

Sémantique des représentations de connaissance
notes de cours

Jérôme Euzenat

28 décembre 2025

©Jérôme Euzenat, 1997-2000

Ce document a été édité en 2025 avec les modifications suivantes :

- Compilation avec un L^AT_EX moderne (PDFLaTeX, babel, tikZ, index, natbib, hyperref, encodage UTF-8);
- Correction de nombreuses fautes d'orthographe, en particulier dans la conclusion ;
- Redéfinition des figures 2.3, 2.4 et 4.3 ;
- Signalement (en rouge) de l'indisponibilité des figures 2.5, 2.6, 2.7 2.8 et de la correction des exercices 2.7(6) et 3.4–3.7 ;
- Complétion de la preuve de la proposition 2.13 ;
- Meilleures fontes `monotype`.

Il n'a cependant qu'une portée historique.

Chapitre 1

Introduction

LA REPRÉSENTATION DE CONNAISSANCE s'attache à décrire un domaine de connaissance particulier. Elle utilise pour cela des systèmes ou modèles de représentation de connaissance. Parmi ceux-ci, certains sont bâtis sur un formalisme logique (à leur sujet, on peut consulter [Kayser, 1997]). D'autres sont fondés sur une représentation structurée dont l'une des particularités est de sembler proche d'une «représentation naturelle» du domaine en identifiant chaque objet comme une entité distincte. Ils sont l'objet de ce cours.

La sémantique, issue de la linguistique et de la logique, permet d'attribuer un sens aux énoncés d'un langage. Elle a été utilisée depuis des années pour les langages de programmation. Ce travail constitue un aboutissement indiscutable de l'approche formelle des langages de programmation. Il existe plusieurs types de sémantique : dénotationnelle (qui fait correspondre un objet – par exemple, une fonction – à un programme) ou opérationnelle (qui spécifie comment interpréter – calculer son résultat – un programme). Cependant, des voix s'élèvent pour regretter que la sémantique des langages de programmation n'ait pas eu le succès de la syntaxe formelle des langages de programmation qui est lue non seulement par les programmeurs et compilateurs mais aussi par les utilisateurs des langages de programmation (afin qu'ils appréhendent mieux le sens de leurs programmes).

Ici on cherche à utiliser la sémantique formelle à l'endroit des systèmes de représentation de connaissance. On cherchera une sémantique dénotationnelle simple (c'est-à-dire ne faisant pas intervenir de mathématiques plus élaborées que les notions d'ensemble et de fonctions). Une telle sémantique devrait pouvoir être comprise par les utilisateurs de représentation de connaissance et non seulement par les concepteurs de systèmes.

Depuis le début des années 80, un effort de définition formelle des langages de représentation de connaissance a été entrepris. Cet effort concerne non seulement la syntaxe et la sémantique des langages mais aussi les problèmes de décidabilité et de complexité des opérations disponibles sur ceux-ci. On dispose dorénavant de langages restreints complètement définis et d'outils formels pour caractériser langages et opérations. Il subsiste également un certain nombre de problèmes ouverts et de langages échappant à ces analyses.

On se propose de faire la description des résultats et des problèmes subsistant en mettant l'accent sur les techniques utilisées pour définir la sémantique des représentations de connaissance. Ce cours s'adresse aux étudiants désireux d'étudier et de mettre en œuvre une approche réfléchie de la représentation de connaissance. En particulier les fameux compro-

mis entre complexité des traitements et expressivité des langages seront illustrés sur des cas précis.

Le cours apporte les bases permettant l'utilisation de langages structurés pour l'organisation et la recherche d'information. Les perspectives d'utilisation sont bien entendu la réalisation de bases de connaissance mais aussi l'indexation de documents multi-média, la construction de profils d'utilisateurs évolués ou l'organisation de mémoires d'entreprises. Le développement du *world-wide web* en particulier a mis en évidence la difficulté de rechercher dans une masse d'informations informelle et souligne le besoin d'utiliser des langages formalisés de représentation de connaissance.

1.1 Motivation

Ce cours a de nombreuses motivations : il ne vise pas à faire de chacun un théoricien des représentations de connaissance. Par contre il veut :

- aller plus loin que la présentation d'outils et de techniques d'intelligence artificielle (vus en maîtrise ou DEA) et contribuer à dissocier les modèles de représentation de connaissance des systèmes ;
- donner des exemples d'application de la logique en dehors de la logique elle-même et ainsi permettre de mieux appréhender son utilité ;
- mettre en évidence la multiplicité des approches permettant d'établir la sémantique d'une représentation ; ainsi on pourra :
 - refaire une logique (voir chapitre 2) ;
 - traduire vers une logique déjà établie (voir chapitre 3) ;
 - développer une sémantique purement algébrique (voir chapitre 4) ;
- mettre en évidence les avantages et les défauts de chacune des approches ;
- enfin, et c'est la matière du cours, présenter les fondements de la sémantique des représentations de connaissance.

1.2 Format du cours

Ce document reprend les notes du cours de DEA dispensé par l'auteur entre 1996 et 1999 au DEA d'informatique, système et communication de l'université Joseph-Fourier (Grenoble 1). Il s'articule autour de la présentation de l'approche de la sémantique dans trois types de systèmes de représentation de connaissance (logiques terminologiques, graphes conceptuels et Ψ -termes) entrecoupés d'«interludes» consacrés à quelques notions intervenant dans la sémantique des représentations de connaissance.

Le cours s'adresse tant à ceux qui seront amenés à concevoir des systèmes de représentation de connaissance que ceux qui devront les utiliser. Il vient en complément d'un cours général et traditionnel sur l'utilisation des techniques de représentation de connaissance (traditionnellement donné en maîtrise).

Le cours idéal est constitué de 8 séances d'2h. En règle générale, les séances sont distribuées ainsi (si possible dans le même ordre) :

- 0h30** Introduction
- 5h00** Logiques de description
- 5h00** Graphes conceptuels
- 5h00** Ψ -termes
- 0h30** Conclusion et aspects dynamiques

Le cours a été dispensé à des étudiants de DEA mais aucune difficulté ne l'empêche d'apparaître en second cycle. Ceci dit, une certaine familiarité avec les outils de l'informatique est nécessaire afin de percevoir l'utilité de son contenu. Il est clair que la maîtrise de certains autres cours peut aider à la compréhension de celui-ci. Ainsi, la connaissance de la logique et de la notion de complétude est très utile (elle n'est indispensable que pour suivre la démonstration du théorème 3.16). La connaissance de la sémantique des langages de programmation et principalement la sémantique dénotationnelle est aussi un avantage. Cependant, aucune de ces notions n'est indispensable. Les notions non explicités dans le présent ouvrage sont marquées par le symbole $*$.

Un certain nombre d'exercices sont proposés au lecteur. Ils sont indiqués à l'aide de l'échelle de difficulté suivante :

- * facile, du niveau des exemples donnés dans le texte ;
- ** moyenne, du niveau des conséquences données dans le texte (on pourra par ailleurs les démontrer puisque leurs démonstrations ne sont pas données) ;
- *** difficile, du niveau des théorèmes donnés dans le texte.

Le cours s'appuie sur quelques références très complètes qui sont signalées au début de chaque chapitre. Elles peuvent être obtenue facilement à l'adresse électronique suivante : <https://exmo.inria.fr/teaching/src/>

1.3 Remerciements et retours

L'auteur remercie les étudiants ayant participé au cours correspondant à ce document. La motivation et l'implication de nombre d'entre eux a été très profitable. Une mention particulière est destinée à Laurent Tardif et Claire Lecourtier qui ont bien voulu me communiquer leurs notes de cours.

Tout problème ou demande de clarification peut être adressé par courrier électronique à l'auteur :

Jerome.Euzenat@inria.fr

Table des matières

1	Introduction	3
1.1	Motivation	4
1.2	Format du cours	4
1.3	Remerciements et retours	5
2	Logiques terminologiques	9
2.1	Exemple introductif	10
2.2	Le formalisme terminologique \mathcal{FLN}	12
2.3	Le formalisme assertionnel \mathcal{AF}	21
2.4	Calcul de subsomption	25
2.5	Conclusion	42
2.6	Exercices	45
	Modèles et modèle	49
3	Graphes conceptuels	53
3.1	Exemple introductif	54
3.2	Graphes conceptuels	55
3.3	\mathcal{S} -projection et \mathcal{S} -morphisme	60
3.4	Opérations	61
3.5	Sémantique	65
3.6	Complexité	70
3.7	Conclusion	70
3.8	Exercices	71
	Déclarativité et procéduralité	75
4	Ψ-termes	77
4.1	Exemple introductif	78
4.2	Syntaxe	79
4.3	Interprétation	87
4.4	Catégorie OSF	92
4.5	Conclusion	95
4.6	Exercices	96
	Compositionnalité	99

4.7	Compositionnalité	99
4.8	Modularité	99
5	Conclusions et perspectives	101
5.1	Conclusions	101
5.2	Sémantique des systèmes d'objets	102
5.3	Aspects dynamiques des représentations de connaissance	103
	Correction des exercices	105
6.2	Logiques terminologiques	105
6.3	Graphes conceptuels	117
6.4	Ψ -termes	127
	Table des figures	135
	Bibliography	137
	Index	138

Chapitre 2

Logiques terminologiques

LES LOGIQUES TERMINOLOGIQUES ont été diversement dénommées (logiques de description qui semble être l'acception actuelle mais aussi langages basé sur les termes. . .).

Ces systèmes peuvent être considérés comme des logiques dont les constructions principales sont des termes et non des formules. Le seul prédicat utilisé dans les formules est le prédicat de subsomption. Présenter ce système permet de rappeler la manière traditionnelle de définir la sémantique d'une logique tout en présentant une logique qui n'a rien de classique. L'une des leçons à en retenir est que lorsque l'on décrit un langage, il est possible, si l'on s'y prend rigoureusement de se démarquer de la logique tout en restant dans le cadre bien défini pour sa sémantique : la théorie des modèles.

L'exposé des logiques terminologiques présenté ici est principalement inspiré de la thèse de Bernhard Nebel [1990]. On pourra consulter l'introduction plus détaillée et plus récente d'Amedeo Napoli [1997] sur le sujet.

Références

Bernhard Nebel. *Reasoning and revision in hybrid representation systems. Lecture Notes in Artificial Intelligence 422*. Springer Verlag, Berlin (DE), 1990

Amedeo Napoli. Une introduction aux logiques de description. Rapport de Recherche 3314, INRIA Lorraine, Nancy (FR), 1997. <ftp.inria.fr//INRIA/publication/publi-ps-gz/RR/RR-3314.ps.gz>

Aperçu

2.1	Exemple introductif	10
2.2	Le formalisme terminologique \mathcal{FLN}	12
2.3	Le formalisme assertional \mathcal{AF}	21
2.4	Calcul de subsomption	25
2.5	Conclusion	42
2.6	Exercices	45

2.1 Exemple introductif

Un premier exemple va permettre d'appréhender l'utilisation des logiques de descriptions. Imaginons qu'une application manipule la description de la structure d'une entreprise et classe les employés et les équipes. C'est une application typique des bases de données mais dans le cas présent, on veut être capable d'inférer à partir des données. C'est-à-dire, par exemple, de déduire qu'un employé est de sexe féminin ou qu'un groupe de travail correspond à certain critères.

Il faudra donc représenter dans cette exemple les employés et les équipes. L'énoncé suivant donne un rapide aperçu des connaissances à représenter.

Exemple 2.1. Voici la définition informelle d'un univers que l'on voudrait représenter :

Un Homme est un Humain.

Une Femme est un Humain.

Aucun Homme n'est une Femme et *vice versa*.

Une Équipe est (définie comme) un Ensemble, avec au moins 2 membres qui sont tous Humains.

Une Petite-Équipe est (définie comme) une Équipe avec au plus 5 membres.

Une Équipe-moderne est (définie comme) une Équipe

avec au plus 4 membres et

avec au moins 1 chef, qui est un membre, et tous les leaders sont des Femmes.

Cet énoncé introduit un certain nombre de termes (au sens de la terminologie et non de la logique) : Homme, Humain. . . Ces termes sont décrits par des assertions (souvent gouvernées par la copule "est un") dont la partie droite est une expression, souvent composée. L'organisation de ces termes entre eux ressemble donc à une hiérarchie. Les expressions composées font intervenir d'autres termes (membre, chef. . .) qui mettent en relation les premiers. Ils seront appelés relations.

Avant d'introduire le langage des logiques terminologiques il est possible de représenter ceci schématiquement comme sur la figure 2.1. On y distingue les termes représentés par des rectangles, la hiérarchie est matérialisée par des flèches verticales, les relations par des flèches ?

L'exemple présenté ci-dessus introduit une *terminologie*. C'est-à-dire un ensemble de termes génériques concernant un domaine à représenter. Une représentation d'une situation particulière va faire intervenir des *individus* qui dénotent un objet particulier (par exemple, une personne particulière alors que le terme Homme dénote un ensemble de personnes). Ces individus vont être positionnés par rapport à la connaissance terminologique (ainsi, on dira de l'individu HARRY qu'il est un Homme).

Une des innovations des logiques de description était de distinguer ces deux types de connaissances qui seront qualifiés respectivement de terminologique et d'assertionnel. Le formalisme est décomposé en deux parties, une partie dite terminologique (ou T-Box) concernant les grandes catégories impliquées et leur définitions et une partie assertionnelle (ou A-Box) concernant les individus. Cette distinction est présente en base de données où l'on distingue couramment le Schéma des données, dans les langages orientés objets où l'on distingue les classes des instances mais beaucoup moins dans les logiques où les formules et les termes ne sont pas séparés suivant leur généralité (on retrouvera cela dans les §3 et

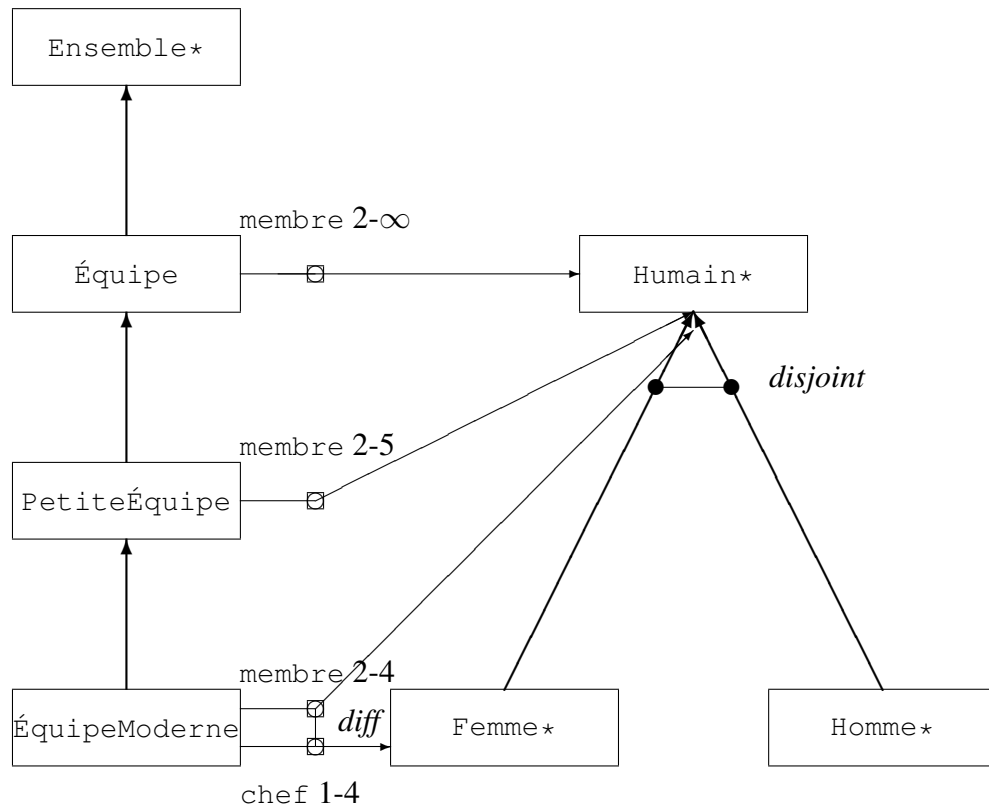


FIGURE 2.1 : Description graphique de l'exemple.

4). Les travaux récents sur les logiques terminologiques ont tendance à abandonner cette dichotomie.

Une fois le domaine de l'application représenté, on cherche à réaliser un certain nombre d'opérations et de déductions à partir d'une telle description :

l'inférence de spécialisation : une *ÉquipeModerne* est une *PetiteÉquipe*,

le test de cohérence : si l'on définit une *ÉquipeLoufoque* comme une *ÉquipeModerne* possédant au moins 53 *chef* alors on pourra déduire qu'il ne peut exister aucune *ÉquipeLoufoque* (et que ce concept est donc incohérent).

le raisonnement sur les individus : TEAM-A est une *ÉquipeModerne* rassemblant au plus trois membres dont deux, DICK et HARRY sont des Hommes, alors le troisième est une Femme et est le chef. Si un troisième membre est un Homme, alors la description est inconsistante.

Ces opérations semblent naturelles à première vue et l'on voit tout l'intérêt de disposer d'un système capable de les mettre en œuvre. Les logiques terminologiques ont été créées dans le but de les réaliser. Le formalisme va être présenté en détail afin de se convaincre de trois propriétés :

- les résultats de ces opérations sont bien correctes vis-à-vis de l'interprétation qui doit être faite du langage ;
- si le résultat d'une telle inférence est vrai dans l'interprétation, alors les mécanismes mis en œuvre pour réaliser les inférences permettent bien de le déduire ;
- le calcul du résultat des opérations est bien réalisable dans un temps acceptable.

Ces trois questions concernent la correction, la complétude et la complexité du formalisme considéré. Elles font partie du schéma classique d'une logique quelconque (voir figure 2.2).

La réponse à ces deux questions est très utile à tous les utilisateurs potentiels du langage. Cependant, comme nous le verrons, ce n'est pas si aisé.

système déductif	syntaxe	sémantique
	langage	
schémas d'axiomes	axiomes	
règles de déduction \vdash		règles d'interprétation \models
théorèmes	≡	conséquences

FIGURE 2.2 : Le schéma classique de description d'une logique

Dans la suite sera défini un langage (§ 2.2.1) (un alphabet et une grammaire). La sémantique (§2.2 et 2.3) étudiera alors le sens des expressions de ce langage à l'aide d'une théorie des modèles (voir figure 2.2). Cela mènera à spécifier les mécanismes (appareil déductif, c'est-à-dire un ensemble d'axiomes et de règles d'inférence) pour dériver ce qui est valide (§2.4). Le couple langage-appareil déductif constitue un système formel. Il sera alors possible d'évaluer la décidabilité du système formel développé et la complexité de la déduction à mettre en œuvre (§2.4.4). Ceci nous amènera à aborder les compromis à développer entre expressivité et complexité (§2.4.5).

2.2 Le formalisme terminologique \mathcal{FLN}

On introduit tout d'abord le formalisme terminologique \mathcal{FLN} destiné à décrire la partie terminologique des logiques de description. Ce formalisme est nommé \mathcal{TF} dans [Nebel, 1990].

2.2.1 Syntaxe

On peut noter que la syntaxe utilisée, contrairement à celle des langages à objets, est une syntaxe à base de formules indépendantes. Ainsi ce qui concerne une même entité pourra se trouver dans diverses assertions au sein de la définition d'une terminologie.

Définition 2.1 (Syntaxe de \mathcal{FLN}).

$$\langle terminology \rangle ::= \{ \langle term-introduction \rangle \mid \langle restriction \rangle \}^*$$

$$\langle term-introduction \rangle ::= \langle concept-introduction \rangle \\ \mid \langle role-introduction \rangle$$

$$\langle concept-introduction \rangle ::= \langle atomic-concept \rangle \doteq \langle concept \rangle \\ \mid \langle atomic-concept \rangle \dot{\leq} \langle concept \rangle \\ \mid \langle atomic-concept \rangle \leq \text{Anything}$$

$$\langle role-introduction \rangle ::= \langle atomic-role \rangle \doteq \langle role \rangle \\ \mid \langle atomic-role \rangle \dot{\leq} \langle role \rangle \\ \mid \langle atomic-role \rangle \leq \text{anyrelation}$$

$$\langle concept \rangle ::= \langle atomic-concept \rangle \\ \mid \text{' (' and } \langle concept \rangle \text{+ ')} \\ \mid \text{' (' all } \langle role \rangle \langle concept \rangle \text{')} \\ \mid \text{' (' atleast } \langle number \rangle \langle role \rangle \text{')} \\ \mid \text{' (' atmost } \langle number \rangle \langle role \rangle \text{')}$$

$$\langle role \rangle ::= \langle atomic-role \rangle$$

$$\langle restriction \rangle ::= \langle atomic-concept \rangle \dot{\oplus} \langle atomic-concept \rangle$$

$$\langle number \rangle ::= \langle non-negative-integer \rangle$$

$$\langle atomic-role \rangle ::= \langle identifier \rangle$$

$$\langle atomic-concept \rangle ::= \langle identifier \rangle$$

Le langage est fondé sur la séparation entre deux types d'entités : les concepts (correspondant aux classes) et les rôles (correspondants aux attributs). Ces entités sont introduites de manière indépendantes et peuvent constituer deux hiérarchies distinctes.

Une terminologie est constituée de deux ensembles de déclarations : des déclarations de restriction et des introductions de termes.

Cette syntaxe est donc fondée sur très peu de constructions :

and la conjonction de concepts

all la restriction de valeurs

atleast restriction de cardinalité

atmost

\doteq introduction de définition

$\dot{\leq}$ introduction de descripton

\oplus introduction d'exclusion

Anything concept maximal (ne souffrant aucune restriction)

anyrelation relation maximale

Exemple 2.2 (\mathcal{FLN} -terminologie correspondant à l'exemple introductif [Nebel, 1990]).

L'exemple de l'Équipe-moderne de [Nebel, 1990] est donné ici dans la syntaxe de \mathcal{FLN} :

Humain	$\dot{\leq}$	Anything
Homme	$\dot{\leq}$	Humain
Femme	$\dot{\leq}$	Humain
Homme	\oplus	Femme
Ensemble	$\dot{\leq}$	Anything
membre	$\dot{\leq}$	anyrelation
Équipe	\doteq	(and Ensemble (all membre Humain) (atleast 2 membre))
Petite-Équipe	\doteq	(and Équipe (atmost 5 membre))
chef	$\dot{\leq}$	membre
Équipe-Moderne	\doteq	(and Équipe (atmost 3 membre) (atleast 1 chef) (all chef Femme))

Définition 2.2 (Sous-ensembles de \mathcal{FLN}).

\mathcal{N}_C ensemble des concepts atomiques;

\mathcal{N}_R ensemble des rôles atomiques ($\mathcal{N}_C \cap \mathcal{N}_R = \emptyset$);

$\mathcal{N} = \mathcal{N}_C \cup \mathcal{N}_R$;

\mathcal{FLN}_C ensemble des concepts (règle concept de la définition 2.1));

\mathcal{FLN}_R ensemble des rôles;

$\mathcal{FLN}_T = \mathcal{FLN}_C \cup \mathcal{FLN}_R$ (règle role de la définition 2.1);

Exemple 2.3 (Sous ensembles de \mathcal{FLN} correspondant à l'exemple introductif). On a les sous-ensembles suivants :

$\mathcal{N}_C = \{\text{Human, Man, Woman, Set, Team, Small-Team, Modern-Team}\}$

$\mathcal{N}_R = \{\text{member, leader}\}$

$\mathcal{FLN}_C = \{(\text{and Man Woman}), (\text{all member Human}), (\text{atleast 2 member}), (\text{atmost 5 member}), (\text{atmost 3 member}), (\text{atleast 1 leader}), (\text{all leader Woman}), (\text{and Set (all member Human) (atleast 2 member)}), (\text{and Set (all member Human), ...}) \cup \mathcal{N}_C$

$\mathcal{FLN}_R = \mathcal{N}_R$

Définition 2.3 (Graphe terminologique). On définit l'arbre étiqueté $G_t = \langle n_t, N_t, A_t, \lambda \rangle$ associé à un terme t par :

- un nœud correspondant au nom du concept (resp. role) atomique dans le cas d'un concept (resp. role) atomique ($\langle a, \{a\}, \emptyset, \emptyset \rangle$ pour $t=a$);

- un nœud dont les fils sont les arbres étiquetés associés aux arguments de ce constructeur et dont les arcs sont étiquetés par le constructeur $(\langle n_t, \{n_t\} \cup \bigcup_{i=1}^p N_p, \{\langle n_t, n_{t_i} \rangle_{i=1}^p \cup \bigcup_{i=1}^p A_p, \bigcup_{i=1}^p \{\langle \langle n_t, n_{t_i} \rangle, \text{op} \rangle\} \rangle$ pour $t = (\text{opt}_1 \dots t_p)$.

Soit un ensemble T d'expressions obéissant à la syntaxe donnée à la définition 2.1, on définit le *graphe terminologique* de T par le plus petit graphe à arcs étiqueté $G_T = \langle N, A, \lambda \rangle$ tel que :

- $\mathcal{N} \subseteq N$;
- si $a \dot{=} t \in T$ alors G_t est un sous-graphe de G_T et $\exists e = \langle a, n_t \rangle \in A$ et $\lambda(e) = \dot{=}$.

La relation induite par le graphe terminologique réduite aux noms de concepts est nommée relation de dépendance et notée \ll . On appelle *cycle terminologique* toute suite de nœuds $\{n_0, \dots, n_p\}$ telle que $n_0 = n_p$ et $\forall i \in 1 \dots p, \langle n_{i-1}, n_i \rangle \in A$.

Définition 2.4 (\mathcal{FLN} -terminologie). Une \mathcal{FLN} -terminologie est un ensemble d'expressions obéissant à la syntaxe donnée à la définition 2.1 dans lequel :

- chaque atome n'est introduit qu'une seule fois ;
- il n'y a pas de cycles terminologiques.

Les termes sont donc les descriptions engendrées à l'aide des constructeurs à partir des atomes.

Définition 2.5 (Niveau). Soit T une terminologie et $\langle N, A \rangle$ son graphe terminologique, on définit la fonction $niv : N \times \mathcal{L} \rightarrow N$ par $niv(a, T) = \max(0, \max_{a; \langle n, a \rangle \in A} niv(n, T) + 1)$.

Conséquence 2.1. niv est bien fondée pour les terminologies sans cycles.

2.2.2 Sémantique

Le but de la sémantique est d'établir une correspondance entre des objets syntaxiques tels que les termes présentés plus haut et ce qu'ils peuvent dénoter. Pour cela on définit le domaine de dénotation (qui dans notre cas sera un ensemble quelconque) et une fonction permettant d'associer la dénotation du terme dans cet ensemble.

Définition 2.6 (\mathcal{FLN} -extension). Soit \mathcal{D} un ensemble (nommé domaine) et

$$\begin{aligned} \mathcal{E} : \mathcal{FLN}_C &\longrightarrow 2^{\mathcal{D}}, \\ \mathcal{FLN}_R &\longrightarrow 2^{\mathcal{D} \times \mathcal{D}} \end{aligned}$$

\mathcal{E} est une fonction d'extension de \mathcal{FLN} (\mathcal{FLN} -extension) si et seulement si :

$$\begin{aligned}\mathcal{E}(\text{(and } c_1 \dots c_n)) &= \bigcap_{i \in 1..n} \mathcal{E}(c_i) \\ \mathcal{E}(\text{(all } r \ c)) &= \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(c)\} \\ \mathcal{E}(\text{(atleast } n \ r)) &= \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \geq n\} \\ \mathcal{E}(\text{(atmost } n \ r)) &= \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq n\} \\ \mathcal{E}(\text{Anything}) &= \mathcal{D} \\ \mathcal{E}(\text{anyrelation}) &= \mathcal{D} \times \mathcal{D}\end{aligned}$$

Définition 2.7 (\mathcal{FLN} -structure). Soit T une terminologie (suivant la syntaxe donnée au §2.2.1), \mathcal{D} et \mathcal{E} comme dans la définition 2.6, \mathcal{E} est une \mathcal{FLN} -extension par rapport à T et $\langle \mathcal{D}, \mathcal{E} \rangle$ une \mathcal{FLN} -structure pour T si et seulement si :

$\forall a \in \mathcal{N}, c, c' \in \mathcal{N}_C$ et primitives et $t \in \mathcal{FLN}_T$:

$$\begin{aligned}(a \doteq t) \in T &\Rightarrow \mathcal{E}(a) = \mathcal{E}(t) \\ (a \dot{\leq} t) \in T &\Rightarrow \mathcal{E}(a) \subseteq \mathcal{E}(t) \\ (c \dot{\oplus} c') \in T &\Rightarrow \mathcal{E}(c) \cap \mathcal{E}(c') = \emptyset\end{aligned}$$

Les structures ne correspondent pas immédiatement aux interprétations de la logique car on voudra construire des interprétations sur des assertions individuelles.

Définition 2.8 (\mathcal{FLN} -subsumption). Un terme t est subsumé par un terme t' dans une terminologie T (noté $t \preceq_T t'$) si et seulement si pour toute \mathcal{FLN} -structure $\langle \mathcal{D}, \mathcal{E} \rangle$ de T , $\mathcal{E}(t) \subseteq \mathcal{E}(t')$

Conséquence 2.2. \preceq_T est une relation transitive et reflexive.

Définition 2.9 (Équivalence, exclusion, incohérence). Soient les termes t et t' dans une terminologie T , les notions suivantes sont définies si pour toute \mathcal{FLN} -structure $\langle \mathcal{D}, \mathcal{E} \rangle$ de T :

$$\begin{aligned}t \approx_T t' &\stackrel{\text{def}}{=} \mathcal{E}(t) = \mathcal{E}(t') && \text{(équivalence)} \\ t \oplus_T t' &\stackrel{\text{def}}{=} \mathcal{E}(t) \cap \mathcal{E}(t') = \emptyset && \text{(exclusion)} \\ t \perp_T &\stackrel{\text{def}}{=} \mathcal{E}(t) = \emptyset && \text{(incohérence)}\end{aligned}$$

Il faut noter la différence entre les définitions 2.7 et 2.9 : la première donne les contraintes pesant sur les extensions pour être des structures d'une terminologies alors que la seconde définit les conséquences propres à toutes les structures (c'est-à-dire toutes les extensions respectant les contraintes en question). Ainsi, si $(a \doteq t) \in T$ entraîne $a \approx_T t$, l'inverse n'est pas vrai.

Notation (Nothing). Nothing peut-être pris comme l'abréviation de $(\text{and } (\text{atleast } 1 \ r) \ (\text{atmost } 0 \ r))$.

On peut en effet, établir qu'avec cette définition :

$$\begin{aligned}
\mathcal{E}(\text{Nothing}) &= \mathcal{E}((\text{and } (\text{atleast } 1 \text{ r}) (\text{atmost } 0 \text{ r}))) \\
&= \mathcal{E}((\text{atleast } 1 \text{ r})) \wedge \mathcal{E}((\text{atmost } 0 \text{ r})) \\
&= \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \geq 1\} \\
&\quad \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq 0\} \\
&= \{x \in \mathcal{D} \mid 0 \geq \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \geq 1\} \\
&= \emptyset
\end{aligned}$$

Conséquence 2.3. Si l'on introduit le concept *Nothing* tel que $\mathcal{E}(\text{Nothing}) = \emptyset$ alors on peut définir :

$$\begin{aligned}
t \approx_T t' &\text{ssi } t \preceq_T t' \text{ et } t' \preceq_T t \\
t \oplus_T t' &\text{ssi } (\text{and } t t') \preceq_T \text{Nothing} \\
t \perp_T &\text{ssi } t \preceq_T \text{Nothing}
\end{aligned}$$

ou alternativement

$$t \preceq_T t' \text{ssi } t \approx_T (\text{and } t t')$$

Comme toutes les opérations désirées sont réductibles à la subsonction on ne s'intéressera plus qu'à cette dernière puisque si l'on sait tester la subsonction, on sait tester toutes les autres propriétés.

2.2.3 Forme normale

On introduit ici une forme normale pour les terminologie qui permet de simplifier le traitement de celles-ci car elle supprime un certain nombre de constructeurs et d'introduceurs (sans pour autant trop augmenter la taille des terminologies).

La mise en forme normale d'une terminologie consiste à transformer toutes les introductions de concepts qui ne sont pas des définitions par des définitions. Pour cela on ajoute un ensemble de concepts (resp. de rôles) atomiques qui doivent représenter la *composante primitive* de ces concepts primitifs. Les concepts seront alors définis en fonction de cette composante primitive. De même, pour supprimer les déclarations d'exclusion on introduit un opérateur de négation (*anot*) restreint aux concepts primitifs. (structure un peu plus et introduire le *androle*).

On peut, par ailleurs, supprimer *AnyThing* et *anyrelation* qui deviennent inutile.

Le nouveau langage ainsi défini est nommé \mathcal{ALNR} (il était nommé \mathcal{NTF} dans [Nebel, 1990] mais été rebaptisé par la suite). L'exemple 2.4 donne un aperçu de ce que donne la transformation.

Exemple 2.4 (Mise en forme normale). La terminologie :

$$\begin{array}{ll}
\text{Man} & \dot{\leq} \text{Human} \\
\text{Woman} & \dot{\leq} \text{Human} \\
\text{Man} & \oplus \text{Woman}
\end{array}$$

devient

$$\begin{aligned} \text{Man} & \doteq (\text{and Human } \overline{\text{Man}}) \\ \text{Woman} & \doteq (\text{and Human } \overline{\text{Woman}} \text{ (anot } \overline{\text{Man}})) \end{aligned}$$

Les deux dernières expressions reflètent pas la symétrie qui prévaut dans la description initiale entre les deux termes définis mais on se convaincra, à l'aide de l'interprétation, que cette symétrie est restituée.

Définition 2.10 (Syntaxe des terminologies en forme normale).

$$\begin{aligned} \langle \text{terminology} \rangle & ::= \langle \text{term-introduction} \rangle^* \\ \langle \text{term-introduction} \rangle & ::= \langle \text{concept-introduction} \rangle \\ & \quad | \langle \text{role-introduction} \rangle \\ \langle \text{concept-introduction} \rangle & ::= \langle \text{atomic-concept} \rangle \doteq \langle \text{concept} \rangle \\ \langle \text{role-introduction} \rangle & ::= \langle \text{atomic-role} \rangle \doteq \langle \text{role} \rangle \\ \langle \text{concept} \rangle & ::= \langle \text{atomic-concept} \rangle \\ & \quad | \langle \text{primitive-concept-component} \rangle \\ & \quad | \text{' (' anot } \langle \text{primitive-concept-component} \rangle \text{')}' \\ & \quad | \text{' (' and } \langle \text{concept} \rangle \text{' + ')}' \\ & \quad | \text{' (' all } \langle \text{role} \rangle \langle \text{concept} \rangle \text{')}' \\ & \quad | \text{' (' atleast } \langle \text{number} \rangle \langle \text{role} \rangle \text{')}' \\ & \quad | \text{' (' atmost } \langle \text{number} \rangle \langle \text{role} \rangle \text{')}' \\ \langle \text{role} \rangle & ::= \langle \text{atomic-role} \rangle \\ & \quad | \langle \text{primitive-role-component} \rangle \\ & \quad | \text{' (' androle } \langle \text{role} \rangle \text{' + ')}' \\ \langle \text{number} \rangle & ::= \langle \text{non-negative-integer} \rangle \\ \langle \text{atomic-role} \rangle & ::= \langle \text{identifier} \rangle \\ \langle \text{atomic-concept} \rangle & ::= \langle \text{identifier} \rangle \\ \langle \text{primitive-concept-component} \rangle & ::= \langle \text{identifier} \rangle \\ \langle \text{primitive-role-component} \rangle & ::= \langle \text{identifier} \rangle \end{aligned}$$

Définition 2.11 (Sous-ensembles de $\mathcal{ALN}\mathcal{R}$).

$$\begin{aligned} \overline{\mathcal{N}_C} & \text{ ensemble des concepts atomiques;} \\ \overline{\mathcal{N}_R} & \text{ ensemble des rôles atomiques } (\overline{\mathcal{N}_C} \cap \overline{\mathcal{N}_R} = \emptyset); \\ \overline{\mathcal{N}} & \overline{\mathcal{N}_C} \cup \overline{\mathcal{N}_R}; \end{aligned}$$

$\mathcal{ALNR}_C, \mathcal{ALNR}_R, \mathcal{ALNR}_T$ as usual.

Exemple 2.5 (Sous ensembles de \mathcal{ALNR} correspondant à l'exemple introductif). On a les sous-ensembles suivants :

$$\overline{\mathcal{N}_C} = \{\overline{\text{Man}}, \overline{\text{Woman}}, \overline{\text{Team}}, \overline{\text{Small} - \text{Team}}, \overline{\text{Modern} - \text{Team}}\} \cup \mathcal{N}_C$$

$$\overline{\mathcal{N}_R} = \{\overline{\text{leader}}\} \cup \mathcal{N}_R$$

$$\overline{\mathcal{FLN}_C} = \{(\text{all member Human}), (\text{atleast 2 member}), (\text{atmost 5 member}), (\text{atmost 3 member}), (\text{atleast 1 leader}), (\text{all leader Woman}), (\text{and Set (all member Human) (atleast 2 member)}), (\text{and Set (all member Human), ...}) \cup \overline{\mathcal{N}_C}$$

$$\overline{\mathcal{FLN}_R} = \{(\text{androle member leader}), (\text{androle member } \overline{\text{leader}}), \dots\} \cup \overline{\mathcal{N}_R}$$

TNORM(T)

1 **if** $c \in \mathcal{N}_C$

2 **then switch**

3 **case** $(c \dot{=} t) \in T : (c \dot{=} t) \in \overline{T}$

4 **case** $(c \dot{\leq} t) \in T : (c \dot{=} (\text{and } \bar{c} t P)) \in \overline{T}$

5 **case default** : $(c \dot{=} (\text{and } \bar{c} P)) \in \overline{T}$

6 avec $P = \emptyset$ si $(c \dot{\oplus} c') \notin T$

7 et $P = (\text{anot } c_1), \dots (\text{anot } c_n)$ si $(c \dot{\oplus} c_1), \dots (c \dot{\oplus} c_n) \in T$

8 avec $c_1, \dots c_n$ primitifs

9 **if** $r \in \mathcal{N}_R$

10 **then switch**

11 **case** $(r \dot{=} t) \in T : (r \dot{=} t) \in \overline{T}$

12 **case** $(r \dot{\leq} t) \in T : (r \dot{=} (\text{androle } \bar{r} t)) \in \overline{T}$

13 **case default** : $(r \dot{=} \bar{r}) \in \overline{T}$

14 avec suppression de toutes les occurences d'Anything ou anyrelation

Algorithme 2.1 (Algorithme de mise en forme normale).

Conséquence 2.4 (Préservation de l'acyclicité). La transformation d'une \mathcal{FLN} -terminologie T vers \overline{T} est une terminologie sans cycle.

Démonstration. 1) FLN-term = sans cycle

2) ajout uniquement de primitifs non définis et de anot primitifs non définis. □

Proposition 2.5 (Complexité de TNORM). La transformation de T vers \overline{T} est linéaire* et la taille de la terminologie résultante est au plus le double de celle de la terminologie initiale.

Définition 2.12 (\mathcal{ALNR} -extension). Soit \mathcal{D} un ensemble (nommé domaine) et

$$\begin{aligned} \mathcal{E} : \mathcal{ALNR}_C &\longrightarrow 2^{\mathcal{D}}, \\ \mathcal{ALNR}_R &\longrightarrow 2^{\mathcal{D} \times \mathcal{D}} \end{aligned}$$

\mathcal{E} est une fonction d'extension de \mathcal{ALNR} (\mathcal{ALNR} -extension) si et seulement si :

$$\begin{aligned}\mathcal{E}(\text{(and } c_1 \dots c_n)) &= \bigcap_{i=1}^n \mathcal{E}(c_i) \\ \mathcal{E}(\text{(all } r \ c)) &= \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(c)\} \\ \mathcal{E}(\text{(atleast } n \ r)) &= \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \geq n\} \\ \mathcal{E}(\text{(atmost } n \ r)) &= \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq n\} \\ \mathcal{E}(\text{(anot } c)) &= \mathcal{D} \setminus \mathcal{E}(c) \\ \mathcal{E}(\text{(androle } r_1 \dots r_n)) &= \bigcap_{i=1}^n \mathcal{E}(r_i)\end{aligned}$$

Définition 2.13 (\mathcal{ALNR} -structure). Soit \bar{T} une terminologie (suivant la syntaxe donnée au §2.10), \mathcal{D} et \mathcal{E} comme dans la définition 2.12, \mathcal{E} est une \mathcal{ALNR} -extension par rapport à T et $(\langle \mathcal{D}, \mathcal{E} \rangle)$ une \mathcal{ALNR} -structure pour T si et seulement si :
 $\forall a \in \bar{\mathcal{N}}$ et $t \in \mathcal{ALNR}_T$:

$$(a \dot{=} t) \in T \Rightarrow \mathcal{E}(a) = \mathcal{E}(t)$$

Le théorème suivant constitue un premier aboutissement de la sémantique des langages de représentation de connaissance. Il permet en effet d'établir que l'on peut changer la représentation syntaxique sans que ceci ne modifie le sens de la représentation.

Théorème 2.6 (Équivalence des structures sémantiques). Soit T une \mathcal{FLN} -terminologie et \bar{T} sa forme normale, alors pour toute \mathcal{FLN} -structure $(\langle \mathcal{D}, \mathcal{E} \rangle)$ de T , il existe une \mathcal{ALNR} -structure $(\langle \bar{\mathcal{D}}, \bar{\mathcal{E}} \rangle)$ de \bar{T} et vice-versa telle que pour tout $t \in \mathcal{FLN}_T$:

$$\mathcal{E}(t) = \bar{\mathcal{E}}(t)$$

Remarque.

$\bar{\mathcal{E}}(t)$ n'est pas correct : il y a en plus `Anything` et `anyrelation`. `Anything` sera remplacé par `(atleast 0 c)` de même pour `anyrelation` (on fera coïncider les interprétations avec $\mathcal{D} \times \mathcal{D}$).

Démonstration. \Rightarrow Soit $(\langle \mathcal{D}, \mathcal{E} \rangle)$ structure sémantique de T . Supposons que $\mathcal{D} = \bar{\mathcal{D}}$, $\bar{\mathcal{E}}(\bar{a}) = \mathcal{E}(a)$ et $\bar{\mathcal{E}}(a) = \mathcal{E}(a)$ pour tout terme atomique a (`Anything` et `anyrelation` sont alors compris comme des concepts et rôles atomiques de \mathcal{ALNR}_T). Alors,

1. $\bar{\mathcal{E}}$ peut être étendu en une \mathcal{ALNR} -extension de sorte que $\forall t \in \mathcal{ALNR}_T, \bar{\mathcal{E}}(t) = \mathcal{E}(t)$, et
2. la définition de \mathcal{ALNR} -structure de \bar{T} (voir définition 2.13) sera respectée parce que :
 - pour les concepts atomiques, \mathcal{E} et $\bar{\mathcal{E}}$ coïncident ($a \dot{=} t \in T \Rightarrow a \dot{=} t \in \bar{T}$ et $\mathcal{E}(a) = \mathcal{E}(t) \Rightarrow \bar{\mathcal{E}}(a) = \bar{\mathcal{E}}(t)$);
 - pour les termes primitifs sans disjonction, $\forall a \dot{\leq} t \in T \Rightarrow a \dot{=} (\text{and } t \ \bar{a}) \in \bar{T}$ et $\mathcal{E}(a) \subseteq \mathcal{E}(t) \Rightarrow \bar{\mathcal{E}}(a) = \bar{\mathcal{E}}(t) \cap \bar{\mathcal{E}}(\bar{a})$;

- pour les termes primitifs avec disjonction ($a \dot{\leq} t \in T$ et $\exists i \in [1 \dots n]$; $a \dot{\oplus} a_i \in T$) alors $\mathcal{E}(a) \subseteq \mathcal{E}(t)$ et $\mathcal{E}(a_i) \subseteq \mathcal{E}(t_i)$ et $\mathcal{E}(a) \cap \mathcal{E}(a_i) = \emptyset$ et $a \dot{=} (\text{and } t \bar{a} \text{ (anot } \bar{a})) \in \bar{T}$ ce qui implique que $\bar{\mathcal{E}}(a) = \bar{\mathcal{E}}(t) \cap \bar{\mathcal{E}}(\bar{a}) \cap \mathcal{D} \setminus \bar{\mathcal{E}}(\bar{a}_i)$

\Leftarrow Soit $\langle \bar{\mathcal{D}}, \bar{\mathcal{E}} \rangle$ structure sémantique de \bar{T} . Soit $\mathcal{D} = \bar{\mathcal{D}}$, $\mathcal{E}(t) = \bar{\mathcal{E}}(t)$ (en ajoutant $\mathcal{E}(\text{Anything}) = \mathcal{D}$ et $\mathcal{E}(\text{anyrelation}) = \mathcal{D} \times \mathcal{D}$), alors :

- \mathcal{E} est une \mathcal{FLN} -extension,
- $a \dot{=} t \in T$ et $a \dot{=} t \in \bar{T}$ par conséquent $\bar{\mathcal{E}}(a) = \bar{\mathcal{E}}(t)$ qui implique $\mathcal{E}(a) = \mathcal{E}(t)$,
- $a \dot{\leq} t \in T$ et $a \dot{=} (\text{and } t \bar{a}) \in \bar{T}$ ce qui s'interprète comme $\bar{\mathcal{E}}(a) = \bar{\mathcal{E}}(t) \cap \bar{\mathcal{E}}(\bar{a})$ ce qui implique que $\mathcal{E}(a) \subseteq \mathcal{E}(t)$,
- $a \dot{\oplus} a' \in T$ et $a \dot{=} (\text{and } t \bar{a} \text{ (anot } \bar{a}')) \in \bar{T}$ ce qui s'interprète comme $\bar{\mathcal{E}}(a) = \bar{\mathcal{E}}(t) \cap \bar{\mathcal{E}}(\bar{a}) \cap \mathcal{D} \setminus \bar{\mathcal{E}}(\bar{a}')$ et $\bar{\mathcal{E}}(\bar{a}') = \bar{\mathcal{E}}(t) \cap \bar{\mathcal{E}}(\bar{a})$ ce qui conduit à $\mathcal{E}(a) \cap \mathcal{E}(a') = \emptyset$.

C'est donc une structure sémantique de T . □

On peut en fait vérifier qu'il y a isomorphisme entre les structures syntaxiques de T et celles de \bar{T} . Ceci sera utile pour la déduction comme on le voit dans la conséquence suivante.

Question intéressante d'un étudiant : est-ce que $\bar{\mathcal{E}}(a) = \bar{\mathcal{E}}(\bar{a})$?

Conséquence 2.7. Pour tous termes $t, t' \in \mathcal{FLN}_T$ et soit T une \mathcal{FLN} -terminologie,

$$t \preceq_T t' \text{ ssi } t \preceq_{\bar{T}} t'$$

2.3 Le formalisme assertional \mathcal{AF}

2.3.1 Syntaxe et sémantique

Afin de décrire les objets à considérer par le système, on utilise un langage différent que pour décrire les terminologies. Une *description du monde* W sera exprimée dans un formalisme dit *assertional* : \mathcal{AF} .

Définition 2.14 (Syntaxe de \mathcal{AF}).

$\langle \text{world-description} \rangle ::= \{ \langle \text{object-description} \rangle \mid \langle \text{relation-description} \rangle \}^*$

$\langle \text{object-description} \rangle ::= ' (' \langle \text{atomic-concept} \rangle \langle \text{object} \rangle ') '$

$\langle \text{relation-description} \rangle ::= ' (' \langle \text{atomic-role} \rangle \langle \text{object} \rangle \langle \text{object} \rangle ') '$
 | $' (' \langle \text{atomic-role} \rangle \langle \text{object} \rangle ' (' \text{atleast } \langle \text{number} \rangle ') ') '$
 | $' (' \langle \text{atomic-role} \rangle \langle \text{object} \rangle ' (' \text{atmost } \langle \text{number} \rangle ') ') '$

Définition 2.15 (Description du monde).

Il existe deux contraintes supplémentaires qui ne s'expriment pas dans la grammaire :

- l’hypothèse de nom unique (*unique name assumption*) qui signifie qu’un nom d’objet désigne toujours la même entité et
- l’hypothèse du monde ouvert (*open world assumption*) qui signifie que l’on ne supposera jamais que le monde est complètement décrit. Ainsi, il est possible que d’autres Human existent tout comme il est possible que les trois membres de TEAM-A soient tous trois leader de l’équipe. Cette dernière hypothèse s’oppose à l’hypothèse du monde clos (utilisée en particulier dans les bases de données) qui spécifie que si une donnée n’a pas été explicitement donnée, elle n’est pas vraie ou à la négation par échec fini (utilisé dans le langage Prolog) qui spécifie que tout ce qui n’est pas déductible dans un laps de temps fini n’est pas vrai.

Définition 2.16 (Sous-ensembles de \mathcal{AF}).

\mathcal{N}_O ensemble des marqueur individuels ;

Exemple 2.6 (Syntaxe de \mathcal{AF}).

```
(Modern-team TEAM-A)
(Man          DICK)
(Human        MARY)
(member       TEAM-A DICK)
(member       TEAM-A HARRY)
(member       TEAM-A MARY)
(member       TEAM-A (atmost 3))
```

L’ensemble \mathcal{N}_O est donc réduit à $\{\text{TEAM-A, DICK, HARRY, MARY}\}$.

Définition 2.17 (\mathcal{AF} -interprétation). Soit \mathcal{D} un ensemble (nommé domaine), soient \mathcal{N}_O , \mathcal{N}_C et \mathcal{N}_R les ensembles d’objets, de concepts et de rôles atomiques, une fonction

$$\begin{aligned} \mathcal{I} : \mathcal{N}_O &\longrightarrow \mathcal{D}, \\ \mathcal{N}_C &\longrightarrow 2^{\mathcal{D}} \\ \mathcal{N}_R &\longrightarrow 2^{\mathcal{D} \times \mathcal{D}} \end{aligned}$$

injective sur \mathcal{N}_O est nommé fonction d’interprétation.

Une paire $\langle \mathcal{D}, \mathcal{I} \rangle$ nommée \mathcal{AF} -interprétation satisfait une description δ (noté $\models_{\langle \mathcal{D}, \mathcal{I} \rangle}^{\mathcal{AF}} \delta$) si et seulement si :

$$\begin{aligned} \models_{\langle \mathcal{D}, \mathcal{I} \rangle}^{\mathcal{AF}} (c \ o) &\text{ ssi } \mathcal{I}(o) \in \mathcal{I}(c) \\ \models_{\langle \mathcal{D}, \mathcal{I} \rangle}^{\mathcal{AF}} (r \ o \ o') &\text{ ssi } \langle \mathcal{I}(o), \mathcal{I}(o') \rangle \in \mathcal{I}(r) \\ \models_{\langle \mathcal{D}, \mathcal{I} \rangle}^{\mathcal{AF}} (r \ o \ (\text{atleast } n)) &\text{ ssi } \|\{\langle \mathcal{I}(o), x \rangle \in \mathcal{I}(r)\}\| \geq n \\ \models_{\langle \mathcal{D}, \mathcal{I} \rangle}^{\mathcal{AF}} (r \ o \ (\text{atmost } n)) &\text{ ssi } \|\{\langle \mathcal{I}(o), x \rangle \in \mathcal{I}(r)\}\| \leq n \end{aligned}$$

L'injectivité de la fonction d'interprétation permet de rendre compte de l'hypothèse de nom unique.

Définition 2.18 (\mathcal{AF} -modèle). Une \mathcal{AF} -interprétation est un \mathcal{AF} -modèle d'une description du monde W si et seulement si elle satisfait toute les assertions de W (on note $\models_{\langle \mathcal{D}, \mathcal{I} \rangle}^{\mathcal{AF}} W$)

Définition 2.19 (Conséquence).

$$W \models^{\mathcal{AF}} \delta \text{ ssi } \forall \langle \mathcal{D}, \mathcal{I} \rangle; \models_{\langle \mathcal{D}, \mathcal{I} \rangle}^{\mathcal{AF}} W, \models_{\langle \mathcal{D}, \mathcal{I} \rangle}^{\mathcal{AF}} \delta$$

2.3.2 Raisonnement terminologico-assertionnel

On va maintenant faire coopérer les raisonnements terminologiques et assertionnels. Ce type de raisonnement est nommé « hybride » dans [Nebel, 1990]. On préférera être plus précis. La possibilité de pouvoir développer deux formalismes indépendamment et de pouvoir les assembler est un exemple de l'aspect modulaire très développé des logiques terminologiques.

Pour cela, il faut mettre en correspondance les concepts et les rôles utilisés dans \mathcal{AF} et \mathcal{FLN} puisque ce sont les seuls éléments en commun. C'est le rôle des définitions 2.20 et 2.22.

Définition 2.20 (\mathcal{AF} -modèles respectant une terminologie). Un \mathcal{AF} -modèle $\langle \mathcal{D}, \mathcal{I} \rangle$ d'une description W respecte la terminologie T s'il existe une \mathcal{FLN} -structure de T $\langle \mathcal{D}, \mathcal{E} \rangle$ telle que la restriction de \mathcal{I} et \mathcal{E} aux concepts et rôles atomiques coïncident :

$$\mathcal{I}|_{\mathcal{N}} = \mathcal{E}|_{\mathcal{N}}$$

Définition 2.21 (Conséquence modulo une terminologie). δ est une conséquence de W modulo la terminologie T (noté $W \models^T \delta$) si et seulement si tout \mathcal{AF} -modèle de W respectant T a pour conséquence δ .

Définition 2.22 (\mathcal{FLN} -structure respectant une description du monde). Une \mathcal{FLN} -structure $\langle \mathcal{D}, \mathcal{E} \rangle$ d'une terminologie T respecte une description du monde W s'il existe un \mathcal{AF} -modèle $\langle \mathcal{D}, \mathcal{I} \rangle$ de W tel que $\mathcal{I}|_{\mathcal{N}} = \mathcal{E}|_{\mathcal{N}}$.

Définition 2.23 (\mathcal{FLN} -subsumption modulo une description du monde). $t \preceq_T^W t'$ si et seulement si pour toute \mathcal{FLN} -structure respectant W , $\mathcal{E}(t) \subseteq \mathcal{E}(t')$.

Proposition 2.8. Pour tout T et W , s'il existe un modèle de W qui respecte T , alors, pour tout $t, t' \in \mathcal{FLN}_T$:

$$t \preceq_T t' \text{ ssi } t \preceq_T^W t'$$

Ce dernier résultat indique que la description du monde n'a aucune influence sur la \mathcal{FLN} -subsumption (à condition qu'il existe un modèle de W qui respecte T). Ainsi, si on rajoute une description du monde arbitrairement grande, cela que compliquera pas le test de \mathcal{FLN} -subsumption.

2.3.3 Sémantique constructive

Définition 2.24 (Affectation primitive). Une affectation primitive est n'importe quelle fonction :

$$\begin{aligned} \mathcal{P} : \overline{\mathcal{N}}_C &\longrightarrow 2^{\mathcal{D}}, \\ \overline{\mathcal{N}}_R &\longrightarrow 2^{\mathcal{D} \times \mathcal{D}} \end{aligned}$$

Théorème 2.9 (Sémantique constructive). Soit \overline{T} une \mathcal{ALNR} -terminologie sans cycle, alors toute affectation primitive peut être étendue inductivement en une unique \mathcal{ALNR} -structure par rapport à \overline{T}

Démonstration. La démonstration peut être obtenue par induction sur le niveau des concepts :

$niv(t, T) = 0$ alors $\mathcal{E}(t) = \mathcal{P}(t)$,

$niv(t, T) > 0$ si \mathcal{E} est construit pour les termes de niveau $niv(t, T) - 1$ alors :

$$\begin{cases} \text{si } t \notin \mathcal{N} \cup \overline{\mathcal{N}} & \text{alors il est construit pour } t \text{ par les équations de la définition} \\ & \text{de fonction d'extension (textdfinition 2.12)} \\ \text{si } t \doteq t' \in T & \text{alors } \mathcal{E}(t) = \mathcal{E}(t') \end{cases}$$

La dernière équation garanti que la fonction d'extension ainsi construite est bien une \mathcal{ALNR} -structure. \square

Exemple 2.7. Si l'on considère la taxonomie suivante :

Camping-car \doteq (and Maison Auto)

Campeur \doteq (and Humain (all domicile Camping-car))

et l'affectation primitive pour le domaine

$$D = \{\text{Pierre, Marie, Paul, MaMaison, MonAuto, MonCampingCar}\}$$

correspond à :

$$\begin{aligned} \mathcal{P}(\text{Auto}) &= \{\text{MonAuto, MonCampingCar}\} \\ \mathcal{P}(\text{Maison}) &= \{\text{MaMaison, MonCampingCar}\} \\ \mathcal{P}(\text{Humain}) &= \{\text{Pierre, Marie}\} \\ \mathcal{P}(\text{domicile}) &= \{\langle \text{Pierre, MonCampingCar} \rangle, \\ &\quad \langle \text{Marie, MonCampingCar} \rangle, \\ &\quad \langle \text{Marie, MaMaison} \rangle, \} \end{aligned}$$

on peut déduire que :

$$\begin{aligned} \mathcal{P}(\text{Camping-car}) &= \mathcal{E}(\text{Maison}) \cap \mathcal{E}(\text{Auto}) = \{\text{MonCampingCar}\} \\ \mathcal{P}(\text{Campeur}) &= \mathcal{E}(\text{Humain}) \\ &\cap \dots \\ &= \{\text{Pierre}\} \end{aligned}$$

L'intérêt de cette sémantique constructive est de montrer que lorsque l'on dispose de celle-ci par un ensemble d'assertion d' \mathcal{AF} , elle détermine une interprétation minimale du monde. Il ne s'agit pas là d'une interprétation unique car les états du monde font l'hypothèse du monde ouvert.

2.4 Calcul de subsomption

La sémantique de la subsomption est complètement définie par les définitions 2.8 et 2.23. Nous allons maintenant examiner le calcul effectif de la subsomption. Pour cela il nous faut passer par :

1. une forme normale pour les terminologies,
2. une expansion des expressions à comparer en fonction de la terminologie,
3. une normalisation de concepts,
4. puis une comparaison.

On définira donc quatre fonctions correspondant à ces actions : $TNORM(T)$, $EXP(t, T)$, $NORM(t)$ et $COMPARE(t, t')$.

La méthode de déduction pour la subsomption présentée ici (il en existe d'autres), est fondée sur la comparaison de termes normalisés. Ainsi, le test de subsomption de deux termes s'exprimera par :

$$SUBSUMPTION-TEST(t, t') = COMPARE(NORM(t), NORM(t'))$$

et le test de subsomption modulo une terminologie s'exprimera par :

$$TSUB(t, t', T) = SUBSUMPTION-TEST(EXP(t, TNORM(T)), EXP(t', TNORM(T)))$$

Le test complet correspondra donc à

$$TSUB(t, t', T) = COMPARE(NORM(EXP(t, TNORM(T))), NORM(EXP(t', TNORM(T))))$$

On va donc décrire les fonctions nécessaires :

EXP retournant un terme $EXP(t, T)$ équivalent à t modulo la terminologie T . On montrera que cette fonction, définie inductivement sur la structure du terme (§2.4.1, termine et est correcte sur toute terminologie sans cycle.

NORM retournant un terme $NORM(t)$ équivalent à t mais plus propice à la comparaison de deux termes. Cette fonction, définie par 8 règles de réécriture (§2.4.2), termine et conflue.

COMPARE décide si un terme est subsumé par un autre. Cette fonction est définie par 10 règles de comparaison (§2.4.3).

Exemple 2.8 (Test de subsomption). Si l'on considère le concept suivant :

```
(and (and Small-Team (all member Woman))
      (atmost 1 (androle leader member)))
```

dont on veut savoir s'il constitue une `Modern-Team` il est nécessaire d'enchaîner les opérations suivantes :

```
COMPARE (NORM (EXP ((and (and Small-Team (all member Woman))
                        (atmost 1 (androle leader member))),  $\bar{T}$ )),
        NORM (EXP (Modern-Team,  $\bar{T}$ )))
```

On pourra ensuite s'intéresser aux propriétés du système ainsi défini (§2.4.4).

2.4.1 Expansion de concept dans une terminologie

Définition 2.25 (fonction d'expansion).

$$\text{EXP}(t, \bar{T}) = \begin{cases} \text{EXP}(u, \bar{T}) & \text{si } t \in \mathcal{N} \text{ et } t \doteq u \in \bar{T} \\ (\text{op } \text{EXP}(t_1) \dots \text{EXP}(t_n)) & \text{si } t = (\text{op } t_1 \dots t_n) \\ t & \text{sinon} \end{cases}$$

Exemple 2.9 (Expansion de concept). Si l'on considère le concept suivant :

```
(and (and Small-Team (all member Woman))
     (atmost 1 (androle leader member)))
```

son expansion peut s'obtenir en appliquant `EXP` :

```
EXP ((and (and Small-Team (all member Woman))
          (atmost 1 (androle leader member))),  $\bar{T}$ )
= (and EXP ((and Small-Team (all member Woman)),  $\bar{T}$ )
     EXP ((atmost 1 (androle leader member)),  $\bar{T}$ ))
= (and (and EXP (Small-Team,  $\bar{T}$ ) EXP ((all member Woman),  $\bar{T}$ ))
     (atmost 1 EXP ((androle leader member),  $\bar{T}$ )))
= (and (and EXP ((and Team (atmost 5 member)),  $\bar{T}$ )
     (all EXP (member,  $\bar{T}$ ) EXP (Woman,  $\bar{T}$ )))
     (atmost 1 (androle EXP (leader,  $\bar{T}$ ) EXP (member,  $\bar{T}$ ))))
= (and (and (and EXP (Team,  $\bar{T}$ ) EXP ((atmost 5 member),  $\bar{T}$ ))
     (all member EXP ((and Human Woman),  $\bar{T}$ )))
     (atmost 1 (androle EXP ((androle member leader),  $\bar{T}$ )
     member)))
= (and (and (and EXP ((and Set (all member Human)
     (atleast 2 member)),  $\bar{T}$ )
     (atmost 5 EXP (member,  $\bar{T}$ )))
     (all member
     (and EXP (Human,  $\bar{T}$ )
     EXP (Woman,  $\bar{T}$ ))))
     (atmost 1 (androle (androle EXP (member,  $\bar{T}$ ) EXP (leader,  $\bar{T}$ ))
     member)))
= (and (and (and (and EXP (Set,  $\bar{T}$ )
     EXP ((all member Human),  $\bar{T}$ )
     EXP ((atleast 2 member),  $\bar{T}$ ))
     (atmost 5 member))
```

$$\begin{aligned}
& (\text{all } \overline{\text{member}} \ (\text{and } \overline{\text{Human}} \ \overline{\text{Woman}})) \\
& (\text{atmost } 1 \ (\text{androle } (\text{androle } \overline{\text{member}} \ \overline{\text{leader}}) \\
& \quad \overline{\text{member}})) \\
= & (\text{and } (\text{and } (\text{and } (\text{and } \overline{\text{Set}} \\
& \quad (\text{all } \text{EXP}(\text{member}, \overline{T}) \ \text{EXP}(\text{Human}, \overline{T})) \\
& \quad (\text{atleast } 2 \ \text{EXP}(\text{member}, \overline{T}))) \\
& \quad (\text{atmost } 5 \ \overline{\text{member}})) \\
& (\text{all } \overline{\text{member}} \ (\text{and } \overline{\text{Human}} \ \overline{\text{Woman}})) \\
& (\text{atmost } 1 \ (\text{androle } (\text{androle } \overline{\text{member}} \ \overline{\text{leader}}) \\
& \quad \overline{\text{member}})) \\
= & (\text{and } (\text{and } (\text{and } (\text{and } \overline{\text{Set}} \ (\text{all } \overline{\text{member}} \ \overline{\text{Human}}) \\
& \quad (\text{atleast } 2 \ \overline{\text{member}})) \\
& \quad (\text{atmost } 5 \ \overline{\text{member}})) \\
& (\text{all } \overline{\text{member}} \ (\text{and } \overline{\text{Human}} \ \overline{\text{Woman}})) \\
& (\text{atmost } 1 \ (\text{androle } (\text{androle } \overline{\text{member}} \ \overline{\text{leader}}) \ \overline{\text{member}}))
\end{aligned}$$

Proposition 2.10 (terminaison de EXP). *EXP termine pour toute terminologie sans cycle.*

Esquisse de démonstration. À partir de la définition, en itérant sur la structure et en balayant les opérateurs (même type de preuve que pour le théorème 2.9). \square

Proposition 2.11. *Soit \overline{T} une \mathcal{ALNR} -terminologie sans cycle et t un terme quelconque, alors pour toute \mathcal{ALNR} -structure $\langle \mathcal{D}, \mathcal{E} \rangle$ de \overline{T} :*

$$\mathcal{E}(t) = \mathcal{E}(\text{EXP}(t, \overline{T}))$$

Démonstration. Par induction, supposons que cela est vrai des sous-termes d'un terme et des termes de niveau inférieur et montrons qu'alors c'est vrai pour le terme. Le cas de base concerne le niveau 0 pour lequel il ne peut exister de termes de niveau inférieur. Tout terme atomique est alors transformé en lui-même (et par conséquent leurs extensions coïncident) et tout opérateur appliqué à des sous-termes est transformé (par EXP) en lui-même appliqué aux résultat de la transformation des sous-termes. Il n'est pas difficile de montrer par induction que ce terme est transformé en lui-même et doit donc avoir la même interprétation.

Pour le pas d'induction, la fonction d'expansion est définie par trois cas :

$\text{EXP}(a, \overline{T}) = \text{EXP}(t, \overline{T})$ si $a \doteq t \in \overline{T}$ Dans ce cas, comme $\langle \overline{\mathcal{D}}, \overline{\mathcal{E}} \rangle$ est une \mathcal{ALNR} -structure de \overline{T} , $\mathcal{E}(a) = \mathcal{E}(t)$ (définition 2.13) et donc, si $\mathcal{E}(\text{EXP}(t, \overline{T})) = \mathcal{E}(t)$ (parce qu'il fait partie du niveau inférieur) le résultat est bien vérifié. La terminologie étant sans cycle, cette expansion est bien fondée.

$\text{EXP}((\text{op } t_1 \dots t_n), \overline{T}) = (\text{op } \text{EXP}(t_1, \overline{T}) \dots \text{EXP}(t_n, \overline{T}))$ l'extension de $\text{EXP}(t, \overline{T})$ est alors complètement définie en fonction de celle de ses sous-termes $\text{EXP}(t_1, \overline{T}) \dots \text{EXP}(t_n, \overline{T})$.

Celle-ci étant égale à celle de $t_1 \dots t_n$ par hypothèse d'induction, on a bien le résultat.

$\text{EXP}(t, \overline{T}) = t$ et donc $\mathcal{E}(t) = \mathcal{E}(\text{EXP}(t, \overline{T}))$. \square

EXP a une structure très répandue en représentation de connaissance en particulier sur des structures non circulaires : c'est l'héritage dans les représentations par objets ; c'est aussi le dépliage ("unfolding") dans les Ψ -termes lorsque l'on cherche à unifier deux termes. C'est une stratégie d'évaluation immédiate que l'on peut opposer à une stratégie d'évaluation paresseuse. Elle a ses avantages et ses inconvénients comme le montre l'exemple 2.10.

Exemple 2.10 (évaluation immédiate ou paresseuse). Si l'on dispose de la terminologie suivante :

A \doteq (and B C)
 B \doteq (and D E)
 C \doteq (and F G)
 D \doteq (and H I)
 ...

Lorsqu'il s'agit de comparer les termes (and A B) et A la stratégie immédiate exige de les développer (ce qui est déjà long) puis de comparer le résultat de l'expansion ce qui peut devenir dramatique. La stratégie paresseuse permet de réduire (and A B) \preceq_A à B \preceq_{Nothing} ce qui est toujours vrai.

La subsomption décrite ci-dessous est différente de la \mathcal{FLN} -subsomption car elle ne dépend plus d'une terminologie.

Définition 2.26 (subsomption). Soient $t, t' \in \mathcal{ALNR}_T$, si $t \preceq_{\emptyset} t'$ on parlera de *subsomption* et on écrira $t \preceq t'$.

Théorème 2.12. Soit T une \mathcal{FLN} -terminologie sans cycle, pour tous termes $t, t' \in \mathcal{FLN}_T$:

$$t \preceq_T t' \text{ ssi } \text{EXP}(t, \bar{T}) \preceq \text{EXP}(t', \bar{T})$$

Esquisse de démonstration. On passe aux extensions (définition 2.8) puis on utilise la proposition 2.11 et le théorème 2.6. Il faut aussi noter que si T est une \mathcal{FLN} -terminologie sans cycle, alors \bar{T} est une \mathcal{ALNR} -terminologie sans cycle (la démonstration se fait en considérant les différents constructeurs).

On montre que pour toute terminologie T , les structures de $T \cup \{a \doteq t\}$ sont celles de T à $\mathcal{E}(a) = \mathcal{E}(t)$ prêt et $a \notin \text{EXP}(t)$. \square

Ce dernier théorème permet d'utiliser COMPARE pour effectuer la comparaison de deux termes indépendamment de la terminologie considérée.

2.4.2 Normalisation de termes

Les termes sont ensuite normalisés, c'est-à-dire que l'on en supprime les and (resp. androle) imbriqués dans d'autres and (resp. androle). De plus, toutes les contraintes d'un même type (all, atmost et atleast) sur le même rôle sont collectées.

Définition 2.27 (Algorithme de normalisation de termes). Soit $t \in \mathcal{ALNR}_T$, la fonction NORM est donnée par l'ensemble de règles de réécriture* suivantes :

réduction-expansion

$$\frac{(\text{and} \dots (\text{and } c \ c') \dots)}{(\text{and} \dots c \ c' \dots)} \quad [\text{and-reduction}]$$

$$\frac{(\text{androle} \dots (\text{androle } r \ r') \dots)}{(\text{androle} \dots r \ r' \dots)} \quad [\text{androle-reduction}]$$

$$\frac{(\text{all } r \ c)}{(\text{all } r \ (\text{and } c))} \quad \begin{array}{l} c \neq (\text{and} \dots) \\ c \neq \text{Nothing} \end{array} \quad [\text{and-all-expansion}]$$

factorisation

$$\frac{(\text{and} \dots (\text{all } r (\text{and } c c')) \dots (\text{all } r (\text{and } c'' c''')) \dots)}{(\text{and} \dots (\text{all } r (\text{and } c c' c'' c''')) \dots)}$$
 [all-factorisation]
$$\frac{(\text{and} \dots (\text{atmost } n r) \dots (\text{atmost } n' r) \dots)}{(\text{and} \dots (\text{atmost } \min(n, n') r) \dots)}$$
 [atmost-factorisation]
$$\frac{(\text{and} \dots (\text{atleast } n r) \dots (\text{atleast } n' r) \dots)}{(\text{and} \dots (\text{atleast } \max(n, n') r) \dots)}$$
 [atmost-factorisation]

introduction

$$\frac{(\text{and} \dots a \dots (\text{anot } a) \dots)}{\text{Nothing}} a \in \mathcal{N}_C$$
 [and-bottom-introduction]
$$\frac{(\text{and} \dots (\text{atmost } n r) \dots (\text{atleast } n' r') \dots)}{\text{Nothing}} n < n', r' \preceq r$$
 [card-bottom-introduction]
$$\frac{(\text{all } r \text{ Nothing})}{(\text{atmost } 0 r)}$$
 [bottom-introduction]
$$\frac{(\text{and} \dots (\text{all } r \text{ Nothing}) \dots (\text{atleast } n r') \dots)}{\text{Nothing}} n > 0, r' \preceq r$$
 [role-bottom-propagation]
$$\frac{(\text{and} \dots \text{Nothing} \dots)}{\text{Nothing}}$$
 [bottom-propagation]

On notera au passage le recours au test de subsomption au sein même de l'algorithme. Cependant, ce test concerne les rôles et comme il ne peut qu'y avoir un constructeur `androle`, dans la portée duquel se trouvent des rôles atomiques, la récursion s'arrête là. Ceci peut être problématique dans des langages disposant de constructeurs de rôles plus élaborés.

Exemple 2.11 (Normalisation de concept). Si l'on considère le concept suivant (on a supprimé les mentions aux parties primitives des concepts) :

```
(and (and (and (and Set (all member Human)) (atleast 2 member))
      (atmost 5 member))
     (all member (and Human Woman)))
(atmost 1 (androle (androle member leader) member))
```

sa normalisation est obtenue en considérant les règles suivantes :

```
(and (and (and (and Set (all member Human) (atleast 2 member))
      (atmost 5 member))
     (all member (and Human Woman)))
(atmost 1 (androle (androle member leader) member))
```

[and – reduction] sur l'intérieur

```
(and (and (and Set (all member Human) (atleast 2 member)
      (atmost 5 member))
     (all member (and Human Woman)))
(atmost 1 (androle (androle member leader) member)))
```

[androle – reduction] sur l'intérieur

```

    (and (and (and Set (all member Human) (atleast 2 member)
                  (atmost 5 member))
          (all member (and Human Woman)))
      (atmost 1 (androle member leader member)))
[and – reduction] sur l'intérieur
    (and (and Set (all member Human) (atleast 2 member)
                  (atmost 5 member)
          (all member (and Human Woman)))
      (atmost 1 (androle member leader member)))
[and – reduction] sur l'intérieur
    (and Set (all member Human) (atleast 2 member)
              (atmost 5 member)
              (all member (and Human Woman))
              (atmost 1 (androle member leader member)))
[all – factorisation]
    (and Set
      (all member (and Human (and Human Woman)))
      (atleast 2 member)
      (atmost 5 member)
      (atmost 1 (androle member leader member)))
[and – reduction]
    (and Set
      (all member (and Human Human Woman))
      (atleast 2 member)
      (atmost 5 member)
      (atmost 1 (androle member leader member)))

```

On devrait montrer la terminaison et la confluence de cet ensemble de règles de réécriture.

Proposition 2.13 (Terminaison de la normalisation). *Soit t un \mathcal{ALNR} -terme, alors $NORM(t)$ termine.*

Démonstration. Chacune des règles, sauf deux, réduit strictement la complexité de l'expression à normaliser en réduisant le nombre de connecteurs `and`, `androle`, `all`, `atmost` et `atleast`. Donc à partir d'un terme fini, elles ne pourront s'appliquer qu'un nombre fini de fois. Les deux seules exceptions sont `and-all-expansion` et `bottom-introduction`. La première ne peut s'appliquer qu'une fois par `all` (elle ne peut s'appliquer sur le résultat de l'ajout d'un `and`). Comme aucune autre règle n'introduit de `all` sur lequel elle puisse s'appliquer, elle ne pourra s'appliquer qu'un nombre fini de fois à partir d'un terme fini. La seconde introduit un connecteur `atmost` et ne peut se déclencher qu'autant de fois qu'il y a de `all` dans le terme initial car aucune règle ne produit de `all` qui n'étaient pas présents (`and-all-expansion` préserve un `all` présent et `all-factorisation` en réduit le nombre). Le nouveau connecteur `atmost` ne peut, au pire, qu'être utilisé au déclenchement de `card-bottom-introduction` le supprimant ou de `at-most factorisation`, le réduisant. Donc, à partir d'un terme fini, $NORM$ termine. \square

Proposition 2.14 (Confluence de la normalisation). *Soit t un \mathcal{ALNR} -terme, alors $NORM(t)$ retourne un unique résultat quelque soit l'ordre d'application des règles.*

Démonstration. Pour cela il faut montrer qu'à chaque fois que plusieurs règles sont applicables, l'application de l'une préserve les conditions d'applications des autres ou donne le même résultat que les autres (le cas de [bottom – propagation]). Pour cela on étudie le comportement de chaque règle en supposant que c'est elle qui est choisie :

[réduction/expansion] Ces règles préservent les conditions d'application de leurs concurrentes, y compris elles-mêmes, car elles ne détruisent pas de conditions d'autres règles ;

[all – factorisation] Cette règle préserve les conditions d'application de ses concurrentes, y compris elle-même, car elle ne détruit pas de conditions d'autres règles ;

[atmost – factorisation] Cette règle entre en conflit avec [card – bottom – introduction]. Mais elle en préserve les conditions d'activation car si, dans [card – bottom – introduction] $n < n'$ pour un n et un n' , l'application de cette règle ne consisterait qu'à retenir un n plus petit ce qui préserve la condition d'activation de [card – bottom – introduction] ;

[atleast – factorisation] Cette règle entre en conflit avec [card – bottom – introduction] et [role – bottom – propagation] ; le cas de la première se règle par le même argument que ci-dessus celui de la seconde par le fait que sa condition est que $n > 0$ or l'application de cette règle pourrait substituer à n une valeur supérieure par conséquent toujours positive ;

[bottom – introduction] Cette règle est en conflit avec [role – bottom – propagation] dont elle détruit les conditions d'application. Mais si cette dernière règle s'applique, alors $n > 0$ ce qui permet à la règle [card – bottom – introduction] de s'appliquer et de donner le même résultat que celui donné par l'application de [role – bottom – propagation] ;

[card – bottom – introduction] , [role – bottom – propagation], [and – bottom – propagation], [bottom – propagation] Ces règles détruisent les conditions d'applications de presque toutes les règles. Mais comme toutes ces règles retournent le même résultat cela ne pose pas de problème. D'autre part, comme on a montré ci-dessus que toutes les autres règles préservent les éventuelles conditions d'applications de ces règles, le résultat final sera toujours `Nothing` et la confluence est donc prouvée. \square

Remarque. La forme normalisée est équivalente à la manière dont une classe serait décrite dans un langage objet :

```
[Class t superclasses: c1 ... cn
  R1: C1 ^ [m1 M1]
  ...
  Rp: Cp ^ [mp Mp]
```

Elle en diffère principalement car les R_i peuvent être des conjonctions de rôles.

Proposition 2.15 (Correction de la normalisation). *Soit t un $\mathcal{ALN}\mathcal{R}$ -terme, alors pour toute $\mathcal{ALN}\mathcal{R}$ -structure $\langle \mathcal{D}, \mathcal{E} \rangle$, $\mathcal{E}(t) = \mathcal{E}(\text{NORM}(t))$.*

Démonstration. Pour chaque règle, il suffit de montrer que l'extension de la partie haute est égale à celle de la partie basse. Ainsi,

$$\begin{aligned} \text{[and-reduction]} \quad & \mathcal{E}(\text{(and } X \text{ (and } c \text{ } c') \text{ } Y)) \\ &= \mathcal{E}(X) \cap \mathcal{E}(\text{(and } c \text{ } c')) \cap \mathcal{E}(Y) \\ &= \mathcal{E}(X) \cap \mathcal{E}(c) \cap \mathcal{E}(c') \cap \mathcal{E}(Y) \\ &= \mathcal{E}(\text{(and } X \text{ } c \text{ } c' \text{ } Y)) \end{aligned}$$

$$\begin{aligned} \text{[androle-reduction]} \quad & \mathcal{E}(\text{(androle } X \text{ (androle } r \text{ } r') \text{ } Y)) \\ &= \mathcal{E}(X) \cap \mathcal{E}(\text{(androle } r \text{ } r')) \cap \mathcal{E}(Y) \\ &= \mathcal{E}(X) \cap \mathcal{E}(r) \cap \mathcal{E}(r') \cap \mathcal{E}(Y) \\ &= \mathcal{E}(\text{(androle } X \text{ } r \text{ } r' \text{ } Y)) \end{aligned}$$

$$\begin{aligned} \text{[and-all-expansion]} \quad & \mathcal{E}(\text{(all } r \text{ } c)) \\ &= \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(c)\} \\ &= \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \cap \mathcal{E}(c)\} \\ &= \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(\text{(and } c))\} \\ &= \mathcal{E}(\text{(all } r \text{ (and } c)) \end{aligned}$$

$$\begin{aligned} \text{[all-factorisation]} \quad & \mathcal{E}(\text{(and } X \text{ (all } r \text{ (and } c \text{ } c') \text{ } Y \text{ (all } r \text{ (and } c'' \text{ } c''') \text{ } Z))}) \\ &= \mathcal{E}(X) \cap \mathcal{E}(\text{(all } r \text{ (and } c \text{ } c')) \cap \mathcal{E}(Y) \cap \\ & \quad \mathcal{E}(\text{(all } r \text{ (and } c'' \text{ } c''')) \cap \mathcal{E}(Z)) \\ &= \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(\text{(and } c \text{ } c'))\} \cap \mathcal{E}(Y) \\ & \quad \cap \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(\text{(and } c'' \text{ } c'''))\} \cap \mathcal{E}(Z) \\ &= \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \\ & \quad \Rightarrow y \in \mathcal{E}(\text{(and } c \text{ } c')) \wedge y \in \mathcal{E}(\text{(and } c'' \text{ } c'''))\} \cap \mathcal{E}(Y) \cap \mathcal{E}(Z) \\ &= \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \\ & \quad \Rightarrow y \in \mathcal{E}(\text{(and } c \text{ } c')) \cap \mathcal{E}(\text{(and } c'' \text{ } c'''))\} \cap \mathcal{E}(Y) \cap \mathcal{E}(Z) \\ &= \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \\ & \quad \Rightarrow y \in (\mathcal{E}(c) \cap \mathcal{E}(c')) \cap (\mathcal{E}(c'') \cap \mathcal{E}(c'''))\} \cap \mathcal{E}(Y) \cap \mathcal{E}(Z) \\ &= \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(\text{(and } c \text{ } c' \text{ } c'' \text{ } c'''))\} \\ & \quad \cap \mathcal{E}(Y) \cap \mathcal{E}(Z) \\ &= \mathcal{E}(X) \cap \mathcal{E}(\text{(all } r \text{ (and } c \text{ } c' \text{ } c'' \text{ } c''')) \cap \mathcal{E}(Y) \cap \mathcal{E}(Z) \\ &= \mathcal{E}(\text{(and } X \text{ (all } r \text{ (and } c \text{ } c' \text{ } c'' \text{ } c''') \text{ } Y \text{ } Z)) \end{aligned}$$

$$\begin{aligned}
[\text{card-bottom-introduction}] \quad & \mathcal{E}((\text{and } X (\text{atmost } n \ r) \ Y (\text{atleast } n' \ r') \ Z)) \\
& = \mathcal{E}(X) \cap \mathcal{E}((\text{atmost } n \ r)) \cap \mathcal{E}(Y) \cap \mathcal{E}((\text{atleast } n' \ r')) \\
& \quad \cap \mathcal{E}(Z) \\
& = \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq n\} \cap \mathcal{E}(Y) \cap \\
& \quad \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r')\}\| \geq n'\} \cap \mathcal{E}(Z) \\
& = \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid n' \leq \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq n\} \cap \mathcal{E}(Y) \\
& \quad \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r')\}\| \geq n'\} \cap \mathcal{E}(Z) \\
& \quad (\text{parce que } r' \preceq r) \\
& = \mathcal{E}(X) \cap \emptyset \cap \mathcal{E}(Y) \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r')\}\| \geq n'\} \\
& \quad \cap \mathcal{E}(Z) (\text{parce que } n < n') \\
& = \emptyset \\
& = \mathcal{E}(\text{Nothing})
\end{aligned}$$

$$\begin{aligned}
[\text{bottom-introduction}] \quad & \mathcal{E}((\text{all } r \ \text{Nothing})) \\
& = \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(\text{Nothing})\} \\
& = \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \emptyset\} \\
& = \{x \in \mathcal{D} \mid \nexists y, \langle x, y \rangle \in \mathcal{E}(r)\} \\
& = \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq 0\} \\
& = \mathcal{E}((\text{atmost } 0 \ r))
\end{aligned}$$

$$\begin{aligned}
[\text{role-bottom-propagation}] \quad & \mathcal{E}((\text{and } X (\text{all } r \ \text{Nothing}) \ Y (\text{atleast } n' \ r') \ Z)) \\
& = \mathcal{E}(X) \cap \mathcal{E}((\text{all } r \ \text{Nothing})) \cap \mathcal{E}(Y) \\
& \quad \cap \mathcal{E}((\text{atleast } n' \ r')) \cap \mathcal{E}(Z) \\
& = \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(\text{Nothing})\} \\
& \quad \cap \mathcal{E}(Y) \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r')\}\| \geq n'\} \cap \mathcal{E}(Z) \\
& = \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq 0\} \cap \mathcal{E}(Y) \\
& \quad \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r')\}\| \geq n'\} \cap \mathcal{E}(Z) \\
& = \mathcal{E}(X) \cap \{x \in \mathcal{D} \mid n' \leq \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq 0\} \cap \mathcal{E}(Y) \\
& \quad \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r')\}\| \geq n'\} \cap \mathcal{E}(Z) \\
& \quad (\text{parce que } r' \preceq r) \\
& = \mathcal{E}(X) \cap \emptyset \cap \mathcal{E}(Y) \cap \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r')\}\| \geq n'\} \\
& \quad \cap \mathcal{E}(Z) (\text{parce que } n' < 0) \\
& = \emptyset \\
& = \mathcal{E}(\text{Nothing})
\end{aligned}$$

□

2.4.3 Comparaison des formes normales

Définition 2.28 (Algorithme de comparaison de termes). Soit $t, t' \in \mathcal{ALNR}_T$, la fonction COMPARE est donnée par l'ensemble de clauses suivantes. Dans chaque cas seule une des

clauses est applicable, il n'y a pas de choix dans l'application des clauses (sauf pour le `and` mais l'ordre n'importe alors pas).

$\frac{\text{COMPARE}(\text{Nothing}, c)}{\text{vrai}}$		[Nothing \preceq]
$\frac{\text{COMPARE}(c, c)}{\text{vrai}}$	$c \in \overline{\mathcal{N}}$	[PRIM \preceq]
$\frac{\text{COMPARE}(d, (\text{and } c_1 \dots c_n))}{\text{COMPARE}(d, c_1) \wedge \dots \text{COMPARE}(d, c_n)}$		[\preceq and][\preceq androle]
$\frac{\text{COMPARE}((\text{and } d_1 \dots d_n), c)}{\text{COMPARE}(d_1, c) \vee \dots \text{COMPARE}(d_n, c)}$	$c \neq (\text{and } \dots)$	[and \preceq][androle \preceq]
$\frac{\text{COMPARE}((\text{anot } d), (\text{anot } c))}{\text{COMPARE}(c, d)}$		[anot \preceq]
$\frac{\text{COMPARE}((\text{all } r_d d), (\text{all } r_c c))}{\text{COMPARE}(r_c, r_d) \wedge \text{COMPARE}(d, c)}$		[all \preceq]
$\frac{\text{COMPARE}((\text{atmost } 0 r_d), (\text{all } r_c c))}{\text{COMPARE}(r_c, r_d)}$		[atmost \preceq all]
$\frac{\text{COMPARE}((\text{atleast } n_d r_d), (\text{atleast } n_c r_c))}{\text{COMPARE}(r_d, r_c)}$	$n_c \leq n_d$	[atleast \preceq]
$\frac{\text{COMPARE}((\text{atmost } n_d r_d), (\text{atmost } n_c r_c))}{\text{COMPARE}(r_c, r_d)}$	$n_d \leq n_c$	[atmost \preceq]
faux		[ELSE]

Exemple 2.12 (Comparaison de termes). Si l'on considère le concept suivant :

```
(and Set
  (all member (and Human Human Woman))
  (atleast 2 member)
  (atmost 5 member)
  (atmost 1 (androle member leader member)))
```

à comparer avec le concept :

```
(and Set
  (all member Human)
  (atleast 2 member)
  (atmost 3 member)
  (atleast 1 (androle leader member))
  (all (androle leader member) Woman))
```

dont on peut vérifier qu'il s'agit de la normalisation de `Modern-Team`. Il est nécessaire d'opérer la dérivation suivante :

```
COMPARE((and Set
  (all member (and Human Human Woman))
  (atleast 2 member)
  (atmost 5 member)
```

```

        (atmost 1 (androle member leader member))),
    (and Set
      (all member Human)
      (atleast 2 member)
      (atmost 3 member)
      (atleast 1 (androle leader member))
      (all (androle leader member) Woman)))
= COMPARE((and Set
  (all member (and Human Human Woman))
  (atleast 2 member)
  (atmost 5 member)))
  (atmost 1 (androle member leader member))),
  Set)
= COMPARE(Set,Set) = vrai.
^ COMPARE((and Set
  (all member (and Human Human Woman))
  (atleast 2 member)
  (atmost 5 member)))
  (atmost 1 (androle member leader member))),
  (all member Human))
= COMPARE((all member (and Human Human Woman)), (all member Human))
= COMPARE(member,member)
= vrai.
^ COMPARE((and Human Human Woman), (and Human))
= COMPARE(Human,Human)
= vrai.
^ COMPARE((and Set
  (all member (and Human Human Woman))
  (atleast 2 member)
  (atmost 5 member)))
  (atmost 1 (androle member leader member))),
  (atleast 2 member))
= COMPARE((atleast 2 member), (atleast 2 member))
= COMPARE(member,member)
= vrai.
^ COMPARE((and Set
  (all member (and Human Human Woman))
  (atleast 2 member)
  (atmost 5 member)))
  (atmost 1 (androle member leader member))),
  (atmost 3 member))
= COMPARE((atmost 5 member), (atmost 3 member))
= faux.
^ COMPARE((and Set
  (all member (and Human Human Woman))
  (atleast 2 member)
  (atmost 5 member)))
  (atmost 1 (androle member leader member))),

```

```

      (atleast 1 (androle leader member))
= faux.
 $\wedge$  COMPARE((and Set
      (all member (and Human Human Woman))
      (atleast 2 member)
      (atmost 5 member)))
      (atmost 1 (androle member leader member)),
      (all (and leader member) Woman))
= COMPARE((all member (and Human Human Woman)),
      (all (and leader member) Woman))
= COMPARE((and leader member), member)
= COMPARE(member, member)
= vrai.
 $\wedge$  COMPARE((and Human Human Woman), Woman)
= COMPARE(Woman, Woman)
= vrai.

```

On se rend compte que le résultat est «faux».

2.4.4 Correction, complexité, complétude

Proposition 2.16 (Correction de COMPARE). *Si $\text{COMPARE}(t, t')$ alors $t \preceq t'$*

Démonstration. Si $\text{COMPARE}(t, t')$ retourne *vrai* alors tous les appels sur les sous-expressions retournent *vrai* (et au moins l'un des appels dans le cas de la disjonction). On montre cela par induction sur l'ensemble des règles pour toute fonction d'extension \mathcal{E} :

[Nothing \preceq] $\mathcal{E}(\text{Nothing}) = \emptyset \subseteq \mathcal{E}(c)$

[PRIM \preceq] $\mathcal{E}(c) \subseteq \mathcal{E}(c)$

[\preceq and] si $\forall c_i \mathcal{E}(d) \subseteq \mathcal{E}(c_i)$ alors $\mathcal{E}(d) \subseteq \bigcap_i \mathcal{E}(c_i) = \mathcal{E}(c)$

[and \preceq] si $\exists d_i; \mathcal{E}(d_i) \subseteq \mathcal{E}(c)$ alors $\mathcal{E}(d) = \bigcap_i \mathcal{E}(d_i) \subseteq \mathcal{E}(c)$

[anot \preceq] si $\mathcal{E}(c') \subseteq \mathcal{E}(d')$ alors $\mathcal{E}(d) = \mathcal{D} \setminus \mathcal{E}(d') \subseteq \mathcal{D} \setminus \mathcal{E}(c') = \mathcal{E}(c)$

[all \preceq] si $\mathcal{E}(r_c) \subseteq \mathcal{E}(r_d)$ et $\mathcal{E}(d') \subseteq \mathcal{E}(c')$
alors $\forall x, y, (\langle x, y \rangle \in \mathcal{E}(r_c) \Rightarrow y \in \mathcal{E}(c))$
 $\Rightarrow (\langle x, y \rangle \in \mathcal{E}(r_d) \Rightarrow y \in \mathcal{E}(d))$
alors $\mathcal{E}(\text{all } r_d \ d') \subseteq \mathcal{E}(\text{all } r_d \ c') \subseteq \mathcal{E}(\text{all } r_c \ c')$

[atmost \preceq all] si $\mathcal{E}(r_c) \subseteq \mathcal{E}(r_d)$ sachant que $\mathcal{E}(\text{atmost } r \ 0) \subseteq \mathcal{E}(\text{all } r \ \text{Nothing})$
et $\mathcal{E}(\text{Nothing}) \subseteq \mathcal{E}(c')$
alors on peut appliquer la démonstration précédente.

[atleast \preceq] si $\mathcal{E}(r_c) \subseteq \mathcal{E}(r_d)$ et $n_c \leq n_d$
alors $\mathcal{E}(\text{atleast } n_d \ r_d) \subseteq \mathcal{E}(\text{atleast } n_c \ r_d) \subseteq \mathcal{E}(\text{atleast } n_c \ r_c)$

[atmost \preceq] similaire □

Notation (Taille d'un terme). Soit t un \mathcal{FLN} -terme, on dénotera par $|t|$ le nombre d'occurrences d'un terme atomique dans t .

Exemple 2.13 (Taille d'un terme). $|(\text{atmost } 1 (\text{androle member leader member}))| = 3$,
 $|(\text{and } (\text{all member } (\text{and Human Human Woman})) (\text{all } (\text{and leader member } \text{Woman})))| = 7$.

Proposition 2.17 (Complexité du test de subsomption). *SUBSUMPTION-TEST* est en $O(|C|^2 + |D|^2)^{*1}$.

Démonstration.

sur les rôles La forme des description de rôle est soit un rôle atomique, soit un androle contenant des rôles atomiques.

NORM ne nécessite qu'un passage sur chaque description de rôle éliminant les androle ($O(|C| + |D|)$);

COMPARE seules trois règles peuvent s'appliquer :

[**PRIM** \preceq] en temps constant ;

[androle \preceq] relance $|C|$ COMPARE avec des c_i atomiques ;

[\preceq androle] au pire $|D|$ COMPARE avec des d_i atomiques.

Le travail est donc fait en $O(|C| \times |D|)$.

sur les concepts

NORM – and-reduction, and-all-expansion, bottom-introduction et bottom-propagation sont en $O(|C| + |D|)$;

– all-factorisation, atmost-factorisation, atleast-factorisation et and-bottom-introduction sont en $O(|C|^2 + |D|^2)$;

– card-bottom-introduction et role-bottom-propagation fait $O(|C|^2 + |D|^2)$ appels à *TSUB* sur des rôles ce qui reste en $O(|C|^2 + |D|^2)$;

COMPARE 1. S'il n'y a pas de all, alors

– [\preceq and] relance $|C|$ COMPARE avec $c_i \neq (\text{and } \dots)$;

– [and \preceq] relance $|D|$ COMPARE avec $d_i \neq (\text{and } \dots)$;

– [anot \preceq], [Nothing \preceq] et [**PRIM** \preceq] sont en temps constant ;

– [atleast \preceq] et [atmost \preceq] font appel à la comparaison de rôles ($O(|C| \times |D|)$).

On arrive donc à $O(|C| \times |D|)$ soit à cause de ce dernier soit à cause de and.

2. Si l'on suppose que COMPARE fait du $O(|C| \times |D|)$ pour une imbrication de k all, ceci reste vrai pour une imbrication de $k + 1$ all. \square

Proposition 2.18 (Incomplétude). *COMPARE* est incomplète.

Contre-exemple.

$$\begin{aligned} & (\text{and } (\text{all } (\text{androle } r \text{ p}) \text{ A}) \\ & (\text{all } (\text{androle } r \text{ q}) (\text{anot } \text{A}))) \preceq (\text{atleast } 3 \text{ r}) \\ & (\text{atleast } 2 (\text{androle } r \text{ q})) \\ & (\text{atleast } 2 (\text{androle } r \text{ p}))) \end{aligned}$$

\square

L'incomplétude de COMPARE peut être simplement notée dans la théorie des ensembles : si $\exists d_i \subseteq a$ alors $d_1 \wedge \dots \wedge d_n \subseteq a$, mais l'inverse n'est pas vrai. Or, la règle [and \preceq] teste la comparaison de cette manière. Ceci est illustré par la figure 2.3.

¹[Nebel, 1990] donne $O((|D| + |C|)^2)$ mais la démonstration donnée ici semble correcte.

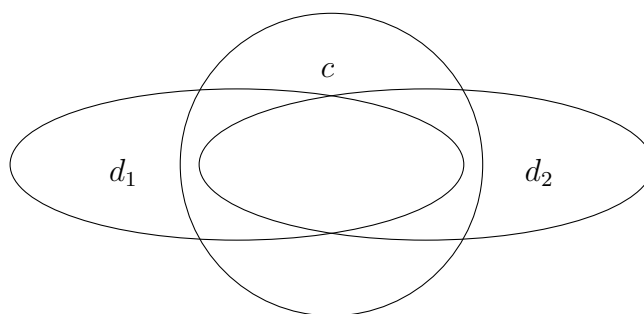


FIGURE 2.3 : Contre-exemple à COMPARE.

Proposition 2.19 (Décidabilité). *La subsomption de termes dans \mathcal{ALNR} est décidable*.*

Démonstration. voir [Nebel, 1990, pp.82–87]. □

Proposition 2.20 (Complexité). *La subsomption de termes dans \mathcal{ALNR} est co-NP-difficile*.*

Ébauche de démonstration. La proposition s’obtient en réduisant le complémentaire d’un problème NP-complet (en l’occurrence le *set-splitting problem*) à un test de subsomption en temps polynomial.

- difficile de voir si le problème est dans NP ;
- complémentaire : le problème où l’on répond à la négation de la question posée dans le problème original. □

Parler de la complexité de l’expansion.

2.4.5 Complexité et expressivité

Bien que décourageante en théorie, la complexité obtenue se révèle plutôt praticable lorsque l’on veut utiliser de tels algorithmes pour des utilisations précises. Cependant, si les performances des algorithmes se révèlent trop désastreuses, deux techniques classiques peuvent être utilisées afin de contourner ce problème :

- réduire l’expressivité du langage utilisé (ainsi, à partir de \mathcal{FLN} , le langage \mathcal{FL}^- qui ne contient pas *atleast* et *atmost* mais *some* peut être envisagé – voir figure 2.4. L’algorithme présenté ci-dessus est toujours correct et polynomial mais il est en plus complet – voir proposition 2.21) ;
- accepter d’utiliser un algorithme incomplet tel que celui présenté plus haut.

Proposition 2.21 (Complétude de SUBSUMPTION-TEST pour \mathcal{FL}^-). *SUBSUMPTION-TEST est un algorithme complet pour le test de subsomption de termes dans \mathcal{FL}^- .*

Démonstration. □

Conséquence 2.22 (Complexité du test de subsomption pour \mathcal{FL}^-). *La subsomption de termes dans \mathcal{FL}^- est polynomiale*.*

Les concepteurs de logiques terminologiques ont toujours cherché à concilier ce fameux compromis entre expressivité et complexité. On peut voir le résultat de cette démarche sur la figure 2.4). Cependant, les problèmes étant difficiles, les résultats obtenus sont, au mieux des systèmes décidables avec vérificateurs incomplets, au pire des systèmes polynomiaux.

Au pire, car ils tombent sous le coup des critiques de Jon Doyle et Ramesh Patil : ces systèmes sont si peu expressifs qu'ils ne permettent pas de résoudre les problèmes et que les développeurs sont contraints de les résoudre par eux-même, à côté.

Par ailleurs, les résultats empiriques obtenus par Bernhard Nebel montrent qu'expérimentalement le comportement des classifieurs implémentés est polynomial sinon linéaire.

Définition 2.29 (Le formalisme universel \mathcal{U}).

$$\begin{aligned} \langle \text{concept} \rangle &::= \langle \text{atomic-concept} \rangle \\ &| \text{'(' and } \langle \text{concept} \rangle + \text{'')} \\ &| \text{'(' or } \langle \text{concept} \rangle + \text{'')} \\ &| \text{'(' not } \langle \text{concept} \rangle \text{'')} \\ &| \text{'(' all } \langle \text{role} \rangle \langle \text{concept} \rangle \text{'')} \\ &| \text{'(' atleast } \langle \text{number} \rangle \langle \text{role} \rangle \text{'')} \\ &| \text{'(' atmost } \langle \text{number} \rangle \langle \text{role} \rangle \text{'')} \\ &| \text{'(' rvm } \langle \text{role} \rangle \langle \text{role} \rangle \text{'')} \\ &| \text{'(' sd } \langle \text{concept} \rangle \langle \text{binding} \rangle + \text{'')} \end{aligned}$$

$$\begin{aligned} \langle \text{binding} \rangle &::= \text{'(' } \subseteq \langle \text{role} \rangle \langle \text{role} \rangle \text{'')} \\ &| \text{'(' } \supseteq \langle \text{role} \rangle \langle \text{role} \rangle \text{'')} \end{aligned}$$

$$\begin{aligned} \langle \text{role} \rangle &::= \langle \text{atomic-role} \rangle \\ &| \text{'(' androle } \langle \text{role} \rangle + \text{'')} \\ &| \text{'(' orrole } \langle \text{role} \rangle + \text{'')} \\ &| \text{'(' notrole } \langle \text{role} \rangle \text{'')} \\ &| \text{'(' comp } \langle \text{role} \rangle + \text{'')} \\ &| \text{self} \\ &| \text{'(' inv } \langle \text{role} \rangle \text{'')} \\ &| \text{'(' range } \langle \text{role} \rangle \langle \text{concept} \rangle \text{'')} \\ &| \text{'(' trans } \langle \text{role} \rangle \text{'')} \end{aligned}$$

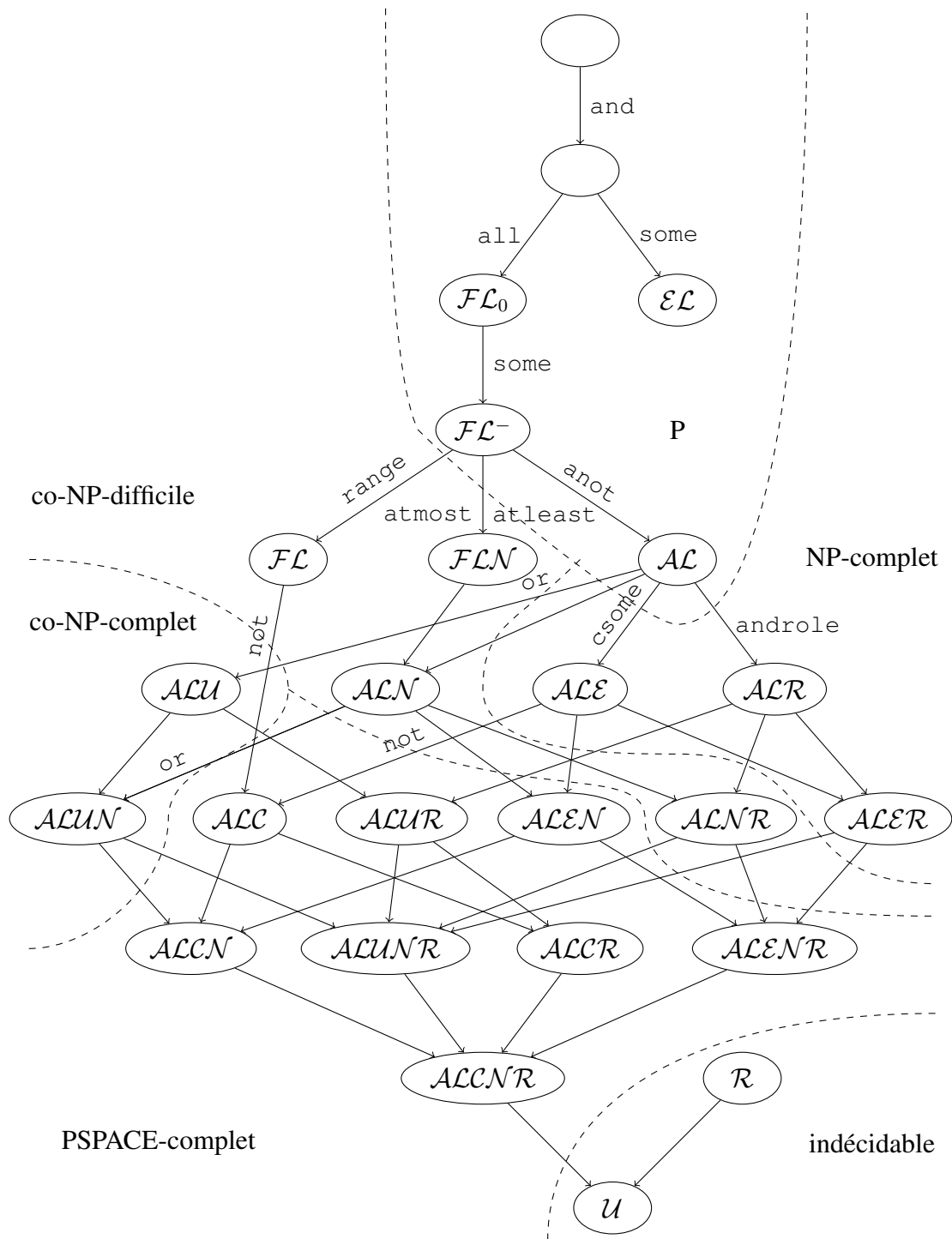


FIGURE 2.4 : Hiérarchie de systèmes terminologiques et leurs classes de complexité pour la subsumption de termes.

Notation (Nouveaux constructeurs).

$$\begin{aligned}
(\text{single } r) &\stackrel{\text{def}}{=} (\text{atmost } 1 \ r) \\
(\text{some } r) &\stackrel{\text{def}}{=} (\text{atleast } 1 \ r) \\
(\text{csofme } r \ c) &\stackrel{\text{def}}{=} (\text{atleast } 1 \ (\text{range } r \ c)) \\
(\text{catleast } n \ r \ c) &\stackrel{\text{def}}{=} (\text{atleast } n \ (\text{range } r \ c)) \\
(\text{catmost } n \ r \ c) &\stackrel{\text{def}}{=} (\text{atmost } n \ (\text{range } r \ c)) \\
(\text{domain } r \ c) &\stackrel{\text{def}}{=} (\text{inv } (\text{range } (\text{inv } r) \ c)) \\
(\text{lrvm } rl_1 \ rl_2) &\stackrel{\text{def}}{=} (\text{rvm } (\text{comp } rl_1) (\text{comp } rl_2)) \\
(\text{sameas } rl_1 \ rl_2) &\stackrel{\text{def}}{=} (\text{and } (\text{rvm } (\text{comp } rl_1) (\text{comp } rl_2)) \\
&\quad (\text{rvm } (\text{comp } rl_2) (\text{comp } rl_1)))
\end{aligned}$$

Définition 2.30 (Sémantique de \mathcal{U}).

$$\begin{aligned}
\mathcal{E}(\text{Anything}) &= \mathcal{D} \\
\mathcal{E}(\text{NoThing}) &= \emptyset \\
\mathcal{E}(\text{anyrelation}) &= \mathcal{D} \times \mathcal{D} \\
\mathcal{E}(\text{(and } c_1 \dots c_n)) &= \bigcap_{i \in 1 \dots n} \mathcal{E}(c_i) \\
\mathcal{E}(\text{(or } c_1 \dots c_n)) &= \bigcup_{i \in 1 \dots n} \mathcal{E}(c_i) \\
\mathcal{E}(\text{(not } c)) &= \mathcal{D} - \mathcal{E}(c) \\
\mathcal{E}(\text{(all } r \ c)) &= \{x \in \mathcal{D} \mid \forall y, \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(c)\} \\
\mathcal{E}(\text{(atleast } n \ r)) &= \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \geq n\} \\
\mathcal{E}(\text{(atmost } n \ r)) &= \{x \in \mathcal{D} \mid \|\{y \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(r)\}\| \leq n\} \\
\mathcal{E}(\text{(rvm } r_1 \ r_2)) &= \{x \in \mathcal{D} \mid \forall y \in \mathcal{D} : \langle x, y \rangle \in \mathcal{E}(r_1) \Rightarrow \langle x, y \rangle \in \mathcal{E}(r_2)\} \\
\mathcal{E}(\text{(sd } c \ b_1 \dots b_n)) &= \{x \in \mathcal{D} \mid \exists y \in \mathcal{E}(c) : \langle x, y \rangle \in \bigcap_{i \in 1 \dots n} \mathcal{E}(b_i)\} \\
\mathcal{E}(\text{(}\subseteq \ r_1 \ r_2)) &= \{\langle x, y \rangle \in (\mathcal{D} \times \mathcal{D}) \mid \forall z \in \mathcal{D} \langle x, z \rangle \in \mathcal{E}(r_1) \Rightarrow \langle y, z \rangle \in \mathcal{E}(r_2)\} \\
\mathcal{E}(\text{(}\supseteq \ r_1 \ r_2)) &= \{\langle x, y \rangle \in (\mathcal{D} \times \mathcal{D}) \mid \forall z \in \mathcal{D} \langle y, z \rangle \in \mathcal{E}(r_2) \Rightarrow \langle x, z \rangle \in \mathcal{E}(r_1)\} \\
\mathcal{E}(\text{(androle } r_1 \dots r_n)) &= \bigcap_{i \in 1 \dots n} \mathcal{E}(r_i) \\
\mathcal{E}(\text{(orrole } r_1 \dots r_n)) &= \bigcup_{i \in 1 \dots n} \mathcal{E}(r_i) \\
\mathcal{E}(\text{(notrole } r)) &= (\mathcal{D} \times \mathcal{D}) - \mathcal{E}(r) \\
\mathcal{E}(\text{(comp } r_1 \dots r_n)) &= \mathcal{E}(r_1) \circ \dots \circ \mathcal{E}(r_n) \\
\mathcal{E}(\text{(self)}) &= \{\langle x, x \rangle \mid x \in \mathcal{D}\} \\
\mathcal{E}(\text{(inv } r)) &= \{\langle x, y \rangle \mid \langle y, x \rangle \in \mathcal{E}(r)\} \\
\mathcal{E}(\text{(range } r \ c)) &= \mathcal{E}(r) \cap (\mathcal{D} \times \mathcal{E}(c)) \\
\mathcal{E}(\text{(trans } r)) &= (\mathcal{E}(r))^+
\end{aligned}$$

On peut alors comparer les systèmes dans le tableau 2.1.

2.5 Conclusion

On a présenté le formalisme des logiques de descriptions qui présente les particularités suivantes :

- il est défini comme une logique ;
- un système déductif (incomplet) lui est associé ;
- il est très modulaire (dans la sémantique mais aussi la définition de la déduction) ;
- il est extrêmement expressif ;

- il a donné lieu à de nombreuses implémentations différentes et un certain nombre d'applications.

Parmi les lacunes que l'on peut trouver aux logiques de descriptions :

- toute l'étude présentée ignore les problèmes de cycles ;
- les langages retenus pour expérimenter sont relativement pauvres (par rapport au calcul des prédicats par exemple) ;

2.5.1 À propos des cycles

Le traitement des cycles dans les terminologies est un problème réel. D'une part, on peut trouver des taxonomies « naturelles » contenant des cycles telle que celles des exemples 2.14 et 2.15.

Exemple 2.14 (Terminologie des êtres humains).

```
Humain      ≐ (and Anything (all parent Humain))
Homme       ≐ (and Humain (all soeur Femme))
Femme      ≐ (and Humain (all frere Homme))
```

Exemple 2.15 (Terminologie des arbres binaires).

```
branch      ≐ anyrelation
Tree        ≐ (all branch Tree)
BinaryTree ≐ (and Tree
              (atmost 2 branch)
              (all branch BinaryTree))
```

Ces deux exemples posent problème parce que la terminologie, à elle seule, ne permet pas de les fonder. Il suffit, en effet, pour leur donner une interprétation finie de créer un individu initial (un humain sans parents ou un arbre sans branches). Il peut d'ailleurs y avoir plusieurs individus initiaux. Il y a donc en général plusieurs interprétations satisfaisant les définitions.

La fonction d'interprétation est considérée comme la solution d'une équation $I = F(I)$ avec F une fonction (appelée opérateur de point fixe) qui calcule I en fonction d'une I initiale (qui peut-être donnée par l'affectation initiale de la sémantique constructive). Une solution de cette équation est une interprétation telle que l'équation soit vérifiée. On peut imaginer qu'il existe de nombreuses solutions quand il en existe une. Pour cela, deux sortes de sémantiques peuvent être imaginées pour ce type de taxonomie :

- Une sémantique du plus petit point fixe qui restreint chaque interprétation à son extension minimale. Elle accepte les définitions circulaires (`BinaryTree`), respecte la sous-propriété des terminologies sans cycle et permet d'avoir $\text{BinaryTree} \leq \text{TernaryTree}$. Mais elle ne permet pas les objets circulaires ou infinis. Elle convient bien pour la circularité de type `parent` pour peu qu'il existe un objet initial mais pas à celle de type `frere/soeur`.

- Une sémantique du plus grand point fixe qui étend chaque interprétation à son extension maximale. Elle accepte les définitions circulaires (Homme) et respecte la subsomption des terminologies sans cycle. Mais elle a pour modèles des arbres binaires circulaires et donne la même interprétation à deux concepts homomorphes définis sous le même primitif (comme Homme et Femme dans l'exemple 2.14).

Une définition plus moderne des logiques de descriptions utilise comme seul opérateur d'introduction l'implication généralisée (CGI pour 'conceptual generalized implication') qui permet d'affirmer d'un terme est subsumé par un autre sans restriction sur la forme des termes à gauche et à droite de l'introducteur. Il est alors autorisé de disposer de plusieurs introduction d'un même terme. Ceci offre de grands avantages, en particulier la possibilité de confronter plusieurs définitions concurrente d'un même terme. Cette définition s'accorde des cycles qui sont en général utilisés pour exprimer l'équivalence entre deux termes, même complexes.

2.6 Exercices

Exercice 2.1 (*). Soient les assertions de subsomption suivantes :

- (and Rectangle Losange) \preceq Rectangle
- (all angle Droit) \preceq rectangle
- (atleast 3 angle) \preceq (atleast 4 angle)
- (some angle) \preceq (atleast 1 angle)
- (atleast 1 angle) \preceq (some angle)
- (and (atleast 4 angle) (atmost 4 angle) (all angle Droit))
 \preceq (and (some angle) (all angle Droit))

1. dites si elles vous semblent vérifiées ;
2. utilisez l'algorithme de SUBSUMPTION-TEST pour l'établir.

Exercice 2.2 (*). Soit la terminologie suivante (taxonomie imaginaire des judokas) exprimée dans \mathcal{ALNR} :

```

Combat       $\dot{\preceq}$  Anything
AuxPoints    $\dot{\preceq}$  Combat
ParIpon      $\dot{\preceq}$  Combat
AuxPoints    $\dot{\oplus}$  ParIpon
Judoka       $\dot{\preceq}$  Anything
combat       $\dot{\preceq}$  anyrelation
victoire     $\dot{\preceq}$  combat
defaite      $\dot{\preceq}$  combat
Competiteur  $\dot{=}$  (and Judoka (atleast 1 victoire))
Maitre       $\dot{=}$  (and Judoka (atleast 11 victoire)
                (all victoire ParIpon))
Disciple     $\dot{=}$  (and Judoka (atleast 1 victoire)
                (all defaite AuxPoints))

```

Refaire tous les exemples qui ont été donnés dans le cours en terme d'application des algorithmes :

1. Donnez ses sous-ensembles \mathcal{N}_C , \mathcal{N}_R et \mathcal{N}_O ;
2. Normalisez la terminologie;
3. Calculez l'expansion de l'expression (and Competiteur (all combat AuxPoints)) dans la terminologie;
4. Normalisez l'expression ainsi obtenue;
5. Que peut-on en conclure pour la comparaison de cette expression avec les concepts `Disciple` et `Maitre`?

Exercice 2.3 ().** Montrez que dans la définition 2.27 la règle `role-bottom-propagation` est redondante.

Exercice 2.4 ().** Vérifier que \mathcal{ALC} et \mathcal{ALUE} sont équivalents (par les constructeurs).

Exercice 2.5 ().** Vérifier qu'une logique terminologique disposant des seuls constructeurs `all`, `or` et `not` permettent la construction de `range`.

Exercice 2.6 ().** Montrez que l'expansion (par `EXP`) d'un terme dans une \mathcal{ALNR} -terminologie peut rendre un terme de taille exponentielle en fonction de la terminologie.

Exercice 2.7 (*)**. Refaire le travail de ce chapitre en introduisant la composition de rôles :

$$\mathcal{E}(\text{(comp } r_1 \dots r_n)) = \mathcal{E}(r_1) \circ \dots \mathcal{E}(r_n)$$

permettant d'exprimer :

```
grandmere ≐ (comp mere parent)
GrandMere ≐ (and Femme (atleast 1 (comp enfant enfant)))
```

On procédera ainsi :

1. Introduire `comp` dans la syntaxe de \mathcal{FLN} ;
2. Introduire `comp` dans sa sémantique (fonction d'extension);
3. Considérer `comp` dans l'algorithme de mise en forme normale (et l'introduire dans la forme normale);
4. Établir les propriétés de complexité de l'algorithme et d'équivalence des formes obtenues;
5. Prendre `comp` en compte dans les algorithmes `EXP`, `NORM` et `COMPARE`;
6. Établir les propriétés de ces algorithmes.

Exercice 2.8 (*). On se place dans le contexte d'une logique terminologique construite à l'aide des constructeurs `all`, `and` et `range`. La sémantique du constructeur de rôles `range` est donnée dans la définition 2.30. Cette logique sera nommée \mathcal{FL}^{--} . On considèrera la terminologie munie d'une unique clause :

```
T={ chef ≐ musicien }
```

et les termes suivants² :

$t_1 \doteq (\text{and Orchestre (all chef Pianiste)})$

$t_2 \doteq (\text{and Orchestre (all chef Méléphobe)}$

$(\text{all (range musicien Méléphobe) Pianiste}))$

1. donnez une définition en langage naturel des termes t_1 et t_2 ;
2. donnez les sous-ensembles \mathcal{N}_C et \mathcal{N}_R de \mathcal{FL}^{--} ;
3. donnez l'extension des termes t_1 et t_2 en fonction de l'extension des concepts et rôles atomiques ;
4. montrez à l'aide de cette interprétation que $\mathcal{E}(t_2) \subseteq \mathcal{E}(t_1)$ pour toute \mathcal{FL}^{--} -extension \mathcal{E} pour T (on pourra faire un dessin de la situation si on le désire, mais ce n'est pas très simple) ;
5. que peut-on penser de $t_2 \preceq_T t_1$?

Exercice 2.9 ().** On se propose, à l'instar de ce qui se fait dans les graphes conceptuels, de traduire les termes de la logique terminologique \mathcal{FL}^{--} de l'exercice précédent en calcul des prédicats. Pour cela on va associer à chaque concept atomique un prédicat monadique (d'arité 1) et à chaque rôle atomique un prédicat diadique (d'arité 2). On va ensuite définir la fonction ϕ paramétrée, par une variable si son argument est un concept et par deux variables si c'est un rôle, de telle sorte que :

$$\phi_x(t_1) = \forall y, \text{chef}(x, y) \Rightarrow \text{Pianiste}(y)$$

$$\phi_x(t_2) = [\forall y, \text{chef}(x, y) \Rightarrow \text{Melophobe}(y)]$$

$$\wedge [\forall y, (\text{musicien}(x, y) \wedge \text{Melophobe}(y)) \Rightarrow \text{Pianiste}(y)]$$

1. donnez une définition analytique de ϕ_x et $\phi_{x,y}$;
2. montrez par induction sur vos définitions que pour toute \mathcal{FL}^{--} -structure $\langle \mathcal{D}, \mathcal{E} \rangle$ de T ,

$$\text{si } a \in \mathcal{E}(t) \text{ alors } A_T \models \phi_a(t)$$

si

$$\forall c \in \mathcal{N}_C, \forall a, a \in \mathcal{E}(c) \stackrel{\text{def}}{=} A_T \models \phi_a(c)$$

$$\text{et } \forall r \in \mathcal{N}_R, \forall a, b, \langle a, b \rangle \in \mathcal{E}(r) \stackrel{\text{def}}{=} A_T \models \phi_{a,b}(r);$$

3. que dire de “si $t \preceq_T t'$ alors $A_T \models \forall x, \phi_x(t) \Rightarrow \phi_x(t')$ ” ?
4. qu'en est-il dans le sens inverse ?
5. on construira la formule associée à une assertion à l'aide de cette fonction ϕ . Par exemple,

$$\phi(t \preceq t') \stackrel{\text{def}}{=} \forall x, \phi_x(t) \Rightarrow \phi_x(t')$$

Donnez les règles de construction pour les autres symboles en vous appuyant sur la définition 2.7 du cours (attention, il faut distinguer concepts et rôles).

6. construisez ainsi l'ensemble d'axiomes A_T qui va être associé à la terminologie T ;

²Car le chef n'aime pas la musique, ni le BACK, ni le CLASSIC (air connu)

7. en s'appuyant sur les définition 2.9 et conséquence 2.3 du cours, donnez la définition d'équivalence, d'exclusion et d'incohérence en fonction de la traduction vers le calcul des prédicats (mais sans utiliser le concept `Nothing`);

Exercice 2.10 (*)**. À partir d'ici on ne considère que \leq et \doteq pour définir les \mathcal{FL}^{--} -terminologies. On cherche maintenant à construire une procédure de décision pour des formules du type $t \preceq_T t'$, c'est-à-dire capable de prouver $A_T \models \forall x, \phi_x(t) \Rightarrow \phi_x(t')$.

1. donnez une preuve logique valide de $A_T \models \forall x, \phi_x(t_2) \Rightarrow \phi_x(t_1)$;
2. peut-on réutiliser les résultats concernant les graphes conceptuels et pourquoi ?
3. montrez que pour toute description de *concept* t , $\phi_x(t)$ peut être mis sous la forme ³ :

$$\bigwedge_{i=1}^n \phi_x(c_i) \wedge \bigwedge_{i=1}^m \forall y, [\phi_{x,y}(r_i) \wedge \bigwedge_{j=1}^{p_i} \phi_y(c'_{i,j})] \Rightarrow \phi_y(c'_i)$$

où les c_i sont des concepts atomiques, les r_i des rôles atomiques et les $c'_{i,j}$ et c'_i des descriptions de concepts quelconques ;

4. exprimez deux règles de réécritures permettant de transformer les formules du type présenté ci-dessus de manière à ce que :

règle-r=r si deux conjoints concernent le même rôle du concept alors un seul conjoint subsistera et sa partie droite (ce qui se trouve après le $\phi_{x,y}(r_i)$) sera conjointe à celle de l'autre conjoint (exemple : $\forall y, r(x, y) \wedge h(y)$ et $\forall y, r(x, y) \wedge h'(y)$ donnera $\forall y, r(x, y) \wedge h'(y) \wedge h(y)$);

règle-r≤r' si un conjoint concerne un rôle subsumé par celui d'un autre conjoint alors la partie droite du premier conjoint est à conjindre à la partie droite du second (exemple : $\forall y, r(x, y) \wedge h(y)$ et $\forall y, r'(x, y) \wedge h'(y)$ avec $r \leq r'$ donnera $[\forall y, r(x, y) \wedge h'(y) \wedge h(y)] \wedge [\forall y, r'(x, y) \wedge h'(y)]$)

Justifiez rapidement pourquoi ces règles sont correctes (respectent la sémantique du calcul des prédicats), confluent et terminent (si l'on marque les applications de la règle $r \leq r'$) pour des termes sans cycle et ne laissent qu'un seul conjoint par rôle ;

5. montrez que l'on peut maintenant considérer chaque conjoint indépendamment des autres pour tester la subsomption (c'est-à-dire comparer chaque conjoint du subsumé potentiel avec ceux du subsumant potentiel). Proposer alors un ensemble de règles (du type de la définition 2.28 du cours) permettant de le faire (elles seront inévitablement récursives).
6. Quels sont les avantages et les inconvénients respectifs des logiques terminologiques par rapport aux graphes conceptuels au regard de la procédure que l'on vient de suivre ?

³C'est-à-dire qu'il existe une transformation de $\phi_x(t)$ valide pour le calcul des prédicats ou au moins valide pour le type de formules considérées.

Modèles et modèle

On s'interroge sur l'utilisation du mot modèle dans divers domaines. En particulier on peut distinguer des modèles dans l'activité de modélisation et dans la théorie des modèles en logique (celle qui est utilisée dans les autres chapitres).

FIGURES NON DISPONIBLES

Modélisation

Le modèle est une représentation d'un domaine avec toutes les simplifications (et toutes les erreurs) que cela suppose. On peut par exemple modéliser la surface d'une automobile et l'écoulement de l'air sur cette surface afin d'améliorer sa consommation. Le résultat en sera un schéma d'écoulement en fonction de la vitesse.

La modélisation met en jeu trois entités : un modélisateur, une situation à modéliser et le modèle lui-même. Dans le contexte de la représentation de connaissance, le modélisateur correspond à celui qui alimente une base de connaissance, le modèle à la base de connaissance et la situation à modéliser à ce que représente la base.

Un modèle est bien plus utile lorsqu'il ne fait pas que décrire ce qui est observé mais lorsqu'il prédit ce qui ne l'a pas encore été. Ceci ouvre la voie en sciences expérimentales à la notion de falsifiabilité (un bon modèle peut être invalidé par une expérience cruciale ne réalisant pas ses prédictions) et en ingénierie à l'utilisation de modèles afin de tester une conception particulière.

Théorie des modèles

En théorie des modèles, un modèle est une interprétation satisfaisant un ensemble d'assertions. En particulier, cette formulation n'envisage aucun rapport avec une quelconque réalité. Seule l'expression syntaxique (l'ensemble d'assertions) initiale compte.

En théorie des modèles, on se place plutôt du point de vue du développeur de systèmes de gestion de bases de connaissance et non plus du point du modélisateur. Il importe à celui-ci non plus que le résultat d'une simulation soit valide dans une situation particulière mais dans toutes celles qui pourraient être modélisées par l'ensemble d'assertions.

FIGURE 2.5 :

FIGURE 2.6 :

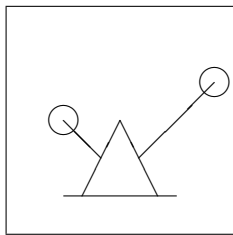
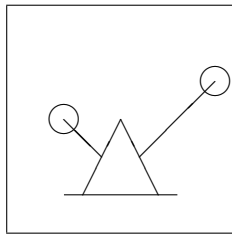


FIGURE 2.7 :

Synthèse

Les deux approches envisagées ci-dessus sont manifestement différentes. De plus, on voit bien que le mot “modèle” ne revêt pas la même signification dans les deux cas. En fait, ce qui est un modèle en modélisation correspond à l’ensemble d’assertions initiale dans la théorie des modèles.

Les deux approches nous intéressent : la théorie des modèles parce qu’elle est constamment utilisée ici pour donner la sémantiques des langages utilisés pour décrire un domaine et la modélisation car c’est pour l’activité de modélisation que l’on crée ces langages.

à première vue, on vient de mettre en évidence que les modèles ne recouvrent pas la même chose dans les deux domaines. Mais la question se pose : est-il possible de réconcilier ces deux approches au delà du vocabulaire.

Réconciliation 1

Puisqu’un modèle est une abstraction, il peut être l’abstraction de nombreuses situations (un groupe en mathématiques est l’abstraction de $\langle \mathbb{N}, +, 0 \rangle$ mais aussi de $\langle \mathbb{Z}, *, 1 \rangle$ ou $\langle \mathbb{B}, \wedge, \top \rangle$). D’autre part, puisque la simulation est fondée *uniquement sur l’abstraction*, si elle est valide pour sa concrétisation, elle le sera pour toutes ses concrétisations possibles.

Il est donc possible de réconcilier les deux schémas comme sur la figure 2.7.

Le fonctionnement de la modélisation informatique doit donc être la même que celle des systèmes logiques. La différence essentielle est que la théorie des modèles, ne pouvant s’appuyer sur le modélisateur, est destinée à s’assurer que le résultat d’une manipulation de l’abstraction est bien valide dans toutes ses concrétisations possibles.

FIGURE 2.8 :

Réconciliation 2

La notion d'approximation a ceci d'intéressant qu'elle ré-introduit la concrétisation dans la sphère du syntaxique au lieu de la laisser à la sémantique.

Modèles et révision

Il est intéressant de conserver la notion de monde privilégié lorsque l'on fait de la révision. Considérer tous les modèles (au sens de la théorie des modèles) lorsque l'on fait de la révision logique n'est sans doute pas la bonne solution puisque le modélisateur pense à ce qu'il modélise et non à l'ensemble de tous les modèles.

Chapitre 3

Graphes conceptuels

L'EXPOSÉ QUI SUIT DES GRAPHES CONCEPTUELS est principalement fondé sur le traitement formel des graphes conceptuels proposés dans [Chein and Mugnier, 1992] et prolongé dans [Mugnier and Chein, 1996]. Le lecteur désirant une courte introduction au formalisme des graphes conceptuels pourra consulter avec profit [Way, 1992].

Références

Michel Chein and Marie-Laure Mugnier. **Conceptual graphs: fundamental notions.** *Revue d'intelligence artificielle*, 6(4):365–406, 1992

Marie-Laure Mugnier and Michel Chein. **Représenter des connaissances et raisonner avec des graphes.** *Revue d'intelligence artificielle*, 10(1):7–56, 1996

Eileen Way. Conceptual graphs: fundamental notions. *Journal of experimental and theoretical artificial intelligence*, 4(2):75–84, 1992

Aperçu

3.1	Exemple introductif	54
3.2	Graphes conceptuels	55
3.3	\mathcal{S} -projection et \mathcal{S} -morphisme	60
3.4	Opérations	61
3.5	Sémantique	65
3.6	Complexité	70
3.7	Conclusion	70
3.8	Exercices	71

Les graphes conceptuels ont tout d'abord été introduits dans [Sowa, 1984]. La présentation qui en est faite ici, diverge sur plusieurs points :

- on n'introduira pas la notion de contexte ou cadre ;
- la notion de base canonique est remplacée par celle de graphe étoile ;

- on ne prend pas en compte la possibilité d'utiliser des ensembles de référents en position référent ;
- on ne considère pas les extensions sur les définitions :
 - l'introduction de types ($\kappa(\bar{x})$ is G ou $\kappa = \lambda\bar{x}.G$) ;
 - la hiérarchisation de ces types ;
 - les opérations sur les types : expansion, contraction (voir [Sowa, 1984] §3.16) et localisation (voir thèse Leclère).

Elle diffère aussi de [Chein and Mugnier, 1992] par l'introduction d'innovations de [Mugnier and Chein, 1996] (comme la non connexité des \mathcal{S} -graphes) et par l'omission des notions suivantes :

- partitions compatibles (§6) ;
- classes d'équivalence (§7) ;
- Join et Iso-join (§8).

3.1 Exemple introductif

Exemple 3.1 (utilisation de graphe conceptuel). La figure 3.1 représente un graphe que l'on interprétera par «un singe nommé Paul mangeant une noix avec une cuillère faite de sa coquille».

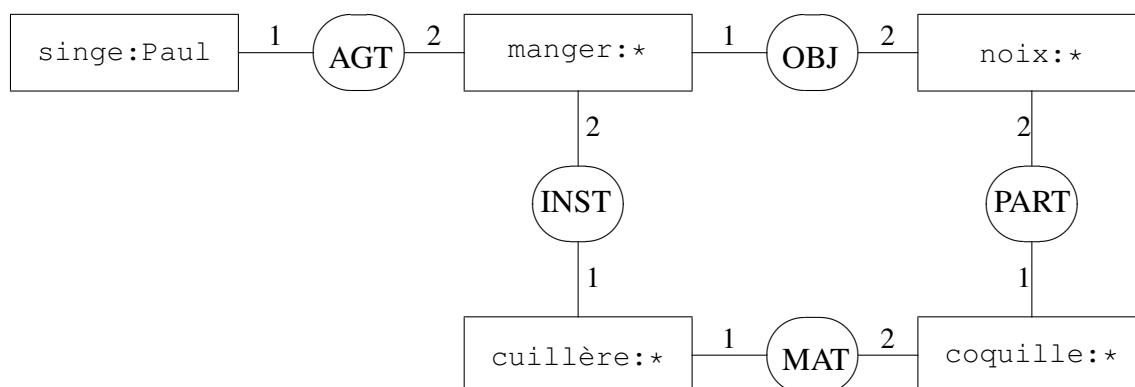


FIGURE 3.1 : Description graphique de l'exemple (d'après [Sowa, 1984]). Les symboles AGT, OBJ, INST, PART, MAT correspondent respectivement aux catégories pragmatico-grammaticales agent, objet, instrument, partie, matière.

On peut remarquer dans ce graphe trois types d'entités :

- des nœuds arrondis contenant un symbole ;
- des nœuds rectangulaires contenant deux symboles séparés par “ :” ;

- des arcs entre un nœud arrondi et un nœud rectangulaire (comme on le verra plus loin, les arcs devraient être identifiés par des numéros dépendant des nœuds arrondis).

Les nœuds rectangulaires correspondent à des objets et les nœuds arrondis à des relations entre objets. La fonction des objets liés est donné par le numéro de l'arc. Ces nœuds correspondent donc respectivement aux concepts et rôles des logiques terminologiques. On peut aussi distinguer parmi les nœuds rectangulaires ceux dont le second symbole est * des autres. Ceux-ci représentent des objets génériques (correspondant aux concepts des logiques terminologiques) alors que les autres correspondent à des individus nommés (ils correspondent alors aux *marqueurs individuels* des logiques terminologiques, voir § 2.3). En fait, les nœuds génériques (contenant un *) sont différents des concepts en ce qu'il sont dans le graphe pour représenter *un* individu particulier (par exemple, la noix que le singe mange et non pas l'ensemble des noix). Ils sont donc quantifiés existentiellement.

Montrer les relations : généralité, instanciation.

3.2 Graphes conceptuels

Malgré leur apparente simplicité, les graphes conceptuels constituent une classe de graphe relativement complexe. Ceci entraîne l'insertion d'une multitude de notions préliminaires. La formulation la plus simple d'un graphe conceptuel est certainement celle d'un multi-hyper-graphe étiqueté sur les arrêtes et les nœuds.

Définition 3.1 (Graphe bipartie*). Une structure $\langle N_1, N_2, E \rangle$ avec $N_1 \cap N_2 = \emptyset$ et $E \subseteq N_1 \times N_2$ est un graphe bipartie dont N_1 et N_2 sont les deux ensembles de nœuds et E l'ensemble des arcs.

Remarque. Le graphe $\langle N_1 \cup N_2, E \rangle$ est naturellement associé au graphe bipartie $\langle N_1, N_2, E \rangle$.

Exemple 3.2 (Graphe bipartie).

- $\langle \{manger, singe\}, \{AGT, OBJ\}, \{(AGT, manger), (AGT, singe)\} \rangle$ est un graphe bipartie ;
- $\langle \emptyset, \{AGT, OBJ\}, \{(AGT, OBJ)\} \rangle$ n'est pas un graphe bipartie car il y a un arc entre deux éléments du même ensemble ;
- $\langle \{manger, singe\}, \emptyset, \{(manger, singe)\} \rangle$ n'est pas un graphe bipartie pour la même raison ;

Définition 3.2 (Graphe étiqueté*). Un graphe étiqueté $\langle N, E, \lambda \rangle$ est composé d'un graphe $\langle N, E \rangle$ et d'une application $\lambda : E \rightarrow L$ (où L est un ensemble d'étiquettes).

Définition 3.3 (Multi-ensemble*). Un multi-ensemble est un ensemble dans lequel un élément peut apparaître plusieurs fois..

Définition 3.4 (Multi-graphe*). Une structure $\langle N, E \rangle$ telle que E soit un multi-ensemble sur $N \times N$ est un multi-graphe.

Définition 3.5 (Multi-graphe étiqueté*). Un multi-graphe étiqueté $\langle N, E, \lambda \rangle$ est un multi-graphe $\langle N, E \rangle$ associé à une application $\lambda : E \rightarrow L$ (où L est un ensemble d'étiquettes) telle que les éléments de E associés au même couple de nœuds sont étiquetés par des étiquettes différentes.

Définition 3.6 (Graphe étoile). Soient deux ensembles disjoints T_C et T_R un graphe étoile pour une relation $r \in T_R$ est un multi-graphe bipartie étiqueté connexe* $\langle N, \{r\}, E, \lambda \rangle$ où $N \subseteq T_C$ et les arcs sont étiquetés par une suite d'entiers croissante (la contrainte la plus importante est que la fonction d'étiquetage soit une bijection).

On peut noter immédiatement pourquoi un tel graphe est appelé graphe étoile : tous les nœuds concepts sont en effet reliés à un même nœud relation (car il y en a qu'un et que le graphe est connexe et biparti).

Notation. Dans la suite on omettra l'étiquetage des arcs par λ . Il s'agira simplement de ce souvenir que les arcs sont discernables et étiquetés d'une manière liée aux graphes de relation.

Définition 3.7 (Support). Un support \mathcal{S} est un quintuplet $\langle T_C, T_R, M, ::, B \rangle$ tel que :

T_C est l'ensemble des types de concepts ;

T_R est l'ensemble des types de relations ;

M est un ensemble énumérable de marqueurs contenant :

- * le marqueur générique ;
- \emptyset le marqueur absurde ;

$::$ est un prédicat de conformité sur $M \times T_C$;

B est un ensemble de graphes étoiles sur T_C, T_R ;

et :

- $\langle T_C, \leq \rangle$ est un ordre partiel* fini avec \top et \perp comme supremum et infimum ;
- $T_R = \cup_{i=2}^{+\infty} T_R^i$ où $\forall i, j, i \neq j \Rightarrow T_R^i \cap T_R^j = \emptyset$ et $\langle T_R^i, \leq_R^i \rangle$ y est un ordre partiel* fini avec \top^i comme supremum ;
- $T_C \cap T_R = \emptyset$;
- il existe une bijection $\beta : T_R \rightarrow B$ telle que $\forall r \in T_R, \beta(r)$ soit un graphe étoile pour r et $\forall r, r' \in T_R^i, r \leq_R^i r' \Rightarrow \forall j \in 1 \dots i, \beta_j(r) \leq \beta_j(r')$;
- $\langle M, \sqsubseteq \rangle$ soit un treillis* construit de telle sorte que $\forall m \in M - \{*, \emptyset\}, \emptyset \sqsubseteq m \sqsubseteq *$ et que tous les éléments de $M - \{*, \emptyset\}$ soient incomparables ;
- $\forall m \in M, \forall t, t' \in T_C,$

$$\begin{aligned} m &:: \top, & (t' \leq t) \wedge (m :: t') &\Rightarrow m :: t \\ m &\not:: \perp, \\ \emptyset &\not:: t, & t &\neq \perp, \equiv * :: t. \end{aligned}$$

Notation. On notera par $\beta_i(r)$ le nœud concept associé à l'arc numéroté par i dans le graphe étoile associé à r par β .

Exemple 3.3 (Support). La figure 3.2 représente graphiquement le support suivant :

$$\begin{aligned} & \langle \{\text{objet, figure, polygone, \dots, sol}\}, \\ & \quad \{\text{sur, acote, adroite, agauche}\}, \{*, \text{a, b, c, g, \dots } \emptyset\}, \\ & \quad \{\langle *, \top \rangle, \langle \text{a, carré} \rangle, \dots \langle \text{g, sol} \rangle, \}, \\ & \langle \langle \{\text{objet}\}, \{\text{sur}\}, \{\langle \text{objet, sur} \rangle\}, \langle \{\text{figure}\}, \{\text{agauche}\}, \{\langle \text{figure, agauche} \rangle\} \rangle \rangle \end{aligned}$$

On a ainsi : $\beta_1(\text{sur}) = \beta_2(\text{sur}) = \text{objet}$ et $\beta_1(\text{agauche}) = \beta_1(\text{adroite}) = \beta_1(\text{acote}) = \beta_2(\text{agauche}) = \beta_2(\text{adroite}) = \beta_2(\text{acote}) = \text{figure}$.

Définition 3.8 (\mathcal{S} -graphe). Un \mathcal{S} -graphe $G = \langle R, C, U, l \rangle$ est une structure telle que :

- $\langle R, C, U \rangle$ soit un multi-graphe bipartie étiqueté* fini avec $C \neq \emptyset$;
- l soit une fonction

$$\begin{aligned} l : R &\longrightarrow T_R \\ r &\longmapsto \text{type}(r) \\ C &\longrightarrow T_C \times (M - \{\emptyset\}) \\ c &\longmapsto \langle \text{type}(c), \text{ref}(c) \rangle \end{aligned}$$

- $\forall c \in C, \text{ref}(c) :: \text{type}(c)$;
- $\forall r \in R,$

1. les arcs sortants de r sont totalement ordonnés de 1 à $\text{degré}(r)$ (et leurs extrémités sont nommées $G_i(r)$);
2. $\text{degré}(r) = \text{degré}(\beta(\text{type}(r)))$;
3. $\forall i \in 1, \dots, \text{degré}(r), \text{type}(G_i(r)) \leq \beta_i(\text{type}(r))$

On peut noter qu'un \mathcal{S} -graphe n'est pas forcément connexe (il s'agit donc plus généralement d'un ensemble de graphes connexes).

Notation (concept individuel, concept générique). Si $\text{ref}(c) \in M - \{*, \emptyset\}$, c est appelé un concept individuel ; si $\text{ref}(c) = *$, c est appelé un concept générique.

Définition 3.9 (\mathcal{S} -sous-graphe). Soit un \mathcal{S} -graphe $G = \langle R, C, U, l \rangle$, $G' = \langle R', C', U', l' \rangle$ est un \mathcal{S} -sous-graphe de (noté $G' \subseteq G$) si et seulement si :

- c' est un \mathcal{S} -graphe ;
- $R' \subseteq R$;
- $C' \subseteq C$;
- $U' \subseteq U|_{R' \times C'}$;
- $l' = l|_{R' \times C'}$.

Conséquence 3.1.

$\forall \langle t, m \rangle; t \in T_C$ et $m \in M - \{\emptyset\}$, $\langle t, m \rangle$ est un \mathcal{S} -graphe.

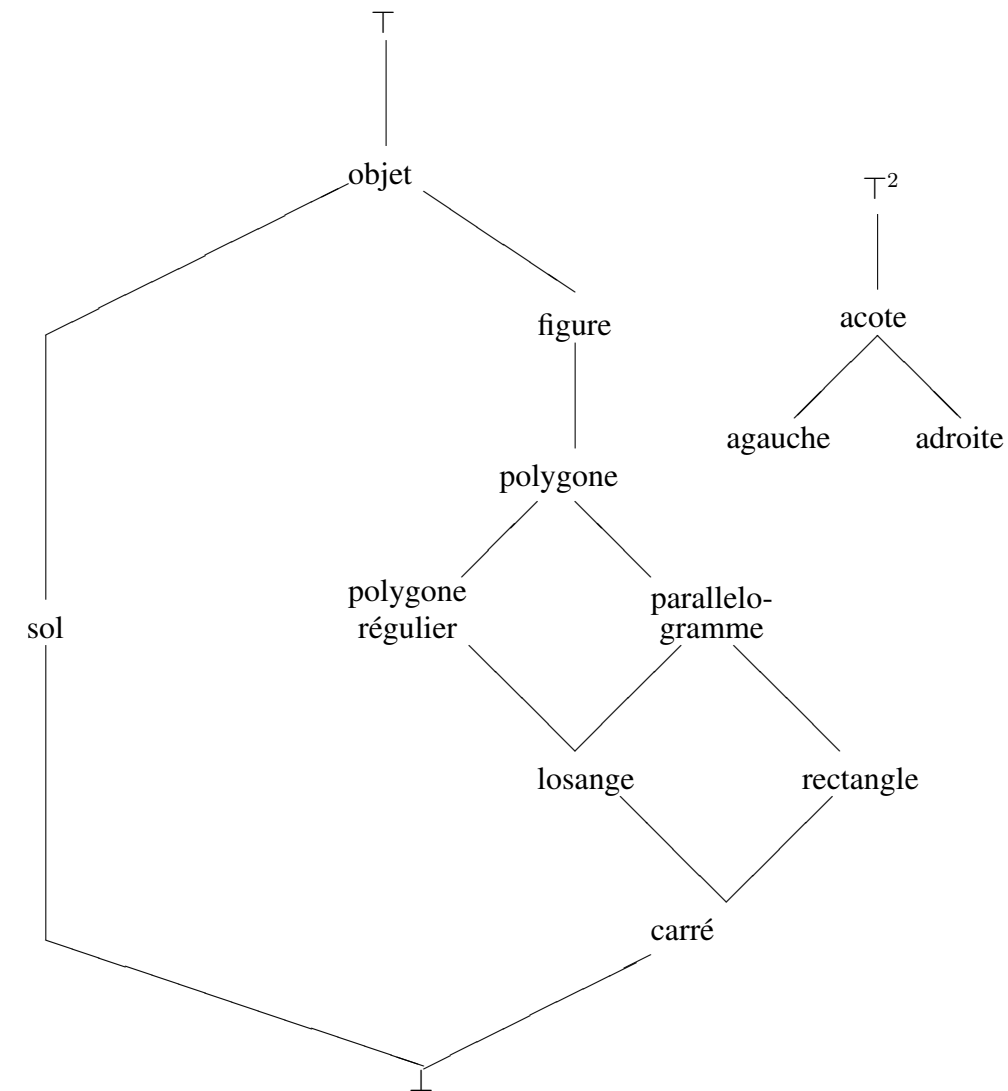


FIGURE 3.2 : Représentation graphique d'un \mathcal{S} -support (d'après [Chein and Mugnier, 1992]). En haut, le graphe des types ; en bas les graphes étoile de *sur* et à gauche.

Exemple 3.4 (\mathcal{S} -graphe). La figure 3.3 représente graphiquement les \mathcal{S} -graphes dont le support est donné en exemple 3.3 suivants :

$$G = \langle \{r_1, r_2\}, \{o_1, o_2, o_3\}, \\ \{\langle r_1, o_1 \rangle, \langle r_1, o_2 \rangle, \langle r_2, o_2 \rangle, \langle r_2, o_3 \rangle\}, \\ \{\langle r_1, \text{sur} \rangle, \langle r_2, \text{sur} \rangle, \langle o_1, \langle \text{polygone}, * \rangle \rangle, \langle o_2, \langle \text{carré}, * \rangle \rangle, \langle o_3, \langle \text{sol}, \text{g} \rangle \rangle\} \rangle,$$

$$H = \langle \{r_1, r_2, r_3, r_4\}, \{o_1, o_2, o_3, o_4, o_5\}, \\ \{\langle r_1, o_1 \rangle, \langle r_1, o_2 \rangle, \langle r_2, o_2 \rangle, \langle r_2, o_3 \rangle, \langle r_3, o_4 \rangle, \langle r_3, o_5 \rangle, \langle r_4, o_5 \rangle, \langle r_4, o_3 \rangle\}, \\ \{\langle r_1, \text{sur} \rangle, \langle r_2, \text{sur} \rangle, \langle r_3, \text{sur} \rangle, \langle r_4, \text{sur} \rangle, \langle o_1, \langle \text{polygone}, * \rangle \rangle, \langle o_2, \langle \text{carré}, * \rangle \rangle, \\ \langle o_3, \langle \text{sol}, \text{g} \rangle \rangle, \langle o_4, \langle \text{figure}, * \rangle \rangle, \langle o_5, \langle \text{rectangle}, * \rangle \rangle\} \rangle,$$

$$K = \langle \{r_1, r_2, r_3, r_4, r_5\}, \{o_1, o_2, o_3, o_5\}, \\ \{\langle r_1, o_1 \rangle, \langle r_1, o_2 \rangle, \langle r_2, o_2 \rangle, \langle r_2, o_3 \rangle, \langle r_3, o_1 \rangle, \langle r_3, o_5 \rangle, \langle r_4, o_5 \rangle, \langle r_4, o_3 \rangle, \langle r_5, o_2 \rangle, \langle r_5, o_5 \rangle\}, \\ \{\langle r_1, \text{sur} \rangle, \langle r_2, \text{sur} \rangle, \langle r_3, \text{sur} \rangle, \langle r_4, \text{sur} \rangle, \langle r_5, \text{agauche} \rangle, \\ \langle o_1, \langle \text{carré}, \text{a} \rangle \rangle, \langle o_2, \langle \text{carré}, \text{b} \rangle \rangle, \langle o_3, \langle \text{sol}, \text{g} \rangle \rangle, \langle o_5, \langle \text{carré}, \text{c} \rangle \rangle\} \rangle$$

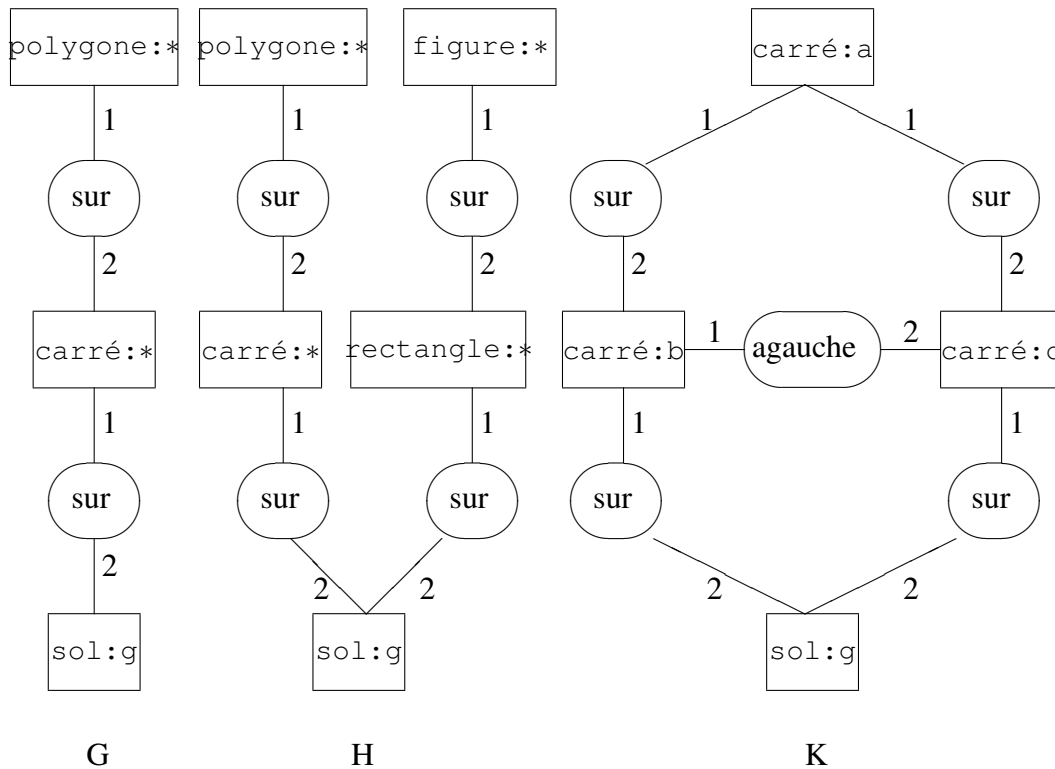


FIGURE 3.3 : Représentation graphique d'un \mathcal{S} -graphe (d'après [Chein and Mugnier, 1992])

Définition 3.10 (\mathcal{S} -graphe en forme normale). Soit un \mathcal{S} -graphe $G = \langle R, C, U, l \rangle$ est dit sous forme normale si et seulement si $\forall c, c' \in C, ref(c) \neq ref(c')$.

3.3 \mathcal{S} -projection et \mathcal{S} -morphisme

Définition 3.11 (\mathcal{S} -projection). Une \mathcal{S} -projection entre deux \mathcal{S} -graphes $\langle R, C, U, l \rangle$ et $\langle R', C', U', l' \rangle$ est une paire d'applications

$$\langle f : R \longrightarrow R', \\ g : C \longrightarrow C' \rangle$$

telle que :

- $\forall r \in R, \forall i \in 1, \dots, \text{degré}(r), g(G_i(r)) = G'_i(f(r))$;
- $\forall r \in R, l'(f(r)) \leq_R l(r)$;
- $\forall c \in C, \text{type}'(g(c)) \leq \text{type}(c)$ et $\text{ref}'(g(c)) \sqsubseteq \text{ref}(c)$.

La projection plaque un graphe sur un autre tout en vérifiant :

- la spécialisation des étiquettes sur les concepts et les relations ;
- la préservation de l'ordre des arcs.

Exemple 3.5. Dans les graphes de l'exemple 3.4 (figure 3.3), on a les projections suivantes :

- G dans H ;
- H dans G ;
- H dans K (de 4 manières différentes) ;
- G dans K (de deux manières différentes) ;
- c 'est tout.

Remarquer qu'une projection peut être multiple, elle se contente de conserver la structure du graphe initial, de restreindre le type des nœuds et permet d'avoir d'autres parties dans le graphe. Par contre elle conserve effectivement toutes les contraintes (par exemple, il n'est pas possible d'inverser les deux sur dans la projection de G dans H).

Définition 3.12 (\mathcal{S} -morphisme). Un \mathcal{S} -morphisme est une \mathcal{S} -projection satisfaisant :

$$\forall c \in C, l'(g(c)) = l(c) \text{ et } \forall r \in R, l'(f(r)) = l(r).$$

Définition 3.13 (\mathcal{S} -isomorphisme). Un \mathcal{S} -isomorphisme est une \mathcal{S} -projection bijective (pour f et g).

Exemple 3.6. Dans les graphes de l'exemple 3.4 (figure 3.3), il existe un seul \mathcal{S} -morphisme entre G et H et aucun \mathcal{S} -isomorphisme.

Conséquence 3.2. Un \mathcal{S} -isomorphisme est un \mathcal{S} -morphisme.

Proposition 3.3 (Stabilité des \mathcal{S} -morphisms). L'ensemble des \mathcal{S} -morphisms (resp. \mathcal{S} -projections, \mathcal{S} -isomorphisms, \mathcal{S} -isoprojections) est stable par composition.

Démonstration. La composition d'application est une application et :

- si $\forall r \in R, \forall i \in 1, \dots, \text{degré}(r), g(G_i(r)) = G'_i(f(r))$ et $\forall r' \in R', \forall i \in 1, \dots, \text{degré}(r'), g'(G'_i(r')) = G''_i(f'(r'))$ alors $\forall r \in R, \forall i \in 1, \dots, \text{degré}(r), g' \circ g(G_i(r)) = g'(G'_i(f(r))) = G''_i(f'(f(r))) = G''_i(f' \circ f(r))$;
- si $\forall r \in R, l'(f(r)) = (\text{resp. } \leq_R) l(r)$ et $\forall r' \in R', l''(f'(r')) = (\text{resp. } \leq_R) l'(r')$ alors $\forall r \in R, l''(f' \circ f(r)) = l'(f(r)) = (\text{resp. } \leq_R) l(r)$;
- si $\forall c \in C, l'(g(c)) = (\text{resp. } \leq) l(c)$ et $\forall c' \in C', l''(g'(c')) = (\text{resp. } \leq) l'(c')$ alors $\forall c \in C, l''(g' \circ g(c)) = l'(g(c)) = (\text{resp. } \leq) l(c)$. \square

3.4 Opérations

Les notions de projections et de morphismes ont été définies de manière non opératoire. Il est possible de définir des opérations sur les \mathcal{S} -graphes.

Définition 3.14 (Règles de spécialisation).

Somme disjointe Somme de deux \mathcal{S} -graphes disjoints

$$R \cap R' = \emptyset, C \cap C' = \emptyset$$

$$\left. \begin{array}{l} \langle R, C, U, l \rangle \\ \langle R', C', U', l' \rangle \end{array} \right\} \text{Somme disjointe} \longmapsto \langle R \cup R', C \cup C', U \cup U', l \cup l' \rangle$$

Suppression Suppression de relations jumelles

$$\exists r \neq r'; \text{type}(r) = \text{type}(r') \text{ et } \forall i \in 1, \dots, \text{degré}(r), G_i(r) = G_i(r')$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Suppression}} \langle R' = R - \{r'\}, C, U|_{R'}, l|_{R' \cup C} \rangle$$

Restriction Restriction de concepts

$$\exists c \in C; l(c) = \langle t, m \rangle, t' \leq t \text{ ou } t' = t \text{ et } m' < m \text{ et } m' :: t'$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Restriction}(c)} \langle R, C, U, l - \{\langle c, \langle t, m \rangle\}\} \cup \{\langle c, \langle t', m' \rangle\}\} \rangle$$

Restriction Restriction de relation (il faut respecter les contraintes des graphes-étoiles)

$$\exists r \in R; t' \leq_R l(r) \text{ et } \forall i \in 1 \dots \text{degré}(r), \text{type}(G_i(r)) \leq \beta_i(t')$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Restriction}(r)} \langle R, C, U, l - \{\langle r, l(r) \rangle\} \cup \{\langle r, t' \rangle\} \rangle$$

Fusion Fusion de deux nœuds (joint interne) $\exists c, c' \in C; l(c) = l(c'), \exists r_i \in R; \langle r_i, c' \rangle \in U$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Fusion}} \langle R, C - c', U - \{\langle r_i, c' \rangle\} \cup \{\langle r_i, c \rangle\}, l - \{\langle c', l(c') \rangle\} \rangle$$

On peut remarquer qu'il est inutile d'autoriser l'«unification» d'étiquette dans Fusion car la restriction est autorisée dans Restriction.

Exemple 3.7. Il est possible de spécialiser H en G de la figure 3.3 par la séquence d'opérations suivantes :

Restriction (figure:*) vers polygone:*

Restriction (rectangle:*) vers carre:*

Fusion (polygone:*,polygone:*)

Fusion (carre:*,carre:*)

Suppression (sur,sur);

Suppression (sur,sur);

Définition 3.15 (\mathcal{S} -Spécialisation). Soient G et H deux \mathcal{S} -graphes, G est une \mathcal{S} -spécialisation de H s'il peut être obtenu de H par une séquence d'applications des règles de spécialisation.

Notation. La spécialisation sera notée \preceq_S .

La clôture* d'un ensemble de \mathcal{S} -graphes E par spécialisation sera notée $\Sigma(E)$.

Définition 3.16 (Règles de généralisation).

Addition Addition d'une relation jumelle

$$\exists r \in R;$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Addition}} \langle R \cup \{r'\}, C, U \cup \{\langle r', G_i(r) \rangle\}_{i=1 \dots \text{degré}(r)}, l \cup \{\langle r', l(r) \rangle\} \rangle$$

Extension Extension de l'étiquette d'un concept (doit respecter les contraintes de type provenant de la base B)

$$\exists c \in C; l(c) = \langle t, m \rangle, m \sqsubseteq m', t \leq t' \text{ et } \forall r \in R, \exists i; c = G_i(r) \Rightarrow t' \leq \beta_i(\text{type}(r))$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Extension}(c)} \langle R, C, U, l - \langle c, \langle t, m \rangle \rangle \cup \langle c, \langle t', m' \rangle \rangle \rangle$$

Extension Extension de l'étiquette d'une relation

$$\exists r \in R; l(r) \leq_R t$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Extension}(r)} \langle R, C, U, l - \langle r, l(r) \rangle \cup \langle r, t \rangle \rangle$$

Séparation Séparation d'un concept en deux

$$\exists c \in C; \exists \{r_i\}_{i \in I} \subseteq R; \{\langle r_i, c \rangle\}_{i \in I} \subseteq U$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Séparation}} \langle R, C \cup \{c'\}, U - \{\langle r_i, c \rangle\}_{i \in I} \cup \{\langle r_i, c' \rangle\}_{i \in I}, l \cup \{\langle c', l(c) \rangle\} \rangle$$

Décomposition Séparation d'une composante déconnectée du graphe

$$\exists C' \subseteq C; \exists R' \subseteq R; U = U|_{R' \times C'} \cup U|_{R - R' \times C - C'},$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Décomposition}} \langle R', C', U|_{R' \times C'}, l|_{R' \cup C'} \rangle$$

Exemple 3.8. Il est possible de généraliser G en H de la figure 3.3 par la séquence d'opérations suivantes :

Addition (sur,sur);

Addition (sur,sur);

Séparation (carré:*,carré:*);

Séparation (polygone:*,polygone:*);

Extension (carré:*) vers rectangle:*

Extension (polygone:*) vers figure:*

Définition 3.17 (\mathcal{S} -Généralisation). Soient G et H deux \mathcal{S} -graphes, H est une \mathcal{S} -généralisation de G s'il peut être obtenu de G par une séquence d'applications des règles de généralisation.

Notation. La clôture* d'un ensemble de \mathcal{S} -graphes E par généralisation sera notée $\Gamma(E)$.

Proposition 3.4. Soient G et H deux \mathcal{S} -graphes, G est une spécialisation de H si et seulement si H est une généralisation de G .

Esquisse de démonstration. En inversant les séquences d'opérations. \square

Conséquence 3.5. Soient G et H deux \mathcal{S} -graphes, $G \preceq_{\mathcal{S}} H$ si et seulement si $H \in \Gamma(\{G\})$.

Remarque. Ce n'est pas la cas de $G \in \Sigma(\{H\})$

Proposition 3.6. L'ensemble des \mathcal{S} -graphes est exactement $\Sigma(\check{B} \cup \{[\top : *]\})$ où \check{B} est l'ensemble des graphes étoiles de \mathcal{S} dans lesquels les nœuds étiquetés par $t \in T_C$ ont l'étiquette $t : *$ et $[\top : *]$ correspond à $\langle \emptyset, \{n\}, \emptyset, \{ \langle n, \langle \top, * \rangle \} \rangle$.

Démonstration. Soit un \mathcal{S} -graphe G , s'il contient des nœuds concepts déconnectés, ceux-ci peuvent être obtenus par copie (somme disjointe) puis Restriction de $[\top : *]$ qui peut toujours engendrer l'étiquette correspondante. Pour chaque nœud relation dans le graphe, celui-ci peut être obtenu par copie (somme disjointe) du graphe étoile correspondant à son type puis par Restriction des étiquettes de chaque nœud concept vers l'étiquette désirée. Ensuite, on procède à la Fusion des nœuds correspondant au même nœud concept. Ceci est toujours possible. \square

Lemme 3.7. Soient G et H deux \mathcal{S} -graphes, s'il existe une projection de H vers G , alors $G \preceq_{\mathcal{S}} H$.

Démonstration. Soit $\pi = \langle f, g \rangle$ une projection de H vers $G = \langle R_G, C_G, U_G, l_G \rangle$,

$\forall c \in C_G$ si $g^{-1}(c) = \{c_1 \dots c_n\}$ alors $c_1 \dots c_n$ peuvent être fusionnés (d'abord par un Restrict pour égaliser leurs étiquettes à celle de r , puis par un Fusion) en un unique nœud ayant l'étiquette de c (si $n = 1$, la fusion n'est qu'une restriction). Ceci est une séquence S_1 d'application des règles de spécialisation permettant de passer de H à un graphe H^1 ayant les mêmes nœuds concept et les mêmes étiquettes pour ceux-ci que G .

$\forall r \in R_G$ si $f^{-1}(r) = \{r_1 \dots r_n\}$ alors les r_i peuvent être fusionnés (d'abord par un Restrict pour égaliser leurs étiquettes à celle de r les contraintes sont satisfaites car les nœuds concepts ont été restreints auparavant et que G est un \mathcal{S} -graphe, puis l'on peut appliquer Suppression $n-1$ fois (car $\forall j \in 1 \dots n, \forall i \in 1 \dots \text{degré}(r_j), H_i^1(f(r_j)) = g(G_i(r_j))$) ce qui définit une nouvelle séquence S_2 d'application des règles de spécialisation permettant de passer de H^1 à un graphe H^2 ayant les mêmes nœuds relation que G reliés aux mêmes nœuds concepts.

Ici on dispose donc d'un graphe H^2 isomorphe à une partie de G : il faut donc ajouter ce qu'il y manque. Soit $G' = \pi(H)$ isomorphe à H^2 , un sous- \mathcal{S} -graphe de H , soit K l'ensemble des $\langle R_G - R_{G'}, C_G|_{R_G - R_{G'}}, U_G|_{R_G - R_{G'}} \rangle$ toute partie connexe de K est un \mathcal{S} -graphe qui peut être joint (par Somme disjointe) à G' puis le résultat peut-être fusionné (par Fusion sur les nœuds de $C_K - C_G$). Ceci forme une séquence S_3 d'application des règles de spécialisation permettant de passer de G' à G .

Il est donc possible d'obtenir G à partir de H en appliquant la séquence de règles de spécialisation $S_1 S_2 S_3$. \square

Lemme 3.8. Soient G et H deux \mathcal{S} -graphes, si $G \preceq_{\mathcal{S}} H$ alors à toute séquence de spécialisation de H vers G peut être associée une projection de H vers G .

Démonstration. Soit $(H \Rightarrow)G^0S_0G^1S_1 \dots S_{k-1}G^k(= G)$ une séquence de spécialisations élémentaires. À chaque S_i est associée une projection $\pi_i : G^i \rightarrow G^{i+1}$ décomposée en $\pi_i = \langle f_i, g_i \rangle$ et définie comme suit :

Suppression (r, r') $g_i = id$ et $f_i = id$ sauf pour r et r' défini tel que $f_i(r) = f_i(r')$. On a bien alors

$$\begin{aligned} \forall j \in 1 \dots \text{degré}(r), & & G_j^i(r) &= G_j^i(r') \\ \Rightarrow & & f_i(G_j^i(r)) &= f_i(G_j^i(r')) \\ \Rightarrow & & G_j^{i+1}(g_i(r)) &= G_j^{i+1}(g_i(r')). \end{aligned}$$

Restriction (c) $f_i = id$ et $g_i = id$. $\text{type}_{i+1}(c) \leq \text{type}_i(c)$ et $\text{ref}_{i+1}(c) \sqsubseteq \text{ref}_i(c)$ on a donc bien (pour $f_i = id$ et $g_i = id$) $\text{type}(g_i(c)) \leq \text{type}(c)$ et $\text{ref}(g_i(c)) \sqsubseteq \text{ref}(c)$.

Restriction (r) $f_i = id$ et $g_i = id$. $l_{i+1}(r) \leq_R l_i(r)$ on a donc bien (pour $f_i = id$ et $g_i = id$) $l(f_i(r)) \leq_R l(r)$.

Fusion (c, c') (avec $l(c) = l(c')$) $f_i = id$ et $g_i = id$ sauf pour c et c' où $g_i(c) = g_i(c')$. Ceci conserve bien $\forall r; G_j^i(r) = c, g_i(G_j^{i+1}(r)) = G_j^i(f_i(r)) = g_i(c)$.

$\pi = \pi_{k-1} \circ \dots \circ \pi_1$ est une projection de H vers G . □

Au passage, cette façon de faire permet d'établir la présence du problème de l'existence d'une projection dans NP. En effet, le nombre minimal d'étapes nécessaires pour passer de H à G est linéairement fonction du nombre de nœuds dans le graphe initial.

Théorème 3.9. Soient G et H deux \mathcal{S} -graphes, il existe une projection de H vers G si et seulement si $G \preceq_{\mathcal{S}} H$.

Démonstration. Suit immédiatement des lemmes 3.7 et 3.8. □

Conséquence 3.10. $\preceq_{\mathcal{S}}$ est un préordre*.

Contre-exemple à l'anti-symétrie. Sur la figure 3.3 on peut voir deux graphes tels que $G \preceq_{\mathcal{S}} H$ et $H \preceq_{\mathcal{S}} G$ bien que les deux graphes soient distincts. □

Cette dernière conséquence pose problème car elle révèle une difficulté à normaliser les graphes et à prouver rapidement qu'ils sont équivalents, c'est-à-dire qu'ils se projettent l'un dans l'autre. Chein and Mugnier [1992] proposent un traitement bien plus complet des classes d'équivalences pour les graphes conceptuels : les graphes irredondants.

Définition 3.18 (\mathcal{S} -graphe irredondant). Un \mathcal{S} -graphe est dit irredondant s'il ne se projette pas dans un de ses sous- \mathcal{S} -graphes propre.

Le graphe H de la figure 3.3 est effectivement irredondant. On montre qu'il existe un unique \mathcal{S} -graphe irredondant dans toute classe d'équivalence. Celui-ci peut alors jouer le rôle du représentant de la classe. L'ensemble de ces graphes muni de la relation de spécialisation forme alors un treillis.

3.5 Sémantique

Contrairement au chapitre précédent, la sémantique est donnée par une traduction vers le calcul des prédicats.

Définition 3.19 (S-formule associée à un S-graphe). Soit un support $\mathcal{S} = \langle T_C, T_R, M, ::, B \rangle$, on considère un calcul des prédicats basé sur un ensemble de constantes $K \supseteq M - \{*, \emptyset\}$, un ensemble de symboles de prédicats $P \supseteq T_C \cup T_R$ (les prédicats associés à T_C étant monadiques – d’arité 1 – et ceux associés à T_R étant polyadiques) et un ensemble de variables V . Soit un S-graphe $\langle R, C, U, l \rangle$ on définit :

$$\begin{aligned} \iota : C &\longrightarrow V \cup K \text{ injective sur } V \\ c &\longmapsto \begin{cases} x \in V \text{ si } \text{ref}(c) = * \\ \text{ref}(c) \in K \text{ sinon} \end{cases} \\ \phi : C \cup R \cup \mathcal{S}\text{-graphe} &\longrightarrow \mathcal{CP}1^* \\ c &\longmapsto \text{type}(c)(\iota(c)) \\ r &\longmapsto \text{type}(r)(\iota(G_1(r)), \dots, \iota(G_{\text{degré}(r)}(r))) \\ G &\longmapsto \exists_{c \in C; \text{ref}(c)=*} \iota(c) \bigwedge_{c \in C} \phi(c) \wedge \bigwedge_{r \in R} \phi(r) \end{aligned}$$

Exemple 3.9 (S-formule associée à un S-graphe). Le graphe de l’exemple initial se traduit par :

$$\begin{aligned} &\exists x, y, z, w; \\ &Singe(paul) \wedge Manger(x) \wedge Noix(y) \wedge Cuillere(z) \wedge Coquille(w) \wedge Agent(paul, x) \\ &\quad \wedge Objet(y, x) \wedge Partie(w, y) \wedge Matériel(w, z) \wedge Instrument(z, x) \end{aligned}$$

Le graphe G de la figure 3.3 se traduit par :

$$\exists x, y; sol(g) \wedge polygone(x) \wedge carré(y) \wedge sur(x, y) \wedge sur(y, g)$$

et le graphe H par :

$$\begin{aligned} &\exists x, y, z, w; polygone(x) \wedge figure(y) \wedge carré(z) \wedge rectangle(w) \wedge sol(g) \\ &\quad \wedge sur(x, z) \wedge sur(z, g) \wedge sur(y, w) \wedge sur(w, g) \end{aligned}$$

Définition 3.20 (Axiomes associés au support). Soit un support $\mathcal{S} = \langle T_C, T_R, M, ::, B \rangle$ on lui associe l’ensemble d’axiomes $A_{\mathcal{S}}$ composé de $\forall x, t(x) \Rightarrow t'(x)$ pour tout $t, t' \in T_C$ tels que $t \leq t'$ et de $\forall x_1, \dots, x_n, r(x_1, \dots, x_n) \Rightarrow r'(x_1, \dots, x_n)$ pour tout $r, r' \in T_R^n$ tels que $t \leq_R^n t'$.

Exemple 3.10 (Axiomes associés à un support). Les axiomes associés au support de l'exemple 3.3 (voir figure 3.2) sont :

$$\begin{array}{ll}
\forall x, \text{objet}(x) \Rightarrow \text{top}(x) & \forall x, \text{figure}(x) \Rightarrow \text{objet}(x) \\
\forall x, \text{polygone}(x) \Rightarrow \text{figure}(x) & \forall x, \text{sol}(x) \Rightarrow \text{objet}(x) \\
\forall x, \text{polygone} - \text{regulier}(x) \Rightarrow \text{polygone}(x) & \forall x, \text{parallélogramme}(x) \Rightarrow \text{polygone}(x) \\
\forall x, \text{losange}(x) \Rightarrow \text{polygone} - \text{regulier}(x) & \forall x, \text{losange}(x) \Rightarrow \text{parallélogramme}(x) \\
\forall x, \text{rectangle}(x) \Rightarrow \text{parallélogramme}(x) & \forall x, \text{carré}(x) \Rightarrow \text{losange}(x) \\
\forall x, \text{carré}(x) \Rightarrow \text{rectangle}(x) & \forall x, \text{bottom}(x) \Rightarrow \text{carré}(x) \\
\forall x, \text{bottom}(x) \Rightarrow \text{sol}(x) &
\end{array}$$

Définition 3.21 (\mathcal{S} -formule). Une \mathcal{S} -formule est une formule pouvant être associée par ϕ à un \mathcal{S} -graphe.

Conséquence 3.11. L'ensemble des \mathcal{S} -formules est $\phi(\Sigma(\check{B} \cup \{[\top : *]\}))$.

Théorème 3.12 (Correction).

$$\text{Si } G \preceq_{\mathcal{S}} H \text{ alors } A_{\mathcal{S}} \models \phi(G) \Rightarrow \phi(H).$$

Démonstration. Soit $(H =)H^0 S_0 H^1 S_1 \dots S_{k-1} H^k (= G)$ une séquence de spécialisations élémentaires. On montre que pour chacune, la formule $\phi(H^{i+1}) \Rightarrow \phi(H^i)$ est une conséquence de $A_{\mathcal{S}}$.

Somme disjointe (H^i, H') $\phi(H^i) = \exists x_1, \dots, x_n, \psi$ et $\phi(H') = \exists x_{n+1}, \dots, x_m, \psi'$ alors $\phi(H^{i+1}) = \exists x_1, \dots, x_n, x_{n+1}, \dots, x_m \psi \wedge \psi'$ (car les variables sont engendrées par des nœuds marqués universellement et que les deux graphes ne partagent aucun nœuds) et, par conséquent, $\models \phi(H^{i+1}) \Rightarrow \phi(H^i)$.

Suppression (r, r') avec $\text{type}(r) = \text{type}(r')$ et $\forall j \in 1 \dots \text{degré}(r), H_j^i(r) = H_j^i(r')$. $\phi(H^i)$ contient deux occurrences de $\text{type}(l(r))(\iota(H_1^i(r)), \dots, \iota(H_{\text{degré}(r)}^i(r)))$ alors que $\phi(H^{i+1})$ n'en contient qu'une. Mais, $\models \phi(r) \wedge \phi(r') \Rightarrow \phi(r)$.

Restriction (c) Il faut distinguer entre la restriction de marqueur et la restriction de type (elles peuvent apparaître simultanément mais sont valides pour des raisons différentes).

- $l^i(c) = \langle t, * \rangle$ et $l^{i+1}(c) = \langle t, a \rangle$. $\phi(H^i)$ contient $t(x)$ et $\phi(H^{i+1})$ contient $t(a)$ et x est remplacé par a dans toutes ses occurrences dans $\phi(H^{i+1})$. On a bien $\models \phi(H^{i+1}) \Rightarrow \phi(H^i)$ car ce qui est vrai de tous les x est aussi vrai pour une constante (règle de substitution).
- $l^i(c) = \langle t, m \rangle$ et $l^{i+1}(c) = \langle t', m \rangle$ de sorte que $t' \leq t$. $\phi(H^i)$ contient $t(\iota(m))$, $\phi(H^{i+1})$ contient $t'(\iota(m))$ à la place, mais $A_{\mathcal{S}}$ contient $\forall x, t'(x) \Rightarrow t(x)$. Par conséquent, $A_{\mathcal{S}} \models \phi(H^{i+1}) \Rightarrow \phi(H^i)$.

Restriction (r) Ici, un seul type de restriction est possible, $l^i(r) = t$ et $l^{i+1}(r) = t'$ de sorte que $t' \leq_R t$. $\phi(H^i)$ contient $t(\iota(x_1), \dots, \iota(x_n))$, $\phi(H^{i+1})$ contient $t'(\iota(x_1), \dots, \iota(x_n))$ à la place, mais $A_{\mathcal{S}}$ contient $\forall x_1, \dots, x_n, t'(x_1, \dots, x_n) \Rightarrow t(x_1, \dots, x_n)$. Par conséquent, $A_{\mathcal{S}} \models \phi(H^{i+1}) \Rightarrow \phi(H^i)$.

Fusion (c, c') Si la fusion a porté sur des nœuds c et c' dont les étiquettes respectives sont :

$\langle t, a \rangle$ alors $\phi(H^{i+1}) = \phi(H^i)$ à une répétition de conjoints prêt ;
 $\langle t, * \rangle$ alors, si la variable associée c (resp. c') est x (resp. x'), il s'en suit que $\phi(H^{i+1}) = \phi(H^i)[x/y, x'/y]$.

On a bien, par conséquent, $\models \phi(H^{i+1}) \Rightarrow \phi(H^i)$.

Comme \Rightarrow est transitive, on obtient $A_S \models \phi(G) \Rightarrow \phi(H)$. \square

Définition 3.22 (\mathcal{S} -substitution). Soient O_C (resp. O_R) l'ensemble des occurrences de formules atomiques prédiquées par des éléments de T_C (resp. T_R). Une \mathcal{S} -substitution σ est une application telle que :

$$\begin{aligned} \text{Si } k \in K \quad \sigma(k) &= k \\ \text{Si } x \in V \quad \sigma(x) &\in V \cup K \\ \text{Si } t(x) \in O_C \quad \sigma(t(x)) &= t'(\sigma(x)); t' \leq t \\ \text{Si } r(x_1, \dots, x_n) \in O_R \quad \sigma(r(x_1, \dots, x_n)) &= r'(\sigma(x_1), \dots, \sigma(x_n)); r' \leq_R r \end{aligned}$$

Soit g une \mathcal{S} -formule, $\sigma(g)$ est obtenue en remplaçant tout terme par sa substitution et en supprimant les quantifications superflues.

Définition 3.23 (\mathcal{S} -substitution valide). Une \mathcal{S} -substitution σ est valide pour une \mathcal{S} -formule g si et seulement si pour tout $r(x_1, \dots, x_n)$ dans $\sigma(g)$,

$$\forall i \in 1 \dots \text{degrés}(r), \exists t(x_i) \text{ dans } \sigma(g) \text{ tel que } t \leq \beta_i(r)$$

Conséquence 3.13. Les \mathcal{S} -substitutions valides sont des opérations internes de l'ensemble des \mathcal{S} -formules.

Exemple 3.11 (\mathcal{S} -substitution). La formule de l'exemple 3.9 étant notée f on peut noter par $\sigma = [\text{Noix}/\text{NoixGrenobloise}, \text{Manger}/\text{Déguster}, w/k]$ une substitution telle que :

$$\begin{aligned} \sigma(f) &= \exists x, y, z, k; \\ &\text{Singe}(\text{paul}) \wedge \text{Déguster}(x) \wedge \text{NoixGrenobloise}(y) \wedge \text{Cuillère}(z) \wedge \text{Coquille}(k) \\ &\wedge \text{Agent}(\text{paul}, x) \wedge \text{Objet}(y, x) \wedge \text{Partie}(k, y) \wedge \text{Matériel}(k, z) \wedge \text{Instrument}(z, x) \end{aligned}$$

De même, si l'on applique la substitution $\sigma = [x/\dot{x}, y/\dot{y}, z/\dot{z}, w/\dot{w}]$ au $\phi(H)$ de l'exemple 3.9 on obtient :

$$\begin{aligned} \text{polygone}(\dot{x}) \wedge \text{figure}(\dot{x}) \wedge \text{carré}(\dot{y}) \wedge \text{rectangle}(\dot{y}) \wedge \text{sol}(g) \\ \wedge \text{sur}(\dot{x}, \dot{y}) \wedge \text{sur}(\dot{y}, g) \wedge \text{sur}(\dot{x}, \dot{y}) \wedge \text{sur}(\dot{y}, g) \end{aligned}$$

Proposition 3.14. Soient G et H deux \mathcal{S} -graphes en forme normale, il existe une projection de H vers G si et seulement si il existe une \mathcal{S} -substitution valide de $\phi(H)$ vers $\phi(G)$.
(réécrite) : $\forall H, G. G \preceq_S H$ ssi $\exists \sigma; \sigma(\phi(H)) \subseteq \phi(G)$

Esquisse de démonstration. \Rightarrow S'il existe une projection, c'est que l'on peut isoler G' comme l'image de H par cette projection. Les contraintes portant sur la substitution sont celles qui concernent les étiquettes des concepts et des relations. Ce sont exactement

les contraintes qui pèsent sur la substitution. Cependant, si H n'est pas en forme normale, il se peut que deux nœuds avec le même marqueur individuel se projettent dans le même nœud de G . Les deux nœuds étant traduits en deux termes distincts (même s'ils sont équivalents) ne pourront être transformés en un même terme par une simple substitution (même s'ils sont identiques). Il se peut alors qu'il n'y ait pas de substitution alors que les deux formules sont logiquement équivalentes.

⇐ S'il existe une telle substitution, alors il est possible d'associer à chaque nœud de H un nœud de G' (donc de G) de telle sorte que son étiquette puisse se substituer en celle du nœud correspondant. Comme les contraintes pesant sur la substitution valide sont celles qui pèsent sur la projection, la proposition est démontrée. \square

Lemme 3.15. *Pour toutes \mathcal{S} -formules h et g , si $A_{\mathcal{S}} \models g \Rightarrow h$ alors $\exists \sigma; \sigma(h) \in g$.*

Démonstration. Soient $h = \phi(H) = \exists x_1 \dots x_p, (t_1^h \wedge \dots t_m^h \wedge r_1^h \wedge \dots r_n^h)$ avec $p \leq m$ et $g = \phi(G) = \exists y_1 \dots y_q, (t_1^g \wedge \dots t_f^g \wedge r_1^g \wedge \dots r_k^g)$ avec $q \leq f$ où t^h et t^g (resp. r^h et r^g) sont les atomes associés aux concepts (resp. relations). $A_{\mathcal{S}} \models g \Rightarrow h$ équivaut à $A_{\mathcal{S}} \wedge g \wedge \neg h \models \square$. La forme clausale* (conjonction de disjonction d'atomes ou de leur négation) de $A_{\mathcal{S}} \wedge g \wedge \neg h$ contient les formules suivantes :

- la clause associée à $\neg h$ qui est $\neg t_1^h \vee \dots \neg t_m^h \vee \neg r_1^h \vee \dots \neg r_n^h$ contenant les variables $x_1 \dots x_p$;
- les clauses de $A_{\mathcal{S}}$ qui sont déjà sous forme clausale ($\neg t'(x) \vee t(x)$ et $\neg r'(x_1, \dots, x_n) \vee r(x_1, \dots, x_n)$);
- les clauses associées à g ($C(g)$) qui consistent à chaque fois en un atome où les variables sont remplacées par des *constantes de Skolem* ou *constantes témoins** (c'est-à-dire des constantes introduites pour l'occasion et qui n'apparaissent nulle part ailleurs; elles seront notées à l'aide d'un point).

Comme on est en présence de clauses de Horn*, la clause vide peut être obtenue par *unit-resolution** à partir de cet ensemble de clauses. La résolution ne pouvant s'appliquer qu'entre :

- un $\neg t_i^h$ de $\neg h$ et
 - un t_j^g de $C(g)$ (avec la substitution σ_i) ou
 - un $t_1(v)$ obtenu par une chaîne de résolutions entre un $t_j^g(v)$ de $C(g)$ et des $t_k(x) \wedge \neg t'_{k+1}(x)$ de $A_{\mathcal{S}}$ (avec $t_j^g = t'_{max}$). On y associe alors une substitution $\sigma_i = [t'_{max}/t_1]o\sigma$ où σ est la substitution nécessaire à l'unification.
- un $\neg r_i^h$ de $\neg h$ et
 - un r_j^g de $C(g)$ (avec la substitution σ_{m+i}) ou
 - un r_p obtenu par résolution entre un r_j^g de $C(g)$ et un $r_p \wedge \neg r_j^g$ de $A_{\mathcal{S}}$ (et comme T_R est fini et ordonné cela s'arrête).
 - un $r_1(\bar{v})$ obtenu par une chaîne de résolutions entre un $r_j^g(\bar{v})$ de $C(g)$ et des $r_k(\bar{x}) \wedge \neg r'_{k+1}(\bar{x})$ de $A_{\mathcal{S}}$ (avec $r_j^g = r'_{max}$). On y associe alors une substitution $\sigma_{m+i} = [r'_{max}/r_1]o\sigma$ où σ est la substitution nécessaire à l'unification.

On a donc une preuve quasi-linéaire (en peigne) dont le squelette est ordonnée par $t_1^h, \dots, t_m^h, r_1^h, \dots, r_n^h$. La composition de ces substitutions est une substitution telle qu'il existe une partie g' de g (celle intervenant dans la preuve) telle que $\sigma_1 \circ \dots \circ \sigma_{m+n}(h) = g'$ \square

Cette sémantique et sa preuve de complétude fournissent une méthode opérationnelle pour les tests de projection.

Exemple 3.12. Illustrons cette démonstration sur les graphes H et G de l'exemple 3.4 (figure 3.3). A_S est donné dans l'exemple 3.10 et $\phi(H)$ et $\phi(G)$ dans l'exemple 3.9. $\neg\phi(G)$ s'exprime donc par :

$$\begin{aligned} \exists x, y, z, w; \neg\text{polygone}(x) \vee \neg\text{figure}(y) \vee \neg\text{carré}(z) \vee \neg\text{rectangle}(w) \vee \neg\text{sol}(g) \\ \vee \neg\text{sur}(x, z) \vee \neg\text{sur}(z, g) \vee \neg\text{sur}(y, w) \vee \neg\text{sur}(w, g) \end{aligned}$$

Pour chacun de ces atomes, montrons comment il est résolu :

- $\neg\text{sol}(g)$ avec $\text{sol}(g)$;
- $\neg\text{polygone}(x)$ avec $\text{polygone}(\dot{x})$ ($\sigma = [x/\dot{x}]$);
- $\neg\text{figure}(y)$ avec $\text{polygone}(\dot{x})$ ($\sigma = [y/\dot{x}]$) via une résolution intermédiaire avec $\neg\text{polygone}(x) \vee \text{figure}(x)$ (avec $\sigma = [\text{figure}/\text{polygone}]$);
- $\neg\text{carré}(z)$ avec $\text{carré}(\dot{y})$ ($\sigma = [z/\dot{y}]$);
- $\neg\text{rectangle}(w)$ avec $\text{carré}(\dot{y})$ ($\sigma = [w/\dot{y}]$) via une résolution intermédiaire avec $\neg\text{carré}(x) \vee \text{rectangle}(x)$ (avec $\sigma = [\text{rectangle}/\text{carré}]$);
- $\neg\text{sur}(x, z)$ avec $\text{sur}(\dot{x}, \dot{y})$ ($\sigma = [x/\dot{x}, z/\dot{y}]$);
- $\neg\text{sur}(z, g)$ avec $\text{sur}(\dot{y}, g)$ ($\sigma = [z/\dot{y}]$);
- $\neg\text{sur}(y, w)$ avec $\text{sur}(\dot{x}, \dot{y})$ ($\sigma = [y/\dot{x}, w/\dot{y}]$);
- $\neg\text{sur}(w, g)$ avec $\text{sur}(\dot{y}, g)$ ($\sigma = [w/\dot{y}]$);

Les substitutions étant compatibles, c'est une preuve valide.

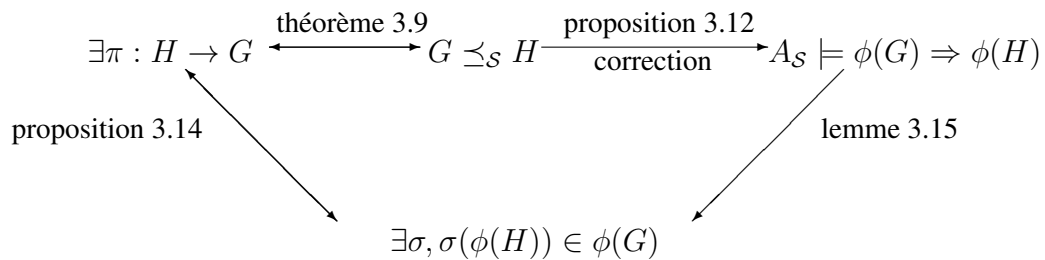
Théorème 3.16 (Complétude).

$$\forall H, G, G \preceq_S H \text{ ssi } A_S \models \phi(G) \Rightarrow \phi(H)$$

Démonstration. \Rightarrow La première partie a été démontrée par le théorème de correction (3.12).

\Leftarrow Le lemme 3.15 indique que si $A_S \models \phi(G) \Rightarrow \phi(H)$ alors il existe une substitution permettant de transformer $\phi(H)$ en une partie de $\phi(G)$ et la proposition 3.14 indique que ceci équivaut à dire que $G \preceq_S H$ (en utilisant le théorème 3.9). \square

L'ensemble des propositions produites s'articule ainsi :



Une sémantique ensembliste a été ébauchée dans [Mugnier and Chein, 1996].

3.6 Complexité

Problème (Projection). Existe-t-il une \mathcal{S} -projection de H vers G ?

Proposition 3.17 (Complexité de la projection). *Le problème de la projection est NP-complet*.*

Esquisse de démonstration. Le problème peut être vérifié en temps linéaire comme on l'a vu plus haut, il est donc dans NP. Par ailleurs, on peut transformer la recherche d'un morphisme de graphe orienté dans le problème de la projection en temps linéaire (en transformant tout nœud en un nœud concept étiqueté par $\langle \top, * \rangle$ et tout arc en un nœud relation étiqueté r dont le premier arc va vers le nœud concept correspondant à la source et le second à celui correspondant à la cible). Le problème de morphisme de graphe orienté contient le problème de la clique (un graphe $\langle N, E \rangle$ contient-il un sous-ensemble N' de N de taille supérieure ou égale à k telle que $N' \times N' \subseteq E$) qui lui est NP-complet. \square

Problème (Projection injective). H est-il \mathcal{S} -isomorphe à un \mathcal{S} -sous-graphe de G ?

Proposition 3.18 (Complexité de la projection injective). *Le problème de la projection injective est NP-complet*.*

Problème (Isoprojection). H est-il \mathcal{S} -isomorphe à G ?

Proposition 3.19 (Complexité de l'isoprojection). *Le problème de l'isoprojection est NP-complet*.*

Ces problèmes se résolvent en temps polynomial si les graphes sont des arbres et T_R et T_C sont sans structure.

Quelques liens avec la combinatoire dans [Mugnier and Chein, 1996].

3.7 Conclusion

On a présenté le formalisme des graphes conceptuels qui présente les particularités suivantes :

- il est fondé directement sur la structure mathématique de graphes et non sur un langage linéaire ;
- sa sémantique a été donnée par traduction et non par la théorie des modèles ;
- le langage est relativement expressif ;
- le langage présenté ne contient pas de cycles (car les relations sont contraintes par les concepts mais non l'inverse) : cela contribue à sa décidabilité et à une faible complexité ;
- il peut être étendu, en particulier vers la notion de contexte (voir figure 3.4).

On peut critiquer l'impossibilité d'avoir des graphes contenant des circuits.

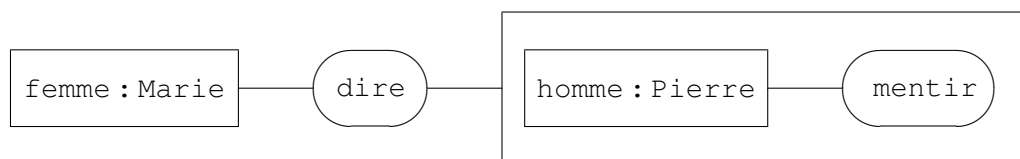


FIGURE 3.4 : Un exemple de graphe emboîté pouvant signifier « Marie dit que Pierre ment ».

3.8 Exercices

Exercice 3.1 (*). Soient les graphes conceptuels de la figure 3.5 : Refaire tous les exemples qui ont été donnés dans le cours en terme d'application des algorithmes :

1. Donnez un support plausible pour ces graphes ;
2. Dites quel graphe se projette dans quel autre ;
3. Donnez une suite d'opérations de spécialisation permettant de passer de H à G ;
4. Donnez l'ensemble d'axiomes associé au support ;
5. Donnez $\phi(H)$;
6. Donnez une substitution permettant de passer de $\phi(H)$ à une sous-partie de $\phi(G)$;
7. Fournissez une preuve par réfutation de $A_S \models \phi(G) \Rightarrow \phi(H)$.

Exercice 3.2 (*). On considère les graphes conceptuels de la figure 3.6 :

1. Donnez une signification en langage naturel à ces graphes.
2. Dites quels graphes se projettent dans quels autres graphes.
3. Donnez la suite des opérations de généralisation nécessaires pour passer de K à G .

Exercice 3.3 ()**. On se propose de considérer la projection entre des graphes dont la signature (graphe étoile) des relations n'est qu'incomplètement respectée (cette opération permet d'obtenir pour les \mathcal{S} -graphes le comportement présenté brièvement dans [Sowa, 1984] pp93-94). Pour cela, on va reconstruire progressivement les résultats obtenus en cours. Dans le cadre de l'exercice précédent, on se propose de redéfinir B (l'ensemble des graphes-étoile ou base) de telle sorte que le graphe étoile correspondant à « vendre » ait quatre voisins. On veut, par contre, pouvoir projeter K dans I en introduisant des arêtes autorisées par B , mais manquantes dans la description initiale (celle de K). Ceci est illustré par la figure 3.7.

1. Donnez dans ce cadre une nouvelle définition d'un \mathcal{S} -graphe.
2. Donnez la nouvelle définition de \mathcal{S} -projection.
3. Modifiez les règles de spécialisation de \mathcal{S} -graphes. L'ensemble des \mathcal{S} -graphes correspond-il encore à $\Sigma(B \cup \{[\top : *]\})$?
4. Donnez la séquence d'opérations de spécialisation permettant de passer de K à I .
5. Vérifiez l'équivalence entre spécialisation et projection. Indépendamment de la démonstration, on attachera de l'importance à ce que les deux notions correspondent effectivement.

Exercice 3.4 (*)**. On s'attache maintenant à donner une sémantique aux opérations définies plus haut.

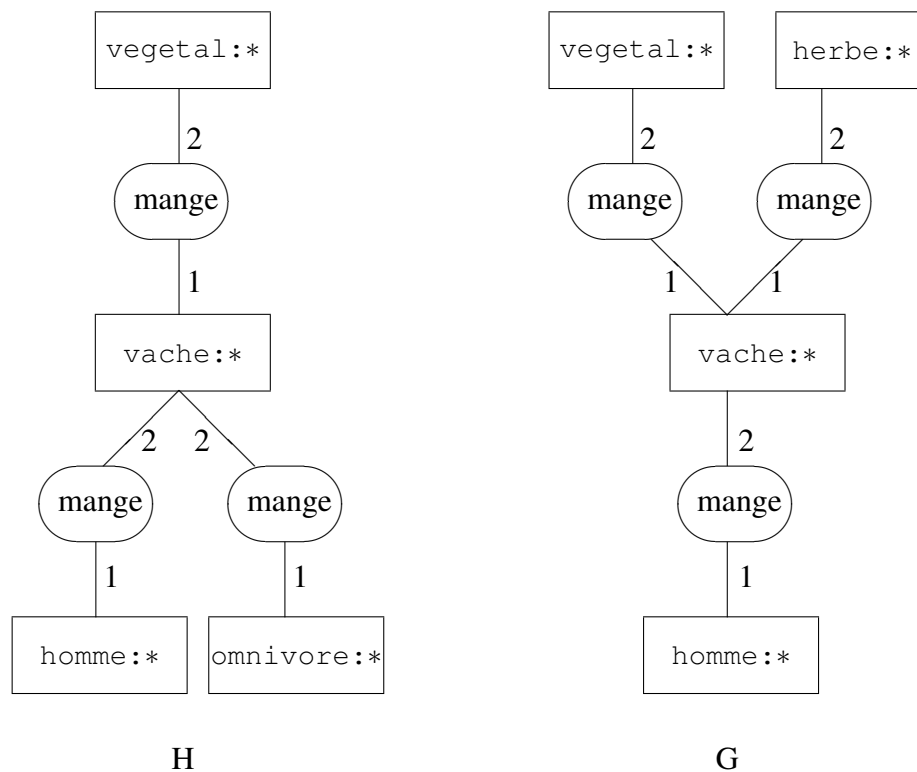


FIGURE 3.5 : Graphes de l'exercice 3.1

1. Donnez intuitivement (avec des mots) le sens de l'utilisation d'une relation incomplète (ou par exemple donnez intuitivement le sens du graphe K dans le nouveau cadre, c'est-à-dire en faisant apparaître les arcs non présents).
2. En quoi la traduction en logique pose-t-elle problème? Quel méthode utiliseriez vous pour y remédier dans le cadre de la logique classique?
3. Modifiez la définition de ϕ de manière à tenir compte des relations incomplètes. Est-il nécessaire de modifier les axiomes associés au support?
4. Donnez l'expression de $\phi(K)$ et $\phi(I)$ et vérifiez votre réponse précédente.
5. L'utilisation d'une logique dont les prédicats sont des \mathcal{OSF} -termes (voir chapitre 4) permettrait-elle de simplifier le travail? Expliquez comment.
6. Qu'est-il nécessaire de modifier dans les démonstrations pour obtenir la correction et la complétude entre spécialisation et implication dans la théorie associée au support?
7. N'y avait-il pas une manière bien plus simple de réaliser tout le travail entrepris jusqu'ici?

Exercice 3.5 (*)**. Généralisez les étiquettes sur les arcs par un ensemble d'étiquette quelconque à la place d'une numérotation. Quel problème cela pose-t-il vis-à-vis de la traduction vers la logique?

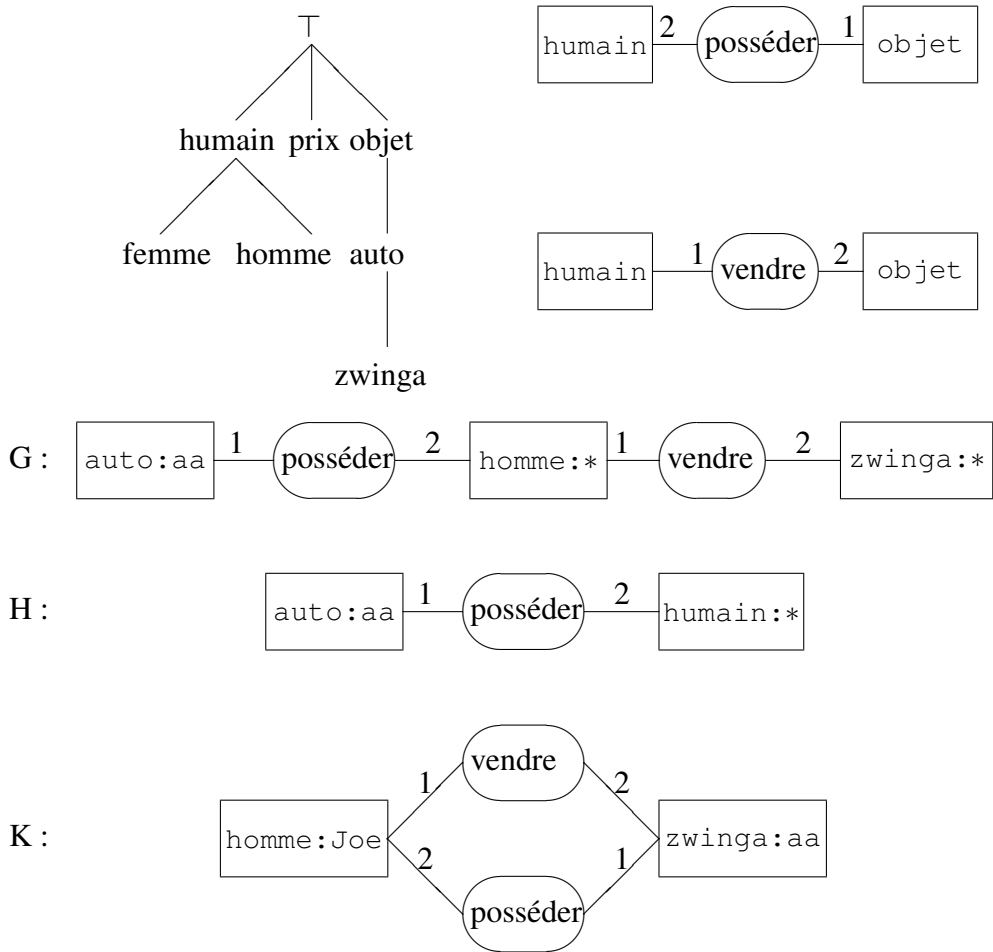
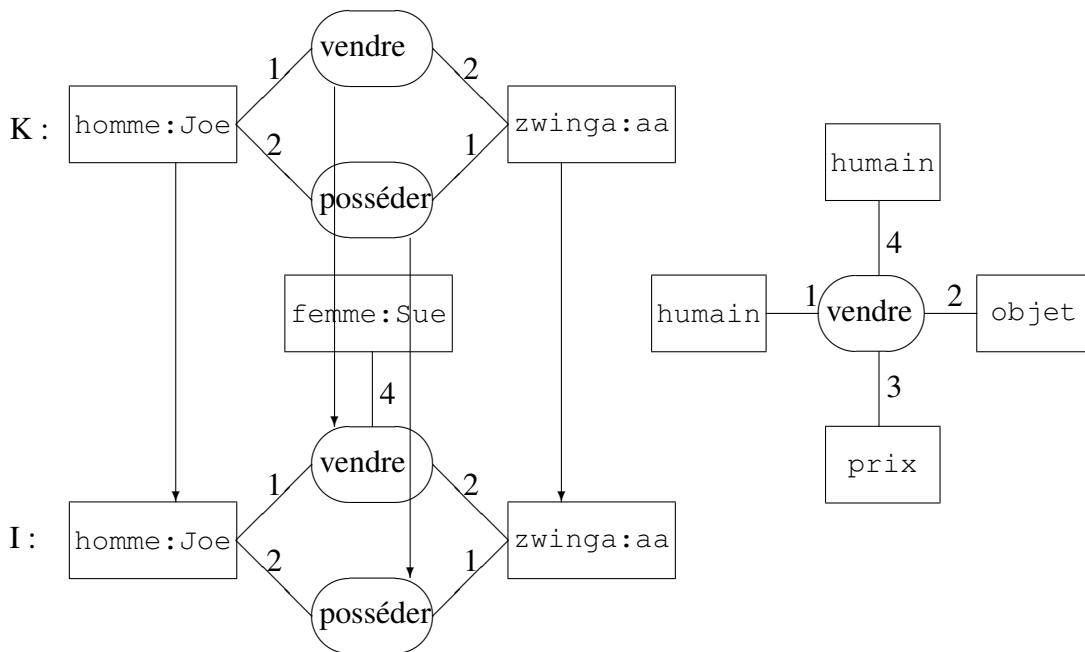


FIGURE 3.6 : Treillis de type, base et \mathcal{S} -graphes G , H et K .

FIGURE 3.7 : Projection partielle de K dans I .

Déclarativité et procéduralité

Premiers éléments

La notion de déclarativité existe tant dans les langages de programmation que dans les langages de représentation de connaissance.

Déclaratif (le quoi) : le programmeur spécifie ce qu'est le résultat (par ses propriétés), l'ordinateur doit le trouver.

Procédural (le comment) : le programmeur spécifie comment obtenir un résultat et l'ordinateur n'a qu'à obéir.

Une même connaissance peut s'énoncer à la fois de manière procédurale et déclarative. Ainsi, pour reprendre l'exemple du §2.1, on peut énoncer que :

- « Une équipe moderne comprend au plus quatre membres et son chef est une femme »,
ou
- « Si une équipe moderne comprend trois hommes alors elle possède un quatrième membre qui est une femme et dirige l'équipe » qui en est dérivé.

Elle a menée dans ces derniers à la fameuse controverse entre le déclaratif et le procédural.

L'important dans cette distinction est que la forme déclarative ne préjuge pas de l'utilisation de la connaissance (elle peut être exploitée de multiples manières) alors que la connaissance procédurale est, en quelque sorte, figée dans une utilisation particulière. On pourra aussi dire que le sens de la première est donnée de manière *dénotationnelle* alors que celui de la seconde ne l'est que de manière *opérationnelle*.

Déclarativité et compilation

Que ce soit en psychologie ou en informatique, on oppose souvent déclarativité de la représentation de connaissance et compilation. En effet, la compilation de la connaissance a pour but de la transformer sous une forme qui lui permette d'être utilisée efficacement. Par conséquent, elle est n'a plus intérêt à être sous une forme indépendante de l'utilisation. Elle n'est donc plus déclarative.

La déclarativité et la loi

Texte no 207 adopté en première lecture « Petite loi » le 9 décembre 1998 par l'assemblée nationale relative au pacte civil de solidarité.

On peut distinguer trois aspects dans le texte qui suit :

Article 1er

Le livre Ier du code civil est complété par un titre XII ainsi rédigé :

«

TITRE XII - DU PACTE CIVIL DE SOLIDARITÉ

Art. 515-1. Un pacte civil de solidarité peut être conclu par deux personnes physiques majeures, de sexe différent ou de même sexe, pour organiser leur vie commune.

Art. 515-2. À peine de nullité, il ne peut y avoir de pacte civil de solidarité :
... »

...

Un aspect déclaratif qui sera le contenu du code civil et un aspect procédural qui sera la manière d'insérer le contenu dans le code civil. On peut remarquer que les deux éléments ne sont pas mélangés. Le premier sert à manipuler le second qui ne fait jamais référence au premier.

Ceci correspond à un certain nombre de langages de représentation de connaissance qui, à l'instar de l'approche fonctionnelle, distinguent les expressions du langage (et leur sémantique) des expressions sur le langage (et leur sémantique).

Chapitre 4

Ψ -termes

LE FORMALISME DES Ψ -TERMES est exposé ici en s'appuyant principalement sur [Aït-Kaci and Podelski, 1993]. Les Ψ -termes sont à la base du langage LIFE (pour “Logic, inheritance, functions and equations”) qui intègre la programmation :

**logique
fonctionnelle
avec contraintes**

Il est aussi partiellement inspiré des travaux en traitement des langues naturelles avec les grammaires de traits (“feature grammars” [Smolka, 1992]).

Références

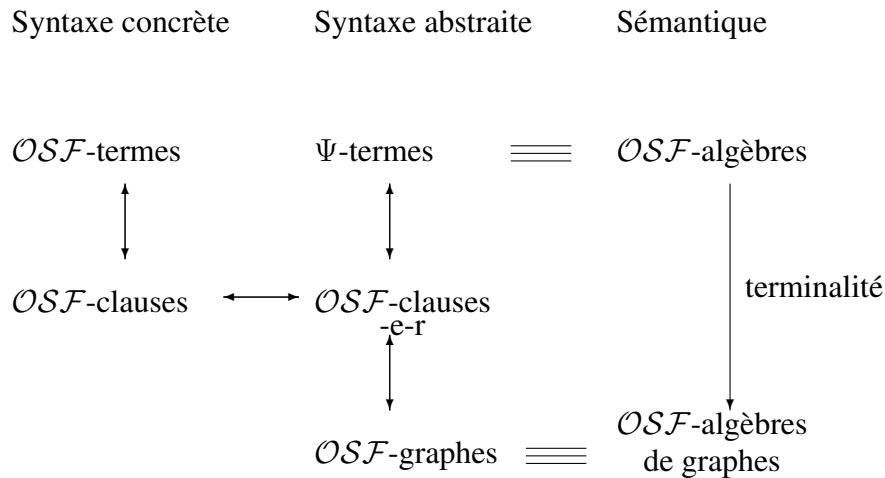
Hassan Aït-Kaci and Andreas Podelski. Towards a meaning of life. *Journal of logic programming*, 16(3):195–234, 1993

Gerd Smolka. Feature constraints logics for unification grammars. *Journal of logic programming*, 12(1):324–343, 1992

Aperçu

4.1	Exemple introductif	78
4.2	Syntaxe	79
4.3	Interprétation	87
4.4	Catégorie OSF	92
4.5	Conclusion	95
4.6	Exercices	96

Globalement, le plan du cours est le suivant :



4.1 Exemple introductif

Exemple 4.1 (Ψ -terme).

```

person( name => id( last => X:string ),
        home => address( city => cityname ),
        father => person( name => id( last => X ) ) )

student( name => id( first => string, last => X:string ),
         home => Y:address( city => paris ),
         father => person( name => id( last => X ),
                          home => Y ) )

```

Cet exemple décrit une personne comme ayant un nom qui est un identifiant qui lui-même contient une dernière partie qui est une chaîne de caractères, ayant ces coordonnées à une adresse dont la ville est un nom de ville et ayant un père qui est une personne dont l'identifiant correspondant au nom à la même dernière partie.

On veut pouvoir déduire que $\text{student} \leq \text{person}$ à condition que $\text{paris} \leq \text{cityname}$.

À noter :

- on retrouve des sortes, des attributs et des variables ;
- les descriptions sont ici récursives ;
- les Ψ -termes sont une généralisation des termes du premier ordre (dans le sens où $p(t_1, \dots, t_n)$ peut être représenté par $p(1 \rightarrow t_1, \dots, n \rightarrow t_n)$);
- on retrouve une dualité entre subsomption et unification ;

4.2 Syntaxe

4.2.1 OSF-termes

L'acronyme OSF signifie "order-sorted features"; il exprime que les objets manipulés sont munis de traits ("features") et liés à des classes ("sort") organisés par une relation d'ordre ("order") qui correspondra à la taxonomie des logiques terminologiques par exemple.

Définition 4.1 (OSF-signature). Une OSF-signature est un quadruplet $\langle S, \leq, \wedge, F \rangle$ tel que :

- S est un ensemble de types contenant \top et \perp ;
- \leq est un ordre partiel décidable sur S tel que \top (resp. \perp) en soit le plus grand (resp. plus petit) élément;
- $\langle S, \leq, \wedge \rangle$ soit un demi-treillis inférieur (avec \wedge pour plus grand minorant);
- F est un ensemble de traits;

Exemple 4.2 (OSF-signature). Pour l'exemple 4.1, une signature pourrait être :

- $S = \{\text{person, id, string, paris, address, cityname, student, } \top, \perp\}$;
- $\leq = \{\langle \text{paris, cityname} \rangle, \langle \text{student, person} \rangle\}$ (si l'on omet les relations triviales impliquant \perp et \top);
- \wedge est simplement induit par l'ordre partiel;
- $F = \{\text{name, first, home, father, city, last}\}$.

Définition 4.2 (OSF-terme). Soit V un ensemble énumérable de variables, un OSF-terme est une expression de la forme :

$$\psi = X : s(l_1 \rightarrow \psi_1, \dots, l_n \rightarrow \psi_n)$$

où :

- $n \geq 0$;
- $X \in V, s \in S, l_1, \dots, l_n \in F$;
- ψ_1, \dots, ψ_n sont des OSF-termes;

Définition 4.3 (Syntaxe des OSF-termes).

$\langle \text{term} \rangle ::= \langle \text{variable} \rangle ' : ' \langle \text{sorte} \rangle ' (' \langle \text{feature} \rangle^* ') '$

$\langle \text{feature} \rangle ::= \langle \text{featurename} \rangle \rightarrow \langle \text{term} \rangle$

Définition 4.4 (Variables d'un OSF-terme).

$$\text{Var}(X : s(l_1 \rightarrow \psi_1, \dots, l_n \rightarrow \psi_n) = \{X\} \cup \bigcup_{i=1}^n \text{Var}(\psi_i)$$

Exemple 4.3 (OSF-terme).

```
X:person( name => N:T( first => F:string ),
          name => M:id( last => S:string ),
          spouse => P:person( name => I:id( last => S:T ),
                              spouse => X:T ) )
```

pour lequel $Var(\psi) = \{X, N, F, M, S, P, I\}$. On peut remarquer sur l'exemple que les \mathcal{OSF} -termes permettent la redondance d'attributs (`name`) et l'utilisation de variables (`M` et `N` pour le trait `name`) et de types (`T` et `id` pour le trait `name`) différents.

Définition 4.5 (Ψ -terme ou \mathcal{OSF} -terme en forme normale). Un \mathcal{OSF} -terme $\psi = X : s(l_1 \rightarrow \psi_1, \dots, l_n \rightarrow \psi_n)$ est en forme normale si :

- $s \neq \perp$;
- l_1, \dots, l_n sont distincts deux à deux ;
- ψ_1, \dots, ψ_n sont en forme normale.
- pour tout $Y \in Var(\psi)$, il existe au plus une occurrence de Y dans laquelle il est variable racine d'un \mathcal{OSF} -terme non trivial (c'est-à-dire différent de $Y : \top$).

Exemple 4.4 (Ψ -terme).

```
X:person( name => id( first => string, last => S ),
          spouse => person( spouse => X,
                           name => id( last => S:string ) ) )
```

est une forme normale de l' \mathcal{OSF} -terme de l'exemple 4.3.

Définition 4.6 (Ψ -terme ordonné). Un Ψ -terme $\psi = X : s(l_1 \rightarrow \psi_1, \dots, l_n \rightarrow \psi_n)$ est dit ordonné (par un ordre total \ll sur F) ssi

- pour tout $Y \in Var(\psi)$, l'occurrence de Y dans laquelle il est variable racine d'un Ψ -terme non trivial est la première ;
- $l_1 \ll l_2 \ll \dots \ll l_n$;
- ψ_1, \dots, ψ_n sont des Ψ -ordonnés.

Conséquence 4.1. *Il existe un unique Ψ -terme ordonné correspondant à un Ψ -terme.*

Exemple 4.5 (Ψ -terme ordonné).

```
X:person( name => id( first => string, last => S:string ),
          spouse => person( name => id( last => S ),
                           spouse => X ) )
```

est le Ψ -terme ordonné correspondant à celui de l'exemple 4.4 qui n'est pas ordonné.

4.2.2 \mathcal{OSF} -clauses

Les termes sont ici exprimés sous forme de clauses de manière à pouvoir leur appliquer des opérations de simplifications.

Définition 4.7 (*OSF*-contrainte). Une *OSF*-contrainte est une expression d'une des formes suivantes :

- $X : s$,
- $X \doteq Y$,
- $X.l \doteq Y$

avec $X, Y \in V, l \in F$ et $s \in S$.

Définition 4.8 (*OSF*-clause). Une *OSF*-clause est une conjonction (qui peut être vide) d'*OSF*-contraintes.

Définition 4.9 (Dissolution d'un *OSF*-terme). Soit un *OSF*-terme $\psi = X : s(l_1 \rightarrow \psi_1, \dots, l_n \rightarrow \psi_n)$, sa dissolution $\phi(\psi)$ est l'*OSF*-clause définie par :

$$\phi(\psi) = X : s \wedge X.l_1 \doteq \text{root}(\psi_1) \wedge \dots \wedge X.l_n \doteq \text{root}(\psi_n) \wedge \phi(\psi_1) \wedge \dots \wedge \phi(\psi_n)$$

avec $\text{root}(X : s(l_1 \rightarrow \psi_1, \dots, l_n \rightarrow \psi_n)) = X$.

Exemple 4.6 (*OSF*-clause). Soit ψ le premier *OSF*-terme de l'exemple 4.3,

$$\begin{aligned} \phi(\psi) = & X : \text{person} \wedge X.\text{name} \doteq N \wedge N : \top \wedge \\ & N.\text{first} \doteq F \wedge F : \text{string} \wedge \\ & X.\text{name} \doteq M \wedge M : \text{id} \wedge M.\text{last} \doteq S \wedge S : \text{string} \wedge \\ & X.\text{spouse} \doteq P \wedge P : \text{person} \wedge \\ & P.\text{name} \doteq I \wedge I : \text{id} \wedge I.\text{last} \doteq S \wedge S : \top \wedge \\ & P.\text{spouse} \doteq X \wedge X : \top \end{aligned}$$

Définition 4.10 (atteignabilité). Soit une *OSF*-clause ϕ on définit une relation binaire $\xrightarrow{\phi}$ (lire ϕ -atteignable) sur $\text{Var}(\phi)$ ainsi :

- $X \xrightarrow{\phi} X$,
- $X \xrightarrow{\phi} Y$ si $X.l \doteq Z \in \phi$ et $Z \xrightarrow{\phi} Y$ ou $X \doteq Y \in \phi$.

On appelle $\dot{\phi}_X$ la clause ϕ restreinte aux variables atteignables depuis X .

Définition 4.11 (*OSF*-clause enracinée). Une *OSF*-clause ϕ est dite enracinée par une variable X (et notée ϕ_X) si $\dot{\phi}_X = \phi$.

Notation (Dissolution enracinée). On désignera par ϕ^R la dissolution enracinée, c'est-à-dire une fonction de dissolution retournant une *OSF*-clause enracinée.

Attention, une *OSF*-clause peut avoir plusieurs *OSF*-termes (et même plusieurs Ψ -termes ordonnés) équivalents. Ainsi l'*OSF*-clause

$$Q : \text{person} \wedge P : \text{person} \wedge Q.\text{son} \doteq P \wedge P.\text{father} \doteq Q$$

correspond-t-elle aux deux Ψ -termes ordonnés :

$$P : \text{person} (\text{father} \Rightarrow Q : \text{person} (\text{son} \Rightarrow P)) \text{ et } Q : \text{person} (\text{son} \Rightarrow P : \text{person} (\text{father} \Rightarrow Q))$$

Définition 4.12 (*OSF-clause résolue*). Une OSF-clause ϕ est dite résolue (solved) si pour toute variable X , ϕ contient :

- au plus une contrainte de sorte $X : s$ (avec $s \neq \perp$);
- au plus une contrainte de trait $X.l \doteq Y$ ($\forall l \in F$);
- aucune contrainte d'égalité.

Exemple 4.7 (*OSF-clause résolue*). La clause de l'exemple 4.6 n'est pas résolue. Elle est enracinée en X et en P . Par contre la clause suivante est résolue :

$$\begin{aligned} X : person \wedge X.name \doteq N \wedge N : id \wedge \\ N.first \doteq F \wedge F : string \wedge \\ N.last \doteq S \wedge S : string \wedge \\ X.spouse \doteq P \wedge P : person \wedge \\ P.name \doteq I \wedge I : id \wedge I.last \doteq S \wedge \\ P.spouse \doteq X \end{aligned}$$

Elle est enracinée en X , mais pas en S ni en P .

Définition 4.13. Soit ϕ_X une OSF-clause résolue et enracinée, on définit l'OSF-terme $\psi(\phi_X)$ par :

$$\psi(\phi_X) = X : s(l_1 \rightarrow \psi(\phi'_{Y_1}), \dots, l_n \rightarrow \psi(\phi'_{Y_n}))$$

où ϕ_X contient $X : s$ (sinon $X : \top$), $X.l_i \doteq Y_i \in \phi_X$ et $\phi' = \phi - \{X : s\}$.

Exemple 4.8 (*OSF-clause résolue et enracinée*). L'OSF-terme correspondant à la clause de l'exemple 4.7 enraciné en X peut-être celui de l'exemple 4.4 ou 4.5.

Proposition 4.2 (**Équivalence entre OSF-clause résolue enracinée et Ψ -terme ordonné**). Étant défini un ordre sur les éléments de F et X une variable, la correspondance entre Ψ -termes ordonnés et OSF-clauses résolues enracinées en X est une bijection. C'est-à-dire que

$$\phi_X = \phi(\psi(\phi_X)) \text{ et } \psi = \psi(\phi(\psi))$$

Démonstration. La différence essentielle entre Ψ -termes ordonnés et OSF-clauses réside d'une part dans la commutativité de la conjonction qui fait que les contraintes des clauses ne sont pas ordonnées (alors que dans le terme ils apparaissent dans un ordre précis) et d'autre part dans le fait que suivant l'ordre dans lequel la clause sera transformée en terme, ce ne sera pas la même occurrence d'un sous-terme qui sera non-triviale. Soit un ordre \ll défini sur F , celui-ci devrait permettre de remédier à cela.

Soit une clause enracinée résolue ϕ_X , on cherche à montrer que $\phi(\psi(\phi_X)) = \phi_X$. On peut procéder par induction sur la structure des formules :

cas de base $\phi(\psi(X : \top)) = \phi(X : \top) = X : \top$.

induction

$$\begin{aligned} & \phi(\psi(X : s \wedge X.l_i \doteq Y_i \wedge \phi')) \\ &= \phi(X : s(l_i \rightarrow \psi(\phi'_{Y_i}))) \\ &= X : s \wedge X.l_i \doteq Y_i \wedge \phi(\psi(\phi'_{Y_i})) \end{aligned}$$

Le résultat intermédiaire est bien un Ψ -terme (car la clause initiale ne contient qu'une seule contrainte $X : s$ et une seule contrainte $X.l = Y$) non forcément ordonné. La clause finale sera donc toujours enracinée en X , ne contiendra aucune contrainte d'égalité car aucune n'est introduite lors de la dissolution et ne contiendra qu'une contrainte de trait $X.l = Y$ pour un X et un l donnés). L'hypothèse d'induction indique qu'il s'agit de la clause initiale.

À l'inverse, soit un Ψ -terme ordonné ψ on cherche à montrer que $\psi(\phi(\psi)) = \psi$. On peut aussi procéder par induction sur la structure des formules :

cas de base $\psi(\phi(X : \top)) = \psi(X : \top) = X : \top$.

induction

$$\begin{aligned} & \psi(\phi(X : s(l_i \rightarrow \phi_i))) \\ &= \psi(X : s \wedge X.l_i \doteq Y_i \wedge \phi') \\ &= X : s(l_i \rightarrow \phi_i) \end{aligned}$$

Dans ce second cas, il faut simplement s'assurer, lors de l'utilisation de ψ , que les traits sont examinés dans l'ordre prescrit par \ll . On aura alors bien un Ψ -terme ordonné, c'est-à-dire ψ . \square

Définition 4.14 (Normalisation d'une \mathcal{OSF} -clause). La normalisation d'une \mathcal{OSF} -clause se fait en lui appliquant jusqu'à clôture les règles* suivantes :

$$\begin{array}{ll} \frac{\phi \wedge X : \perp}{X : \perp} & \text{(type inconsistant)} \\ \frac{\phi \wedge X : s \wedge X : s'}{\phi \wedge X : s \wedge s'} & \text{(intersection de types)} \\ \frac{\phi \wedge X.l \doteq Y \wedge X.l \doteq Y'}{\phi \wedge X.l \doteq Y \wedge Y' \doteq Y} & \text{(substituton de traits)} \\ \frac{\phi \wedge X \doteq Y}{\phi[X/Y] \wedge X \doteq Y} X \in Var(\phi) & \text{(élimination de variables)} \end{array}$$

La forme normale de ϕ est notée $\bar{\phi}$.

Exemple 4.9 (Normalisation d'une \mathcal{OSF} -clause). Soit la clause de l'exemple 4.6, sa normalisation donnera celle de l'exemple 4.7 (à laquelle il faut ajouter un ensemble d'égalités inutiles). Cette normalisation s'opère par les transformations suivantes :

- (intersection de types) sur X ($\top \wedge person = person$)
- (intersection de types) sur S ($\top \wedge string = string$)
- (substitution de traits) sur $name$ $M \doteq N$
- (élimination de variables) M pour N
- (intersection de type) sur N ($\top \wedge id = id$)

On peut remarquer que la normalisation laisse l'ensemble des contraintes d'égalités dans le résultat. Il suffirait de ne pas remettre les égalités dans la règle d'(élimination de variables) pour que le résultat soit une \mathcal{OSF} -clause résolue.

Il faut aussi noter que les clauses de type $X \doteq Y$ ne sont pas équivalentes à $Y \doteq X$. Si c'était le cas, le système de réécriture ne terminerait pas et pourrait passer son temps à

remplacer X par Y et Y par X . Au lieu de cela, la condition $X \in Var(\phi)$ permet de garantir qu'une fois que X a été remplacé par Y , il ne le sera pas une seconde fois (il n'y a plus rien à remplacer).

Théorème 4.3 (Terminaison, confluence et résultat de la normalisation). *La normalisation d'OSF-clauses :*

- *termine** (finitement);
- *conflue** (modulo renommage des variables);
- *retourne soit une clause inconsistante ($X : \perp$), soit une OSF-clause résolue et une conjonction de contraintes d'égalité dont les variables en partie gauche n'apparaissent pas dans le reste de la clause.*

Démonstration.

terminaison Il faut noter que, mis à part la première règle, les deux suivantes ont besoin pour fonctionner de les deuxièmes et troisièmes règles consomment des contraintes (elles éliminent définitivement des contraintes de type $X : s$ et $X.l \doteq Y$) alors que la dernière consomme des variables (elle fait disparaître définitivement la variable X ailleurs que dans la clause $X \doteq Y$). Il faut donc faire attention à ce que la consommation d'un de ces objets ne conduise pas à condamner une règle déclenchable. Pour chaque règle :

type inconsistant fait terminer trivialement l'exécution.

intersection de type L'intersection de types fait strictement décroître la taille des clauses et aucune autre règle ne produit les contraintes qu'elle consomme.

substitution de traits Ne peut s'appliquer qu'un nombre fini de fois puisqu'elle fait décroître le nombre de descriptions de traits qu'elle consomme.

élimination de variable ne peut plus s'appliquer sur la variable X une fois appliquée (à cause de la garde sur son appartenance à $Var(\phi)$ alors que X est éliminé de la clause par la règle). Les seules contraintes d'égalité pouvant être rajouté par le système le sont par la (substitution de traits), comme le nombre d'égalité au départ est fini et que le nombre d'application de cette règle l'est aussi, l'(élimination de variable) ne peut s'appliquer qu'un nombre fini de fois.

confluence La confluence est triviale puisque les règles ne se déclenchent que sur des éléments distincts (à l'exception des deux premières). Les conflits entre les deux premières règles (et entre diverses applications de la première ne posent pas de problème puisque les sortes sont structurées dans un treillis dont le plus petit élément est \perp). Les conflits internes entre diverses applications des deux règles suivantes correspondent à la question quelle variable éliminer et sont reflétées dans les ensembles d'égalités liées au résultat.

forme Soit une inconsistance de type est détecté et la propriété est vérifiée, soit il ne l'est pas et reste à prouver que :

il existe au plus une contrainte de sorte c'est assuré par la règle d'(intersection de type) avec la contrainte supplémentaire que la sorte résultante n'est pas \perp ce qui déclencherait la règle de (type inconsistant) ;

- il existe au plus une contrainte de trait** (par couple variable/trait) ce qui est assuré par la règle de (substitution de traits);
- il ne subsiste plus de contrainte d'égalité** n'a pas besoin d'être satisfaite puisque la propriété à démontrer les autorise;
- les variables en partie gauche n'apparaissent pas** est assuré par la règle d'(élimination de variable) qui sinon se déclencherait. \square

Notation. La forme normale d'une \mathcal{OSF} -clause ϕ débarrassée des contraintes d'égalité sera notée $\overline{\phi}$.

4.2.3 \mathcal{OSF} -graphes

Définition 4.15 (Graphe enraciné). Un graphe orienté* est dit enraciné* s'il existe un nœud privilégié nommé la racine depuis lequel tout autre nœud du graphe est accessible.

Définition 4.16 (\mathcal{OSF} -graphe). Un \mathcal{OSF} -graphe $g = \langle N, E, \lambda_N, \lambda_E, X \rangle$ est un multi-graphe étiqueté orienté enraciné tel que :

- $\lambda_N : N \longrightarrow S - \{\perp\}$;
- $\lambda_E : E \longrightarrow F$;
- $X \in N$ est la racine de g ;
- $\forall \langle n, n' \rangle, \langle n, n'' \rangle \in E$, si $\lambda_E(\langle n, n' \rangle) = \lambda_E(\langle n, n'' \rangle)$ alors $n' = n''$.

Remarque. Ici, λ_E correspond à la définition du λ des graphes étiquetés alors que λ_N correspond à la fonction l des graphes conceptuels.

Définition 4.17 (\mathcal{OSF} -graphe associé à un Ψ -terme). À tout Ψ -terme ψ on associe un unique \mathcal{OSF} -graphe $G(\psi)$ dans lequel un nœud est associé à chaque variable, tel que :

1. si $\psi = X : s$ alors $G(\psi) = \langle \{X\}, \emptyset, \{\langle X, s \rangle\}, \emptyset, X \rangle$
2. si $\psi = X : s(l_1 \rightarrow \psi_1, \dots, l_n \rightarrow \psi_n)$ et $G(\psi_i) = g_i = \langle N_i, E_i, \lambda_{N_i}, \lambda_{E_i}, X_i \rangle$ alors $G(\psi) = \langle N, E, \lambda_N, \lambda_E, X \rangle$ tel que :

- $N = \{X\} \cup \bigcup_i N_i$;
- $E = \bigcup_i \{\langle X, X_i \rangle\} \cup E_i$;
- $\lambda_N(n) = \begin{cases} s & \text{si } n = X \\ \bigwedge_{i=1}^n \lambda_{N_i}(n) & \text{si } n \in \bigcup_i N_i \neq X \end{cases}$;
- $\lambda_E(e) = \begin{cases} l_i & \text{si } e = \langle X, X_i \rangle \\ \lambda_{E_i}(e) & \text{si } e \in E_i \end{cases}$;

Exemple 4.10 (*OSF-graphe associé à un OSF-terme*). L'OSF-graphe associé à l'OSF-terme de l'exemple 4.4 est le suivant :

$$\langle \begin{aligned} & \{n_1, n_2, n_3, n_6, n_4, n_5\}, \\ & \{\langle n_1, n_2 \rangle, \langle n_2, n_3 \rangle, \langle n_2, n_6 \rangle, \langle n_1, n_4 \rangle, \langle n_4, n_5 \rangle, \langle n_5, n_6 \rangle, \langle n_5, n_1 \rangle\}, \\ & \{\langle n_1, person \rangle, \langle n_2, id \rangle, \langle n_3, string \rangle, \langle n_6, string \rangle, \langle n_4, person \rangle, \langle n_5, id \rangle\}, \\ & \{\langle \langle n_1, n_2 \rangle, name \rangle, \langle \langle n_2, n_3 \rangle, first \rangle, \langle \langle n_2, n_6 \rangle, last \rangle, \langle \langle n_1, n_4 \rangle, spouse \rangle, \\ & \quad \langle \langle n_4, n_5 \rangle, name \rangle, \langle \langle n_5, n_6 \rangle, last \rangle, \langle \langle n_4, n_1 \rangle, spouse \rangle\} \end{aligned} \rangle$$

Il est aussi bien décrit par le graphe de la figure 4.1.

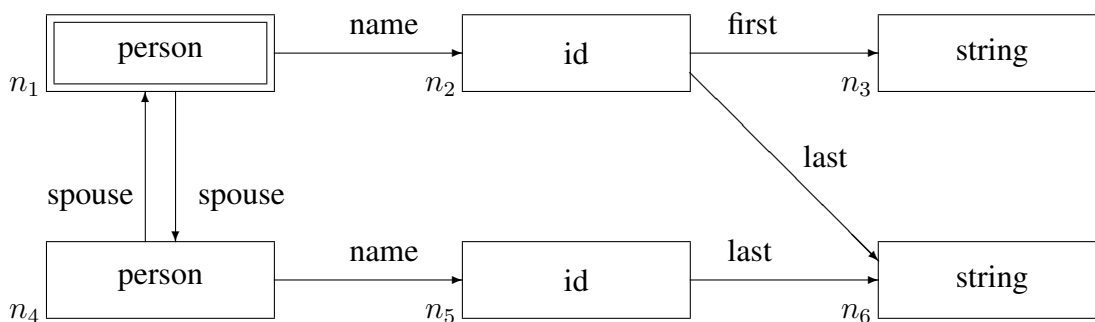


FIGURE 4.1 : Description graphique du graphe de l'exemple 4.10

Définition 4.18 (*OSF-terme associé à un OSF-graphe*). À tout OSF-graphe g on associe un OSF-terme $\psi(g)$ tel que si X est la racine de g telle que $\lambda_N(X) = s$ et $l_1, \dots, l_n \in F$ sont les étiquettes distinctes deux à deux des arcs sortant de X ,

$$\psi(g) = X : s(l_1 \rightarrow \psi(g_1(X)), \dots, l_n \rightarrow \psi(g_n(X)))$$

où $g_i(X)$ est le sous-graphe de g accessible à partir de $l_i^g(X)$ dans lequel $\lambda_N(X) = \top$, tout $\langle X, X' \rangle$ est supprimé de E et tous les sous-graphes accessibles à partir de $l_1^g(X), \dots, l_{i-1}^g(X)$ sont remplacés par $\{\{l_i^g(X)\}, \emptyset, \{\langle l_i^g(X), \top \rangle\}, \emptyset, l_i^g(X)\}$.

Conséquence 4.4. L'OSF-terme associé à un OSF-graphe est un Ψ -terme.

L'unicité du terme associé à un OSF-graphe dépend donc de l'ordre d'exploration du graphe. Celui-ci peut être simplement fixé en imposant un ordre total sur les éléments de F .

Définition 4.19 (*Ψ -terme associé à un OSF-graphe*). À tout OSF-graphe g on associe un unique Ψ -terme $\psi^{\ll}(g)$ tel que $\psi^{\ll}(g) = \psi(g)$ et les traits sont ordonnés par \ll .

Proposition 4.5 (*Correspondance entre OSF-graphes et Ψ -termes ordonnés*). Soit un ordre \ll sur les éléments de F , la correspondance entre Ψ -termes ordonnés et OSF-graphes est une bijection.

Démonstration. Soit un ordre \ll défini sur F , l' \mathcal{OSF} -terme correspondant à un \mathcal{OSF} -graphe enraciné est un unique terme en forme normale (définition 4.18). De même, étant donné un \mathcal{OSF} -terme en forme normale dont les étiquettes respectent \ll , il lui correspond un unique \mathcal{OSF} -graphe (définition 4.17). Il faut montrer que $\psi(G(\psi)) = \psi$ et que $G(\psi(g)) = g$.

$\psi^{\ll}(G(\psi)) = \psi$. Par construction, soit $\psi = X : s(l_1 \rightarrow \psi_1, \dots, l_n \rightarrow \psi_n)$, alors $\psi^{\ll}(G(\psi)) = X' : s'(l'_1 \rightarrow \psi'_1, \dots, l'_n \rightarrow \psi'_n)$ où

- X' est la racine de $G(\psi)$ à savoir X ;
- s' est $\lambda_N(G(X)) = \lambda_N(X)$ à savoir s ;
- l'_1, \dots, l'_n sont les étiquettes des arcs sortants de $G(X)$ (ordonnées par \ll) qui correspondent aux l_1, \dots, l_n du graphe initial (car pour un \mathcal{OSF} -terme en forme normale il existe une unique occurrence d'une variable (ici X) racine d'un \mathcal{OSF} -terme non trivial ;
- g'_1, \dots, g'_n sont les \mathcal{OSF} -graphes correspondant à $\psi^{\ll}(G(\psi_1)), \dots, \psi^{\ll}(G(\psi_n))$ et pour les mêmes raisons que ci-dessus, ils sont égaux à ψ_1, \dots, ψ_n . Si une variable Y est déjà apparue lors de la construction, elle est justement remplacée par $Y : \perp$ parce qu'elle n'est pas plus développée dans ψ (ce qui correspond à la position du terme développé dans un Ψ -terme ordonné.

$G(\psi(g)) = g$. Par construction, soit $g = \langle N, E, \lambda_N, \lambda_E, X \rangle$, alors $G(\psi(g)) = \langle N', E', \lambda_{N'}, \lambda_{E'}, X' \rangle$ où :

- X' est la racine de $\psi(g)$ c'est-à-dire la racine X de g ;
- N' est l'ensemble des variables dans $\psi(g)$ qui correspondent exactement aux nœuds de N (car le graphe étant enraciné tous les nœuds sont atteignables ; $\lambda_{N'}$ étiquette chacun de ces nœuds avec leur sorte dans $\psi(g)$ (et il n'y a qu'un seul endroit dans $\psi(g)$ où cette sorte n'est pas \top car c'est un terme en forme normale), mais cette sorte est exactement la valeur de λ_N (définition 4.18) ;
- E' contient tous les traits qui sont présents dans $\psi(g)$ et ses sous-termes mais ceux-ci correspondent exactement aux arcs de g (définition 4.18) ; ils sont étiquetés par le nom du trait correspondant (définition 4.17) qui correspond à son tour à l'étiquette du nœud initial par λ_E (définition 4.18). \square

Théorème 4.6 (Équivalence syntaxique). *Il y a une correspondance un à un entre \mathcal{OSF} -graphes, \mathcal{OSF} -clauses enracinées résolues et Ψ -termes ordonnés.*

Démonstration. Cette proposition est une conséquence directe de la bijection entre Ψ -termes ordonnés et \mathcal{OSF} -clauses enracinées résolues (théorème 4.2) d'une part et entre Ψ -termes ordonnés et \mathcal{OSF} -graphes (proposition 4.5) d'autre part. \square

4.3 Interprétation

4.3.1 \mathcal{OSF} -structure

Définition 4.20 (\mathcal{OSF} -algèbre). Une \mathcal{OSF} -algèbre* sur une \mathcal{OSF} -signature $\langle S, \leq, \wedge, F \rangle$ est une structure $A = \langle D^A, \{s^A\}_{s \in S}, \{l^A\}_{l \in F} \rangle$ telle que :

- D^A soit un ensemble non vide ;
- $\forall s \in S, s^A \subseteq D^A$ avec $\top^A = D^A$ et $\perp^A = \emptyset$.
- $(s \wedge s')^A = s^A \cap s'^A$;
- $l^A : D^A \longrightarrow D^A$ est une fonction totale unaire.

Le fait que les traits soient interprétés comme des fonctions totales unaires permet d'avoir $l^1.l^2 \dots l^n$ toujours définie. Si l'on ne se soucie pas de la composition de traits, alors on peut rendre les fonctions partielles.

Exemple 4.11 (\mathcal{OSF} -algèbre).

On utilise ci-dessous $\#$ pour dénoter tous les éléments qui n'ont pas de valeur spécifique.

$$\begin{aligned}
 A = \langle & \\
 & \{ pierre, janine, paul, ida, id_1, id_2, id_3, id_4, \\
 & Dupond, Durand, Pierre, Paul, Janine, Valérie, \iota \}, \\
 & person^A = \{ pierre, janine, paul, ida \}, \\
 & id^A = \{ id_1, id_2, id_3, id_4 \} \\
 & string^A = \{ Dupond, Durand, Pierre, Paul, Janine, Valérie \}, \\
 & name^A = \{ \langle pierre, id_1 \rangle, \langle janine, id_2 \rangle, \langle paul, id_3 \rangle, \langle ida, id_4 \rangle, \langle \#, \iota \rangle \} \\
 & spouse^A = \{ \langle pierre, janine \rangle, \langle janine, pierre \rangle, \langle \#, \iota \rangle \} \\
 & first^A = \{ \langle id_1, Pierre \rangle, \langle id_2, Janine \rangle, \langle id_3, Paul \rangle, \langle id_4, Valérie \rangle, \langle \#, \iota \rangle \} \\
 & last^A = \{ \langle id_1, Dupont \rangle, \langle id_2, Dupont \rangle, \langle id_3, Durand \rangle, \langle id_4, Durand \rangle, \langle \#, \iota \rangle \} \}
 \end{aligned}$$

est une \mathcal{OSF} -algèbre pour l' \mathcal{OSF} -signature :

$$\langle S = \{ person, id, string, \top, \perp \}, \leq = \{ \langle \perp, \# \rangle, \langle \#, \top \rangle \}, F = \{ name, first, spouse, last \} \rangle$$

. L'entité ι sert ici de puit à tout le graphe : tous ses traits ont pour valeur ι .

Définition 4.21 (Assignment). Soit A une \mathcal{OSF} -algèbre et V un ensemble de variables, une assignation sur A est une fonction $\alpha : V \longrightarrow D^A$.

Notation. L'ensemble des assignations sur A est noté $Val(A)$.

Exemple 4.12 (Assignment). Soit l'ensemble $V = \{ X, N, F, P, S, I \}$,

$$\alpha = \{ \langle X, pierre \rangle, \langle N, id_1 \rangle, \langle F, Pierre \rangle, \langle P, janine \rangle, \langle S, Dupont \rangle, \langle I, id_2 \rangle \}$$

et

$$\beta = \{ \langle X, paul \rangle, \langle N, id_3 \rangle, \langle F, Paul \rangle, \langle P, ida \rangle, \langle S, Durand \rangle, \langle I, id_4 \rangle \}$$

sont deux assignations pour l' \mathcal{OSF} -algèbre de l'exemple 4.11.

On introduit ci-dessous les \mathcal{OSF} -algèbres de graphe dont le domaine est un ensemble de graphes. Ils jouent un rôle semblable à celui de l'univers de termes de Herbrand dans le calcul des prédicats : servir à la construction de modèles canoniques auxquels tous les autres doivent être homomorphes.

Les graphes dénotent alors aussi bien des éléments du domaine (eux-mêmes) que l'ensemble des graphes qu'ils approximent. La solution la plus générale est immédiatement extraite de l' \mathcal{OSF} -graphe, toutes les autres en sont des approximations endomorphes.

Définition 4.22 (\mathcal{OSF} -algèbre de graphe). Une \mathcal{OSF} -algèbre de graphe est une \mathcal{OSF} -algèbre G dont le domaine D^G est un ensemble d' \mathcal{OSF} -graphes et

$$s^G = \{g = \langle N, E, \lambda_N, \lambda_E, X \rangle \mid \lambda_N(X) \leq s\}$$

et

$$l^G(g = \langle N, E, \lambda_N, \lambda_E, X \rangle) = \begin{cases} \langle N|_Y, E|_Y, \lambda_N, \lambda_E, Y \rangle & \text{si } \lambda_E(\langle X, Y \rangle) = l \text{ pour un } \langle X, Y \rangle \in E \\ \langle \{Z_{l,g}\}, \emptyset, \{\langle Z_{l,g}, \top \rangle\}, \emptyset, Z_{l,g} \rangle & \text{où } Z_{l,g} \in V \setminus N \text{ sinon.} \end{cases}$$

Une \mathcal{OSF} -algèbre de graphe est donc l'algèbre dont le éléments sont des \mathcal{OSF} -graphes, l'interprétation d'une sorte est l'ensemble des graphes dont la racine a une étiquette au moins aussi précise qu'elle et l'interprétation d'un trait une fonction qui associe à un graphe le sous-graphe connexe accessible par le trait à partir de la racine (ou un graphe *trivial* fait d'une unique variable de sorte \top si la racine n'a pas d'arc étiqueté par ce trait).

Exemple 4.13 (\mathcal{OSF} -algèbre de graphe). Soient les graphes de la figure 4.2, ils constituent le domaine d'une \mathcal{OSF} -algèbre de graphes définie par :

$$A = \langle \begin{aligned} & \{g_1, g_2, g_3, g_4, g_5, g_6, g_7\}, \\ & \{person^A = \{g_5, g_6\}, id^A = \{g_3, g_4\}, string^A = \{g_1, g_2\}\}, \\ & \{name^A = \{\langle g_5, g_4 \rangle, \langle g_6, g_3 \rangle, \langle \#, g_7 \rangle\}, spouse^A = \{\langle g_5, g_6 \rangle, \langle g_6, g_5 \rangle, \langle \#, g_7 \rangle\}, \\ & first^A = \{\langle g_4, g_1 \rangle, \langle \#, g_7 \rangle\}, last^A = \{\langle g_4, g_2 \rangle, \langle g_3, g_2 \rangle, \langle \#, g_7 \rangle\}\} \end{aligned} \rangle$$

On note par \mathcal{G} l' \mathcal{OSF} -algèbre de graphe dont le domaine est l'ensemble de tous les \mathcal{OSF} -graphes.

Définition 4.23 (\mathcal{G}). Soit P l'ensemble de tous les \mathcal{OSF} -termes sous forme normale, on lui associe l' \mathcal{OSF} -algèbre de graphes \mathcal{G} caractérisée par :

$$\begin{aligned} D^{\mathcal{G}} &= \{G(\psi) \mid \psi \in P\} \\ s^{\mathcal{G}} &= \{G(X : s'(\dots)) \mid s' \leq s\} \\ l^{\mathcal{G}}(G(X : s(\dots, l \rightarrow \psi', \dots))) &= \begin{cases} G(X : s(\dots, l \rightarrow \psi', \dots)) & \text{si } X = root(\psi') \\ G(\psi') & \text{sinon} \end{cases} \\ l^{\mathcal{G}}(G(\psi)) &= G(Z_{l, G(\psi)} : \top) \text{ où } Z_{l, G(\psi)} \notin Var(\psi) \end{aligned}$$

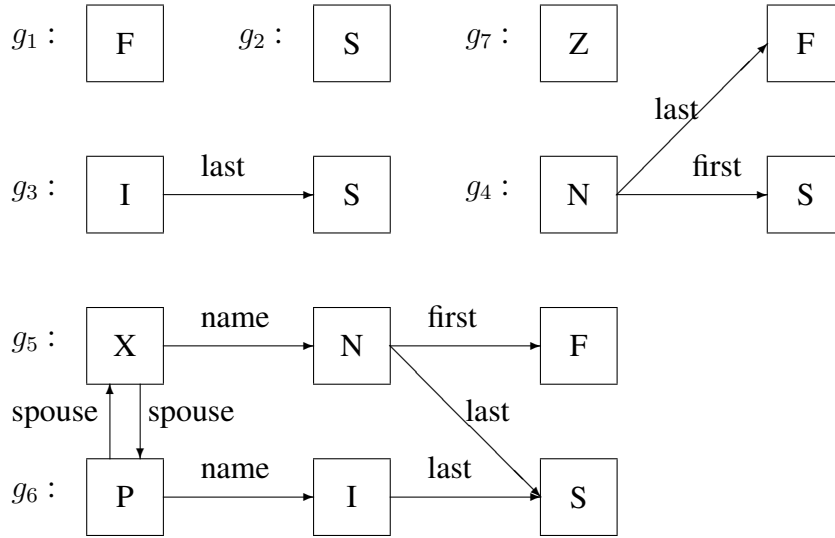


FIGURE 4.2 : Description graphique des graphes de l'exemple 4.13

4.3.2 Satisfaction d'une \mathcal{OSF} -clause

Définition 4.24 (Satisfaction d'une \mathcal{OSF} -clause). Soit A une \mathcal{OSF} -algèbre et α une assignation sur A , la satisfaction de la clause ϕ dans l'interprétation A sous l'assignation α (noté $A, \alpha \models \phi$) est définie par :

$$\begin{aligned}
 A, \alpha \models X : s & \text{ssi } \alpha(X) \in s^A \\
 A, \alpha \models X \doteq Y & \text{ssi } \alpha(X) = \alpha(Y) \\
 A, \alpha \models X.l \doteq Y & \text{ssi } l^A(\alpha(X)) = \alpha(Y) \\
 A, \alpha \models \phi \wedge \phi' & \text{ssi } A, \alpha \models \phi \text{ et } A, \alpha \models \phi'
 \end{aligned}$$

Exemple 4.14 (Satisfaction d'une \mathcal{OSF} -clause). L' \mathcal{OSF} -clause de l'exemple 4.7 est satisfaite dans l' \mathcal{OSF} -algèbre de l'exemple 4.11 sous l'assignation α de l'exemple 4.12 mais pas sous l'assignation β .

Proposition 4.7. $A, \alpha \models \phi$ ssi $A, \alpha \models \bar{\phi}$

Démonstration. Ce résultat s'obtient en montrant que chacune des règles de normalisation (définition 4.14) préserve la propriété.

$$\begin{array}{l}
 \frac{A, \alpha \models \phi \text{ et } \alpha(X) \in \emptyset}{\alpha(X) \in \emptyset} \quad \text{(type inconsistant)} \\
 \frac{\alpha(X) \in s^A \text{ et } \alpha(X) \in s'^A}{\alpha(X) \in s^A \cap s'^A} \quad \text{(intersection de types)} \\
 \frac{l^A(\alpha(X)) = \alpha(Y) \text{ et } l^A(\alpha(X)) = \alpha(Y')}{l^A(\alpha(X)) = \alpha(Y) \text{ et } \alpha(Y') = \alpha(Y)} \quad \text{(substituton de traits)} \\
 \frac{A, \alpha \models \phi \text{ et } \alpha(X) = \alpha(Y)}{A, \alpha \models \phi[X/Y] \text{ et } \alpha(X) = \alpha(Y)} \quad \text{(élimination de variables)}
 \end{array}$$

Chaque passage de la barre (dans n'importe quel sens) est effectivement une égalité valide dans la théorie des ensembles (la première l'est car, comme il ne peut exister d'assignation qui fasse qu'un objet appartienne à l'ensemble vide, la conjonction initiale est fausse). \square

Définition 4.25 (\mathcal{OSF} -algèbres admissibles pour une \mathcal{OSF} -clause). Soit ϕ une \mathcal{OSF} -clause résolue, toute \mathcal{OSF} -algèbre A est dite admissible pour ϕ s'il existe une assignation $\alpha \in Val(A)$ telle que :

$$A, \alpha \models \phi$$

Définition 4.26 (\mathcal{OSF} -implication). Soient ϕ et ϕ' deux \mathcal{OSF} -clauses, ϕ implique ϕ' (noté $\phi \Rightarrow \phi'$) si et seulement si :

$$\forall A, \alpha \text{ tels que } A, \alpha \models \phi \\ \exists \alpha' \text{ tel que } \forall X \in Var(\phi) \cap Var(\phi'), \alpha(X) = \alpha'(X) \text{ et } A, \alpha' \models \phi'$$

Définition 4.27 (\mathcal{OSF} -implication enracinée). Soient ϕ_X et $\phi'_{X'}$ deux \mathcal{OSF} -clauses enracinées sans variables communes,

$$\phi_X \Rightarrow \phi'_{X'} \text{ ssi } \phi \Rightarrow \phi'[X/X']$$

4.3.3 Dénotation des \mathcal{OSF} -termes

Définition 4.28 (Dénotation sous une assignation). Soit A une \mathcal{OSF} -algèbre et α une assignation, la dénotation $[\psi]^{A,\alpha}$ d'un \mathcal{OSF} -terme $\psi = X : s(l_i \rightarrow \psi_i)$ sous une assignation α est définie par :

$$[\psi]^{A,\alpha} = \{\alpha(X)\} \cap s^A \cap \bigcap_{i=1}^n (l_i^A)^{-1}([\psi_i]^{A,\alpha})$$

où $l^{-1}(S) = \{x | \exists y \in S; y = l(x)\}$.

Proposition 4.8. Soit A une \mathcal{OSF} -algèbre, $\forall \alpha \in Val(A)$:

$$[\psi]^{A,\alpha} = \begin{cases} \{\alpha(X)\} & \text{si } A, \alpha \models \phi(\psi) \\ \emptyset & \text{sinon.} \end{cases}$$

Démonstration. Par la définition 4.28, $[\psi]^{A,\alpha}$ ne peut être plus grand que $\{\alpha(X)\}$ (celui-ci étant un singleton). Si $A, \alpha \models \phi(\psi)$ alors $\alpha(X) \in s^A$ et, en particulier, si $\psi = X : \perp$ alors $\alpha(X) \in D^A$. Par ailleurs, $Y.l_i = X \Rightarrow \alpha(X) \in (l_i^A)^{-1}([\psi_i]^{A,\alpha})$ (si la proposition est vraie) et, par conséquent, $[\psi]^{A,\alpha} = \{\alpha(X)\}$. Si ce n'est pas le cas, alors $\{\alpha(X)\} \cap s^A \cap \bigcap_{i=1}^n (l_i^A)^{-1}([\psi_i]^{A,\alpha}) = \emptyset$. \square

Définition 4.29 (Dénotation). Soit A une \mathcal{OSF} -algèbre et ψ un \mathcal{OSF} -terme, la dénotation de ψ (notée $[\psi]^A$) est l'ensemble :

$$[\psi]^A = \bigcup_{\alpha \in Val(A)} [\psi]^{A,\alpha}$$

À l'instar des graphes conceptuels, les \mathcal{OSF} -termes ne considèrent jamais les individus en tant que tels. Cette propriété peut sembler héritée de la programmation logique où les assertions sont quantifiées universellement. Ceci les distingue des logiques terminologiques ou de la programmation par objets.

Exemple 4.15 (Dénotation). La dénotation de l' \mathcal{OSF} -terme de l'exemple 4.4 dans l' \mathcal{OSF} -algèbre de l'exemple 4.11 sous l'assignation α de l'exemple 4.12 est $\{pierre\}$, sous l'assignation β , elle est \emptyset .

La dénotation du même terme dans la même algèbre est $\{pierre, janine\}$

Conséquence 4.9.

$$[\psi]^A = \{\alpha(X) \mid \alpha \in Val(A) \text{ et } A, \alpha \models \phi(\psi)\}$$

Définition 4.30 (Subsomption d' \mathcal{OSF} -termes). Soient ψ et ψ' deux \mathcal{OSF} -termes, ψ est subsumé par ψ' (noté $\psi \preceq \psi'$) si et seulement si pour toute \mathcal{OSF} -algèbre A , $[\psi]^A \subseteq [\psi']^A$.

4.3.4 Algèbre de graphe canonique

Définition 4.31 (Algèbre de graphe canonique). Soit ϕ une \mathcal{OSF} -clause résolue, la sous-algèbre $\mathcal{G}[D^{G,\phi}]$ de l' \mathcal{OSF} -algèbre de graphe G engendrée par $D^{G,\phi} = \{G(\phi(X)) \mid X \in Var(\phi)\}$ de tous les sous-graphes maximalement connexes de $G(\phi)$ est nommée l'algèbre de graphes canonique de ϕ .

Exemple 4.16 (Algèbre de graphe canonique). L' \mathcal{OSF} -algèbre de graphes de l'exemple 4.13 est l' \mathcal{OSF} -algèbre de graphes canonique pour l' \mathcal{OSF} -terme de l'exemple 4.4.

Conséquence 4.10. L' \mathcal{OSF} -algèbre de graphe canonique engendrée par ϕ est admissible pour ϕ .

4.4 Catégorie OSF

Définition 4.32 (\mathcal{OSF} -homomorphisme). Un \mathcal{OSF} -homomorphisme entre deux \mathcal{OSF} -algèbres A et B est une fonction $\gamma : D^A \rightarrow D^B$ telle que :

- $\forall d \in D^A, \gamma(l^A(d)) = l^B(\gamma(d))$;
- $\gamma(s^A) \subseteq s^B$.

Le rôle de l'homomorphisme va donc être :

- d'identifier deux variables (comme dénotant le même élément du domaine),
- d'associer une sorte plus spécifique à une variable.

On retrouve donc les opérations de projection dans les graphes conceptuels où il est possible de fusionner deux graphes en identifiant deux nœuds et de restreindre la sorte associée à un nœud. L'opération restante de suppression de relations redondante peut être considérée comme prise en compte par les procédures de normalisation qui ont été déployées jusqu'à présent.

Exemple 4.17 (OSF-homomorphisme). La fonction $\gamma : D \rightarrow D$ définie par :

$$\begin{aligned} \gamma(g_1) &= Pierre, \gamma(g_2) = Dupont, & \gamma(g_3) &= id_2, \\ \gamma(g_4) &= id_1, \gamma(g_5) = pierre, & \gamma(g_6) &= janine, \\ \gamma(g_7) &= \iota \end{aligned}$$

est un OSF-homomorphisme entre l'OSF-algèbre de graphe de l'exemple 4.13 et l'OSF-algèbre de l'exemple 4.11.

Définition 4.33 (Catégorie OSF). La catégorie OSF est la catégorie* formée des OSF-algèbres comme catégories et des OSF-homomorphismes comme morphismes.

Définition 4.34 (Approximation endomorphe). Soit une OSF-algèbre A , un préordre \sqsubseteq_A est défini tel que pour tout élément d et e de D^A , d approxime e (noté $d \sqsubseteq_A e$) si et seulement si $\exists \gamma : A \rightarrow A$ tel que $e = \gamma(d)$.

Conséquence 4.11 (Solution canonique). Toute OSF-clause résolue ϕ_X est satisfaisable dans l'OSF-algèbre de graphe canonique \mathcal{G} sous toute assignation $\alpha \in Val(\mathcal{G})$ telle que $\alpha(X) = G(\phi_X)$.

Théorème 4.12 (Faible initialité*). Les solutions d'une OSF-clause résolue ϕ dans toute OSF-algèbre A admissible pour ϕ peuvent être obtenues par un OSF-homomorphisme à partir de l'OSF-algèbre de graphe canonique engendrée par ϕ .

Démonstration. La raison en est que, pour toute assignation $\alpha \in Val(A)$ telle que $A, \alpha \models \phi$ il existe un OSF-homomorphisme $\gamma : \mathcal{G}[D^{\mathcal{G}, \phi}] \rightarrow A$ tel que $\alpha(X) = \gamma(G(\phi_X))$. En effet, soit α telle que $A, \alpha \models \phi$, l'homomorphisme γ peut être défini en posant $\gamma(G(\phi_X)) = \alpha(X)$ et en faisant correspondre (par γ) à toute partie de $G(\phi_X)$ la partie correspondante par α dans A . Ceci permet bien de définir complètement γ . Ceci respecte bien la définition d'homomorphisme puisque pour tout $g' = l^{\mathcal{G}}(G(\phi_X))$:

$g' = G(Z : \top)$ avec $Z \notin Var(\phi)$ alors $\gamma(Z) \in \top^B$;
 $g' = G(\phi_Y)$ avec $Y \in Var(\phi)$, alors $\lambda_N^A(\alpha(X)) = \alpha(Y)$ ce qui signifie que pour tout $g \in D^{\mathcal{G}, \phi}$ de la forme $g = G(\phi_X)$ il est vrai que $\gamma(l^{\mathcal{G}}(g)) = l^A(\gamma(g))$. Si $G(\phi_X) \in s^{\mathcal{G}}$ alors ϕ doit contenir une contrainte de type $X : s'$ (avec $s' \leq s$) et $\alpha(X) \in s'^A$ ce qui signifie que si $g \in s^{\mathcal{G}}$ alors $\gamma(g) \in s^A$. \square

Proposition 4.13. Soient A et B deux OSF-algèbres et $\gamma : A \rightarrow B$ un OSF-homomorphisme, soit ϕ une OSF-clause telle qu'il existe $\alpha \in Val(A)$ telle que $A, \alpha \models \phi$, alors pour tout $\beta \in Val(B)$ tel que $\beta = \gamma \circ \alpha$, $B, \beta \models \phi$.

Démonstration. Il s'agit de montrer qu'un homomorphisme préserve les solutions.

$A, \alpha \models X : s$ signifie que $\alpha(X) \in s^A$ et donc $\gamma \circ \alpha(X) \in s^B$ (définition 4.32) ;
 $A, \alpha \models X \doteq Y$ signifie que $\alpha(X) = \alpha(Y)$ et donc $\gamma \circ \alpha(X) = \gamma \circ \alpha(Y)$;
 $A, \alpha \models X.l = Y$ signifie que $l^A(\alpha(X)) = \alpha(Y)$ et donc $l^B(\gamma \circ \alpha(X)) = \gamma(l^A(\alpha(X))) = \gamma(\alpha(Y)) = \gamma \circ \alpha(Y)$ (définition 4.32) ;
 $A, \alpha \models \phi' \wedge \phi''$ signifie que $A, \alpha \models \phi'$ et $A, \alpha \models \phi''$ et donc $\gamma \circ \alpha(X) \in s^B$ suite aux résultats ci-dessus. \square

Théorème 4.14 (Faible finalité* de \mathcal{G}). *Il existe un \mathcal{OSF} -homomorphisme γ depuis toute \mathcal{OSF} -algèbre vers l' \mathcal{OSF} -algèbre de graphe \mathcal{G} .*

Démonstration. Pour tout élément $d \in D^A$, on définit une variable X_d qui correspondra à un nœud. Chaque variable X_d est étiquetée avec la plus grande sous-sortie commune de toutes les sortes telles que $d \in s^A$ (et donc $\gamma(d) \in (s \wedge \dots)^{\mathcal{G}} \subseteq s^{\mathcal{G}}$). Pour tout d' tel que $l^A(d) = d'$, on ajoute un arc de X_d vers $X_{d'}$ étiqueté par l . Le résultat est donc un \mathcal{OSF} -graphe. $\gamma(d)$ se définit alors comme son sous-graphe connexe maximal enraciné en X_d . Comme c'est un \mathcal{OSF} -graphe, il appartient à \mathcal{G} . On a alors clairement un homomorphisme car $\forall d \in D^A$, $\gamma(l^A(d)) = X_{d'}$ et $l^{\mathcal{G}}(\gamma(d)) = l^{\mathcal{G}}(X_d) = X_{d'}$. \square

Conséquence 4.15 (Canonicité de \mathcal{G}). *Une \mathcal{OSF} -clause est satisfaisable si et seulement si elle est satisfaisable dans l' \mathcal{OSF} -algèbre de graphe \mathcal{G} .*

Conséquence 4.16 (Solutions canoniques principales). *L' \mathcal{OSF} -graphe $G(\phi(X))$ approxime tout autre graphe g affecté à la variable X comme solution d'une \mathcal{OSF} -clause ϕ .*

C'est-à-dire que la solution $\alpha \in Val(G)$ telle que $\alpha(X) = G(\phi(X))$ est une solution principale de ϕ dans l' \mathcal{OSF} -algèbre \mathcal{G} .

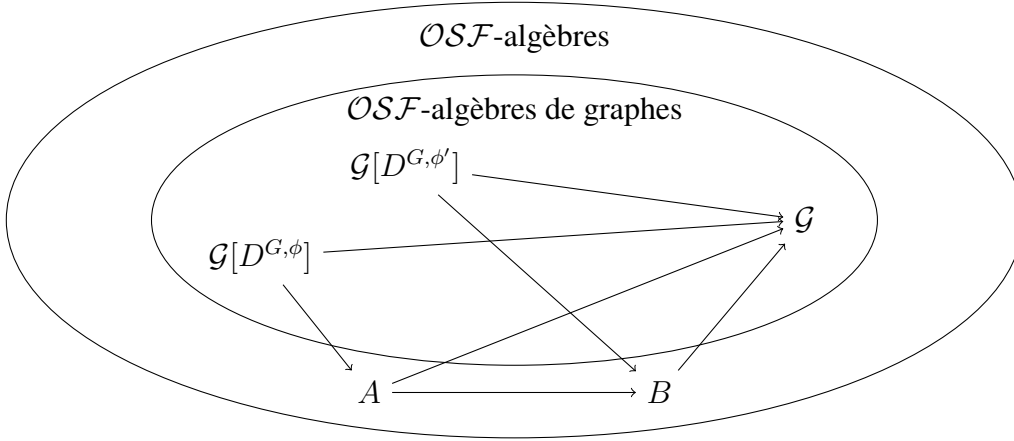


FIGURE 4.3 : Résumé de tous les ensembles et morphismes manipulés.

Conséquence 4.17. *Soit un \mathcal{OSF} -terme en forme normale ψ et le graphe correspondant $G(\psi) \in D^{\mathcal{G}}$, alors sa dénotation est :*

$$[\psi]^A = \{\gamma(G(\psi)) \mid \gamma : \mathcal{G} \longrightarrow A \text{ est un } \mathcal{OSF}\text{-homomorphisme}\}$$

Conséquence 4.18. *Soit un \mathcal{OSF} -terme en forme normale ψ et le graphe correspondant $g = G(\psi) = G(\phi(\psi))$, alors sa dénotation est :*

$$[\psi]^A = \{\alpha(X) \mid A, \alpha \models \phi(\gamma(g)) \text{ et } \gamma : \mathcal{G} \longrightarrow \mathcal{G} \text{ est un } \mathcal{OSF}\text{-endomorphisme}\}$$

Conséquence 4.19. *La dénotation d'un \mathcal{OSF} -terme en forme normale ψ dans l' \mathcal{OSF} -algèbre de graphe est l'ensemble des \mathcal{OSF} -graphes que l' \mathcal{OSF} -graphe $G(\psi)$ approxime :*

$$[\psi]^{\mathcal{G}} = \{g \in D^{\mathcal{G}} \mid G(\psi) \sqsubseteq_{\mathcal{G}} g\}$$

Conséquence 4.20.

$$d \sqsubseteq_A d' \text{ ssi } \forall \psi \text{ OSF-terme; } d \in [\psi]^A \Rightarrow d' \in [\psi]^A$$

Théorème 4.21 (Équivalences sémantiques).

Soient ψ, ψ' OSF-termes en forme normale,
 g, g' OSF-graphes,
 $\phi_X, \phi'_{X'}$ OSF-clauses résolues enracinées

se correspondant respectivement (au sens du théorème 4.6) alors les propositions suivantes sont équivalentes :

- i $g \sqsubseteq_G g'$;
- ii $\psi \preceq \psi'$;
- iii $\phi'_X \Rightarrow \phi_X$;
- iv $[\psi]^G \subseteq [\psi']^G$.

Démonstration. Ceci découle directement des conséquences 4.9, 4.17 et 4.19. □

Conséquence 4.22. Soient deux OSF-clauses résolues enracinées ϕ et ϕ'

$$\phi' \Rightarrow \phi \text{ ssi } \phi = \phi(\gamma(G(\phi')))$$

pour un OSF-endomorphisme γ .

Conséquence 4.23. Soient deux OSF-graphes g_1 et g_2 , soit, s'il existe, $g = G(\text{Solved}(\phi(g_1) \wedge \phi(g_2)))$ alors $g \sqsubseteq_G g_1$ et $g \sqsubseteq_G g_2$ et il est le principal OSF-graphe avec cette propriété (c'est-à-dire qu'il approxime tous les autres).

4.5 Conclusion

Le formalisme des Ψ-termes possède un certain nombre de caractéristiques propres :

- c'est un formalisme comparable aux logiques de descriptions mais conçu comme une extension de Prolog ;
- il est comparable aux graphes conceptuel en ce sens que les Ψ-termes sont des graphes (mais ces graphes sont limités à un graphe enraciné) ;
- on y autorise les descriptions cycliques ;
- la notion de morphisme et d'approximation permet d'intégrer élégamment la notion de solutions principales ;
- sur un plan plus pratique, des techniques particulières sont développées pour se jouer de la semi-décidabilité.

4.6 Exercices

Exercice 4.1 (*). Soient les \mathcal{OSF} -termes suivants :

```
X:projet( chef => personne ( projet => X,
                           diplome => Y:phD ),
         budget => bilan )

projet_bien_gere( budget => bilan( entrees => Z:entier,
                                   sorties => Z ))

projet_moderne( chef => femme )
```

Refaire tous les exemples qui ont été donnés dans le cours en termes d'application des algorithmes.

1. Donnez une \mathcal{OSF} -signature correspondant à ces termes ;
2. Donnez les \mathcal{OSF} -termes correspondant à ces termes ainsi que les \mathcal{OSF} -termes normalisés ; à chaque fois, spécifiez l'ensemble Var ;
3. Donnez les \mathcal{OSF} -clauses correspondant aux \mathcal{OSF} -termes ; vérifiez que ces clauses sont enracinées dans la variable racine ;
4. Normalisez ces \mathcal{OSF} -clauses à l'aide des règles de la définition 4.14 ; vérifiez que les clauses obtenues correspondent aux égalités près aux clauses obtenues par dissolution des \mathcal{OSF} -termes normalisés ;
5. Tracez les \mathcal{OSF} -graphes correspondant aux \mathcal{OSF} -termes initiaux ;
6. Extrayez les \mathcal{OSF} -termes associés aux \mathcal{OSF} -graphes précédents et vérifiez que vous obtenez les \mathcal{OSF} -termes initiaux.

Exercice 4.2 (*). On désire comparer les graphes conceptuels et les \mathcal{OSF} -termes.

1. Soit l' \mathcal{OSF} -terme suivant :

```
X:personne( conjoint => britannique( age => 32 ),
           possede => Y:auto,
           conjoint => personne( conduit => Y ))
```

Traduisez le en l' \mathcal{OSF} -graphe correspondant puis en un graphe conceptuel similaire.

2. Mettez en correspondance les divers éléments que l'on trouve dans les deux formalismes : syntaxe (y compris support/signature), sémantique, opérations.

Exercice 4.3 (*)**. Montrez l'équivalence entre \mathcal{OSF} -clauses enracinées résolues et les \mathcal{OSF} -graphes sans faire intervenir les \mathcal{OSF} -termes.

Est-il nécessaire de faire intervenir un ordre sur F ? Pourquoi ?

Exercice 4.4 (*). Soit la description de concept suivante :

```
humain  $\hat{=}$  (and (all pere homme) (all mere femme)
              (all enfant humain) (all fils homme) (all fille femme))
```

1. Quel problème pose-t-elle vis-à-vis de la définition donnée pour les logiques terminologiques ?
2. Implique-t-elle que les éléments de l'extension d'humain doivent être circulaires ou que l'extension doit être infinie ? Dites pourquoi ?

Exercice 4.5 (*). On considère la logique terminologique $\mathcal{FL}^=$ dont les constructeurs sont `and`, `all` et `eqrvm`, telle que l'on puisse décrire les concepts suivants :

```

homme   ⊑h humain
femme   ⊑h humain
homme   ⊕h femme
fils    ⊑e enfant
fille   ⊑e enfant
fils    ⊕e fille''
t       ≐ (and humain (eqrvm (pere enfant) (mere enfant)))
t'      ≐ (and homme (eqrvm (pere fils) (mere fils)))
t''     ≐ (and homme (eqrvm (pere fils) (fils pere)))
t'''    ≐ (and homme (eqrvm (mere fils) (fils pere)))
    
```

où `eqrvm` s'interprète comme l'égalité de ses deux membres qui eux s'interprètent comme des chaînes de rôles : $(z \ x \ y)$ signifie "les z des x des y".

1. Donnez une interprétation en langue naturelle des concepts `t` à `t'''` ;
2. Est-ce que certains concepts en subsument d'autres ? Les donner. Que penser de $(\text{and } t' \ t'')$ par rapport à `t'''` ? Que dire de $(\text{and } (\text{all } a \ (\text{eqrvm } b \ c)))$ par rapport à $(\text{eqrvm } (a \ b) \ (a \ c))$?
3. Soit la fonction d'extension

$$\mathcal{E}((\text{eqrvm}(r_1 \dots r_n)(r'_1 \dots r'_m))) = \{o'_0 \in D \mid \forall o_1, \dots, o_n \in D; \forall i \in [1 \dots n] \langle o_{i-1}, o_i \rangle \in \mathcal{E}(r_i), \\ \exists o'_1, \dots, o'_m \in D; \forall i \in [1 \dots m] \langle o'_{i-1}, o'_i \rangle \in \mathcal{E}(r'_i) \wedge o_n = o'_m\}$$

montrez le bien fondé de votre réponse précédente à l'aide des extensions ;

4. Ajoutez les règles de comparaison au système de COMPARE et montrez qu'elles sont justifiées par les extensions.

Exercice 4.6 ().** Le but de cette partie est de mettre en évidence le rapport entre les termes de $\mathcal{FL}^=$ et les *OSF* termes.

1. Indiquez quels éléments vous semblent correspondre dans les deux systèmes et quels éléments vous semblent différer ;
2. Exprimez les *OSF*-termes de l'exemple 4.1 sous la forme de $\mathcal{FL}^=$ -termes ;
3. Faites correspondre une *OSF*-signature à une $\mathcal{FL}^=$ -terminologie, que manque-t-il ?

4. Dessinez l' \mathcal{OSF} -graphe correspondant aux termes t et t'' et donnez les \mathcal{OSF} -clauses correspondant aux termes t' et t''' ;
5. Donnez une fonction de traduction d'un $\mathcal{FL}^=$ -terme en un \mathcal{OSF} -graphe ; Que penser de la traduction d'un \mathcal{OSF} -graphe en un $\mathcal{FL}^=$ -terme ?
6. Donnez une fonction de traduction d'un $\mathcal{FL}^=$ -terme en une \mathcal{OSF} -clause.

Exercice 4.7 (*)**. On cherche maintenant à utiliser les outils développés pour les \mathcal{OSF} -termes dans le cadre des $\mathcal{FL}^=$ -termes. On suppose pour cela un ordre \ll sur les traits dans l' \mathcal{OSF} -signature et on réduit les rôles intervenants dans les eqvrm à être des attributs (c'est-à-dire des rôles à une valeur unique).

1. Précisez l' \mathcal{OSF} -signature donnée précédemment au 3c.
2. Est-il nécessaire d'ajouter des règles pour la résolution d' \mathcal{OSF} -clauses ? Si oui dites lesquelles, montrez qu'elles sont correctes et qu'elles sont complètes.
3. L'utilisation des règles de résolution des \mathcal{OSF} -clauses a-t-elle une utilité pour les $\mathcal{FL}^=$ -termes ? Quels tests donnés dans les définitions 2.8 (subsomption) et 2.9 (équivalence, incohérence, exclusion) peut-elle servir à réaliser ?

Compositionnalité

4.7 Compositionnalité

Dans les langages considérés par la logique, il existe une constante ; la sémantique est compositionnelle. C'est-à-dire que le sens des expressions est presque toujours calculé en fonction uniquement du sens des sous-expressions ($I(a \star b) = I(a) \star I(b)$).

Ce n'est pas forcément le cas dans le traitement du langage où le sens peut être évalué en fonction d'éléments extérieurs (que l'on peut appeler contexte).

Afin qu'un ordinateur puisse calculer correctement par rapport à une telle manière de définir la sémantique, il est nécessaire de rendre ce contexte objectif. C'est le cas dans les logiques modales où le sens s'apprécie en fonction du contenu des formules et d'un ensemble de mondes possibles.

4.8 Modularité

La modularité telle que considérée ici est en rapport avec la construction des langages. Elle permet de définir ces langages à partir d'un ensemble de constructeurs qui peuvent être différents suivant les langages. Ajouter un constructeur dans un langage doté d'une sémantique compositionnelle n'est en général pas très difficile puisqu'il s'agit de fournir la règle d'interprétation du nouveau constructeur.

Cette modularité a été poussée très loin dans les logiques terminologiques. Elle est non seulement utilisée dans la définition de la sémantique mais aussi dans la description de certains algorithmes de test de subsomption où une règle est ajoutée par constructeurs.

Chapitre 5

Conclusions et perspectives

5.1 Conclusions

Ici, il faudrait comparer les formalismes présentés ainsi que la manière d'établir la sémantique.

En ce qui concerne les formalismes, il semble clair que les logiques terminologiques ont une place à part car le formalisme est beaucoup plus extensible que les autres. Le compromis expressivité-efficacité y prend tout son sens. Il existe cependant différentes dimensions pour comparer les langages présentés :

terminologie/assertion Les logiques de descriptions sont principalement terminologique et offrent le support minimal pour l'aspect assertionnel ; les graphes conceptuels sont très peu terminologiques (surtout, quand comme nous l'avons fait, on oublie les types de concepts définis et les motifs) en particulier parce que tout graphe s'interprète existentiellement ; les Ψ -termes sont uniquement terminologiques (les entités syntaxiques ne s'interprètent que comme des ensembles).

classificateur/relationnel À l'inverse, les graphes conceptuels offrent la richesse de proposer des relations polyadiques alors que les autres formalismes sont plus orientés vers la définition de classes, ce qui peut se nuancer dans le cadre des Ψ -termes.

Mais de nombreuses autres différences font que la combinatoire des langages possibles est très importante :

arité des relations alors qu'elle est de deux dans les logiques terminologiques et les Ψ -termes, elle est quelconque dans les graphes conceptuels.

définition des attributs alors qu'elle est possible dans les logiques terminologiques et les Ψ -termes, elle est restreinte dans les graphes conceptuels (tant que les définitions ne sont pas prises en compte). Ceci a une incidence sur la possibilité de tester une subsomption entre attributs.

enracinement et connexité des objets l'enracinement est obligatoire dans les logiques terminologiques et les Ψ -termes, mais n'est pas prise en compte dans les graphes conceptuels. Ceci a une incidence sur le test de subsomption.

individus les logiques de descriptions identifient clairement un niveau assertionnel, les graphes conceptuels permettent de nommer les nœuds manipulés alors que les Ψ -termes ne conçoivent les individus que comme la limite des descriptions.

quantification Les assertions sont quantifiées universellement dans les logiques de description mais les quantificateurs existentiels apparaissent dans les définitions, elles sont universelles dans les Ψ -termes alors que les graphes conceptuels sont quantifiés existentiellement.

En ce qui concerne la définition de la sémantique, les techniques utilisables pour élaborer une sémantique dénotationnelle simple des représentations de connaissance par objets sont nombreuses et présentent une certaine variabilité. Il est important de faire intervenir diverses considérations dans le choix d'une approche ou d'une autre :

intelligibilité : au premier chef il est important que les «utilisateurs» à qui est destinée la sémantique, c'est-à-dire non seulement les développeurs de systèmes de représentation de connaissance mais surtout les développeurs de systèmes à base de connaissance, soient capable de la comprendre. Dans le cas contraire, les systèmes ne seront pas utilisés correctement.

modularité : Il faut pouvoir ajouter ou retirer facilement une composante dans un système sans avoir à reconstruire l'ensemble de la sémantique et à démontrer de nouveau ses propriétés.

adéquation : Il est important d'utiliser pour exprimer la sémantique un formalisme adéquat, c'est-à-dire ni trop puissant, ni trop étriqué.

5.2 Sémantique des systèmes d'objets

- signaler l'analogie entre les méthodes de spécification de systèmes à objets (UML, OMT) et les petits diagrammes que nous avons été amenés à tracer ici ou là. Ces méthodes manquent cruellement d'une véritable sémantique : elle n'est pas loin.
- les langages de programmation par objets ont un noyau apparemment commun avec les systèmes présentés ici. Il est important de pouvoir confronter la sémantique des systèmes de représentation de connaissance avec celle qui peut être donnée à un langage de programmation par objets. Pour cela, il est possible de considérer certains travaux aboutis :

[Ducournau, 1996] reprend la sémantique telle qu'elle est donnée ici pour les logiques terminologiques (chapitre 2) et l'adapte à de nombreuses constructions que l'on trouve dans les langages orientés-objets et les langages de représentation de connaissance par objets ;

[Kifer et al., 1995] formule une sémantique très élaborée qui devrait convenir à de nombreux langages orientés-objets ;

[Abadi and Cardelli, 1996] s'intéresse à une abstraction des objets, les types, et développe une sémantique opérationnelle et dénotationnelle pour le typage des objets.

La grande différence que l'on peut observer est la complexité des sémantiques utilisées par rapport à celles qui ont été présentées ici.

5.3 Aspects dynamiques des représentations de connaissance

Un sujet qui n'a pas été abordé dans ce court cours est l'aspect dynamique d'une représentation de connaissance. En effet, si la sémantique permet de fixer le cadre de ce que signifie une description exprimée dans l'un de ces formalismes, rien ne vient régir l'évolution de cette description. Une telle description peut évoluer pour plusieurs raisons :

l'objet décrit évolue règles dues à cette évolution.

la description est complétée juste moins de modèles.

la description est erronée

Dans les trois cas, un certain nombre de règles devraient permettre de décrire cette évolution et de garantir qu'elle évolue dans un cadre bien déterminé. Si la correction d'erreurs est le cadre de relativement nombreux travaux (en particulier dans le cadre de la révision de connaissance), l'aspect évolution mériterait des traitements plus systématiques.

Notions abordées

Voici la liste des notions abordées au cours de l'exposé et dont la définition n'est pas donnée (les items suivis d'une étoile sont présentés dans [Arnold and Guessarian, 1992]) :

- BNF ou “Backus-Naur form” (§2.2.2) ;
- Graphes (§3.2, 4.2.3, *) ;
- Ordres/trellis (§3.2, 4.2.1, *) ;
- Restriction ($|$) (§2.3.2) ;
- Réécriture (§2.4.2, 4.2.2) ;
- Complexité (§2.5, 3.6, *) ;
- Point fixe (§2.6.1, *) ;
- Calcul des prédicats/Herbrand/Skolem (§3.5, *) ;
- Catégories (§4.4) ;

Correction des exercices

Les exercices sont repris ici avec leurs énoncés mais sans leurs figures originales.

6.2 Logiques terminologiques

Exercice 2.1 (*). Soient les assertions de subsomption suivantes :

1. $(\text{and Rectangle Losange}) \preceq \text{Rectangle}$
2. $(\text{all angle Droit}) \preceq \text{Rectangle}$
3. $(\text{atleast 3 angle}) \preceq (\text{atleast 4 angle})$
4. $(\text{some angle}) \preceq (\text{atleast 1 angle})$
5. $(\text{atleast 1 angle}) \preceq (\text{some angle})$
6. $(\text{and (atleast 4 angle) (atmost 4 angle) (all angle Droit)}) \preceq (\text{and (some angle) (all angle Droit)})$

1. dites si elles vous semblent vérifiées ;

Seules la seconde et la troisième ne sont pas vérifiées.

2. utilisez l'algorithme de SUBSUMPTION-TEST pour l'établir.

```
SUBSUMPTION-TEST((and Rectangle Losange),Rectangle)
= COMPARE(NORM((and Rectangle Losange)),NORM(Rectangle))
= COMPARE((and Rectangle Losange),Rectangle)
= COMPARE(Rectangle,Rectangle) V COMPARE(Losange,Rectangle)
= vrai V ...
= vrai
```

```
SUBSUMPTION-TEST((all angle Droit),Rectangle)
= COMPARE(NORM((all angle Droit)),NORM(Rectangle))
= COMPARE((all angle Droit),Rectangle)
= faux (pas de règle applicable)
```

```
SUBSUMPTION-TEST((atleast 3 angle),(atleast 4 angle))
= COMPARE(NORM((atleast 3 angle)),NORM((atleast 4 angle)))
= COMPARE((atleast 3 angle),(atleast 4 angle))
= faux (pas de règle applicable, atleast $\preceq$  ne satisfait pas
la condition)
```

```
SUBSUMPTION-TEST((some angle),(atleast 1 angle))
```

```

= SUBSUMPTION-TEST((atleast 1 angle),(atleast 1 angle))
= COMPARE(NORM((atleast 1 angle)),NORM((atleast 1 angle)))
= COMPARE((atleast 1 angle),(atleast 1 angle))
= COMPARE(angle,angle) (1≤1)
= vrai

SUBSUMPTION-TEST((atleast 1 angle),(some angle))
= SUBSUMPTION-TEST((atleast 1 angle),(atleast 1 angle))
= vrai (résultat précédent)

SUBSUMPTION-TEST((and (atleast 4 angle) (atmost 4 angle)
                    (all angle Droit)),
                  (and (some angle) (all angle Droit)))
= SUBSUMPTION-TEST((and (atleast 4 angle) (atmost 4 angle)
                    (all angle Droit)),
                  (and (atleast 1 angle) (all angle Droit)))
= COMPARE(NORM((and (atleast 4 angle) (atmost 4 angle),
                    (all angle Droit)),
              NORM((and (atleast 1 angle) (all angle Droit))))
= COMPARE((and (atleast 4 angle) (atmost 4 angle)
              (all angle (and Droit))),
          (and (atleast 1 angle) (all angle (and Droit))))
= COMPARE((and (atleast 4 angle) (atmost 4 angle)
              (all angle (and Droit))),
          (atleast 1 angle))
∧ COMPARE((and (atleast 4 angle) (atmost 4 angle)
              (all angle (and Droit))),
          (all angle (and Droit)))
= (COMPARE((atleast 4 angle),(atleast 1 angle))∨
   COMPARE((atmost 4 angle),(atleast 1 angle))∨
   COMPARE((all angle (and Droit)),(atleast 1 angle)))
∧
  (COMPARE((atleast 4 angle),(all angle (and Droit)))∨
   COMPARE((atmost 4 angle),(all angle (and Droit)))∨
   COMPARE((all angle (and Droit)),(all angle (and Droit))))
= COMPARE((atleast 4 angle),(atleast 1 angle))∧
  COMPARE((all angle (and Droit)),(all angle (and Droit)) (car 1≤1))
= COMPARE(angle,angle)∧
  COMPARE(angle,angle)∧ COMPARE((and Droit),(and Droit))
= vrai ∧ vrai ∧ COMPARE((and Droit),Droit)
= COMPARE(Droit,Droit)
= vrai

```

Exercice 2.2 (*). Soit la terminologie suivante (taxonomie imaginaire des judokas) exprimée dans $\mathcal{ALN}\mathcal{R}$:

```

Combat      ≐ Anything
AuxPoints   ≐ Combat

```

```

ParIpon      ≐ Combat
AuxPoints    ⊕ ParIpon
Judoka       ≐ Anything
combat       ≐ anyrelation
victoire     ≐ combat
defaite      ≐ combat
Competiteur  ≐ (and Judoka (atleast 1 victoire))
Maitre       ≐ (and Judoka (atleast 11 victoire)
               (all victoire ParIpon))
Disciple     ≐ (and Judoka (atleast 1 victoire)
               (all defaite AuxPoints))

```

Refaire tous les exemples qui ont été donnés dans le cours en terme d'application des algorithmes :

1. Donnez ses sous-ensembles \mathcal{N}_C , \mathcal{N}_R et \mathcal{N}_O ;
 $\mathcal{N}_C = \{\text{Combat, AuxPoints, ParIpon, Judoka, Competiteur, Maitre, Disciple}\}$
 $\mathcal{N}_R = \{\text{combat, victoire, defaite}\}$ \mathcal{N}_O n'est pas applicable à une terminologie.
2. Normalisez la terminologie ;

```

AuxPoints    ≐ (and Combat AuxPoints)
ParIpon      ≐ (and Combat ParIpon (anot AuxPoints))
victoire     ≐ (androle combat victoire)
defaite      ≐ (androle combat defaite)
Competiteur  ≐ (and Judoka (atleast 1 victoire))
Maitre       ≐ (and Judoka (atleast 11 victoire)
               (all victoire ParIpon))
Disciple     ≐ (and Judoka (atleast 1 victoire)
               (all defaite AuxPoints))

```

3. Calculez l'expansion de l'expression (and Competiteur (all combat AuxPoints)) dans la terminologie ;

```

EXP((and Competiteur (all combat AuxPoints)),  $\bar{T}$ )
= (and EXP(Competiteur,  $\bar{T}$ ), EXP((all combat AuxPoints),  $\bar{T}$ ))
= (and EXP(Competiteur,  $\bar{T}$ ), EXP((all combat AuxPoints),  $\bar{T}$ ))
= (and EXP((and Judoka (atleast 1 victoire)),  $\bar{T}$ )
   (all EXP(combat,  $\bar{T}$ ) EXP(AuxPoints,  $\bar{T}$ )))
= (and (and EXP(Judoka,  $\bar{T}$ ) EXP((atleast 1 victoire),  $\bar{T}$ ))
   (all EXP(combat,  $\bar{T}$ ) EXP((and Combat AuxPoints),  $\bar{T}$ )))
= (and (and Judoka (atleast 1 EXP(victoire,  $\bar{T}$ ))
   (all combat (and EXP(Combat,  $\bar{T}$ ) EXP(AuxPoints,  $\bar{T}$ ))))
= (and (and Judoka (atleast 1 EXP((androle combat victoire),  $\bar{T}$ ))
   (all combat (and Combat AuxPoints)))
= (and (and Judoka (atleast 1 (androle EXP(combat,  $\bar{T}$ ) EXP(victoire,  $\bar{T}$ ))))
   (all combat (and Combat AuxPoints)))
= (and (and Judoka (atleast 1 (androle combat victoire)))
   (all combat (and Combat AuxPoints)))

```

4. Normalisez l'expression ainsi obtenue ;

$$\begin{aligned} & \text{NORM}((\text{and} (\text{and} \text{Judoka} (\text{atleast } 1 (\text{androle} \text{combat} \overline{\text{victoire}}))) \\ & \quad (\text{all} \text{combat} (\text{and} \text{Combat} \overline{\text{AuxPoints}})))))) \\ = & \text{NORM}((\text{and} \text{Judoka} (\text{atleast } 1 (\text{androle} \text{combat} \overline{\text{victoire}})) \\ & \quad (\text{all} \text{combat} (\text{and} \text{Combat} \overline{\text{AuxPoints}})))))) \\ = & (\text{and} \text{Judoka} (\text{atleast } 1 (\text{androle} \text{combat} \overline{\text{victoire}})) \\ & \quad (\text{all} \text{combat} (\text{and} \text{Combat} \overline{\text{AuxPoints}})))) \end{aligned}$$

5. Que peut-on en conclure pour la comparaison de cette expression avec les concepts *Disciple* et *Maitre* ?

On peut appliquer l'algorithme de comparaison et on obtiendra que :

- $(\text{and} \text{Competiteur} (\text{all} \text{combat} \text{AuxPoints})) \preceq \text{Disciple et}$
- $(\text{and} \text{Competiteur} (\text{all} \text{combat} \text{AuxPoints})) \perp \text{Maitre.}$

Exercice 2.3 ().** Montrez que dans la définition 2.4.2 la règle *role-bottom-propagation* est redondante.

Si *role-bottom-propagation* se déclenche, c'est que le terme comprend une expression du type :

$$(\text{and} \dots (\text{all } r \text{ Nothing}) \dots (\text{atleast } n \ r') \dots)$$

Mais celle-ci peut aussi donner lieu au déclenchement de *bottom-introduction* qui la récriera en :

$$(\text{and} \dots (\text{atmost } 0 \ r) \dots (\text{atleast } n \ r') \dots)$$

Ce dernier terme pourra lui-même être réécrit par *card-bottom-introduction* en :

$$\text{Nothing}$$

car si *role-bottom-propagation* peut se déclencher, alors $n > 0$, ce qui garanti le déclenchement de cette dernière règle. Le résultat est celui donné aussi par *role-bottom-propagation*.

Exercice 2.4 ().** Vérifier que \mathcal{ALC} et \mathcal{ALUE} sont équivalents (par les constructeurs).

Il est aisé de vérifier que l'on obtient la disjonction (*or*) à l'aide de la négation et de la conjonction. Comme en logique on a :

$$(\text{or } c_1 \dots c_n) \equiv (\text{not } (\text{and } (\text{not } c_1) \dots (\text{not } c_n)))$$

De même on peut aisément établir que :

$$(\text{csume } r \ c) \equiv (\text{not } (\text{all } r \ (\text{not } c)))$$

Ceci prouve que \mathcal{ALUE} peut être obtenue à partir de \mathcal{ALC} . à l'inverse, il est moins facile d'établir que la négation peut être obtenue à partir des autres constructeurs. En fait, il est

possible de prouver sur les formules de \mathcal{ALC} que l'on peut toujours remplacer cette négation :

$$\begin{aligned}
 (\text{not } c) &\equiv (\text{anot } c) && \text{pour } c \in \mathcal{N}_C \\
 (\text{not } (\text{anot } c)) &\equiv c \\
 (\text{not } (\text{not } c)) &\equiv c \\
 (\text{not } (\text{all } r c)) &\equiv (\text{c} \text{some } r (\text{not } c)) \\
 (\text{not } (\text{and } c_1 \dots c_n)) &\equiv (\text{or } (\text{not } c_1) \dots \text{not } c_n) \\
 (\text{not } (\text{some } r)) &\equiv (\text{all } r \text{Nothing})
 \end{aligned}$$

Les parties droites ne contiennent que des formules contenant des constructeurs de \mathcal{ALUE} et des `not` plus imbriqués. Ultimement, les `not` subsistants vont porter sur des concepts primitifs et seront remplacés par des `anot`.

Exercice 2.5 ().** Vérifier qu'une logique terminologique disposant des seuls constructeurs `all`, `or` et `not` permettent la construction de `range`.

Il faut remarquer que le constructeur `range` ne peut être utilisé que dans sa portée ou celle du constructeur `all`, ce qui revient à dire qu'il ne peut apparaître que dans la portée du `all`. On peut définir ce contexte par l'équation :

$$(\text{all } (\text{range } r c) d) \equiv (\text{all } r (\text{or } (\text{not } c) d))$$

qui peut être utilisée jusqu'à l'élimination totale du `range`.

On peut noter ici l'intérêt de cette `fac`, on de faire : réduire un constructeur sur les rôles à des constructeurs sur les concepts permet de simplifier les expressions du langage tout en disposant de la même puissance. Bien entendu, cela n'est pas toujours possible.

Exercice 2.6 ().** Montrez que l'expansion (par `EXP`) d'un terme dans une \mathcal{ALNR} -terminologie peut rendre un terme de taille exponentielle en fonction de la terminologie.

Il suffit de construire la terminologie suivante :

$$\begin{aligned}
 a_n &\doteq (\text{and } a_{n-1} a_{n-1}) \\
 \dots & \\
 a_1 &\doteq (\text{and } a_0 a_0)
 \end{aligned}$$

L'expansion du terme a_n est alors composé de 2^n atomes et $2^n - 1$ `and`. Ceci pour une terminologie à n termes (contenant n `and` et $2 \times n$ termes atomiques).

Exercice 2.7 (*)**. Refaire le travail de ce chapitre en introduisant la composition de rôles :

$$\mathcal{E}((\text{comp } r_1 \dots r_n)) = \mathcal{E}(r_1) \circ \dots \mathcal{E}(r_n)$$

permettant d'exprimer :

$$\begin{aligned}
 \text{grandmere} &\doteq (\text{comp mere parent}) \\
 \text{GrandMere} &\doteq (\text{and Femme } (\text{atleast } 1 (\text{comp enfant enfant})))
 \end{aligned}$$

On procédera ainsi :

1. Introduire comp dans la syntaxe de \mathcal{FLN} ;
 comp étant un constructeur de rôle, il faut introduire les rôles construits :
 $\langle \text{role} \rangle := \langle \text{atomic-role} \rangle$
 $\quad \mid \text{' (' comp } \langle \text{role} \rangle + \text{')'}$
2. Introduire comp dans sa sémantique (fonction d'extension) ;
 La réponse fait partie de l'énoncé :

$$\mathcal{E}((\text{comp } r_1 \dots r_n)) = \mathcal{E}(r_1) \circ \dots \circ \mathcal{E}(r_n)$$

3. Considérer comp dans l'algorithme de mise en forme normale (et l'introduire dans la forme normale) ;
 Il n'y a pas a priori de raison de supprimer comp de la forme normale.
4. établir les propriétés de complexité de l'algorithme et d'équivalence des formes obtenues ;
 Ce sont les mêmes problèmes que précédemment.
5. Prendre comp en compte dans les algorithmes EXP , NORM et COMPARE ;
 EXP n'est pas changée par l'introduction de comp qui est un constructeur comme un autre.
 Si l'on veut avoir des performances acceptables dans la suite il est bon de normaliser les termes. Il va donc falloir ajouter des règles à la normalisation. Par exemple, il est possible de décider que les androle doivent toujours englober les comp . Ceci conduit à ajouter les trois règles suivantes :

$$\frac{(\text{comp } \dots (\text{comp } r_1 \dots r_n) \dots)}{(\text{comp } \dots r_1 \dots r_n \dots)} \quad [\text{comp-reduction}]$$

$$\frac{(\text{comp } r)}{r} \quad [\text{comp-cleanup}]$$

$$\frac{(\text{comp } \dots (\text{androle } r_1 \dots r_n) \dots)}{(\text{androle } (\text{comp } \dots r_1 \dots) \dots (\text{comp } \dots r_n \dots))} \quad [\text{comp-androle-normalisation}]$$

En ce qui concerne la comparaison des compositions, elle ne peut se faire qu'entre composition (pour des termes normalisés) :

$$\frac{\text{COMPARE}((\text{comp } r_1 \dots r_n), (\text{comp } s_1 \dots s_n))}{\text{COMPARE}(r_1, s_1) \wedge \text{COMPARE}(r_n, s_n)} \quad [\text{comp } \preceq]$$

6. établir les propriétés de ces algorithmes.

CORRECTION NON DISPONIBLE

Exercice 2.8 (*). On se place dans le contexte d'une logique terminologique construite à l'aide des constructeurs all , and et range . La sémantique du constructeur de rôles range est donnée dans la définition 2.30. Cette logique sera nommée \mathcal{FL}^{--} . On considèrera la terminologie munie d'une unique clause :

$T = \{ \text{chef} \dot{\leq} \text{musicien} \}$

et les termes suivants :

$t_1 \doteq (\text{and Orchestre (all chef Pianiste)})$

$t_2 \doteq (\text{and Orchestre (all chef Mélophobe)} \\ (\text{all (range musicien Mélophobe) Pianiste}))$

1. donnez une définition en langage naturel des termes t_1 et t_2 :
 t_1 est une sorte d'orchestre où tous les chefs sont pianistes ;
 t_2 est une sorte d'orchestre où tous les chefs sont mélophobes (n'aiment pas la musique) et tous les musiciens mélophobes sont pianistes.
2. donnez les sous-ensembles \mathcal{N}_C et \mathcal{N}_R de \mathcal{FL}^{--} :
 $\mathcal{N}_C = \{\text{Orchestre, Pianiste, Mélophobe}\}$
 $\mathcal{N}_R = \{\text{musicien, chef}\}$
 Attention, maintenant, $\mathcal{FL}\mathcal{N}_R$ est différent de \mathcal{N}_R car (range musicien Mélophobe) est un rôle construit.
3. donnez l'extension des termes t_1 et t_2 en fonction de l'extension des concepts et rôles atomiques :

$$\mathcal{E}(t_1) = \mathcal{E}(\text{Orchestre}) \cap \{x \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(\text{chef}) \Rightarrow y \in \mathcal{E}(\text{Pianiste})\}$$

$$\mathcal{E}(t_2) = \mathcal{E}(\text{Orchestre}) \cap \{x \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(\text{chef}) \Rightarrow y \in \mathcal{E}(\text{Mélophobe})\}$$

$$\cap \{x \in \mathcal{D} \mid \langle x, y \rangle \in \{\langle x, y \rangle \mid \langle x, y \rangle \in \mathcal{E}(\text{musicien}) \wedge y \in \mathcal{E}(\text{Mélophobe})\} \\ \Rightarrow y \in \mathcal{E}(\text{Pianiste})\}$$

$$= \mathcal{E}(\text{Orchestre}) \cap \{x \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(\text{chef}) \Rightarrow y \in \mathcal{E}(\text{Mélophobe})\}$$

$$\cap \{x \in \mathcal{D} \mid \exists y, \langle x, y \rangle \in \mathcal{E}(\text{musicien}) \wedge y \in \mathcal{E}(\text{Mélophobe}) \Rightarrow y \in \mathcal{E}(\text{Pianiste})\}$$

4. montrez à l'aide de cette interprétation que $\mathcal{E}(t_2) \subseteq \mathcal{E}(t_1)$ pour toute \mathcal{FL}^{--} -extension \mathcal{E} pour T :

$$\mathcal{E}(t_2) = \mathcal{E}(\text{Orchestre}) \cap \{x \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(\text{chef}) \Rightarrow y \in \mathcal{E}(\text{Mélophobe})\}$$

$$\cap \{\langle x, y \rangle \mid \langle x, y \rangle \in \{\langle x, y \rangle \mid \langle x, y \rangle \in \mathcal{E}(\text{musicien}) \wedge y \in \mathcal{E}(\text{Mélophobe})\} \\ \Rightarrow y \in \mathcal{E}(\text{Pianiste})\}$$

$$= \mathcal{E}(\text{Orchestre}) \cap$$

$$\{x \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(\text{chef}) \Rightarrow y \in \mathcal{E}(\text{Mélophobe})\}$$

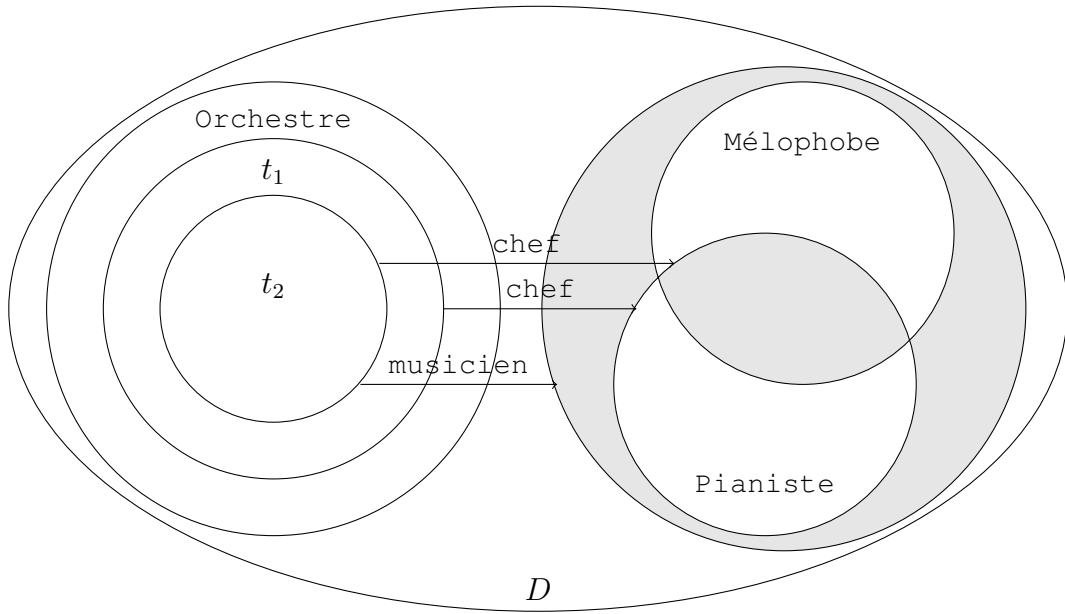
$$\wedge (\langle x, y \rangle \in \{\langle x, y \rangle \mid \langle x, y \rangle \in \mathcal{E}(\text{musicien}) \wedge y \in \mathcal{E}(\text{Mélophobe})\}) \Rightarrow y \in \mathcal{E}(\text{Pianiste})\}$$

Or $\mathcal{E}(\text{chef}) \subseteq \mathcal{E}(\text{musicien})$, donc :

$$\mathcal{E}(t_2) \subseteq \mathcal{E}(\text{Orchestre}) \cap \{x \in \mathcal{D} \mid \langle x, y \rangle \in \mathcal{E}(\text{chef}) \Rightarrow y \in \mathcal{E}(\text{Pianiste})\}$$

$$= \mathcal{E}(t_1)$$

Ceci est décrit par la figure suivante :



5. que peut-on penser de $t_2 \preceq_T t_1$?
Si $\mathcal{E}(t_2) \subseteq \mathcal{E}(t_1)$ alors $t_2 \preceq_T t_1$.

Exercice 2.9 ().** On se propose, à l'instar de ce qui se fait dans les graphes conceptuels, de traduire les termes de la logique terminologique \mathcal{FL}^{--} de l'exercice précédent en calcul des prédicats. Pour cela on va associer à chaque concept atomique un prédicat monadique (d'arité 1) et à chaque rôle atomique un prédicat diadique (d'arité 2). On va ensuite définir la fonction ϕ paramétrée, par une variable si son argument est un concept et par deux variables si c'est un rôle, de telle sorte que :

$$\begin{aligned}\phi_x(t_1) &= \text{Orchestre}(x) \wedge \forall y, \text{chef}(x, y) \Rightarrow \text{Pianiste}(y) \\ \phi_x(t_2) &= \text{Orchestre}(x) \wedge [\forall y, \text{chef}(x, y) \Rightarrow \text{Mélophobe}(y)] \\ &\quad \wedge [\forall y, (\text{musicien}(x, y) \wedge \text{Mélophobe}(y)) \Rightarrow \text{Pianiste}(y)]\end{aligned}$$

1. donnez une définition analytique de ϕ_x et $\phi_{x,y}$:

$$\begin{aligned}\text{si } c \in \mathcal{N}_C, \phi_x(c) &= c(x) \\ \text{si } r \in \mathcal{N}_R, \phi_{x,y}(r) &= r(x, y) \\ \phi_x(\text{all } r \ c) &= \forall y, \phi_{x,y}(r) \Rightarrow \phi_y(c) \\ \phi_x(\text{and } c_1 \dots c_n) &= \phi_x(c_1) \wedge \dots \wedge \phi_x(c_n) \\ \phi_x(\text{range } r \ c) &= \phi_{x,y}(r) \wedge \phi_y(c)\end{aligned}$$

2. montrez par induction sur vos définitions que pour toute \mathcal{FL}^{--} -structure $\langle \mathcal{D}, \mathcal{E} \rangle$ de T ,

$$\text{si } a \in \mathcal{E}(t) \text{ alors } A_T \models \phi_a(t)$$

si

$$\begin{aligned}\forall c \in \mathcal{N}_C, \forall a, a \in \mathcal{E}(c) &\stackrel{\text{def}}{=} A_T \models \phi_a(c) \\ \text{et } \forall r \in \mathcal{N}_R, \forall a, b, \langle a, b \rangle \in \mathcal{E}(r) &\stackrel{\text{def}}{=} A_T \models \phi_{a,b}(r);\end{aligned}$$

Cas de base :

$$a \in \mathcal{E}(c) \stackrel{ind.}{\equiv} \phi_a(c)$$

$$\langle a, b \rangle \in \mathcal{E}(r) \stackrel{ind.}{\equiv} \phi_{a,b}(r)$$

Preuves inductives :

$$a \in \mathcal{E}(\text{and } c_1 \dots c_n) \equiv a \in \mathcal{E}(c_1) \cap \dots \mathcal{E}(c_n)$$

$$\stackrel{ind.}{\equiv} \phi_a(c_1) \wedge \dots \phi_a(c_n)$$

$$\equiv \phi_a(\text{and } c_1 \dots c_n)$$

$$a \in \mathcal{E}(\text{all } r \ c) \equiv a \in \{x \mid \langle x, y \rangle \in \mathcal{E}(r) \Rightarrow y \in \mathcal{E}(c)\}$$

$$\stackrel{ind.}{\equiv} a \in \{x \mid \forall y, \phi_{x,y}(r) \Rightarrow \phi_y(c)\}$$

$$\equiv \forall y, \phi_{a,y}(r) \Rightarrow \phi_y(c)$$

$$\equiv \phi_a(\text{all } r \ c)$$

$$\langle a, b \rangle \in \mathcal{E}(\text{range } r \ c) \equiv \langle a, b \rangle \in \{\langle x, y \rangle \mid \langle x, y \rangle \in \mathcal{E}(r) \wedge y \in \mathcal{E}(c)\}$$

$$\equiv \langle a, b \rangle \in \mathcal{E}(r) \wedge b \in \mathcal{E}(c)$$

$$\stackrel{ind.}{\equiv} \phi_{a,b}(r) \wedge \phi_b(c)$$

$$\equiv \phi_{a,b}(\text{range } r \ c)$$

Il en va de même pour les axiomes car si $(c \dot{\leq} c') \in T$ alors $(\forall x, \phi_x(c) \Rightarrow \phi_x(c')) \in A_T$.

3. que dire de “si $t \dot{\leq}_T t'$ alors $A_T \models \forall x, \phi_x(t) \Rightarrow \phi_x(t')$ ” ?

Cela est clairement vrai car :

$$t \dot{\leq}_T t' \equiv \mathcal{E}(t) \subseteq \mathcal{E}(t') \quad \text{pour tout } \langle \mathcal{D}, \mathcal{E} \rangle \text{ } \mathcal{FL}^{--}\text{-structure de } T$$

$$\equiv \forall x, x \in \mathcal{E}(t) \Rightarrow x \in \mathcal{E}(t') \quad \text{pour tout } \langle \mathcal{D}, \mathcal{E} \rangle \text{ } \mathcal{FL}^{--}\text{-structure de } T$$

$$\equiv A_T \models \forall x, \phi_x(t) \Rightarrow \phi_x(t') \quad \text{par la question précédente}$$

4. qu'en est-il dans le sens inverse ?

La preuve ne faisant appel qu'à des équivalences, sa réciproque est vraie : “si $A_T \models \forall x, \phi_x(t) \Rightarrow \phi_x(t')$ alors $t \dot{\leq}_T t'$ ”

5. on construira la formule associée à une assertion à l'aide de cette fonction ϕ . Par exemple,

$$\phi(t \dot{\leq} t') \stackrel{\text{def}}{=} \forall x, \phi_x(t) \Rightarrow \phi_x(t')$$

Donnez les règles de construction pour les autres symboles :

$$\phi(t \dot{=} t') \stackrel{\text{def}}{=} \forall x, \phi_x(t) \Leftrightarrow \phi_x(t')$$

$$\phi(t \dot{\oplus} t') \stackrel{\text{def}}{=} \forall x, (\phi_x(t) \wedge \neg \phi_x(t')) \vee (\neg \phi_x(t) \wedge \phi_x(t'))$$

$$\phi(t \dot{\leq} t') \stackrel{\text{def}}{=} \forall x, y, \phi_{x,y}(t) \Rightarrow \phi_{x,y}(t')$$

$$\phi(t \dot{=} t') \stackrel{\text{def}}{=} \forall x, y, \phi_{x,y}(t) \Leftrightarrow \phi_{x,y}(t')$$

$$\phi(t \dot{\oplus} t') \stackrel{\text{def}}{=} \forall x, y, (\phi_{x,y}(t) \wedge \neg \phi_{x,y}(t')) \vee (\neg \phi_{x,y}(t) \wedge \phi_{x,y}(t'))$$

6. construisez ainsi l'ensemble d'axiomes A_T qui va être associé à la terminologie T :

$$A_T = \{\forall x, y, \text{chef}(x, y) \Rightarrow \text{musicien}(x, y)\}$$

7. en s'appuyant sur les définition 2.9 et conséquence 2.3 du cours, donnez la définition d'équivalence, d'exclusion et d'incohérence en fonction de la traduction vers le calcul des prédicats (mais sans utiliser le concept `Nothing`) :

$$\begin{aligned} t \approx_T t' &\text{ ssi } A_T \models \forall x, \phi_x(t) \Leftrightarrow \phi_x(t') \\ t \oplus_T t' &\text{ ssi } A_T \models \forall x, (\phi_x(t) \wedge \neg \phi_x(t')) \vee (\neg \phi_x(t) \wedge \phi_x(t')) \\ t \perp_T &\text{ ssi } A_T \models \forall x, \neg \phi_x(t) \end{aligned}$$

ou alternativement

$$t \preceq_T t' \text{ ssi } A_T \models \forall x, \phi_x(t) \Rightarrow \phi_x(t')$$

Exercice 2.10 (*)**. À partir d'ici on ne considère que \preceq et \doteq pour définir les \mathcal{FL}^{--} -terminologies. On cherche maintenant à construire une procédure de décision pour des formules du type $t \preceq_T t'$, c'est-à-dire capable de prouver $A_T \models \forall x, \phi_x(t) \Rightarrow \phi_x(t')$.

1. donnez une preuve logique valide de $A_T \models \forall x, \phi_x(t_2) \Rightarrow \phi_x(t_1)$:

$$\begin{aligned} A_T &\models \forall x, \phi_x(t_2) \\ A_T &\models \forall x, y, (\text{chef}(x, y) \Rightarrow \text{Mélophobe}(y)) \wedge (\text{musicien}(x, y) \wedge \text{Mélophobe}(y) \Rightarrow \text{Pianiste}(y)) \\ A_T &\models \forall x, y, (\text{chef}(x, y) \Rightarrow \text{Mélophobe}(y)) \wedge (\text{chef}(x, y) \wedge \text{Mélophobe}(y) \Rightarrow \text{Pianiste}(y)) \\ A_T &\models \forall x, y, (\neg \text{chef}(x, y) \vee \text{Mélophobe}(y)) \wedge (\neg \text{chef}(x, y) \vee \neg \text{Mélophobe}(y) \vee \text{Pianiste}(y)) \\ A_T &\models \forall x, y, \neg \text{chef}(x, y) \vee (\text{Mélophobe}(y) \wedge (\neg \text{Mélophobe}(y) \vee \text{Pianiste}(y))) \\ A_T &\models \forall x, y, \neg \text{chef}(x, y) \vee (\text{Mélophobe}(y) \wedge \text{Pianiste}(y)) \\ A_T &\models \forall x, y, \neg \text{chef}(x, y) \vee \text{Pianiste}(y) \\ A_T &\models \forall x, y, \text{chef}(x, y) \Rightarrow \text{Pianiste}(y) \\ A_T &\models \forall x, \phi_x(t_1) \end{aligned}$$

On peut même chercher une preuve plus simple :

$$\frac{\frac{\forall x, y, \text{chef}(x, y) \Rightarrow \text{Mélophobe}(y) \quad \forall x, y, \text{chef}(x, y) \Rightarrow \text{musicien}(x, y)}{\forall x, y, \text{chef}(x, y) \Rightarrow \text{Mélophobe}(y) \wedge \text{musicien}(x, y)} \quad \forall x, y, \text{Mélophobe}(y) \wedge \text{musicien}(x, y) \Rightarrow \text{Pianiste}(y)}{\forall x, y, \text{chef}(x, y) \Rightarrow \text{Pianiste}(y)}$$

2. peut-on réutiliser les résultats concernant les graphes conceptuels et pourquoi ?

On ne peut réutiliser les résultats obtenus sur les graphes conceptuels parce que l'on n'est plus en présence de formes clausales (et en particulier de clauses de Horn).

3. montrez que pour toute description de *concept* t , $\phi_x(t)$ peut être mis sous la forme ¹ :

$$\begin{aligned} &\bigwedge_{i=1}^n \phi_x(c_i) \\ &\wedge \bigwedge_{i=1}^m \forall y, [\phi_{x,y}(r_i) \wedge \bigwedge_{j=1}^{p_i} \phi_y(c'_{i,j})] \Rightarrow \phi_y(c'_i) \end{aligned}$$

¹C'est-à-dire qu'il existe une transformation de $\phi_x(t)$ valide pour le calcul des prédicats ou au moins valide pour le type de formules considérées.

où les c_i sont des concepts atomiques, les r_i des rôles atomiques et les $c'_{i,j}$ et c'_i des descriptions de concepts quelconques ;

Ceci peut se montrer par induction sur la forme de la formule. Il faut que l'induction se fasse sur la forme des descriptions de concepts et de rôles. Ces derniers ayant la forme :

$$\phi_{x,y}(r) \wedge \bigwedge_{j=1}^p \phi_y(c_j)$$

Cas de base :

$$t \in \mathcal{N}_C \quad \phi_x(t) = t(x)$$

$$r \in \mathcal{N}_R \quad \phi_{x,y}(r) = r(x, y)$$

Preuves inductives (en commençant par les constructeurs de rôles) :

$$\begin{aligned} \phi((\text{range } r \ c)) &= \phi_{x,y}(r) \wedge \phi_y(c) \\ &\stackrel{ind.}{\equiv} \phi_{x,y}(r_i) \wedge \bigwedge_{j=1}^{p_i} \phi_y(c'_{i,j}) \wedge \phi_y(c) \end{aligned}$$

$$\begin{aligned} \phi((\text{and } t_1 \dots t_q)) &= \phi(t_1) \wedge \dots \wedge \phi(t_q) \\ &\stackrel{ind.}{\equiv} \bigwedge_{i=1}^{n_1} \phi_x(c_i) \wedge \bigwedge_{i=1}^{m_1} \forall y, [\phi_{x,y}(r_i) \wedge \bigwedge_{j=1}^{p_i} \phi_y(c'_{i,j})] \Rightarrow \phi_y(c'_i) \\ &\quad \wedge \dots \wedge \bigwedge_{i=1}^{n_q} \phi_x(c_i) \wedge \bigwedge_{i=1}^{m_q} \forall y, [\phi_{x,y}(r_i) \wedge \bigwedge_{j=1}^{p_i} \phi_y(c'_{i,j})] \Rightarrow \phi_y(c'_i) \\ &= \bigwedge_{i=1}^{n_1} \phi_x(c_i) \wedge \dots \wedge \bigwedge_{i=1}^{n_q} \phi_x(c_i) \\ &\quad \wedge \bigwedge_{i=1}^{m_1} \forall y, [\phi_{x,y}(r_i) \wedge \bigwedge_{j=1}^{p_i} \phi_y(c'_{i,j})] \Rightarrow \phi_y(c'_i) \wedge \dots \\ &\quad \wedge \bigwedge_{i=1}^{m_q} \forall y, [\phi_{x,y}(r_i) \wedge \bigwedge_{j=1}^{p_i} \phi_y(c'_{i,j})] \Rightarrow \phi_y(c'_i) \end{aligned}$$

$$\begin{aligned} \phi((\text{all } r \ c)) &= \forall y, \phi_{x,y}(r) \Rightarrow \phi_y(c) \\ &\stackrel{ind.}{\equiv} \forall y, \phi_{x,y}(r_i) \wedge \bigwedge_{j=1}^{p_i} \phi_y(c'_{i,j}) \Rightarrow \phi_y(c) \end{aligned}$$

4. exprimez deux règles de réécriture permettant de transformer les formules du type présenté ci-dessus de sorte que :

règle-r=r si deux conjoints concernent le même rôle du concept alors un seul conjoint subsistera et sa partie droite (ce qui se trouve après le $\phi_{x,y}(r_i)$) sera conjointe à celle de l'autre conjoint (exemple : $\forall y, r(x, y) \wedge h(y)$ et $\forall y, r(x, y) \wedge h'(y)$ donnera $\forall y, r(x, y) \wedge h'(y) \wedge h(y)$);

règle-r≤r' si un conjoint concerne un rôle subsumé par celui d'un autre conjoint alors la partie droite du premier conjoint est à conjointre à la partie droite du second (exemple : $\forall y, r(x, y) \wedge h(y)$ et $\forall y, r'(x, y) \wedge h'(y)$ avec $r \leq r'$ donnera $[\forall y, r(x, y) \wedge h'(y) \wedge h(y)] \wedge [\forall y, r'(x, y) \wedge h'(y)]$)

$$\frac{\forall y, r(x, y) \wedge h(y) \wedge \dots, \forall y, r(x, y) \wedge h'(y)}{\forall y, r(x, y) \wedge h(y) \wedge h'(y) \wedge \dots} \quad [\mathbf{r=r}]$$

$$\frac{\forall y, r(x, y) \wedge h(y) \wedge \dots, \forall y, r'(x, y) \wedge h'(y)}{\forall y, r(x, y) \wedge h(y) \wedge h'(y) \wedge \dots, \forall y, r'(x, y) \wedge h'(y)} \quad r \dot{\leq} r' \quad [\mathbf{r \leq r'}]$$

Justifiez rapidement pourquoi ces règles sont correctes (respectent la sémantique du calcul des prédicats), confluent et terminent (si l'on marque les applications de la règle $r \leq r'$) pour des termes sans cycle et ne laissent qu'un seul conjoint par rôle :

La règle $(r=r')$ est valide dans le calcul des prédicats; la règle $(r \leq r')$ est valide car si $r \dot{\leq} r'$ alors $\forall x, y, r(x, y) \Rightarrow r'(x, y)$. Le système ainsi constitué termine si l'on n'applique pas deux fois la règle $(r \leq r')$ au même couple de formules (la règle $(r=r')$ réduisant toujours les formules) et conflue car la conjonction est commutative. S'il y a deux conjoints pour le même rôle, il est supprimé par la règle $(r=r')$.

5. montrez que l'on peut maintenant considérer chaque conjoint indépendamment des autres pour tester la subsomption (c'est-à-dire comparer chaque conjoint du subsumé potentiel avec ceux du subsumant potentiel). Proposer alors un ensemble de règles (du type de la définition 2.28 du cours) permettant de le faire (elles seront inévitablement récursives).

On peut considérer les conjoints de manière indépendante car pour les c_i atomiques s'il n'en existe pas un dans le candidat subsumant qui le subsume, il ne peut y avoir subsomption. Pour les rôles, chaque rôle contient exactement toutes les contraintes qui pèsent sur lui car elles ont été accumulées par la règle $(r \leq r')$.

On peut donc proposer la procédure suivante. En ce qui concerne la normalisation : mettre les deux termes en forme normale (à l'aide des règles ci-dessus). Le test de subsomption peut alors se faire conjoint par conjoint (il faut que pour chaque conjoint du candidat subsumé il existe un conjoint du candidat subsumant qui le subsume). Les règles utiles pour cela sont les suivantes :

$$\frac{\text{COMPARE}(t, t'_1 \wedge \dots t'_m)}{\text{COMPARE}(t, t'_1) \wedge \dots \text{COMPARE}(t, t'_m)} \quad [\wedge - \text{distrib}]$$

$$\frac{\text{COMPARE}(t_1 \wedge \dots t_n, t')}{\text{COMPARE}(t_1, t') \vee \dots \text{COMPARE}(t_n, t')} \quad [\wedge - \text{elim}]$$

$$\frac{\text{COMPARE}(c, c')}{\text{vrai}} c \dot{\leq} c' \quad [\dot{\leq} - \text{elim}]$$

$$\frac{\text{COMPARE}(\text{Nothing}, t')}{\text{vrai}} \quad [\text{Nothing} - \text{elim}]$$

$$\frac{\text{COMPARE}(\forall y, \phi_{x,y}(r) \wedge \psi_y, \forall y, \phi_{x,y}(r) \wedge \psi'_y)}{\text{COMPARE}(\psi_y, \psi'_y)} \quad [\text{rôle} - \text{elim}]$$

$$\frac{\text{COMPARE}(A \Rightarrow C, A' \Rightarrow C')}{\text{COMPARE}(A', A) \wedge \text{COMPARE}(C, C')} \quad [\Rightarrow - \text{elim}]$$

6. Quels sont les avantages et les inconvénients respectifs des logiques terminologiques par rapport aux graphes conceptuels au regard de la procédure que l'on vient de suivre ?

Il apparaît que les logiques terminologiques sont plus modulaires dans leur traitement (ceci se voit dans la construction de ϕ). Par contre, la structure des graphes conceptuels est plus adaptée à ce traitement : on parvient à des formules du calcul des prédicats assez simples.

6.3 Graphes conceptuels

Exercice 3.1 (*). Soient les graphes conceptuels de la figure 3.5. Refaire tous les exemples qui ont été donnés dans le cours en terme d'application des algorithmes.

1. Donnez un support plausible pour ces graphes ;

$$\langle \{ \text{vegetal}, \text{herbe}, \text{animal}, \text{herbivore}, \text{omnivore}, \text{homme}, \text{vache}, \dots \}, \\ \{ \text{mange} \}, \\ \{ *, \dots \emptyset \}, \\ \{ \}, \\ \{ \langle \{ \top \}, \{ \text{mange} \}, \langle \top, \text{mange} \rangle \} \} \rangle$$

$$\text{avec } \beta_1(\text{mange}) = \beta_2(\text{mange}) = \top.$$

$$\begin{aligned} \text{vegetal} &\leq \text{herbe} \leq \top \\ \text{vache} &\leq \text{herbivore} \leq \text{animal} \leq \top \\ \text{homme} &\leq \text{carnivore} \leq \text{animal} \\ \text{mange} &\leq_R^2 \top^2 \end{aligned}$$

2. Dites quel graphe se projette dans quel autre ;
 - H se projette dans G de deux manières ;
 - G ne se projette pas dans G .
3. Donnez une suite d'opérations de spécialisation permettant de passer de H à G ;

Restriction de omnivore:* en homme:*
 Fusion des deux nœuds concepts homme:*
 Suppression des deux relations mange jumelles pour le nœud obtenu
 Joint externe avec un graphe [vache:*]-mange-[herbe:*]
 Le résultat est G .

4. Donnez l'ensemble d'axiomes associé au support ;

$$\begin{aligned} \forall x, \text{animal}(x) &\Rightarrow \text{top}(x) & \forall x, \text{vegetal}(x) &\Rightarrow \text{top}(x) \\ \forall x, \text{herbivore}(x) &\Rightarrow \text{animal}(x) & \forall x, \text{omnivore}(x) &\Rightarrow \text{animal}(x) \\ \forall x, \text{herbe}(x) &\Rightarrow \text{vegetal}(x) & \forall x, \text{homme}(x) &\Rightarrow \text{omnivore}(x) \\ \forall x, \text{vache}(x) &\Rightarrow \text{herbivore}(x) & \forall x, y, \text{devore}(x, y) &\Rightarrow \text{mange}(x, y) \end{aligned}$$

5. Donnez $\phi(H)$;

$$\begin{aligned} \exists x, y, z, w; \text{vegetal}(w) \wedge \text{vache}(x) \wedge \text{homme}(y) \wedge \text{omnivore}(z) \\ \wedge \text{mange}(y, x) \wedge \text{mange}(z, x) \wedge \text{mange}(x, w) \end{aligned}$$

6. Donnez une substitution permettant de passer de $\phi(H)$ à une sous-partie de $\phi(G)$;
Soit $\phi(G)$ donné par :

$$\begin{aligned} \exists \alpha, \beta, \gamma, \delta; & \text{vegetal}(\alpha) \wedge \text{vache}(\beta) \wedge \text{homme}(\gamma) \wedge \text{herbe}(\delta) \\ & \wedge \text{mange}(\gamma, \beta) \wedge \text{mange}(\beta, \delta) \wedge \text{mange}(\beta, \alpha) \end{aligned}$$

et sa sous-partie utile :

$$\exists \alpha, \beta, \gamma; \text{vegetal}(\alpha) \wedge \text{vache}(\beta) \wedge \text{homme}(\gamma) \wedge \text{mange}(\gamma, \beta) \wedge \text{mange}(\beta, \alpha)$$

La substitution est $[x/\beta, y/\gamma, z/\gamma, w/\alpha, \text{omnivore}(z)/\text{homme}(z)]$.

7. Fournissez une preuve par réfutation de $A_S \models \phi(G) \Rightarrow \phi(H)$.
Pour cela, on doit exprimer $\neg\phi(H)$ comme :

$$\begin{aligned} \forall x, y, z, w; & \neg\text{vegetal}(w) \vee \neg\text{vache}(x) \vee \neg\text{homme}(y) \vee \neg\text{omnivore}(z) \\ & \vee \neg\text{mange}(y, x) \vee \neg\text{mange}(z, x) \vee \neg\text{mange}(x, w) \end{aligned}$$

et $\phi(G)$ comme :

$$\begin{array}{cccc} \text{vegetal}(\dot{\alpha}) & \text{vache}(\dot{\beta}) & \text{homme}(\dot{\gamma}) & \text{herbe}(\dot{\delta}) \\ \text{mange}(\dot{\gamma}, \dot{\beta}) & \text{mange}(\dot{\beta}, \dot{\delta}) & \text{mange}(\dot{\beta}, \dot{\alpha}) & \end{array}$$

La résolution s'applique alors comme suit :

$$\begin{aligned} & \neg\text{vegetal}(w) \text{ avec } \text{vegetal}(\dot{\alpha}) \text{ } (\sigma = [w/\dot{\alpha}]); \\ & \neg\text{vache}(x) \text{ avec } \text{vache}(\dot{\beta}) \text{ } (\sigma = [x/\dot{\beta}]); \\ & \neg\text{homme}(y) \text{ avec } \text{homme}(\dot{\gamma}) \text{ } (\sigma = [y/\dot{\gamma}]); \\ & \neg\text{omnivore}(z) \text{ avec } \text{homme}(\dot{\gamma}) \text{ } (\sigma = [z/\dot{\gamma}]); \end{aligned}$$

à ce moment la résolvante est : $\neg\text{mange}(\dot{\gamma}, \dot{\beta}) \vee \neg\text{mange}(\dot{\gamma}, \dot{\beta}) \vee \neg\text{mange}(\dot{\beta}, \dot{\alpha})$. La suite de la résolution est alors triviale puisqu'il s'agit de pas sans substitution :

$$\begin{aligned} & \neg\text{mange}(\dot{\gamma}, \dot{\beta}) \text{ avec } \text{mange}(\dot{\gamma}, \dot{\beta}); \\ & \neg\text{mange}(\dot{\gamma}, \dot{\beta}) \text{ avec } \text{mange}(\dot{\gamma}, \dot{\beta}); \\ & \neg\text{mange}(\dot{\beta}, \dot{\alpha}) \text{ avec } \text{mange}(\dot{\beta}, \dot{\alpha}); \end{aligned}$$

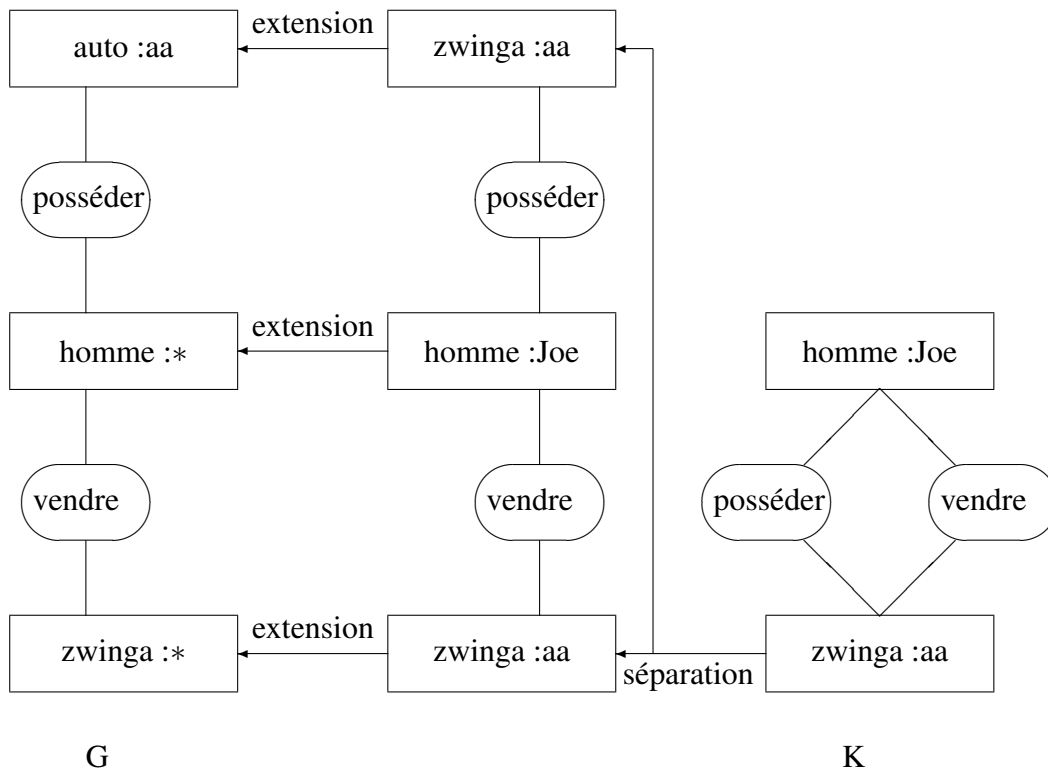
dont la résolvante est la clause vide. On peut noter que la substitution utilisée est bien celle proposée en réponse à la question précédente.

Exercice 3.2 (*). On considère les graphes conceptuels de la figure 3.6 (p. 73).

1. Donnez une signification en langage naturel à ces graphes.
 - G** Un homme possède l'auto aa et vend une zwinga;
 - H** Un humain possède l'auto aa;
 - K** L'homme Joe vend sa zwinga aa.
2. Dites quels graphes se projettent dans quels autres graphes.
 - H dans G;
 - H dans K;

- G dans K;
- c'est tout.

3. Donnez la suite des opérations de généralisation nécessaires pour passer de K à G .



Exercice 3.3 ().** On se propose de considérer la projection entre des graphes dont la signature (graphe étoile) des relations n'est qu'incomplètement respectée (cette opération permet d'obtenir pour les \mathcal{S} -graphes le comportement présenté brièvement dans Sowa [1984] pp93-94). Pour cela, on va reconstruire progressivement les résultats obtenus en cours. Dans le cadre de l'exemple, on se propose de redéfinir B (l'ensemble des graphes-étoile ou base) de telle sorte que le graphe étoile correspondant à ζ vendre \hat{E} ait quatre voisins. On veut, par contre, pouvoir projeter K dans I en introduisant des arêtes autorisées par B , mais manquantes dans la description initiale (celle de K) :

1. Donnez dans ce cadre une nouvelle définition d'un \mathcal{S} -graphe. $\forall r \in R$ les arcs sortants sont totalement ordonnés et indexés par $I(r) \subseteq [1 \dots degré(\beta(type(r)))]$ et $\forall i \in I(r), type(G_i(r)) \leq \beta_i(type(r))$.
2. Donnez la nouvelle définition de \mathcal{S} -projection.

$$\forall r \in R, \forall i \in I(r) \left\{ \begin{array}{l} i \in I'(f(r)) \text{ et} \\ g(G_i(r)) = G'_i(f(r)) \end{array} \right.$$

3. Modifiez les règles de spécialisation de \mathcal{S} -graphes. On modifie la règle **Suppression** et ajoute la règle **Précision** :

Suppression remplacer $\forall i \in 1, \dots, \text{degré}(r_1)$ par $I(r_1) = I(r_2)$ et $\forall i \in I(r_1)$.

Précision Restriction d'étiquette

$$\exists r \in R; \exists i \in [1 \dots \text{degré}(\beta(\text{type}(r)))] - I(r)$$

$$\langle R, C, U, l \rangle \xrightarrow{\text{Précision}} \langle R, C \cup \{c\}, U \cup \{\langle r, c \rangle\}, l \cup \{c, \langle \beta_i(\text{type}(r)), * \rangle\} \rangle$$

L'ensemble des \mathcal{S} -graphes ne correspond alors plus à $\Sigma(B \cup \{\top : *\})$.

4. Donnez la séquence d'opérations de spécialisation permettant de passer de K à I .
5. Vérifiez l'équivalence entre spécialisation et projection. Démontrons que $G \leq H$ si et seulement si il existe une projection de H vers G .

si Soit $\langle f, g \rangle$ une projection de H vers G , on ne considère que la seconde partie de la démonstration du lemme 3.7 (p. 63). Si $f^{-1}(r) = \{r_1 \dots r_n\}$ alors

- (a) On peut saturer toutes les relations $r_1 \dots r_n$ par **Précision**. Elles ont alors la même arité (en fait plutôt que de saturer, on fait en sorte que chaque indice apparaissant dans l'un des r_i apparaisse dans tous).
- (b) On peut, comme dans la première partie de la démonstration du lemme 3.7 (p. 63) fusionner toutes les images des arcs ainsi introduits avec les arcs correspondants (par **Restriction+Fusion**).
- (c) Les r_i .

Ces trois séquences d'opérations remplacent S_2 dans la démonstration initiale.

seulement si Il faut associer une \mathcal{S} -projection élémentaire à la règle **Précision**. $\pi = \langle f, g \rangle = \langle id, id \rangle$. Il n'y a rien à changer, c'est bien une projection qui peut être concaténée aux autres comme dans la démonstration initiale.

Exercice 3.4 (*)**. On s'attache maintenant à donner une sémantique aux opérations définies plus haut.

1. Donnez intuitivement (avec des mots) le sens de l'utilisation d'une relation incomplète : K signifie que l'homme Joe a vendu la zwika aa qu'il possédait à quelqu'un pour un certain prix. Le sens d'une relation incomplète est donc le même que celui de la relation dont tous les arcs atteignent un concept générique du type de celui présent dans le graphe étoile.
2. En quoi la traduction en logique pose-t-elle problème? Quelle méthode utiliseriez vous pour y remédier dans le cadre de la logique classique? La traduction pose problème à cause des arcs absents qui correspondent à des places dans le prédicat. En logique classique, il est très facile de remédier à cela en introduisant de nouvelles variables quantifiées existentiellement (exactement comme précédemment). Ces variables seront restreintes aux types du graphe étoile.

3. Modifiez la définition de ϕ de manière à tenir compte des relations incomplètes.

$$\begin{aligned} \phi : C \cup R \cup \mathcal{S}\text{-graphe} &\longrightarrow \mathcal{CP1} \\ r_i &\longmapsto \begin{cases} \iota(G_i(r)) & \text{si } i \in I(r) \\ x & \text{sinon} \end{cases} \\ r &\longmapsto \text{type}(l(r))(\phi(r_1), \dots, \phi(r_{\text{degré}(r)})) \\ G &\longmapsto \exists_{c \in C} \iota(c) \exists_{r \in R, x \in \text{Var}(r)} x \wedge_{c \in C} \phi(c) \\ &\quad \wedge_{r \in R} \phi(r) \wedge_{r \in R, x \in \text{Var}(r)} \text{type}(G_i(l(\text{type}(r))))(x) \\ &\quad \text{avec } \text{Var}(x) = \{\phi(r_i); i \in [1 \dots \text{degré}(r)] - I(r)\} \end{aligned}$$

Les axiomes associés au support restent les mêmes.

4. Donnez l'expression de $\phi(K)$ et $\phi(I)$ et vérifiez votre réponse précédente.

$$\begin{aligned} \phi(K) &= \exists x, \exists y; \\ &\quad \text{vendre}(\text{Joe}, aa, x, y) \wedge \text{posséder}(aa, \text{Joe}) \wedge \\ &\quad \text{homme}(\text{Joe}) \wedge \text{zwinga}(aa) \wedge \text{Prix}(x) \wedge \text{humain}(y) \\ \phi(I) &= \exists x; \\ &\quad \text{vendre}(\text{Joe}, aa, x, \text{Sue}) \wedge \text{posséder}(aa, \text{Joe}) \wedge \\ &\quad \text{homme}(\text{Joe}) \wedge \text{zwinga}(aa) \wedge \text{Prix}(x) \wedge \text{femme}(\text{Sue}) \end{aligned}$$

On a donc bien $A_S \models \phi(I) \Rightarrow \phi(K)$.

5. L'utilisation d'une logique dont les prédicats sont des \mathcal{OSF} -termes (voir chapitre 4) permettrait de se passer des positions des prédicats et de disposer de relations du type $x : \text{Vente}(\text{vendeur} \rightarrow \text{Joe} : \text{homme}, \text{objet} \rightarrow aa : \text{zwinga})$.
6. Afin d'obtenir la correction et la complétude entre spécialisation et implication dans la théorie associée au support, il est nécessaire de modifier les points suivants :

correction Il est nécessaire de prendre en compte les nouvelles règles. On doit montrer que si le pas de spécialisation est **Précision** de H^j à H^{j+1} alors $A_S \models \phi(H^j) \Rightarrow \phi(H^{j+1})$. Ceci est immédiat car si l'on dispose d'un pas de **Précision**, alors :

$$\begin{aligned} \exists r \in R^j, \exists i \in [1 \dots \text{degré}(\beta(\text{type}(r)))] - j(r) \\ \text{et } i \in j + 1(r); C^{j+1} = C^j \cup \{c\} \\ \text{et } l(c) = \langle \beta_i(\text{type}(r)), * \rangle \\ \text{et } G_i^{j+1}(r) = c \\ \text{alors } \phi(H^j) = \exists x \dots \text{type}(r)(\phi(r_1^j) \dots \phi(r_n^j)) \\ \text{et } \phi(H^{j+1}) = \exists x \dots \text{type}(r)(\phi(r_1^{j+1}) \dots \phi(r_{\text{degré}(r)}^{j+1})) \\ \text{or } \phi(H^j) \equiv \phi(H^{j+1}). \end{aligned}$$

complétude On peut se rendre compte que le graphe associé à une formule est un graphe complet (ce qui est différent d'avec les OSF -termes). Il n'est donc pas nécessaire de modifier la démonstration.

7. Arrivé à ce point, on se rend compte qu'il est possible de normaliser les graphes « étendus » avant de faire le travail. Par exemple, considérer qu'un graphe « étendu » est simplement une abbréviation pour le graphe clos par la règle **Précision**. Cela permettrait de ne pas avoir à introduire **Précision** dans les règles de spécialisation et de ne pas faire intervenir de modifications dans l'aspect logique.

Exercice 3.5 ().** Généralisez les étiquettes sur les arcs par un ensemble d'étiquette quelconque à la place d'une numérotation. Quel problème cela pose-t-il vis-à-vis de la traduction vers la logique ?

Les termes de logique ont leurs arguments numérotés par leur position. Il est donc nécessaire d'introduire un ordre entre les éléments de l'ensemble. Ceci peut être réalisé comme cela est fait dans les Ψ -termes à l'aide d'un ordre total \ll sur les étiquettes (voir définition 4.6).

Exercice 3.6 (*). On considère un langage nommé \mathcal{RDF} (Resource Description Framework) qui se présente comme un ensemble d'assertions de type :

Objet attribut valeur.

Que l'on nomme triplet. Il permet d'exprimer (informellement), par exemple :

L'Union Européenne contient le Royaume-Uni .
Le Royaume-Uni a pour Premier ministre _:1 .

Signifiant que le Royaume-Uni a un Premier ministre.

\mathcal{RDF} a la particularité de disposer de plusieurs syntaxes concurrentes : une syntaxe graphique, une syntaxe en XML, une syntaxe sous forme de triplets que nous considérerons ici.

Le langage de \mathcal{RDF} est défini par la syntaxe suivante (string et integer sont les chaînes de caractères et les entiers, id des ensembles d'identificateurs) :

$\langle graph \rangle ::= \{ \langle triple \rangle^* \}$

$\langle triple \rangle ::= \langle ref \rangle \langle rel \rangle \langle val \rangle \{ . \}$

$\langle val \rangle ::= \langle ref \rangle$
| $\langle literal \rangle$

$\langle ref \rangle ::= \langle uri \rangle$
| $\langle anon \rangle$

$\langle rel \rangle ::= \langle uri \rangle$

$\langle literal \rangle ::= string$
| integer

$\langle uri \rangle ::= id$

$\langle anon \rangle ::= _ : id$

Par exemple, on écrira le graphe suivant :

```
G1={
    Marie écrit _:1 .
    Marie frère _:2 .
    _:2 traduit _:1 .
    Marie chante _:3 .
    _:2 compose _:3 .
}
```

Dans la suite, on manipulera cette syntaxe comme un ensemble de triplets mathématiques, c'est-à-dire un sous-ensemble de $E \times R \times E'$, avec $R \cap (E \cup E') = \emptyset$ (c'est-à-dire qu'un identificateur ne peut simultanément être en position de relation et à une autre position).

1. Énoncer en français la signification de ce dernier exemple ;

Marie a écrit quelque chose que son frère a traduit et a chanté quelque chose que son frère a composé.

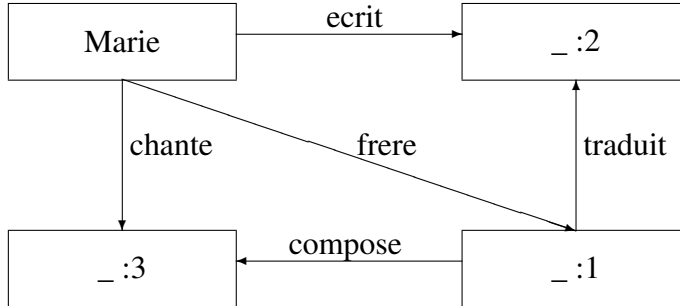
2. Définissez une structure mathématique (au sens ensemble de noeuds-ensemble d'arcs) associée à la production $\langle graph \rangle$ de la grammaire ci-dessus.

On peut utiliser un multi-graphe orienté étiqueté sur les nœuds par des ensembles d'identificateurs (I), de littéraux (L) et de variables (X) et sur les arcs par un ensemble de relations (R). $G = \langle N, N', E, \lambda_N, \lambda_E \rangle$ avec $E \subseteq N \times (N \cup N')$, $\lambda_N : N \rightarrow I \cup X$; $N' \rightarrow L$, $\lambda_E : E \rightarrow R$. λ_N est injective (c'est-à-dire qu'il n'existe qu'un seul nœud portant une étiquette de $I \cup L \cup X$ particulière).

L'exemple sera alors exprimé de la manière suivante :

$$\begin{aligned} & \langle \{n_1, n_2, n_3, n_4\}, \{\} \rangle, \\ & \quad \{ \langle n_4, n_1 \rangle, \langle n_4, n_2 \rangle, \langle n_1, n_2 \rangle, \langle n_4, n_3 \rangle, \langle n_1, n_3 \rangle \}, \\ & \quad \{ \langle n_1, _ : 1 \rangle, \langle n_2, _ : 2 \rangle, \langle n_3, _ : 3 \rangle, \langle n_4, Marie \rangle \}, \\ & \quad \{ \langle \langle n_4, n_1 \rangle, frere \rangle, \langle \langle n_4, n_2 \rangle, ecrit \rangle, \langle \langle n_1, n_2 \rangle, traduit \rangle, \langle \langle n_4, n_3 \rangle, chante \rangle, \\ & \quad \quad \langle \langle n_1, n_3 \rangle, compose \rangle \}, \end{aligned}$$

3. Présentez graphiquement le graphe donné en exemple.



4. Définissez la fonction g et la fonction t qui transforment respectivement un ensemble de triplets en un graphe et un graphe en un ensemble de triplets.

$$t(G) = \{a b c.; \exists \langle n, n' \rangle \in E, \lambda_N(n) = a, \lambda_N(n') = c, \lambda_E(\langle n, n' \rangle) = b\}$$

$g(T)$ est le plus petit graphe $\langle N, N', E, \lambda_N, \lambda_E \rangle$ tel que :

$$\forall a b c \in T, \exists n \in N, \exists n' \in N \cup N'; \langle n, n' \rangle \in E, \lambda_N(n) = a, \lambda_N(n') = c \text{ et } \lambda_E(\langle n, n' \rangle) = b$$

5. Est-ce que $g = t^{-1}$?

Oui, à cause du fait que l'on dispose d'un multigraphe qui permet de représenter dans E chaque triplet et que l'on est capable de reconstruire ce triplet à partir des étiquettes. Lorsque l'ensemble de triplets sera retraduit en graphe, il stocke les triplets dans E et construit l'étiquette en accord avec chaque triplet.

Exercice 3.7 ().** On veut définir la sémantique du langage \mathcal{RDF} . Pour cela on va définir une fonction d'interprétation I vers un domaine concret V et un domaine (plus classique) D ($D \cap V = \emptyset$) de sorte que :

$$\begin{array}{ll} I : \langle ref \rangle & \longrightarrow D \\ & \langle literal \rangle \longrightarrow V \\ & \langle rel \rangle \longrightarrow D \times (D \cup V) \end{array}$$

Une substitution est une fonction $\sigma : \langle anon \rangle \longrightarrow D$. $I\sigma$ est une fonction telle que, $I\sigma(a) = \sigma(a)$ si a est anonyme, $I\sigma(a) = I(a)$ sinon. Un triplet $a r b$. est satisfait par une interprétation I sous une substitution σ , ($I, \sigma \models a r b$.) si et seulement si $\langle I\sigma(a), I\sigma(b) \rangle \in I(r)$.

Une interprétation I est un modèle d'un graphe G , si et seulement si, il existe une substitution σ telle que tous les triplets du graphe sont satisfaits par I sous σ . Un graphe G est conséquence d'un autre graphe G' (noté $G' \models G$) si et seulement si tout modèle de G est un modèle de G' .

1. Existe-t-il des graphes équivalents (l'un est conséquence de l'autre et vice versa) qui ne soient pas identiques (au renommage des identificateurs prêts) ?

Oui, à l'instar des graphes irrédundants des graphes conceptuels (voir §). Par exemple,

H1={
 Marie frère Pierre .
 }

et

H2={
 Marie frère _:1 .
 Marie frère Pierre .
 }

Exercice 3.8 ().** On définit une instance d'un graphe comme un graphe dont des noeuds anonymes ont été remplacés par un identificateur $\langle uri \rangle$ ou un autre nœud anonyme $\langle anon \rangle$. Un sous-graphe d'un graphe est un graphe dont on a supprimé une partie des triplets. La fusion d'un ensemble de graphes est composée de l'ensemble des triplets dont les identificateurs anonymes ont pu être renommés de telle sorte qu'un identificateur anonyme n'apparaisse pas dans deux graphes différents.

1. Donnez une définition formelle de ces trois notions, soit avec la forme graphique, soit avec la forme grammaticale.

Avec la forme graphique :

Soit $Var(G)$ la restriction aux éléments de X de l'image de N par λ_N . Un graphe $G' = \langle N_\sigma, N', E_\sigma, \lambda_{N,\sigma}, \lambda_E \rangle$ est instance d'un graphe $G = \langle N, N', E, \lambda_N, \lambda_E \rangle$ si et seulement si $\exists \sigma : Var(G) \rightarrow X \cup I$ tel que $\lambda_{N,\sigma}$ est la fonction $\sigma \circ \lambda_N$ (où $\sigma(i) = i$ pour $i \in I \cup L$), N_σ (resp. E_σ) est N (resp. E) pour lequel les nœuds ayant la même étiquette par λ_N, σ sont unifiés.

Un sous-graphe G' d'un graphe $G = \langle N, N', E, \lambda_N, \lambda_E \rangle$ est défini par : $\langle N, N', E', \lambda_N, \lambda_E|_{E'} \rangle$ où $E' \subseteq E$.

Soit $\rho : Var(G^2) \rightarrow X - Var(G^1)$ une substitution injective, la fusion $Merge_\rho(G^1, G^2)$ de deux graphes $G^1 = \langle N^1, N^1', E^1, \lambda_{N^1}^1, \lambda_{E^1}^1 \rangle$ et $G^2 = \langle N^2, N^2', E^2, \lambda_{N^2}^2, \lambda_{E^2}^2 \rangle$ est $\langle N^1 \cup N^2, N^1' \cup N^2', E^1 \cup E^2, \lambda_{N^1}^1 \cup \rho(\lambda_{N^2}^2), \lambda_{E^1}^1 \cup \lambda_{E^2}^2 \rangle$ où ρ est étendue de telle sorte que $\rho(\langle n, n' \rangle) = \langle \rho(n), \rho(n') \rangle$ et $\rho(N) = \{\rho(e); e \in N\}$.

Avec la forme grammaticale :

Soit $Var(G)$ l'ensemble des éléments de X apparaissant dans un triplet de G . Un ensemble de triplets G' est instance d'un ensemble de triplets G ssi $\exists \sigma : Var(G) \rightarrow X \cup I$, étendu de sorte que $\sigma(N) = \{\sigma(n) | n \in N\}$, $\sigma(\langle a, b, c \rangle) = \langle \sigma(a), b, \sigma(c) \rangle$ et $\sigma(i) = i$ pour $i \in I \cup L$, tel que $G' = \sigma(G)$.

Un ensemble de triplets G' est un sous-graphe d'un ensemble de triplets G ssi $G' \subseteq G$.

Soit $\rho : Var(G') \rightarrow X - Var(G)$ une substitution injective, la fusion $Merge_\rho(G, G')$ de deux ensembles de triplets G et G' est $G \cup \rho(G')$ où ρ est étendue de telle sorte que $\rho(\langle a b c \rangle) = \langle \rho(a) b \rho(c) \rangle$ et $\rho(G) = \{\rho(t); t \in G\}$.

2. Existe-t-il des graphes équivalents à leur sous-graphe propres ?

Oui, on le voit clairement dans la réponse donnée ci-dessus : H_1 est un sous-graphe propre de H_2 .

Exercice 3.9 (*)**. Le résultat principal sur les graphes \mathcal{RDF} est le suivant :

Interpolation. Un ensemble de graphes \mathcal{RDF} S a pour conséquence un graphe G si, et seulement si, il existe un sous-graphe de la fusion de S instance de G .

Soit le graphe défini par :

```
G2= {
  Marie écrit _:1 .
  Marie frère Pierre .
  Pierre traduit _:1 .
  Marie chante _:1 .
  Pierre compose _:1 .
  Pierre joue _:1 .
}
```

1. Y a-t-il un lien de conséquence entre G_1 et G_2 (ou G_2 et G_1) ? Si oui, préciser par quelles opérations on passe de l'un à l'autre.

G_1 est une conséquence de G_2 . En effet, si l'on considère le graphe G_3 sous-graphe de G_2 sans "Pierre joue _:1", G_3 est instance de G_1 . Il convient simplement de remplacer les nœuds anonymes _:3 par _:1 et _:2 par Pierre dans G_1 pour obtenir G_3 .

G_2 n'est pas conséquence de G_1 . Il faudrait pour cela qu'un sous-graphe de G_1 soit une instance de G_2 , c'est-à-dire qu'il possède un triplet résultat de l'instanciation de "Pierre joue _:1" ce qui est impossible puisqu'il ne contient pas la relation "joue".

2. Démontrez le résultat d'interpolation. (Note : on aura intérêt à décomposer en fonction de chacune des opérations)

On peut prouver l'équivalence pour chacune des opérations impliquées dans la définition.

lemme de l'instance. $\forall g$ instance de $G, g \models G$

Démonstration. Si le graphe g est satisfait par une interprétation I , il existe une substitution σ telle que tous les triplets de g soient satisfaits par I sous σ . Mais, g contient des triplets de G plus des triplets de G auxquels une substitution ρ a été appliquée. I satisfait donc forcément G sous $\sigma\rho$. \square

lemme du sous-graphe. $\forall G'$ sous-graphe de $G, G \models G'$

Démonstration. Si le graphe G est satisfait par une interprétation I , tout sous-graphe, qui a moins de triplets à satisfaire, le sera aussi. \square

Ces deux lemmes doivent être complétés par le lemme de Herbrand.

lemme de la fusion $S \models Merge_\rho(S)$ et $\forall g \in S, Merge_\rho(S) \models g$.

Démonstration. Si le graphe $Merge_\rho(S)$ est satisfait par une interprétation I , il existe une substitution σ , telle que tous les triplets de $Merge_\rho(S)$ soient satisfaits par I sous σ . Mais tous les triplets de g sont présents dans $Merge_\rho(S)$ à un renommage ρ des nœuds anonymes près. Il existe donc une substitution $\sigma\rho$ telle que I satisfasse g sous elle (et donc tous les modèles de $Merge_\rho(S)$ sont des modèles de g). À l'inverse, toute interprétation I telle qu'il existe σ pour lequel I satisfasse S (c'est-à-dire tous les triplets d'un graphe de S), est forcément une interprétation satisfaisant $Merge_\rho(S)$ sous une substitution $\rho\sigma$ où ρ se borne à défaire le renommage (c'est-à-dire qu'il unifie les nœuds qui avaient le même nom dans S). \square

3. Que sont, selon vous, les différences entre ce formalisme et celui des graphes conceptuels ?

La différence principale avec les graphes conceptuels tient dans l'absence de support qui simplifie le traitement. Ce support est introduit dans le langage \mathcal{RDFS} . Une seconde différence est l'absence d'existence directe de relation non binaires. Mais celles-ci peuvent facilement être simulées.

Une autre différence qui n'apparaît pas à cause d'une restriction de l'énoncé est que les identificateurs de relation, en \mathcal{RDF} peuvent aussi être sujet à relations. Ceci rend la sémantique plus complexe mais pas intrinsèquement contradictoire.

6.4 Ψ -termes

Exercice 3.1 (*). Soient les \mathcal{OSF} -termes suivants :

```
X:projet( chef => personne ( projet => X,
                               diplome => Y:phD ),
          budget => bilan )

projet_bien_gere( budget => bilan( entrees => Z:entier,
                                   sorties => Z ))

projet_moderne( chef => femme )
```

Refaire tous les exemples qui ont été donnés dans le cours en termes d'application des algorithmes.

1. Donnez une \mathcal{OSF} -signature correspondant à ces termes ;

$S = \{\text{personne, projet, bilan, phD, projet_bien_géré, entier, projet_moderne, femme, } \top, \perp\}$;
 $\leq = \{\langle \text{femme, personne} \rangle\}$ (si l'on omet les relations triviales impliquant \perp et \top) ;
 \wedge est simplement induit par l'ordre partiel ;
 $F = \{\text{chef, budget, projet, diplome, entrees, sorties}\}$.

2. Donnez les \mathcal{OSF} -termes correspondant à ces termes ainsi que les \mathcal{OSF} -termes normalisés ; à chaque fois, spécifiez l'ensemble Var ;

Voici des \mathcal{OSF} -termes qui sont aussi les Ψ -termes correspondants :

$$\begin{aligned} \psi_1 &= X:\text{projet}(\text{chef} \Rightarrow P:\text{personne}(\text{projet} \Rightarrow X:\top, \\ &\hspace{15em} \text{diplome} \Rightarrow Y:\text{phD}), \\ &\hspace{15em} \text{budget} \Rightarrow B:\text{bilan}) \\ \psi_2 &= X:\text{projet_bien_gere}(\text{budget} \Rightarrow B:\text{bilan}(\text{entrees} \Rightarrow Z:\text{entier}, \\ &\hspace{15em} \text{sorties} \Rightarrow Z:\top)) \\ \psi_3 &= X:\text{projet_moderne}(\text{chef} \Rightarrow P:\text{femme}) \end{aligned}$$

Pour les trois termes, Var correspond respectivement à :

- $Var(\psi_1) = \{X, P, Y, B\}$
- $Var(\psi_2) = \{X, B, Z\}$
- $Var(\psi_3) = \{X, P\}$

3. Donnez les \mathcal{OSF} -clauses correspondant aux \mathcal{OSF} -termes ; vérifiez que ces clauses sont enracinées dans la variable racine ;

- $\phi(\psi_1) = X : \text{projet} \wedge X.\text{chef} \doteq P \wedge P : \text{personne} \wedge P.\text{projet} \doteq X \wedge X : \top \wedge P.\text{diplome} \doteq Y \wedge Y : \text{phD} \wedge X.\text{budget} \doteq B \wedge B : \text{bilan}$ qui est enracinée en X car $P = X.\text{chef}$, $B = X.\text{budget}$ et $Y = X.\text{chef}.\text{diplome}$.
- $\phi(\psi_2) = X : \text{projet_bien_géré} \wedge X.\text{budget} \doteq B \wedge B : \text{bilan} \wedge B.\text{entrees} \doteq Z \wedge Z : \text{entier} \wedge B.\text{sorties} \doteq Z \wedge Z : \top$ qui est enracinée en X car $B = X.\text{budget}$ et $Z = X.\text{budget}.\text{entrees}$.
- $\phi(\psi_3) = X : \text{projet_moderne} \wedge X.\text{chef} \doteq P \wedge P : \text{femme}$ qui est enracinée en X car $P = X.\text{chef}$.

4. Normalisez ces \mathcal{OSF} -clauses à l'aide des règles de la définition 4.14 ; vérifiez que les clauses obtenues correspondent, aux égalités près, aux clauses obtenues par dissolution des \mathcal{OSF} -termes normalisés ;

$$\begin{aligned} \phi(\psi_1) &= X : \text{projet} \wedge X.\text{chef} \doteq P \wedge P : \text{personne} \wedge P.\text{projet} \doteq X \\ &\hspace{10em} \wedge X : \top \wedge P.\text{diplome} \doteq Y \wedge Y : \text{phD} \wedge X.\text{budget} \doteq B \wedge B : \text{bilan} \end{aligned}$$

La seule règle déclenchable est une intersection de types sur X ce qui donne :

$$\begin{aligned} X : \text{projet} \wedge X.\text{chef} \doteq P \wedge P : \text{personne} \wedge P.\text{projet} \doteq X \\ \wedge P.\text{diplome} \doteq Y \wedge Y : \text{phD} \wedge X.\text{budget} \doteq B \wedge B : \text{bilan} \end{aligned}$$

ce qui correspond, une fois appliqué ψ , au Ψ -terme associé à ψ_1 .

$$\begin{aligned} \phi(\psi_2) &= X : \text{projet_bien_géré} \wedge X.\text{budget} \doteq B \wedge B : \text{bilan} \wedge B.\text{entrees} \doteq Z \wedge \\ &\hspace{10em} Z : \text{entier} \wedge B.\text{sorties} \doteq Z \wedge Z : \top \end{aligned}$$

La seule règle déclenchable est une intersection de types sur Z ce qui donne :

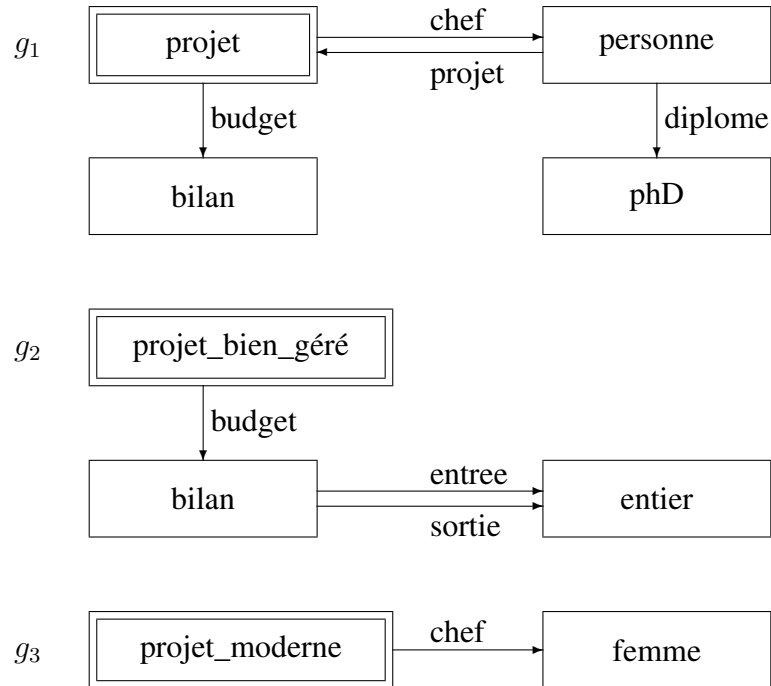
$$X : \text{projet_bien_géré} \wedge X.\text{budget} \doteq B \wedge B : \text{bilan} \wedge B.\text{entrees} \doteq Z \wedge Z : \text{entier} \wedge B.\text{sorties} \doteq Z$$

ce qui correspond, une fois appliqué ψ , au Ψ -terme associé à ψ_2 .

$$\phi(\psi_3) = X : projet_moderne \wedge X.chef \doteq P \wedge P : femme$$

ne permet de déclencher aucune règle et correspond, une fois appliqué ψ , au Ψ -terme associé à ψ_3 .

5. Tracez les OSF -graphes correspondant aux OSF -termes initiaux ;



6. Extrayez les OSF -termes associés aux OSF -graphes précédents et vérifiez que vous obtenez les OSF -termes initiaux.

$$\begin{aligned} & \phi(g_1) \\ &= X:projet(chef => \\ & \phi(\langle \{ P, X, Y \}, \langle \{ P, X \}, \{ P, Y \} \rangle, \langle \{ P, personne \}, \langle X, \top \rangle, \langle \langle P, Y \rangle, phD \rangle \rangle, \langle \langle P, X \rangle, \\ & \quad budget => \phi(\langle \{ B \}, \emptyset, \langle \{ B, bilan \} \rangle, \{ \}, B) \rangle) \\ &= X:projet(chef => P:personne(projet => \phi(\langle \{ X \}, \emptyset, \langle \{ X, \top \} \rangle, \{ \}, X) \rangle, \\ & \quad diplome => \phi(\langle \{ Y \}, \emptyset, \langle \{ Y, phD \} \rangle, \{ \}, Y) \rangle) \\ & \quad budget => B:bilan) \\ &= X:projet(chef => P:personne(projet => X:\top, \\ & \quad diplome => Y:phD) \\ & \quad budget => B:bilan) \end{aligned}$$

$$\begin{aligned} & \phi(g_2) \\ &= X:projet(budget => \\ & \phi(\langle \{ B, Z \}, \langle \{ B, Z \}, \{ B, Z \} \rangle, \langle \{ B, bilan \}, \langle Z, entier \rangle \rangle, \langle \langle \{ B, Z \} entrees \rangle, \langle B, Z \rangle sorties \rangle \rangle) \\ &= X:projet(budget => B:bilan(entrees => \phi(\langle \{ Z \}, \emptyset, \langle \{ Z, entier \} \rangle, \emptyset, Z) \rangle), \end{aligned}$$

```

sorties => φ({Z}, ∅, {⟨ Z, T ⟩}, ∅, Z))
= X:projet( budget => B:bilan( entrees => Z:entier,
                               sorties => Z:T))

```

```

φ(g3)
= X:projet_moderne( chef => φ({P}, ∅, {⟨ P, femme ⟩}, ∅, P))
= X:projet_moderne( chef => P:femme)

```

Exercice 3.2 (*). On désire comparer les graphes conceptuels et les *OSF*-termes.

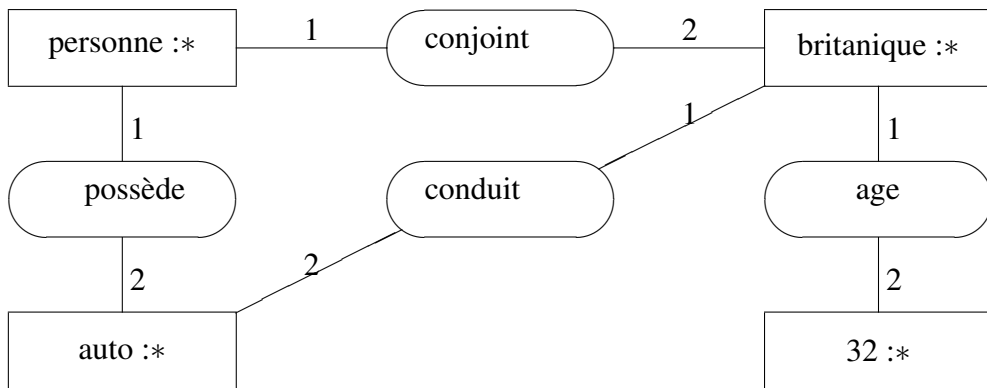
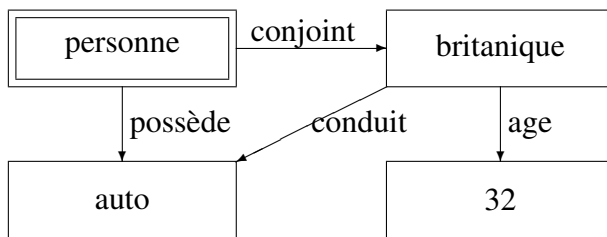
1. Soit l'*OSF*-terme suivant :

```

X:personne( conjoint => britannique( age => 32 ),
            possede => Y:auto,
            conjoint => personne( conduit => Y ))

```

Traduisez le en l'*OSF*-graphe correspondant puis en un graphe conceptuel similaire.



2. Mettez en correspondance les divers éléments que l'on trouve dans les deux formalismes.

	<u>graphes conceptuels</u>	<u>Ψ-termes</u>
	T_C	S
On peut faire le parallèle suivant :	\leq	\leq
	T_R^2	F
	*	implicite.

Les différences syntaxiques principales résident dans la binarité des traits (il est possible de simuler les relations n -aires en les présentant comme des sortes – les prédicats en logique sont ce qui correspond aux relations) et dans l'absence totale d'éléments individuels dans les Ψ -termes.

En ce qui concerne la sémantique, les Ψ -termes, à l'instar des termes de logiques terminologiques sont des définitions de classes qui n'impliquent pas l'existence d'éléments de ces classes (l'ensemble vide est un modèle trivial de tous les termes d'une base). Par contre, les graphes conceptuels ont une interprétation existentielle : ils affirment qu'il existe un individu ayant les caractéristiques du graphe.

Par ailleurs, un \mathcal{OSF} -terme dénote un ensemble d'objets (appartenant à la sorte de tête) alors qu'un graphe conceptuel dénote une situation (l'absence d'exigence de connexité et d'enracinement étant révélateur en ce sens). Ceci se traduit dans la différence de statut : le premier est un terme alors que le second est une formule. Ces différences s'effacent lorsque l'on utilise la notion d'implication (ou de subsomption) pour qualifier le degré de généralité des formules.

Exercice 3.3 (*)**. Montrez l'équivalence entre \mathcal{OSF} -clauses enracinées résolues et les \mathcal{OSF} -graphes sans faire intervenir les \mathcal{OSF} -termes.

Pour cela, il faut définir l' \mathcal{OSF} -clause $\hat{\phi}(g)$ correspondant à un \mathcal{OSF} -graphe g et l' \mathcal{OSF} -graphe $\hat{G}(\phi)$ correspondant à une \mathcal{OSF} -clause ϕ . Les définitions étant données par rapport aux termes, elles sont contraintes par la formulation : $\hat{\phi}(g) = \phi(\psi(g))$ et $\hat{G}(\phi) = G(\psi(\phi))$. Les deux fonctions sont donc les suivantes.

À tout \mathcal{OSF} -graphe $g = \langle N, E, \lambda_N, \lambda_E, X \rangle$ est associé l' \mathcal{OSF} -clause $\hat{\phi}(g)$ définie par :

$$\hat{\phi}(g) = X : \lambda_N(X) \wedge \bigwedge_{i=1}^n \left(X.l_i = \text{root}(g_i(X)) \wedge \hat{\phi}(g_i) \right)$$

telle que $l_1, \dots, l_n \in F$ sont les étiquettes distinctes deux à deux des arcs sortant de X (c'est-à-dire $\{\lambda_E(i)\}_{i=\langle X, X_i \rangle \in E}$) et $g_i(X)$ est le sous-graphe de g accessible à partir de $l_i^g(X)$ dans lequel $\lambda_N(X) = \top$, tout $\langle X, X' \rangle$ est supprimé de E et tous les sous-graphes accessibles à partir de $l_1^g(X), \dots, l_{i-1}^g(X)$ sont remplacés par $\langle \{l_i^g(X)\}, \emptyset, \{\langle l_i^g(X), \top \rangle\}, \emptyset, l_i^g(X) \rangle$.

De même, à toute \mathcal{OSF} -clause ϕ_X est associé l' \mathcal{OSF} -graphe $\hat{G}(\phi) = \langle N, E, \lambda_N, \lambda_E, X \rangle$ tel que : Si $\langle X.l = Y \rangle \in \phi_X$ alors $\langle X, Y \rangle \in E$ et $\lambda_E(\langle X, Y \rangle) = l$ Si $\langle X : s \rangle \in \phi_X$ alors $X \in N$ et $\lambda_N(X) = s$.

Pour ces deux fonctions, on vérifie aisément que la valeur est bien un \mathcal{OSF} -graphe ou une \mathcal{OSF} -clause résolue et enracinée. Il faut maintenant montrer que $g = \hat{G}(\hat{\phi}(g))$ et $\hat{\phi}(\hat{G}(\phi)) = \phi$.

Soit $g = \langle N, E, \lambda_N, \lambda_E, X \rangle$, $\hat{G}(\hat{\phi}(g)) = \langle N', E', \lambda_{N'}, \lambda_{E'}, X' \rangle$. On peut établir que : ²

$$\begin{array}{ll}
 X' = X & \text{car } X' = \text{root}(\phi_X) \\
 N' = \{X \mid (X : s) \in \hat{\phi}(g)\} = N & \text{car } \forall X \in N, (X : \lambda_N(X)) \in \hat{\phi}(g) \\
 \lambda_{N'} = \lambda_N & \text{car } N = N' \text{ et } \lambda_{N'}(X) = \bigwedge_{X:s_i \in N} s_i \\
 = \{\langle X, Y \rangle \mid (X.l = Y) \in \hat{\phi}(g)\} = E & \text{car } \forall \langle X, Y \rangle \in E, (X.\lambda_E(\langle X, Y \rangle) = Y) \in \hat{\phi}(g) \\
 \lambda_{E'} = \lambda_E & \text{car } E = E' \text{ et } \lambda_{E'}(\langle X, Y \rangle) = \bigwedge_{X:s_i \in N} s_i
 \end{array}$$

Est-il nécessaire de faire intervenir un ordre sur F ? Pourquoi ?

L'ordre est inutile dans la démonstration directe car les clauses comme les graphes ne constituent pas (modulo la commutativité de la conjonction) une représentation textuelle (donc ordonnée) et permettent de partager les variables. L'ordonnement est donc inutile.

Examen de 1999-2000 CORRECTION NON DISPONIBLE

Exercice 3.4 (*). Soit la description de concept suivante :

humain $\dot{\leq}$ (and (all pere homme) (all mere femme)
 (all enfant humain) (all fils homme) (all fille femme))

1. Quel problème pose-t-elle vis-à-vis de la définition donnée pour les logiques terminologiques ?
2. Implique-t-elle que les éléments de l'extension d'humain doivent être circulaires ou que l'extension doit être infinie ? Dites pourquoi ?

Exercice 3.5 (*). On considère la logique terminologique $\mathcal{FL}^=$ dont les constructeurs sont and, all et eqrv, telle que l'on puisse décrire les concepts suivants :

homme $\dot{\leq}$ humain
 femme $\dot{\leq}$ humain
 homme $\dot{\oplus}$ femme
 fils $\dot{\leq}$ enfant
 fille $\dot{\leq}$ enfant
 fils $\dot{\oplus}$ fille
 t $\dot{=}$ (and humain (eqrv (pere enfant) (mere enfant)))
 t' $\dot{=}$ (and homme (eqrv (pere fils) (mere fils)))
 t'' $\dot{=}$ (and homme (eqrv (pere fils) (fils pere)))
 t''' $\dot{=}$ (and homme (eqrv (mere fils) (fils pere)))

où eqrv s'interprète comme l'égalité de ses deux membres qui eux s'interprètent comme des chaînes de rôles : $(z \times y)$ signifie "les z des x des y".

1. Donnez une interprétation en langue naturelle des concepts t à t''' ;

²Ce n'est pas les couples qui sont dans E mais des n étiquetés.

2. Est-ce que certains concepts en subsument d'autres ? Les donner. Que penser de $(\text{and } t' \ t'')$ par rapport à t''' ? Que dire de $(\text{and } (\text{all } a \ (\text{eqrvm } b \ c)))$ par rapport à $(\text{eqrvm } (a \ b) \ (a \ c))$?
3. Soit la fonction d'extension

$$\mathcal{E}((\text{eqrvm}(r_1 \dots r_n)(r'_1 \dots r'_m))) =$$

$$\{o'_0 \in D \mid \forall o_1, \dots, o_n \in D; \forall i \in [1 \dots n] \langle o_{i-1}, o_i \rangle \in \mathcal{E}(r_i),$$

$$\exists o'_1, \dots, o'_m \in D; \forall i \in [1 \dots m] \langle o'_{i-1}, o'_i \rangle \in \mathcal{E}(r'_i) \wedge o_n = o'_m\}$$

montrez le bien fondé de votre réponse précédente à l'aide des extensions ;

4. Ajoutez les règles de comparaison au système de COMPARE et montrez qu'elles sont justifiées par les extensions.

Exercice 3.6 ().** Le but de cette partie est de mettre en évidence le rapport entre les termes de $\mathcal{FL}^=$ et les \mathcal{OSF} -termes.

1. Indiquez quels éléments vous semblent correspondre dans les deux systèmes et quels éléments vous semblent différer ;
Concepts récursifs autorisés dans l'un et pas dans l'autre.

2. Exprimez les \mathcal{OSF} -termes de l'exemple 4.1 sous la forme de $\mathcal{FL}^=$ -termes ;

```
k ≐ (and person
      (all name (and id (all last string)))
      (all home (and address (all city cityname)))
      (all father (and person (all name id)))
      (eqrvm (name last) (father name last)))
student ≐ (and k
            (all name (and id
                          (all first string)
                          (all last string)))
            (all home (and address (all city paris)))
            (all father (and k (all name id)))
            (eqrvm (name last) (father name last))
            (eqrvm (home) (father home)))
```

3. Faites correspondre une \mathcal{OSF} -signature à une $\mathcal{FL}^=$ -terminologie, que manque-t-il ?
4. Dessinez l' \mathcal{OSF} -graphe correspondant aux termes t et t'' et donnez les \mathcal{OSF} -clauses correspondant aux termes t' et t''' ;
5. Donnez une fonction de traduction d'un $\mathcal{FL}^=$ -terme en un \mathcal{OSF} -graphe ; Que penser de la traduction d'un \mathcal{OSF} -graphe en un $\mathcal{FL}^=$ -terme ?
6. Donnez une fonction de traduction d'un $\mathcal{FL}^=$ -terme en une \mathcal{OSF} -clause.

Exercice 3.7 (*)**. On cherche maintenant à utiliser les outils développés pour les \mathcal{OSF} -termes dans le cadre des $\mathcal{FL}^=$ -termes. On suppose pour cela un ordre \ll sur les traits dans l' \mathcal{OSF} -signature et on réduit les rôles intervenants dans les eqvm à être des attributs (c'est-à-dire des rôles à une valeur unique).

1. Précisez l' \mathcal{OSF} -signature donnée précédemment au 3c.
2. Est-il nécessaire d'ajouter des règles pour la résolution d' \mathcal{OSF} -clauses ? Si oui dites lesquelles, montrez qu'elles sont correctes et qu'elles sont complètes.
3. L'utilisation des règles de résolution des \mathcal{OSF} -clauses a-t-elle une utilité pour les $\mathcal{FL}^=$ -termes ? Quels tests donnés dans les définitions 2.8 (subsumption) et 2.9 (équivalence, incohérence, exclusion) peut-elle servir à réaliser ?

Table des figures

2.1	Exemple de terminologie	11
2.2	Architecture d'une logique	12
2.3	Contre-exemple à COMPARE	38
2.4	Hierarchie de systèmes terminologiques	40
2.5	50
2.6	50
2.7	51
2.8	52
3.1	Exemple de graphe conceptuel	54
3.2	Exemple de \mathcal{S} -support	58
3.3	Exemples de graphes conceptuels	59
3.4	Exemple de graphe conceptuel emboité	71
3.5	Graphes de l'exercice 3.1	72
3.6	\mathcal{S} -support et graphes de l'exercice 3.2	73
3.7	Graphes de l'exercice 3.3	74
4.1	Exemple d' \mathcal{OSF} -graphe	86
4.2	Exemple de domaine d' \mathcal{OSF} -algèbre de graphe	90
4.3	Ensembles et morphismes	94

Bibliography

- Martín Abadi and Luca Cardelli. *A theory of objects*. Springer Verlag, New-York (NY US), 1996.
- André Arnold and Irène Guessarian. *Mathématiques pour l'informatique*. Masson, Paris (FR), 1992.
- Hassan Aït-Kaci and Andreas Podelski. Towards a meaning of life. *Journal of logic programming*, 16(3):195–234, 1993.
- Michel Chein and Marie-Laure Mugnier. Conceptual graphs: fundamental notions. *Revue d'intelligence artificielle*, 6(4):365–406, 1992.
- Roland Ducournau. Des langages à objets aux logiques terminologiques: les systèmes classificatoires. Rapport de Recherche 96-030, LIRMM, Montpellier (FR), 1996. <ftp.lirmm.fr/pub/LIRMM/papers/1996/RRI-Ducournau-96.ps>.
- Daniel Kayser. *La représentation des connaissances*. Hermès, Paris (FR), 1997.
- Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.
- Marie-Laure Mugnier and Michel Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'intelligence artificielle*, 10(1):7–56, 1996.
- Amedeo Napoli. Une introduction aux logiques de description. Rapport de Recherche 3314, INRIA Lorraine, Nancy (FR), 1997. <ftp.inria.fr//INRIA/publication/publi-ps-gz/RR/RR-3314.ps.gz>.
- Bernhard Nebel. *Reasoning and revision in hybrid representation systems*. Lecture Notes in Artificial Intelligence 422. Springer Verlag, Berlin (DE), 1990.
- Gerd Smolka. Feature constraints logics for unification grammars. *Journal of logic programming*, 12(1):324–343, 1992.
- John Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley, Reading (MA US), 1984.
- Eileen Way. Conceptual graphs: fundamental notions. *Journal of experimental and theoretical artificial intelligence*, 4(2):75–84, 1992.

Index des propositions

- acyclicité (\mathcal{ALNR} -normalisation, TNORM), 19
- \mathcal{ALNR} -subsomption
 - complexité, 38
 - décidabilité, 38
- COMPARE
 - complexité, 37
 - correction, 36
 - incomplétude, 37
- complexité
 - \mathcal{ALNR} -normalisation (TNORM), 19
 - \mathcal{ALNR} -subsomption, 38
 - COMPARE, 37
 - \mathcal{FL} -subsomption, 38
 - \mathcal{S} -isoprojection, 70
 - \mathcal{S} -projection, 70
 - injective, 70
- complétude
 - \mathcal{FL} -subsomption, 38
 - ϕ (traduction de \mathcal{S} -graphes), 69
 - \mathcal{S} -subsomption, 69
- confluence (normalisation d' \mathcal{OSF} -clause), 84
- correction
 - \mathcal{ALNR} -expansion (EXP), 27
 - COMPARE, 36
 - ϕ (traduction de \mathcal{S} -graphes), 66
- correspondance
 - \mathcal{FLN} -structure- \mathcal{ALNR} -structure, 20
 - \mathcal{OSF} -clauses- Ψ -termes, 82
 - \mathcal{OSF} -graphes- Ψ -termes, 87
 - \mathcal{OSF} -graphes- Ψ -termes- \mathcal{OSF} -clauses, 87
 - \mathcal{OSF} -terme en forme normale- \mathcal{OSF} -graphes- \mathcal{OSF} -clauses résolues enracinées, 95
- \mathcal{S} -projection- \mathcal{S} -spécialisation, 64
- \mathcal{S} -projection- \mathcal{S} -substitution, 67
- \mathcal{S} -spécialisation- \mathcal{S} -généralisation, 62
- décidabilité (\mathcal{ALNR} -subsomption), 38
- EXP
 - complexité, 27
 - correction, 27
- finalité (faible, \mathcal{OSF} -algèbre de graphes \mathcal{G}), 94
- \mathcal{FL} -subsomption
 - complexité, 38
 - complétude, 38
- \mathcal{FLN} (taille d'un terme), 36
- incomplétude (COMPARE), 37
- initialité (faible, \mathcal{OSF} -algèbre de graphes canonique), 93
- \mathcal{OSF} -clause normalisation
 - confluence, 84
 - résultat, 84
 - terminaison, 84
- résultat (normalisation d' \mathcal{OSF} -clause), 84
- \mathcal{S} -isoprojection (complexité), 70
- \mathcal{S} -morphisme (stabilité), 60
- \mathcal{S} -projection (complexité), 70
 - injective, 70
- stabilité \mathcal{S} -morphisme, 60
- sémantique constructive, 24
- taille d'un terme, 36
- terminaison
 - \mathcal{ALNR} -expansion (EXP), 27
 - normalisation d' \mathcal{OSF} -clause, 84

TNORM

acyclicité, 19

complexité, 19

Index des termes définis

- admissible (\mathcal{OSF} -algèbre), 91
- \mathcal{AF}
 - interprétation, 22
 - modèle, 23
 - respectant une terminologie, 23
 - syntaxe, 21
 - ensembles, 22
- affectation primitive (\mathcal{ALNR}), 24
- algèbre (\mathcal{OSF}), 88
 - admissible pour une \mathcal{OSF} -clause, 91
 - de graphe, 89
 - canonique, 92
 - engendrée par des termes, 89
- \mathcal{ALNR}
 - extension, 19
 - structure, 20
 - subsomption, 28
 - affectation primitive, 24
 - comparaison de termes, 33
 - ensembles, 18
 - fonction d'expansion, 26
 - normalisation, 28
 - syntaxe, 18
- approximation endomorphe (\mathcal{OSF}), 93
- assertionnelle, 10
- assignation (\mathcal{OSF}), 88
- atteignabilité, 81
- axiomes associés au support, 65
- catégorie \mathcal{OSF} , 93
- clause (\mathcal{OSF}), 81
 - enracinée, 81
- comparaison de termes (\mathcal{ALNR}), 33
- COMPARE (\mathcal{ALNR}), 33
- conséquence
 - (\mathcal{AF}), 23
 - modulo une terminologie (\mathcal{FLN}), 23
- contrainte (\mathcal{OSF}), 81
- description du monde, 21
- dissolution d'un \mathcal{OSF} -terme, 81
- dénotation (\mathcal{OSF}), 91
 - sous une assignation, 91
- enraciné(e)
 - graphe, 85
 - \mathcal{OSF} -clause, 81
 - \mathcal{OSF} -implication, 91
- ensembles
 - (\mathcal{AF}), 22
 - (\mathcal{ALNR}), 18
 - (\mathcal{FLN}), 14
- équivalence (\mathcal{FLN}), 16
- exclusion (\mathcal{FLN}), 16
- EXP (\mathcal{ALNR}), 26
- expansion (fonction, \mathcal{ALNR}), 26
- extension
 - (\mathcal{ALNR}), 19
 - (\mathcal{FLN}), 15
- \mathcal{FLN}
 - extension, 15
 - structure, 16
 - subsomption, 16
 - syntaxe, 13
 - terminologie, 15
 - (sous-ensembles), 14
 - conséquence modulo une terminologie, 23
 - structure respectant une description du monde, 23
 - subsomption modulo une description du monde, 23
- formule

- associée à un \mathcal{S} -graphe (\mathcal{S}), 65
- \mathcal{S} -formule, 66
- graphe
 - algèbre de graphe (\mathcal{OSF}), 89
 - canonique, 92
 - bipartie, 55
 - enraciné, 85
 - multi, 55
 - \mathcal{OSF} -graphe, 85
 - \mathcal{S} -graphe, 57
 - en forme normale, 59
 - \mathcal{S} -sous-graphe, 57
 - terminologique, 14
 - étiqueté, 55
 - étoile (\mathcal{S}), 56
- généralisation
 - \mathcal{S} -graphe, 62
 - règles, 62
- homomorphisme (\mathcal{OSF}), 92
- implication (\mathcal{OSF}), 91
 - enracinée, 91
- incohérence (\mathcal{FLN}), 16
- interprétation
 - (\mathcal{AF}), 22
 - (\mathcal{U}), 41
- irredondant (\mathcal{S} -graphe), 64
- isomorphisme (\mathcal{S} -isomorphisme), 60
- modèle
 - (\mathcal{AF}), 23
 - (\mathcal{FLN}) respectant une terminologie, 23
- monde (description du), 21
- morphisme (\mathcal{S}), 60
- multi-ensemble, 55
- multi-graphe, 55
 - étiqueté, 56
- niveau terminologique, 15
- NORM (\mathcal{ALNR}), 28
- normalisation
 - de concept (\mathcal{ALNR}), 28
 - \mathcal{OSF} -clause, 83
- \mathcal{OSF}
 - algèbre, 88
 - admissible pour une \mathcal{OSF} -clause, 91
 - de graphe, 89
 - clause, 81
 - enracinée, 81
 - normalisation, 83
 - résolue, 82
 - satisfaction, 90
 - contrainte, 81
 - graphe, 85
 - associé à un Ψ -terme, 85
 - homomorphisme, 92
 - implication, 91
 - enracinée, 91
 - signature, 79
 - subsomption, 92
 - syntaxe, 79
 - terme, 79
 - associé à un \mathcal{OSF} -graphe, 86
 - en forme normale, 80
 - approximation endomorphe, 93
 - assignation, 88
 - atteignabilité, 81
 - catégorie \mathcal{OSF} , 93
 - dissolution d'un terme, 81
 - dénotation, 91
 - sous une assignation, 91
 - variables d'un \mathcal{OSF} -terme, 79
- problème
 - \mathcal{S} -isoprojection, 70
 - \mathcal{S} -projection, 70
 - injective, 70
- projection (\mathcal{S} -projection), 60
- Ψ -terme, 80
 - associé à un \mathcal{OSF} -graphe, 87
 - ordonné, 80
- résolue (\mathcal{OSF} -clause), 82
- \mathcal{S}
 - formule, 66
 - graphe, 57
 - en forme normale, 59
 - irredondant, 64

- généralisation, 62
- isomorphisme, 60
- morphisme, 60
- projection, 60
- sous-graphe, 57
- spécialisation, 61
- substitution, 67
 - valide, 67
- formule associée à un \mathcal{S} -graphe, 65
- axiomes associés au support, 65
- graphe étoile, 56
- isoprojection (problème), 70
- projection (problème), 70
 - injective, 70
- support, 56
- satisfaction (\mathcal{OSF} -clause), 90
- signature (\mathcal{OSF}), 79
- spécialisation
 - règles (\mathcal{S}), 61
 - \mathcal{S} -spécialisation, 61
- structure
 - (\mathcal{FLN})
 - respectant une description du monde, 23
 - (\mathcal{ALNR}), 20
 - (\mathcal{FLN}), 16
- subsomption
 - (\mathcal{FLN})
 - modulo une description du monde, 23
 - (\mathcal{ALNR}), 28
- (\mathcal{FLN}), 16
- (\mathcal{OSF}), 92
- substitution (\mathcal{S}), 67
 - valide, 67
- support (\mathcal{S}), 56
- syntaxe
 - (\mathcal{AF}), 21
 - (\mathcal{ALNR}), 18
 - (\mathcal{FLN}), 13
 - (\mathcal{OSF}), 79
 - (\mathcal{U}), 39
- terme
 - \mathcal{OSF} -terme, 79
 - associé à un \mathcal{OSF} -graphe, 86
 - en forme normale, 80
 - Ψ -terme, 80
 - associé a un \mathcal{OSF} -graphe, 87
 - ordonné, 80
- terminologie
 - (\mathcal{FLN}), 15
 - graphe, 14
 - niveau, 15
- terminologique, 10
- \mathcal{U}
 - interprétation, 41
 - syntaxe, 39
- validité (\mathcal{S} -substitution), 67
- variables d'un \mathcal{OSF} -terme, 79
- Ventura, Ray et ses collégiens, 47