}_{OR})$, ce qui contredit le fait que $\overrightarrow{\pi}_{OR}$ est une solution optimale de $\mathcal{P}_{OR}$. Cela prouve au final que $\overrightarrow{\pi}$ est une solution optimale de $\mathcal{P}_{PBI}$.

À l'inverse, la construction d'une solution de $\mathcal{P}_{OR}$ à partir d'une solution $\overrightarrow{\pi}$ de $\mathcal{P}_{PBI}$ peut se faire en projetant $\overrightarrow{\pi}$ la première sur l'ensemble $\mathscr{O}_{OR}$. La preuve est relativement similaire à ce qui précède, donc nous l'omettrons.

En ce qui concerne le langage XOR, on peut vérifier aisément que le remplacement de l'opérateur d'agrégation individuel $\oplus = +$ par $\oplus = \max$ dans la réduction précédente permet d'appliquer cette même réduction à une instance du *Winner Determination Problem* pour laquelle les préférences des agents sont exprimées dans le langage XOR. Pour cette réduction, l'introduction des objets α_i^j n'est même pas nécessaire, car l'opérateur max se charge de rendre les lots mutuellement exclusifs pour le calcul de l'utilité individuelle des agents. ▲

La proposition 3.7 a pour conséquence directe le fait qu'il est possible de traduire en temps et espace polynomiaux toute instance du problème d'enchère combinatoire dont les préférences des agents sont exprimées dans tout langage dérivé des langages OR et XOR (OR-of-XOR, OR*, OR / XOR, etc.) en une instance de notre modèle. En effet, il suffit de traduire dans un premier temps les mises des agents dans le langage OR* ([Nisan, 2000] nous confirme que cette traduction peut se faire en n'utilisant qu'un nombre polynomial d'objets factices), puis de traiter ces mises de la même manière que pour la traduction détaillée dans la preuve de la proposition 3.7.

3.4 Conclusion

Dans ce chapitre, nous avons mis en valeur le phénomène d'explosion combinatoire et défini quelques notions ayant trait à la représentation formelle compacte d'un espace d'alternatives combinatoire (autrement dit, des contraintes restreignant l'ensemble des alternatives admissibles), et des préférences sur cet espace. Nous avons ensuite présenté une vue d'ensemble des langages de représentation compacte classiques utilisés dans la littérature, d'une part pour la représentation des contraintes d'admissibilité de l'espace des alternatives, et d'autre part pour la représentation des préférences des agents sur cet espace.

Nous avons enfin appliqué ces notions à la représentation compacte de deux problèmes de partage de biens indivisibles. Dans un premier temps, nous nous sommes intéressés à la représentation compacte du problème de partage de biens indivisibles avec des préférences dichotomiques, et nous avons montré que les propriétés d'absence d'envie et de Pareto-efficacité pouvaient s'exprimer sous forme logique. Dans un deuxième temps, nous avons défini un cadre formel de représentation compacte dédié au problème de partage de biens indivisibles tel que nous l'avons introduit au chapitre 1 dans la définition 1.37. Ce cadre formel est fondé sur une représentation logique des contraintes d'admissibilité et des préférences des agents. Il s'inspire de la théorie du *welfarisme* pour l'agrégation de ces préférences.

La suite logique de ce travail sur la représentation compacte des problèmes de partage de biens indivisibles est l'analyse de leur complexité théorique. Nous nous pencherons sur cette question dans le chapitre 4. Le chapitre 5 sera quant à lui consacré aux aspects algorithmiques liés au calcul de

solutions optimales au sens du bien-être social, dans le cas particulier de l'ordre de bien-être collectif leximin.