INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE

Une introduction aux logiques de descriptions

Amedeo Napoli

N° 3314

Décembre 1997

THÈME 3

Rapport de recherche

ISSN 0249-6399

Une introduction aux logiques de descriptions

Amedeo Napoli

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet SYCO

Rapport de recherche n° 3314 — Décembre 1997 — 72 pages

Résumé : Ce rapport de recherche présente une introduction élémentaire aux logiques de descriptions, qui forment une famille de langages de représentation de connaissances. Les logiques de descriptions permettent de représenter les connaissances d'un domaine à l'aide de concepts (classes d'individus), de rôles (relations entre classes) et d'individus. Les concepts et les rôles sont organisés en hiérarchies sur lesquelles opèrent les processus de classification et d'instanciation, qui sont à la base du raisonnement terminologique.

Mots-clés : logiques de descriptions, représentation des connaissances, raisonnement, subsomption, classification.

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifique,
615, rue de Jardin Botanique, B.P. 101, 54602 Villers Lès Nancy (France)
Téléphone : 03 83 59 30 30 - International : +33 3 3 83 59 30 30
Télécopie : 03 83 27 83 19 - International : +33 3 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
Téléphone : 03 87 20 35 00 - International : +33 3 87 20 35 00
Télécopie : 03 87 76 39 77 - International : +33 3 87 76 39 77

An Introduction to Description Logics

Abstract: This research report presents description logics, a knowledge representation formalism allowing to represent domain knowledge with concepts (classes of individuals), roles (relations between classes) and individuals. Concepts and roles are organized within hierarchies on which operates terminological reasoning (based on classification and instantiation).

Key-words: description logics, knowledge representation, reasoning, subsumption, classification.

Préambule

Ce rapport de recherche présente une introduction élémentaire aux logiques de descriptions, qui forment une famille de langages de représentation de connaissances. Les logiques de descriptions permettent de représenter les connaissances relatives à un domaine de référence à l'aide de « descriptions » qui peuvent être des concepts, des rôles et des individus. Les concepts modélisent des classes d'individus et les rôles des relations entre classes. Une sémantique est associée aux descriptions par l'intermédiaire d'une fonction d'interprétation. La relation de subsomption permet d'organiser les concepts et les rôles en hiérarchies ; la classification et l'instanciation sont les opérations qui sont alors à la base du raisonnement sur les descriptions, ou raisonnement terminologique. La classification permet de déterminer la position d'un concept et d'un rôle dans leurs hiérarchies respectives, tandis que l'instanciation permet de retrouver les concepts dont un individu est susceptible d'être une instance.

La première partie du rapport de recherche introduit les éléments de base de la théorie des logiques de descriptions. La détection des relations de subsomption repose soit sur l'usage d'algorithmes dits de normalisation–comparaison (NC) soit sur l'usage de la méthode des tableaux sémantiques. Un algorithme NC élémentaire est détaillé dans le texte, tandis qu'un algorithme plus général et plus complet est donné en annexe. La méthode des tableaux sémantiques appliquée dans le cadre des logiques de descriptions est exposée dans le troisième paragraphe. Les aspects pratiques du raisonnement terminologique sont abordés dans le quatrième paragraphe, et donnent lieu à la présentation de deux systèmes terminologiques parmi les plus caractéristiques, CLASSIC et LOOM. Une brève revue de la famille des logiques de descriptions et une analyse de l'apport de ces logiques dans le cadre de la représentation des connaissances font l'objet du cinquième paragraphe. Une conclusion et une annexe contenant le détail de la subsomption dans le langage $\mathcal{ALNR}$ terminent le rapport de recherche.

Note et remerciements

Ce rapport est la version étendue d'un chapitre faisant partie d'un livre intitulé « Langages et modèles à objets : états et perspectives ». Ce livre doit paraître au cours de l'année 1998 dans la collection Didactique de l'INRIA, et il est publié sous la responsabilité de Roland Ducournau (LIRMM, Montpellier), Jérôme Euzenat (INRIA RHÔNE-ALPES, Grenoble), Gérald Masini (CRIN CNRS – INRIA LORRAINE, Nancy), et Amedeo Napoli (CRIN CNRS – INRIA LORRAINE, Nancy).

L'auteur voudrait remercier Roland Ducournau (LIRMM, Montpellier), Jérôme Euzenat (INRIA RHÔNE-ALPES, Grenoble), Gérald Masini (CRIN CNRS – INRIA LORRAINE, Nancy), et Marie-Christine Rousset (LRI, Orsay), qui, par leur lecture attentive, ont grandement contribué à l'amélioration ce rapport de recherche.

Table des matières

1 Introduction **7**

2 Les bases des logiques de descriptions **8**
2.1 Un exemple introductif 8
2.2 La description des concepts et des rôles : syntaxe 10
2.2.1 La famille $\mathcal{AL}$ 10
2.2.2 Les langages $\mathcal{FL}$ et $\mathcal{FL}$- 12
2.3 La description des concepts et des rôles : sémantique 13
2.3.1 La notion d'interprétation dans $\mathcal{ALCNR}$ 13
2.3.2 La notion de satisfiabilité d'un concept 14
2.4 La relation de subsomption 15
2.5 Subsomption et propriétés de treillis 16
2.6 Le niveau terminologique et les méthodes de tests de subsomption 17
2.7 Algorithmes NC pour tester la subsomption 18
2.7.1 La subsomption dans $\mathcal{FL}$- 18
2.7.2 La subsomption dans $\mathcal{ALNR}$ 20
2.8 La notion de base de connaissances terminologique 20
2.8.1 Introduction générale et discussion 20
2.8.2 Exemples d'instanciations 21
2.8.3 Base terminologique et modèle 22
2.8.4 Variantes sur la notion de base terminologique 24
2.9 Un bilan intermédiaire sur le raisonnement terminologique 25
2.9.1 Les opérations associées au raisonnement terminologique . 25
2.9.2 La complexité du raisonnement terminologique 26

3 La méthode des tableaux sémantiques dans les logiques de descriptions **28**
3.1 Brefs rappels sur la méthode des tableaux sémantiques 28
3.2 La satisfiabilité d'une base terminologique 29
3.2.1 Systèmes de contraintes 30
3.2.2 Les règles de décomposition 31
3.2.3 Un exemple d'application des règles de décomposition . . 33
3.2.4 Variante sur le test de satisfiabilité d'une base terminologique 34

4 La pratique du raisonnement terminologique **35**
4.1 Le processus de classification 36
4.2 Règles et requêtes en CLASSIC 37
4.2.1 Les règles 37
4.2.2 Les requêtes 39
4.2.3 Pour conclure sur CLASSIC 40
4.3 L'instanciation en LOOM 40
4.3.1 Le détail du processus d'instanciation en LOOM 40
4.3.2 Pour conclure sur LOOM 43

5 Discussions sur les logiques de descriptions **44**
5.1 La diversité des logiques de descriptions 44
5.2 Les logiques de descriptions et la représentation de connaissances 46

6 Conclusion **51**

7 Annexe : détails sur la subsomption dans $\mathcal{ALNR}$ **53**
7.1 La normalisation des concepts 53
7.2 La comparaison des formes normales 53
7.3 La relation de subsomption et le constructeur restrict 54
7.4 Autres extensions de l'ensemble des constructeurs 57

Index **70**

1 Introduction

Un système à base de connaissances est un programme capable de raisonner sur un domaine d'application pour résoudre un certain problème, en s'aidant de connaissances relatives au domaine étudié. Les connaissances du domaine sont représentées par des entités qui ont une description syntaxique à laquelle est associée une sémantique. Il n'existe pas de méthode universelle pour concevoir des systèmes à base de connaissances, mais un courant de recherche très actif s'est développé autour des idées dont le système KL-ONE [Brachman and Schmolze,1985] est à l'origine. Ce courant de recherche, qui s'est nourri d'études effectuées sur la logique des prédicats, les réseaux sémantiques et les langages de *frames*, a donné naissance à une famille de langages de représentation appelés *logiques de descriptions*, ou encore *logiques terminologiques* [1].

Dans le formalisme des logiques de descriptions, un *concept* permet de représenter un ensemble d'*individus*, tandis qu'un *rôle* représente une relation binaire entre individus. Un concept correspond à une entité générique d'un domaine d'application et un individu à une entité particulière, *instance* d'un concept. Concepts, rôles et individus obéissent aux principes suivants :

- Un concept et un rôle possèdent une *description structurée*, élaborée à partir d'un certain nombre de *constructeurs*. Une sémantique est associée à chaque description de concept et de rôle par l'intermédiaire d'une *interprétation*. Les manipulations opérées sur les concepts et les rôles sont réalisées en accord avec cette sémantique.

- Les connaissances sont prises en compte selon plusieurs niveaux : la représentation et la manipulation des concepts et des rôles relèvent du niveau *terminologique* ; la description et la manipulation des individus relèvent du niveau *factuel* ou niveau des *assertions*. Le niveau terminologique est aussi qualifié de *TBox* et le niveau factuel de *ABox*.

- La relation de *subsomption* permet d'organiser concepts et rôles par niveau de généralité : intuitivement, un concept `C` *subsume* un concept `D` si `C` est plus général que `D` au sens où l'ensemble d'individus représenté par `C` contient l'ensemble d'individus représenté par `D`. Une base de connaissances se compose alors d'une *hiérarchie* de concepts et d'une (éventuelle) hiérarchie de rôles.

- Les opérations qui sont à la base du *raisonnement terminologique* sont la *classification* et l'*instanciation*. La classification s'applique aux concepts, le cas échéant aux rôles, et permet de déterminer la position d'un concept et d'un rôle dans leurs hiérarchies respectives ; la construction et l'évolution de ces hiérarchies est ainsi assistée par le processus de classification.

1. En anglais, ces logiques ont été désignées entre autres par les expressions *description logics*, *terminological logics* et *concept languages*. La première expression semble être devenue leur nom définitif, qui est donc adopté ici.

L'instanciation permet de retrouver les concepts dont un individu est susceptible d'être une instance [2].

La recherche sur les logiques de descriptions est très active aux États Unis, en Allemagne et en Italie, et connaît un intérêt grandissant en France. L'ouvrage de référence est toujours [Nebel,1990a], tandis que [Woods and Schmolze,1992], [Donini *et al.*,1996] et [Donini *et al.*,1997] sont des synthèses très complètes. Ce rapport de recherche est lui aussi une synthèse [3], qui a pour but de mettre en valeur les idées générales sur lesquelles reposent les logiques de descriptions et de montrer la richesse et la rigueur du formalisme associé, ainsi que les possibilités de représentation qui sont offertes.

Après cette introduction, le paragraphe 2 introduit les bases des logiques de descriptions : la syntaxe et la sémantique d'un concept, la relation de subsomption, les notions de bases de connaissances et de raisonnement terminologiques. Il existe deux façons principales d'appréhender la détection des relations de subsomption et le raisonnement terminologique. La première façon s'appuie sur des algorithmes de normalisation–comparaison, appelés ici algorithmes NC. La seconde façon s'appuie sur la méthode des tableaux sémantiques et son application aux logiques de descriptions. Les algorithmes NC sont discutés dans le deuxième paragraphe et la méthode des tableaux sémantiques l'est dans le troisième paragraphe. Dans le paragraphe 4, un point de vue pratique sur le raisonnement terminologique est mis en valeur, et deux systèmes sont examinés pour l'occasion, CLASSIC et LOOM. Le paragraphe 5 propose une vue générale des logiques de descriptions et montre leur apport théorique et pratique en représentation des connaissances. La conclusion est suivie d'une annexe où figurent des compléments algorithmiques sur le test de subsomption.

2 Les bases des logiques de descriptions

2.1 Un exemple introductif

Les entités de base qui sont définies et manipulées dans une logique de descriptions sont les *concepts* et les *rôles*. Un concept dénote un ensemble d'individus – l'*extension* du concept – et un rôle dénote une relation binaire entre individus. Un concept possède une *description structurée* qui se construit à l'aide d'un ensemble de *constructeurs* introduisant les rôles associés au concept et les *restrictions* attachées à ces rôles. Les restrictions portent généralement sur le *co-domaine* du rôle, qui est le concept avec lequel le rôle établit une relation, et la *cardinalité* du rôle, qui fixe le nombre minimal et maximal de *valeurs élémentaires* que peut prendre le rôle. Les valeurs élémentaires sont des instances de concepts ou bien des valeurs qui relèvent des types de bases comme *entier*, *réel*, et *chaîne de caractères*.

Les concepts peuvent être *primitifs* ou *définis*. Les concepts primitifs sont comparables à des atomes et servent de base à la construction des concepts

2. Dans les langages à objets [Masini *et al.*,1989], l'instanciation a un sens différent : elle désigne l'opération qui permet d'engendrer des instances à partir d'une classe.

3. Qui reprend et actualise le rapport de recherche [Napoli and Volle,1993].

```
PERSONNE ≤̇ TOP
ENSEMBLE ≤̇ TOP
HOMME ≤̇ PERSONNE
FEMME ≤̇ (and PERSONNE (not HOMME))
membre ≤̇ toprole
chef ≤̇ membre

ÉQUIPE ≐ (and ENSEMBLE
              (all membre PERSONNE)
              (atleast 2 membre))

PETITE-ÉQUIPE ≐ (and ÉQUIPE
                     (atmost 5 membre))

ÉQUIPE-MODERNE ≐ (and ÉQUIPE
                      (atmost 4 membre)
                      (atleast 1 chef)
                      (all chef FEMME))
```

FIG. 1: *Un ensemble de concepts primitifs, de rôles primitifs et de concepts définis, d'après [Nebel,1990a].*

définis, c'est-à-dire qui possèdent une *définition*. À l'image d'un concept, un rôle peut être primitif ou défini et peut posséder une description structurée, où figurent les propriétés associées au rôle.

À la figure 1, nous donnons un exemple de concepts et de rôles primitifs et définis. Par convention, un nom de concept est en majuscules, un nom de rôle en minuscules, et un nom d'individu commence par une majuscule. Les concepts `PERSONNE` et `ENSEMBLE` sont des concepts primitifs : ils sont introduits par le symbole $\dot{\leq}$ et sont *subsumés* par `TOP`, qui dénote le concept le plus général, appelé aussi la *racine* de la hiérarchie des concepts. Le constructeur `and` indique qu'un concept est construit à partir d'une conjonction de concepts — qui sont les ascendants directs du nouveau concept — et le constructeur `all` précise le co-domaine d'un rôle. Le constructeur `not` exprime la négation et ne s'applique qu'à des concepts primitifs. Les constructeurs `atleast` et `atmost` précisent la cardinalité du rôle auquel ils sont associés, et indiquent respectivement le nombre minimal et maximal de valeurs élémentaires du rôle.

Les concepts `HOMME` et `FEMME` sont primitifs et *incompatibles* : l'ensemble des femmes est dans le complémentaire de celui des hommes. Les trois concepts `ÉQUIPE`, `PETITE-ÉQUIPE` et `ÉQUIPE-MODERNE` sont des concepts définis introduits par le symbole $\doteq$. Une équipe est définie comme un ensemble de personnes qui compte au moins deux membres. Une petite équipe est définie comme une équipe qui compte au plus cinq membres. Une équipe moderne est définie comme une équipe qui compte au plus quatre membres, au moins un chef et dont tous les chefs sont des femmes. Du côté des rôles, `membre` et `chef` sont primitifs : `chef` est subsumé par `membre`, et `membre` par `toprole`, qui dénote le rôle le plus général.

Deux remarques principales peuvent être faites sur ce premier exemple. La première est que la description d'un concept — `TOP` excepté — *dérive* toujours d'un ou de plusieurs autres concepts, qui sont ses *subsumants* dans la hiérarchie des concepts (comme c'est le cas pour les sous-classes qui *spécialisent* les classes dans un langages à objets). La seconde remarque concerne le statut des concepts : certains concepts, comme `ÉQUIPE`, `PETITE-ÉQUIPE` et `ÉQUIPE-MODERNE` possèdent une *définition* ; d'autres, comme `PERSONNE`, `ENSEMBLE`, `HOMME` ou encore `FEMME` sont primitifs. Les caractéristiques associées à un concept primitif sont *nécessaires* : un individu `x` qui est une instance d'un concept primitif `P` possède les caractéristiques de `P` (comme c'est le cas pour les instances des classes dans un langages à objets [Euzenat,1993]). Les caractéristiques associées à un concept défini `D` sont *nécessaires et suffisantes* : un individu `x` qui est une instance d'un concept défini `D` possède les caractéristiques de `D`, et inversement, le fait qu'un individu `y` possède l'ensemble des caractéristiques associées à `D` suffit pour inférer que `y` est une instance de `D`. Cette distinction est à la base du processus de classification, qui est explicité au paragraphe 4.1 : les concepts sont définis de façon déclarative et la mise en place des concepts définis dans la hiérarchie des concepts est effectuée sous le contrôle du processus de classification.

2.2 La description des concepts et des rôles : syntaxe

Il existe plusieurs langages de description de concepts et de rôles. Dans ce qui suit, nous introduisons d'abord un langage minimal appelé $\mathcal{AL}$, qui est enrichi progressivement de nouveaux constructeurs. Le langage $\mathcal{AL}$ s'appuie sur les langages $\mathcal{FL}$ et $\mathcal{FL}$-, présentés ci-dessous, qui sont les langages pour lesquels ont été établis les premiers résultats théoriques sur les logiques de descriptions.

2.2.1 La famille $\mathcal{AL}$

La grammaire de $\mathcal{AL}$ est donnée à la figure 2. Les expressions construites grâce à une telle grammaire sont aussi appelées *expressions conceptuelles*. À la syntaxe *lispienne*, où le nom des constructeurs est donné en toutes lettres et où la notation est préfixée, correspond une syntaxe *allemande*, qui est utilisée dans la plupart des articles théoriques traitant des logiques de descriptions. Dans la suite, les deux syntaxes seront utilisées indifféremment.

Le concept `TOP` ($\top$) dénote le concept le plus général et le concept `BOTTOM` ($\perp$) le concept le plus spécifique. Intuitivement, l'extension de `TOP` inclut tous les individus possibles tandis que celle de `BOTTOM` est vide. Le constructeur `and` ($\sqcap$) permet de définir une conjonction d'expressions conceptuelles. Le constructeur `not` ($\neg$) correspond à la négation et ne porte que sur les concepts primitifs. La *quantification universelle* `all` ($\forall$`r.C`) précise le co-domaine du rôle `r` ; la *quantification existentielle* non typée `some` ($\exists$ `r`) introduit le rôle `r` et affirme l'existence d'(au moins) un couple d'individus en relation par l'intermédiaire de `r`.

C,D $\longrightarrow$	A |	
	Top |	$\top$
	Bottom |	$\bot$
	(and C D) |	$\mathtt{C} \sqcap \mathtt{D}$
	(not A) |	$\neg\mathtt{A}$
	(all r C) |	$\forall\mathtt{r.C}$
	(some r)	$\exists\,\mathtt{r}$
	syntaxe lispienne	syntaxe allemande

FIG. 2: *La grammaire du langage de description de concepts $\mathcal{AL}$, avec les syntaxes lispienne et allemande.* `C` *et* `D` *sont des noms de concepts,* `A` *un nom de concept primitif et* `r` *un nom de rôle primitif.*

Le langage $\mathcal{AL} = \{\top, \bot, \neg\mathtt{A}, \mathtt{C} \sqcap \mathtt{D}, \forall\mathtt{r.C}, \exists\,\mathtt{r}\}$ peut être enrichi des constructeurs suivants :

- La négation de concepts primitifs ou définis, qui est notée (`not C`) ou $\neg\mathtt{C}$. L'extension correspondante de $\mathcal{AL}$ est $\mathcal{ALC} = \mathcal{AL} \cup \{\neg\mathtt{C}\}$.

- La disjonction de concepts, qui est notée (`or C D`) ou $\mathtt{C} \sqcup \mathtt{D}$. L'extension correspondante de $\mathcal{AL}$ est $\mathcal{ALU} = \mathcal{AL} \cup \{\mathtt{C} \sqcup \mathtt{D}\}$.

  Notons les équivalences $\bot \equiv \mathtt{C} \sqcup \neg\mathtt{C}$ et $\mathtt{C} \sqcup \mathtt{D} \equiv \neg(\neg\mathtt{C} \sqcap \neg\mathtt{D})$ [Schmidt-Schauß and Smolka,1991].

- La quantification existentielle typée, qui est notée (`c-some r C`) ou $\exists\mathtt{r.C}$. L'extension correspondante de $\mathcal{AL}$ est $\mathcal{ALE} = \mathcal{AL} \cup \{\exists\mathtt{r.C}\}$.

  La quantification existentielle typée $\exists\mathtt{r.C}$ introduit un rôle `r` de co-domaine `C` et impose l'existence d'(au moins) un couple d'individus `(x,y)` en relation par l'intermédiaire du rôle `r`, où `C` est le type de `y` [Donini *et al.*,1992].

  Notons les équivalences $\exists\,\mathtt{r} \equiv \exists\mathtt{r}.\top$ et $\exists\mathtt{r.C} \equiv \neg(\forall\mathtt{r}.\neg\mathtt{C})$.

- La cardinalité sur les rôles, qui est notée (`atleast n r`) ou $\geq\mathtt{n\,r}$, et (`atmost n r`) ou $\leq\mathtt{n\,r}$. L'extension correspondante de $\mathcal{AL}$ est $\mathcal{ALN} = \mathcal{AL} \cup \{\geq\mathtt{n\,r}, \leq\mathtt{n\,r}\}$.

  Les constructeurs $\geq\mathtt{n\,r}$ et $\leq\mathtt{n\,r}$ fixent la cardinalité — nombre de valeurs élémentaires — minimale et maximale du rôle auquel ils sont associés. En particulier, la construction $(\exists\,\mathtt{r})$ est équivalente à la construction $(\geq\mathtt{1\,r})$.

- La conjonction de rôles, qui est notée (`and` $\mathtt{r_1}$ $\mathtt{r_2}$) ou $\mathtt{r_1} \sqcap \mathtt{r_2}$, les rôles $\mathtt{r_1}$ et $\mathtt{r_2}$ étant primitifs. L'extension correspondante de $\mathcal{AL}$ est $\mathcal{ALR} = \mathcal{AL} \cup \{\mathtt{r_1} \sqcap \mathtt{r_2}\}$.

  Si $\mathtt{r} = \mathtt{r_1} \sqcap \mathtt{r_2}$, alors `r` est un sous-rôle de $\mathtt{r_1}$ et de $\mathtt{r_2}$. Par extension, il est possible de considérer une hiérarchie de rôles comme un ensemble de conjonctions de rôles et de se passer du rôle toprole — ce qui est d'ailleurs le cas pour $\mathcal{ALCNR}$— comme expliqué dans [Nebel,1990a, page 54]. Ainsi, le rôle chef peut être défini par membre $\sqcap$ chef-prim, où chef-prim est un rôle primitif au même titre que membre.

Comme le montrent les équivalences $\mathtt{C} \sqcup \mathtt{D} \equiv \neg(\neg\mathtt{C} \sqcap \neg\mathtt{D})$ et $\exists\mathtt{r.C} \equiv \neg(\forall\mathtt{r.}\neg\mathtt{C})$, la disjonction et la quantification existentielle typée sont disponibles dans un langage $\mathcal{L}$ de la famille $\mathcal{AL}$ dès lors que la négation de concepts (définis) l'est, et réciproquement. Par convention, un langage $\mathcal{L}$ de la famille $\mathcal{AL}$ est supposé contenir tous les constructeurs qui peuvent être obtenus à partir de ses constructeurs de base. Ainsi, le langage $\mathcal{ALCNR} = \{\top, \bot, \neg\mathtt{C}, \mathtt{C} \sqcap \mathtt{D}, \mathtt{C} \sqcup \mathtt{D}, \forall\mathtt{r.C}, \exists\mathtt{r.C}, \geq\mathtt{n\,r}, \leq\mathtt{n\,r}, \mathtt{r}_1 \sqcap \mathtt{r}_2\}$ contient l'ensemble des constructeurs qui viennent d'être présentés (il est aussi d'usage d'employer la lettre $\mathcal{C}$ plutôt que les lettres $\mathcal{UE}$ dans le nom du langage). D'autres équivalences de langages sont discutées par exemple dans [Donini *et al.*,1997] (simulation de la quantification existentielle typée avec la conjonction de rôles) et dans [Baader,1996] (simulation de la négation de concepts primitifs avec les restrictions de cardinalité sur les rôles).

Tous les constructeurs envisageables pour une logique de descriptions n'ont pas été présentés ici et il est toujours possible d'enrichir le langage $\mathcal{ALCNR}$. Une liste très complète de constructeurs est donnée dans la spécification KRSS , pour *Knowledge Representation Specification System*, introduite dans [Baader *et al.*,1990] et reprise dans [Patel-Schneider and Swartout,1993]. Cette spécification sert de référence de base aux chercheurs du domaine, et décrit de façon systématique les différents composants d'une logique de descriptions et un ensemble important de constructeurs, accompagnés de leur sémantique.

2.2.2 Les langages $\mathcal{FL}$ et $\mathcal{FL}$-

Les langages $\mathcal{FL}$ et $\mathcal{FL}$- sont les premières logiques de descriptions[4] à l'aide desquelles ont été établis des résultats théoriques sur la complexité de la subsomption [Brachman and Levesque,1984] [Levesque and Brachman,1987] [Nebel,1988]. Voici la grammaire sur laquelle repose $\mathcal{FL}$:

```
C,D ⟶ A | (and C D) | (all r C) | (some r)
r ⟶ p | (restrict r C)
```

Les symboles A et p désignent respectivement un nom de concept primitif et un nom de rôle primitif. Le constructeur restrict, dont la notation allemande est $\mathtt{r}\,|\mathtt{C}$, enrichit le langage de description des rôles. Il introduit une contrainte sur le co-domaine d'un rôle et peut être rapproché de la quantification existentielle typée (c-some r C). Toutefois, d'une part, (c-some r C) est un constructeur qui s'applique aux concepts tandis que (restrict r C) s'applique aux rôles, et d'autre part, (restrict r C) ne fait que restreindre le co-domaine du rôle r au concept C, sans aucune contrainte d'existence comme celle qui est associée à (c-some r C).

4. Pour la petite histoire, dans [Brachman and Levesque,1984], les logiques de descriptions sont qualifiées de *frame-based description languages*, d'où les initiales $\mathcal{FL}$.

La description D représentant l'expression « une personne qui a des enfants et dont chacun des enfants pauvres est musicien et dont chacun des enfants riches est docteur » est donnée en $\mathcal{FL}$ par :

```
D ≐ (and PERSONNE
         (some enfant)
         (all (restrict enfant PAUVRE) MUSICIEN)
         (all (restrict enfant RICHE) DOCTEUR))
```

Le langage $\mathcal{FL}$- est une simplification de $\mathcal{FL}$ dans laquelle le constructeur `restrict` n'est pas utilisé.

2.3 La description des concepts et des rôles : sémantique

2.3.1 La notion d'interprétation dans $\mathcal{ALCNR}$

À l'instar de la logique classique, une sémantique est associée aux descriptions de concepts et de rôles : les concepts sont interprétés comme des sous-ensembles d'un domaine d'interprétation $\Delta_\mathcal{I}$ et les rôles comme des sous-ensembles du produit $\Delta_\mathcal{I} \times \Delta_\mathcal{I}$. Pour un concept C, $\mathtt{C}^\mathcal{I}$ correspond au sous-ensemble des éléments du domaine $\Delta_\mathcal{I}$ qui appartiennent à l'extension de C, et pour un rôle r, $\mathtt{r}^\mathcal{I}$ correspond au sous-ensemble des couples d'éléments du produit $\Delta_\mathcal{I} \times \Delta_\mathcal{I}$ qui appartiennent à l'extension de r. La définition suivante est donnée dans le cadre du langage $\mathcal{ALCNR}$ et introduit la notion d'*interprétation* dans les logiques de descriptions.

Définition 1 ***(Interprétation)***
Une interprétation $\mathcal{I} = (\Delta_\mathcal{I}, .^\mathcal{I})$ *est la donnée d'un ensemble* $\Delta_\mathcal{I}$ *appelé* domaine de l'interprétation *et d'une fonction d'interprétation* $.^\mathcal{I}$ *qui fait correspondre à un concept un sous-ensemble de* $\Delta_\mathcal{I}$ *et à un rôle un sous-ensemble de* $\Delta_\mathcal{I} \times \Delta_\mathcal{I}$*, de telle sorte que les équations suivantes soient satisfaites :*

$$
\begin{aligned}
\top^\mathcal{I} &= \Delta_\mathcal{I} \\
\bot^\mathcal{I} &= \emptyset \\
(\mathtt{C} \sqcap \mathtt{D})^\mathcal{I} &= \mathtt{C}^\mathcal{I} \cap \mathtt{D}^\mathcal{I} \\
(\mathtt{C} \sqcup \mathtt{D})^\mathcal{I} &= \mathtt{C}^\mathcal{I} \cup \mathtt{D}^\mathcal{I} \\
(\neg \mathtt{C})^\mathcal{I} &= \Delta_\mathcal{I} - \mathtt{C}^\mathcal{I} \\
(\forall \mathtt{r.C})^\mathcal{I} &= \{\mathtt{x} \in \Delta_\mathcal{I} / \forall \mathtt{y} : (\mathtt{x}, \mathtt{y}) \in \mathtt{r}^\mathcal{I} \rightarrow \mathtt{y} \in \mathtt{C}^\mathcal{I}\} \\
(\exists \mathtt{r.C})^\mathcal{I} &= \{\mathtt{x} \in \Delta_\mathcal{I} / \exists \mathtt{y} : (\mathtt{x}, \mathtt{y}) \in \mathtt{r}^\mathcal{I} \wedge \mathtt{y} \in \mathtt{C}^\mathcal{I}\} \\
(\geq \mathtt{n}\, \mathtt{r})^\mathcal{I} &= \{\mathtt{x} \in \Delta_\mathcal{I} / |\{\mathtt{y} \in \Delta_\mathcal{I} / (\mathtt{x}, \mathtt{y}) \in \mathtt{r}^\mathcal{I}\}| \geq n\} \\
(\leq \mathtt{n}\, \mathtt{r})^\mathcal{I} &= \{\mathtt{x} \in \Delta_\mathcal{I} / |\{\mathtt{y} \in \Delta_\mathcal{I} / (\mathtt{x}, \mathtt{y}) \in \mathtt{r}^\mathcal{I}\}| \leq n\} \\
(\mathtt{r_1} \sqcap \cdots \sqcap \mathtt{r_n})^\mathcal{I} &= \mathtt{r_1}^\mathcal{I} \cap \cdots \cap \mathtt{r_n}^\mathcal{I}
\end{aligned}
$$

Intuitivement, les équations précédentes se comprennent comme suit. L'interprétation de $\top$ est le domaine $\Delta_{\mathcal{I}}$ tout entier tandis que celle de $\bot$ se réduit à l'ensemble vide. L'interprétation d'une conjonction (respectivement d'une disjonction) de concepts se ramène à l'intersection (respectivement la réunion) des interprétations des concepts. L'interprétation de la négation d'un concept C se ramène au complémentaire de l'interprétation de C. L'interprétation de $(\forall \mathtt{r.C})$ précise le type du co-domaine du rôle r, tandis que celle de $(\exists \mathtt{r.C})$ affirme l'existence d'un couple d'éléments $(\mathtt{x}, \mathtt{y})$ en relation par l'intermédiaire du rôle r, où C est le type de y. Les interprétations de $(\geq \mathtt{n\,r})$ et $(\leq \mathtt{n\,r})$ précisent le nombre d'éléments de l'ensemble de valeurs élémentaires associées au rôle r. Enfin, l'interprétation d'une conjonction de rôles se ramène à l'intersection des interprétations des rôles.

L'interprétation de la construction $(\exists\,\mathtt{r})$ est un cas particulier de celle de $(\exists \mathtt{r.C})$, où $\mathtt{C} \equiv \top$: $(\exists\,\mathtt{r})^{\mathcal{I}} = \{\mathtt{x} \in \Delta_{\mathcal{I}} / \exists \mathtt{y} : (\mathtt{x}, \mathtt{y}) \in \mathtt{r}^{\mathcal{I}}\}$; il existe au moins un couple $(\mathtt{x}, \mathtt{y})$ tel que x et y soient en relation par l'intermédiaire de r (le type de y est ici quelconque).

Par exemple, étant donné un domaine d'interprétation $\Delta_{\mathcal{I}}$, l'interprétation du concept $\mathtt{C} \doteq$ (and PERSONNE (atleast 1 enfant)), où enfant est un rôle primitif, est donnée par l'intersection des ensembles $\mathtt{PERSONNE}^{\mathcal{I}}$ et $(\mathtt{atleast\ 1\ enfant})^{\mathcal{I}}$. Le dernier ensemble contient les couples d'éléments $(\mathtt{x}, \mathtt{y})$ où $\mathtt{x} \in \Delta_{\mathcal{I}}$ et tels que $|\{\mathtt{y} \in \Delta_{\mathcal{I}} / (\mathtt{x}, \mathtt{y}) \in \mathtt{enfant}^{\mathcal{I}}\}| \geq 1$. Autrement dit, tout x dans $\mathtt{C}^{\mathcal{I}}$ est dans $\mathtt{PERSONNE}^{\mathcal{I}}$ et il est en relation avec au moins un y de $\Delta_{\mathcal{I}}$ par l'intermédiaire du rôle enfant.

Pour être complet, il reste à donner l'interprétation du constructeur restrict associé à $\mathcal{FL}$: $(\mathtt{restrict\ r\ C})^{\mathcal{I}} = \{(\mathtt{x}, \mathtt{y}) \in \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}} / (\mathtt{x}, \mathtt{y}) \in \mathtt{r}^{\mathcal{I}} \wedge \mathtt{y} \in \mathtt{C}^{\mathcal{I}}\}$; le couple (x,y) fait partie de l'extension du rôle r et y fait partie de l'extension du concept C.

La définition 1 montre une façon classique d'associer une sémantique aux concepts et aux rôles de $\mathcal{ALCNR}$. Il existe des façons moins classiques de le faire : ainsi, celle qui est donnée dans [Borgida,1992] s'appuie sur la déduction naturelle ; celle qui est donnée dans [Royer and Quantz,1992] et [Royer and Quantz,1993] s'appuie sur le calcul des séquents.

2.3.2 La notion de satisfiabilité d'un concept

Définition 2 ***(Satisfiabilité d'un concept, équivalence, incompatibilité)***

- *Un concept* C *est* satisfiable *ou* cohérent *si et seulement s'il existe une interprétation $\mathcal{I}$ telle que* $\mathtt{C}^{\mathcal{I}} \neq \emptyset$ *;* C *est* non satisfiable *ou* incohérent *sinon.*

- *Deux concepts* C *et* D *sont dits* équivalents*, ce qui se note* $\mathtt{C} \equiv \mathtt{D}$*, si et seulement si* $\mathtt{C}^{\mathcal{I}} = \mathtt{D}^{\mathcal{I}}$ *pour toute interprétation $\mathcal{I}$.*

- *Deux concepts* C *et* D *sont* incompatibles *ou* disjoints *si et seulement si* $\mathtt{C}^{\mathcal{I}} \cap \mathtt{D}^{\mathcal{I}} = \emptyset$ *pour toute interprétation $\mathcal{I}$.*

Considérons par exemple les expressions conceptuelles suivantes, où les concepts et les rôles employés sont supposés primitifs :

(1) `(and HOMME (some enfant MUSICIEN))`
(2) `(and FEMME (all enfant HOMME))`
(3) `(and FEMME (all enfant (and MUSICIEN HOMME)))`
(4) `(and (atmost 0 r) (atleast 1 r))`
(5) `(some r (and A (not A)))`

Les trois premières expressions conceptuelles sont satisfiables : (1) représente l'ensemble des individus qui sont des hommes et qui ont un enfant musicien ; (2) représente l'ensemble des individus qui sont des femmes et dont tous les enfants sont des garçons ; (3) représente l'ensemble des individus qui sont des femmes et dont tous les enfants sont des garçons et des musiciens. En revanche, les quatrième et cinquième expressions conceptuelles sont non satisfiables : quel que soit le rôle `r`, une instance de l'expression conceptuelle (4) devrait être munie simultanément d'au moins une valeur et d'au plus zéro valeur pour `r` ; l'expression conceptuelle (5) suppose quant à elle l'existence d'un élément dans l'ensemble vide.

Une des particularités de $\mathcal{FL}^-$, qui est un langage pour lequel aucune forme de négation n'est disponible, est que tout concept défini en $\mathcal{FL}^-$ est satisfiable ; un résultat plus fort est démontré dans [Schmidt-Schauß and Smolka,1991] : tout concept de $\mathcal{FL}$ est satisfiable.

2.4 La relation de subsomption

La *subsomption* est la relation qui permet d'organiser les concepts et les rôles[5] en hiérarchies. Intuitivement, un concept `C` *subsume* un concept `D` si et seulement si l'extension de `C` contient nécessairement — quelle que soit l'interprétation choisie — l'extension de `D`. Plus formellement, nous avons :

Définition 3 ***(Subsomption)***
Un concept `D` *est* subsumé *par un concept* `C` *(respectivement* `C` subsume `D`*), ce qui se note* $\mathtt{D} \sqsubseteq \mathtt{C}$ *(respectivement* $\mathtt{C} \sqsupseteq \mathtt{D}$*) si et seulement si* $\mathtt{D}^{\mathcal{I}} \subseteq \mathtt{C}^{\mathcal{I}}$ *pour toute interprétation* $\mathcal{I}$*. Le concept* `C` *est appelé le* subsumant *et* `D` *le* subsumé*.*

Par exemple, un concept est subsumé par tous les concepts qui ont été déclarés comme ses subsumants directs : `PERSONNE` est subsumé par `TOP`, `HOMME` par `PERSONNE`, etc. (voir figure 1). De plus, le concept `PETITE-ÉQUIPE` subsume le concept `ÉQUIPE-MODERNE` car toute instance du second concept vérifie nécessairement toutes les caractéristiques du premier concept, essentiellement le fait qu'une petite équipe compte au plus `5` membres. Par suite, le concept `ÉQUIPE-MODERNE` est un subsumé du concept `PETITE-ÉQUIPE`.

La relation de subsomption est *réflexive* (un concept est subsumé par lui-même) et *transitive* (si `E` est subsumé par `D` et `D` est subsumé par `C` alors `E`

5. La subsomption de rôles n'est véritablement abordée que dans l'annexe 7.3. Un traitement systématique de la subsomption de rôles est donné dans [Royer and Quantz,1993].

est subsumé par C). Elle est aussi *antisymétrique* (si D est subsumé par C et C est subsumé par D, alors C = D) si deux concepts ayant la même extension sont identifiés, ce que nous supposerons. La relation de subsomption est donc une relation d'ordre partiel, qui organise les concepts en une hiérarchie, où tout concept se compose d'une description propre définie par des propriétés *locales* et d'une description *partagée* avec ses subsumants (comme c'est le cas entre une sous-classe et ses super-classes dans un langage à objets). La hiérarchie obtenue possède un élément maximal, TOP, qui est le concept qui subsume tous les autres concepts, et un élément minimal, BOTTOM, qui est subsumé par tous les autres concepts.

Nous pouvons reformuler la définition 2 en termes de subsomption :

- Un concept C est non satisfiable si et seulement si $\mathtt{C} \sqsubseteq \perp$.
- Deux concepts C et D sont équivalents si et seulement si $\mathtt{D} \sqsubseteq \mathtt{C}$ et $\mathtt{C} \sqsubseteq \mathtt{D}$. En particulier, $\mathtt{D} \equiv \mathtt{C} \sqcap \mathtt{D}$ si et seulement si $\mathtt{D} \sqsubseteq \mathtt{C}$.
- Deux concepts C et D sont incompatibles si et seulement si $\mathtt{D} \sqcap \mathtt{C} \sqsubseteq \perp$. Cette vision alternative de l'incompatibilité permet d'exprimer que les concepts HOMME et FEMME sont disjoints de la façon suivante : $\mathtt{HOMME} \sqcap \mathtt{FEMME} \sqsubseteq \perp$.

2.5 Subsomption et propriétés de treillis

Les constructeurs $\sqcap$ et $\sqcup$ obéissent aux règles suivantes (pour plus de détails, voir [Royer and Quantz,1993]) :

- $\mathtt{A} \sqcap \mathtt{A} \equiv \mathtt{A}$ et $\mathtt{A} \sqcup \mathtt{A} \equiv \mathtt{A}$ (idempotence).
- $\mathtt{A} \sqcap \mathtt{B} = \mathtt{B} \sqcap \mathtt{A}$ et $\mathtt{A} \sqcup \mathtt{B} = \mathtt{B} \sqcup \mathtt{A}$ (commutativité).
- $\mathtt{A} \sqcap (\mathtt{B} \sqcap \mathtt{C}) = (\mathtt{A} \sqcap \mathtt{B}) \sqcap \mathtt{C}$ et $\mathtt{A} \sqcup (\mathtt{B} \sqcup \mathtt{C}) = (\mathtt{A} \sqcup \mathtt{B}) \sqcup \mathtt{C}$ (associativité).
- Si $\mathtt{D} \sqsubseteq \mathtt{C}$ et $\mathtt{D} \sqsubseteq \mathtt{E}$, alors $\mathtt{D} \sqsubseteq \mathtt{C} \sqcap \mathtt{E}$.
- Si $\mathtt{D} \sqsubseteq \mathtt{C}$ et $\mathtt{E} \sqsubseteq \mathtt{C}$, alors $\mathtt{D} \sqcup \mathtt{E} \sqsubseteq \mathtt{C}$.
- Si $\mathtt{D} \sqsubseteq \mathtt{C}$, alors $\mathtt{D} \sqcap \mathtt{X} \sqsubseteq \mathtt{C}$, où X est une description quelconque.
- Si $\mathtt{D} \sqsubseteq \mathtt{C}$, alors $\mathtt{D} \sqsubseteq \mathtt{C} \sqcup \mathtt{X}$, où X est une description quelconque.

L'ensemble des descriptions (potentielles) pour $\mathcal{ALCNR}$ est un *treillis* [6], où chaque couple de concepts (A,B) admet une borne supérieure, $\mathtt{A} \sqcup \mathtt{B}$, et une borne inférieure, $\mathtt{A} \sqcap \mathtt{B}$.

Étant donné deux descriptions A et B dans $\mathcal{ALCNR}$ telles que $\mathtt{A} \sqsubseteq \mathtt{B}$, le *complémentaire de* A *relativement à* B, noté comp(A, B), est défini comme la description C la plus générale telle que $\mathtt{A} \equiv \mathtt{B} \sqcap \mathtt{C}$ [Borgida,1995]: comp(A, B) correspond à la connaissance additionnelle minimale qu'il manque à une instance de B pour être une instance de A. Par exemple, comp(PETITE-ÉQUIPE,ÉQUIPE) = (atmost 5 membre).

6. Les treillis sont présentés en détails dans [Barbut and Monjardet,1970] ou [Davey and Priestley,1990] par exemple.

Le complémentaire d'un concept relativement à un autre concept et le plus petit subsumant commun de deux concepts — cette opération est introduite dans [Cohen *et al.*,1992] — sont utilisés dans le cadre des théories de l'apprentissage pour construire et organiser des concepts [Kietz and Morik,1994], ou encore pour établir des similarités entre concepts lors de la phase de remémoration dans un système de raisonnement à partir de cas [Coupey and Salotti,1997] [Coupey *et al.*,1997].

Dans le même ordre d'idées, il faut citer les études générales [Gaines,1993] et [Aït-Kaci,1986], qui portent sur les propriétés de treillis dans les systèmes de représentation de connaissances. La première introduit des algèbres de contraintes et une analyse de CLASSIC est menée dans le cadre de telles algèbres ; la seconde porte sur les caractéristiques générales des systèmes de représentation des connaissances dans le cadre de la théorie des types.

2.6 Le niveau terminologique et les méthodes de tests de subsomption

Classiquement, deux types de déclarations se rencontrent au *niveau terminologique* : les introductions de concepts primitifs (symbole $\dot{\leq}$), et les définitions de concepts (symbole $\doteq$). Chacune de ces déclarations s'interprète comme une *équation terminologique*, qui se compose d'un membre gauche où apparaît le nom du nouveau concept et d'un membre droit où apparaît l'expression subsumant le nouveau concept en cas d'introduction de concept primitif, ou la définition du nouveau concept défini. Du côté des rôles, seules les équations terminologiques introduisant des rôles primitifs sont considérées ici.

En général, ces déclarations s'appuient sur les deux principes suivants : d'une part, un nom de concept n'est utilisé qu'une seule fois en partie gauche d'une équation terminologique, d'autre part, il n'existe pas de *circuit terminologique*. Un circuit terminologique apparaît lorsqu'un concept fait référence directement ou indirectement à lui-même lors de son introduction (le concept apparaît donc simultanément en partie gauche et droite de l'équation terminologique, éventuellement par transitivité). Dans la suite, nous supposons qu'il n'existe pas de circuit terminologique dans les définitions de concepts et de rôles [7]. Si l'absence de circuit terminologique est vérifiée, alors il est possible de substituer tout nom de concept défini par sa définition dans n'importe quelle expression conceptuelle. En particulier, cette opération « de développement des définitions » est à la base du processus de normalisation des expressions conceptuelles qui est présenté ci-après.

La subsomption est la relation fondamentale existant entre les descriptions. Pour tester les relations de subsomption entre les descriptions, deux voies ont été principalement explorées jusqu'à présent : les algorithmes de type normalisation–comparaison, abrégés en *algorithmes NC*, et une méthode dérivée de la méthode des tableaux sémantiques en logique classique [Gochet and Gribomont,1991],

7. Un certain nombre d'études sur les circuits terminologiques existent, et nous nous renvoyons le lecteur intéressé à [Baader,1990] [Nebel,1990a] [Nebel,1991], [Buchheit *et al.*,1993] et [De Giacomo and Lenzerini,1997].

que nous appelons plus simplement dans la suite « méthode des tableaux sémantiques ».

Un algorithme NC s'appuie sur un processus de *normalisation* à l'issue duquel tous les composants d'une description sont développés et factorisés : toute l'information partagée est recopiée dans le concept traité ; les concepts définis sont remplacés par leur définition, et ainsi tous les symboles d'une définition développée dénotent des concepts ou des rôles primitifs. Le processus de normalisation produit les *formes normales* des descriptions, qui sont ensuite effectivement comparées. Toutefois, seules les formes normales des concepts définis doivent être comparées, car les relations de subsomption sont systématiquement spécifiées pour les concepts primitifs. Il existe une certaine dualité entre normalisation et comparaison : plus il y a de travail fait lors de la normalisation, moins il y en aura à faire lors de la comparaison.

Dans la méthode des tableaux sémantiques, la question « est-ce que `C` subsume `D` » est transformée en la question « est-ce que $\mathtt{D} \sqcap \neg\mathtt{C}$ est non satisfiable » — ce qui suppose que le langage de description des concepts est muni de la négation des concepts définis — puis la méthode des tableaux sémantiques est utilisée pour répondre à la dernière question (ce qui est présenté plus en détail au paragraphe 3).

S'il est facile de vérifier la *correction* des algorithmes NC, (la réponse fournie par l'algorithme est en accord avec la sémantique, voir par exemple [Nebel,1990a, page 77]), il est beaucoup plus difficile de prouver qu'un tel algorithme est *complet*, autrement dit que toutes les relations de subsomption valides sont détectées par l'algorithme. Il s'avère qu'en dehors des langages simples comme $\mathcal{FL}$-, la plupart des algorithmes NC sont incomplets, ce qui se montre en exhibant des contre-exemples (comme il en est fourni dans [Heinsohn *et al.*,1994] par exemple). En revanche, il est plus facile de démontrer la correction et la complétude des algorithmes qui s'appuient sur la méthode des tableaux sémantiques, ainsi que d'étudier leur complexité et la décidabilité de la logique associée.

Dans ce qui suit, nous abordons les deux types de techniques, algorithmes NC au paragraphe suivant et dans l'annexe 7, et la méthode des tableaux sémantiques au paragraphe 3.

2.7 Algorithmes NC pour tester la subsomption

2.7.1 La subsomption dans $\mathcal{FL}$-

Nous présentons dans ce paragraphe l'algorithme (de test) de subsomption associé à $\mathcal{FL}$-, qui, malgré sa simplicité, illustre bien les processus de normalisation et de comparaison des formes normales (rappelons que le langage $\mathcal{FL}$- est défini par l'ensemble de constructeurs $\{\mathtt{C} \sqcap \mathtt{D}, \forall \mathtt{r.C}, \exists\, \mathtt{r}\}$).

Le but de la normalisation est de mettre les concepts définis `C` et `D` qui sont comparés sous la forme de conjonctions $\mathtt{C} = (\mathtt{and}\ \mathtt{C}_1\ \mathtt{C}_2 ... \mathtt{C}_n)$ et $\mathtt{D} = (\mathtt{and}\ \mathtt{D}_1\ \mathtt{D}_2 ... \mathtt{D}_m)$. Dans la présentation de l'algorithme, le symbole $\rightarrow$ doit se lire « se réécrit en » :

- *Normalisation* : factorisation des conjonctions.
  $(\mathtt{and}\ \mathtt{C}_1\ (\mathtt{and}\ \mathtt{C}_2\ \mathtt{C}_3)\ \mathtt{C}_4) \rightarrow (\mathtt{and}\ \mathtt{C}_1\ \mathtt{C}_2\ \mathtt{C}_3\ \mathtt{C}_4)$.

```
D ≐ (and HOMME                                   D1
          (some enfant)                          D2
          (all ami (and DOCTEUR                  D3
                        (some spécialité))))

C ≐ (and PERSONNE                                C1
          (all ami DOCTEUR))                     C2
```

FIG. 3: *Le concept* D *est subsumé par le concept* C.

- *Normalisation* : factorisation des co-domaines.
(and (all r (and C_1 C_2)) (all r (and C_3 C_4))) $\rightarrow$
(and (all r (and C_1 C_2 C_3 C_4))).

- *Comparaison* :
après normalisation, C = (and C_1 $C_2 ... C_n$) et D = (and D_1 $D_2 ... D_m$).
D $\sqsubseteq$ C est vrai si et seulement si, pour chaque C_i, $i = 1, .., n$:
  - [concept primitif] : si C_i est un concept primitif, alors il existe D_j dans l'ensemble $\{D_1, D_2, ..., D_m\}$ tel que $D_j = C_i$ ou $D_j \dot{\leq} C_i$ (éventuellement par transitivité).
  - [some] : si C_i = (some r), alors il existe D_j dans $\{D_1, D_2, ..., D_m\}$ tel que $D_j = C_i$.
  - [all] : si C_i = (all r X), alors il existe D_j dans $\{D_1, D_2, ..., D_m\}$ tel que D_j = (all r Y) avec Y $\sqsubseteq$ X.

Un exemple illustrant cet algorithme est donné à la figure 3. Le concept D, qui dénote « un homme ayant des enfants et dont tous les amis sont des docteurs ayant une spécialité », est subsumé par le concept C, qui dénote « une personne dont tous les amis sont docteurs ». Les deux concepts D = (and D_1 D_2 D_3) et C = (and C_1 C_2) sont sous forme normale, et : $D_1 \sqsubseteq C_1$ car HOMME est subsumé par PERSONNE ; $D_3 \sqsubseteq C_2$ car (and DOCTEUR X) est subsumé par DOCTEUR, où X est une description quelconque (ici, X = (some spécialité)).

Deux remarques peuvent être faites : d'une part, la relation de subsomption (and C D) $\sqsubseteq$ C est toujours vérifiée, quelle que soit la description D, d'autre part, le nombre d'éléments de la conjonction D = (and D_1 $D_2 ... D_m$) n'est pas forcément plus grand que celui de la conjonction C = (and C_1 $C_2 ... C_n$), comme le montre la notion d'équivalence dans la définition 2 : D $\sqsubseteq$ C si et seulement si D $\equiv$ (and D C) ; dans ce cas, (and D C) $\sqsubseteq$ D, mais aussi D $\sqsubseteq$ (and D C).

Dans [Nebel,1990a, page 54], et comme cela a déjà été discuté pour les rôles au paragraphe 2.2, les introductions de concepts primitifs sont transformées en définition de concepts de la façon suivante : l'introduction HOMME $\dot{\leq}$ PERSONNE se transforme en la définition HOMME $\doteq$ (and PERSONNE HOMME-PRIM), où HOMME-PRIM est un concept primitif. En s'appuyant sur cette transformation, si, lors de la normalisation, tous les concepts sont remplacés par leur « définition », la clause `[concept primitif]` de l'algorithme ci-dessus peut se réécrire comme suit : si $\mathtt{C_i}$ est un concept primitif, alors il existe $\mathtt{D_j}$ dans l'ensemble $\{\mathtt{D_1}, \mathtt{D_2}, ..., \mathtt{D_m}\}$ tel que $\mathtt{D_j} = \mathtt{C_i}$.

2.7.2 La subsomption dans $\mathcal{ALNR}$

Ce paragraphe complète le précédent et présente les principales règles de subsomption associées à $\mathcal{ALNR}$. Un algorithme détaillé donnant l'ensemble des règles de normalisation et de comparaison des formes normales pour $\mathcal{ALNR}$ figure dans l'annexe 7.

- `[concept ou rôle primitif]` : si C et D dénotent des concepts ou des rôles primitifs, alors $\mathtt{D} \sqsubseteq \mathtt{C}$ si $\mathtt{D} = \mathtt{C}$ ou $\mathtt{D} \stackrel{.}{\leq} \mathtt{E}$ avec $\mathtt{E} \sqsubseteq \mathtt{C}$ (règle identique à celle de l'algorithme de $\mathcal{FL}$-).

- `[not]` : si $\mathtt{C} = \neg\mathtt{C}'$ et $\mathtt{D} = \neg\mathtt{D}'$ sont des négations de concepts primitifs, alors $\mathtt{D} \sqsubseteq \mathtt{C}$ si $\mathtt{C}' \sqsubseteq \mathtt{D}'$. Ainsi, la négation du concept `DOCTEUR` subsume la négation du concept `BACHELIER`, `BACHELIER` subsumant `DOCTEUR` (figure 4, `(1)`). Les cas où les négations sont *croisées*, qui consisteraient à comparer $\mathtt{C} = \neg\mathtt{C}'$ et D, ou bien C et $\mathtt{D} = \neg\mathtt{D}'$, ne sont pas traités ici.

- `[all]` : si $\mathtt{r_C} \sqsubseteq \mathtt{r_D}$ et $\mathtt{D} \sqsubseteq \mathtt{C}$, alors $(\mathtt{all}\ \mathtt{r_D}\ \mathtt{D}) \sqsubseteq (\mathtt{all}\ \mathtt{r_C}\ \mathtt{C})$.

  Par exemple, le concept « personne dont tous les fils sont bacheliers » subsume le concept « personne dont tous les enfants sont docteurs », où le rôle `fils` est subsumé par le rôle `enfant` (figure 4, `(2)`). De façon duale, le concept « personne dont tous les amis ne sont pas docteurs » subsume le concept « personne dont tous les amis ne sont pas bacheliers » (figure 4, `(3)`).

- `[atleast]` : si $\mathtt{n_C} \leq \mathtt{n_D}$ et $\mathtt{r_D} \sqsubseteq \mathtt{r_C}$, alors $(\mathtt{atleast}\ \mathtt{n_D}\ \mathtt{r_D}) \sqsubseteq (\mathtt{atleast}\ \mathtt{n_C}\ \mathtt{r_C})$.

  Par exemple, le concept « personne qui a au moins un enfant » subsume le concept « personne qui a au moins deux filles », où le rôle `fille` est subsumé par le rôle `enfant` (figure 4, `(4)`).

- `[atmost]` : si $\mathtt{n_D} \leq \mathtt{n_C}$ et $\mathtt{r_C} \sqsubseteq \mathtt{r_D}$, alors $(\mathtt{atmost}\ \mathtt{n_D}\ \mathtt{r_D}) \sqsubseteq (\mathtt{atmost}\ \mathtt{n_C}\ \mathtt{r_C})$.

  Par exemple, le concept « personne qui a au plus deux filles » subsume le concept « personne qui a au plus un enfant » (figure 4, `(5)`).

2.8 La notion de base de connaissances terminologique

2.8.1 Introduction générale et discussion

D'une façon générale, et dans les logiques de descriptions en particulier, la représentation de connaissances s'articule autour de deux niveaux :

- Le *niveau terminologique* où sont introduites les définitions de concepts et de rôles : ce niveau est relatif à l'*intension* des concepts et des rôles.

```
(1) (not BACHELIER) ⊑ (not DOCTEUR)
(2) (all enfant DOCTEUR) ⊑ (all fils BACHELIER)
(3) (all ami (not BACHELIER)) ⊑ (all ami (not DOCTEUR))
(4) (atleast 2 fille) ⊑ (atleast 1 enfant)
(5) (atmost 1 enfant) ⊑ (atmost 2 fille)
```

FIG. 4: *Relations de subsomption en $\mathcal{ALNR}$.*

- Le niveau *factuel* ou niveau des *assertions*, où sont introduits les individus et les faits dans lesquels ces individus interviennent : ce niveau, appelé aussi *ABox*, est relatif à l'extension des concepts.

Au niveau factuel figurent essentiellement des spécifications d'instances et des relations entre instances. Ainsi, deux types d'assertions se rencontrent : les premières sont de la forme `C(a)` où `C` est un nom de concept et `a` un nom d'individu ; les secondes sont de la forme `r(a,b)` où `r` est un nom de rôle, et `a` et `b` celui de deux individus. Intuitivement, l'assertion `C(a)` stipule que `a` dénote une instance du concept `C` (`a` est dans l'extension de `C`) ; l'assertion `r(a,b)` stipule que les individus `a` et `b` sont en relation par l'intermédiaire de `r` (le couple `(a,b)` est dans l'extension de `r`).

2.8.2 Exemples d'instanciations

Le but de ce paragraphe est de montrer le degré de subtilité que peut recouvrir l'*instanciation*, qui est une des formes du raisonnement terminologique. L'instanciation consiste à déterminer qu'un objet `o` est instance d'un concept `C`, ou bien encore que l'assertion `C(o)` est vérifiée.

Un ensemble d'assertions est donné en exemple à la figure 5. La définition d'instances repose sur l'*hypothèse du nom unique* — les noms `Colette`, `Antoine` et `Patrick` font référence à des individus différents — et sur l'*hypothèse du monde ouvert*[8] : une instanciation est partielle et non définitive, et elle peut être complétée le cas échéant. Par exemple, il est suffisant de savoir que `Trio-54` est instance du concept `(atmost 3 membre)`, sans avoir à nommer explicitement les membres de `Trio-54`, du moins, tant que cette donnée n'est pas indispensable.

Dans la pratique, plusieurs cas d'instanciation se présentent, dont les plus courants sont expliqués ci-dessous (voir aussi [Nebel,1990a] et comparer avec les règles d'inférence données dans [Ducournau,1996a] et [Ducournau,1996a]) :

- [$\mathtt{I_1}$] : si l'assertion `D(o)` est connue et si `D` est subsumé par `C`, alors l'assertion `C(o)` en découle. Par exemple, `Trio-54`, qui est une instance du concept `ÉQUIPE-MODERNE`, est aussi une instance de `PETITE-ÉQUIPE`, qui lui-même subsume `ÉQUIPE-MODERNE` (voir figure 1).

8. Un fait n'est faux que si cela a été effectivement établi. L'hypothèse du monde ouvert s'oppose à l'*hypothèse du monde clos*, qui stipule que tout fait non déclaré, donc non connu, est considéré comme faux.

```
ÉQUIPE-MODERNE(Trio-54)
HOMME(Antoine)
PERSONNE(Colette)
membre(Trio-54,Antoine)
membre(Trio-54,Patrick)
chef(Trio-54,Colette)
(atmost 3 membre)(Trio-54)
```

FIG. 5: *Un ensemble d'assertions. La dernière assertion signifie en particulier que* `Trio-54` *est un instance de l'expression conceptuelle* `(atmost 3 membre)`.

- $[\mathtt{I}_2]$: si les assertions $\mathtt{D}_1(\mathtt{o})$ et $\mathtt{D}_2(\mathtt{o})$ sont déclarées, et si `C` ne subsume ni $\mathtt{D}_1$ ni $\mathtt{D}_2$, mais subsume (`and` $\mathtt{D}_1$ $\mathtt{D}_2$), alors l'assertion `C(o)` en découle. Par exemple, supposons que $\mathtt{D}_1$ = `DOCTEUR`, $\mathtt{D}_2$ = `MUSICIEN`, `C` = `(and BACHELIER ARTISTE)`, et que `Colette` est une instance de $\mathtt{D}_1$ et de $\mathtt{D}_2$. Dans ce cas, `C` ne subsume ni $\mathtt{D}_1$ ni $\mathtt{D}_2$, mais `C` subsume (`and` $\mathtt{D}_1$ $\mathtt{D}_2$), et `Colette` est une instance de `C`.

- $[\mathtt{I}_3]$: il est possible de profiter du fait qu'un objet $\mathtt{o}_1$ est en relation avec un objet $\mathtt{o}_2$ pour déterminer une instanciation. Ainsi, alors que ce n'est pas explicitement spécifié, `Patrick` et `Antoine`, qui sont membres de `Trio-54`, sont nécessairement des instances de `PERSONNE`, car tous les membres d'une équipe sont des personnes. Cette instanciation repose sur le principe suivant : si une assertion de la forme $\mathtt{r}(\mathtt{o}_1, \mathtt{o}_2)$ est connue, et si $\mathtt{o}_2$ est une instance d'un concept de la forme `(all r C)` ou encore `(all r D)` avec `D` subsumé de `C`, alors $\mathtt{C}(\mathtt{o}_1)$ en découle. Par exemple, `Colette`, qui est déclarée comme `chef` de `Trio-54`, est forcément une instance du concept `FEMME`. Ici, $\mathtt{o}_1$ correspond à `Colette`, $\mathtt{o}_2$ à `Trio-54` et `r` à `chef`. Puisque `Trio-54` est instance de l'expression `(all chef FEMME)`, qui subsume le concept `ÉQUIPE-MODERNE`, l'instanciation `FEMME(Colette)` en découle.

Un traitement systématique de l'instanciation est donné au paragraphe 3, où est étudiée une série de règles qui traitent toutes les constructions possibles de $\mathcal{ALCNR}$.

2.8.3 Base terminologique et modèle

En l'absence de circuit terminologique, il est possible de normaliser les descriptions de façon à ne plus avoir que des concepts primitifs dans une description. Tout se passe alors comme s'il n'existait aucun concept défini : dans ce cas, les assertions peuvent devenir les ingrédients principaux d'une base de connaissances et du raisonnement terminologique.

Pour associer une sémantique aux assertions, la fonction d'interprétation $.^{\mathcal{I}}$ est prolongée aux individus, en faisant correspondre à un individu un élément du domaine d'interprétation $\Delta_{\mathcal{I}}$ — $\mathsf{a}^{\mathcal{I}} \in \Delta_{\mathcal{I}}$ — de telle façon que si $a \neq b$ alors

$\mathtt{a}^{\mathcal{I}} \neq \mathtt{b}^{\mathcal{I}}$; deux individus différents sont interprétés comme des objets différents de $\Delta_{\mathcal{I}}$.

Définition 4 ***(Satisfiabilité d'une assertion, base terminologique et modèle)***

- *Si* $\mathcal{I} = (\Delta_{\mathcal{I}}, .^{\mathcal{I}})$ *est une interprétation, l'assertion* $\mathtt{C(a)}$ *est* satisfaite *par* $\mathcal{I}$, *ou* satisfiable, *si et seulement si* $\mathtt{a}^{\mathcal{I}} \in \mathtt{C}^{\mathcal{I}}$ *; l'assertion* $\mathtt{r(a,b)}$ *est* satisfaite *par* $\mathcal{I}$, *ou* satisfiable, *si et seulement si* $(\mathtt{a}^{\mathcal{I}}, \mathtt{b}^{\mathcal{I}}) \in \mathtt{r}^{\mathcal{I}}$.
- *Une* base (de connaissances) terminologique Σ *pour* $\mathcal{ALCNR}$ *est un ensemble d'assertions de la forme* $\mathtt{C(a)}$ *ou* $\mathtt{r(a,b)}$.
- *Une interprétation* $\mathcal{I}$ *est un* modèle *pour* Σ *si et seulement si toute assertion* α *est satisfaite par* $\mathcal{I}$.
- *Une base terminologique est* satisfiable *si elle admet un modèle.*
- *L'assertion* α *est* logiquement impliquée *par la base* Σ, *ce qui s'écrit* $\Sigma \models \alpha$, *si et seulement si* α *est satisfaite par tout modèle de* Σ.

Nous utiliserons également dans la suite l'expression *système terminologique* pour désigner un système combinant un niveau terminologique et un niveau factuel, avec un langage de description de concepts et de rôles donné. Considérons par exemple la base terminologique Σ suivante :

```
Σ = {enfant(Pierre,Marie)
    (all enfant (not MUSICIEN))(Pierre)
    (and FEMME (some enfant))(Marie)}
```

Intuitivement, les trois assertions indiquent que `Marie` est l'enfant de `Pierre`, que tous les enfants de `Pierre` ne sont pas musiciens et que `Marie` est une femme ayant un enfant. La base Σ est satisfiable, puisqu'il existe une interprétation $\mathcal{I}$ donnée ci-dessous, qui satisfait toutes les assertions figurant dans Σ et qui est donc un modèle pour Σ :

$\Delta_{\mathcal{I}} = \{\mathtt{Pierre,Marie}\}$
$\mathtt{Pierre}^{\mathcal{I}} = \mathtt{Pierre}$
$\mathtt{Marie}^{\mathcal{I}} = \mathtt{Marie}$
$\mathtt{enfant}^{\mathcal{I}} = \{\mathtt{(Pierre,Marie)}\}$
$\mathtt{FEMME}^{\mathcal{I}} = \{\mathtt{Marie}\}$
$\mathtt{MUSICIEN}^{\mathcal{I}} = \emptyset$

L'adjonction d'une assertion comme `MUSICIEN(Marie)` rendrait la base non satisfiable, car $\Sigma \models$ `(not MUSICIEN)(Marie)` (il est possible de montrer ce résultat en employant la règle $\mathtt{I}_3$ explicitée au paragraphe précédent).

2.8.4 Variantes sur la notion de base terminologique

Comme nous l'avons déjà vu, les circuits terminologiques ne sont pas admis en général dans un système terminologique. Lors des introductions de concepts primitifs ($\mathtt{A} \leq \mathtt{B}$) ou de concepts définis ($\mathtt{A} \doteq \mathtt{B}$), tout nom de concept doit d'abord avoir été introduit en partie gauche avant de pouvoir figurer en partie droite d'une introduction. Dans [Buchheit *et al.*,1993] et [De Giacomo and Lenzerini,1997], cette restriction n'est plus obligatoire et les introductions de concepts n'ont qu'une seule forme, à savoir $\mathtt{A} \sqsubseteq \mathtt{B}$, où A et B sont des expressions quelconques pouvant faire intervenir des circuits terminologiques ; une équation comme $\mathtt{A} \sqsubseteq \mathtt{B}$ s'appelle aussi une *inclusion* de concepts. Dans ce cas, une définition $\mathtt{A} \doteq \mathtt{B}$ se ramène à deux inclusions, $\mathtt{A} \sqsubseteq \mathtt{B}$ et $\mathtt{B} \sqsubseteq \mathtt{A}$.

Une base de connaissances se compose alors effectivement d'un niveau terminologique et d'un niveau factuel : une *base terminologique* pour $\mathcal{ALCNR}$ est un couple $\Sigma = (\mathcal{T}, \mathcal{A})$, où $\mathcal{T}$ (pour *TBox*) est un ensemble d'inclusions de concepts et $\mathcal{A}$ (pour *ABox*) un ensemble d'assertions. Une interprétation $\mathcal{I}$ est un *modèle* pour $\Sigma = (\mathcal{T}, \mathcal{A})$ si elle est à la fois un modèle pour $\mathcal{T}$ et pour $\mathcal{A}$. La notion de modèle pour un ensemble d'inclusions de concepts est du même ordre que la notion de modèle pour une base terminologique : une interprétation $\mathcal{I}$ *satisfait* l'inclusion $\mathtt{D} \sqsubseteq \mathtt{C}$ si $\mathtt{D}^{\mathcal{I}} \subseteq \mathtt{C}^{\mathcal{I}}$; l'interprétation $\mathcal{I}$ est un *modèle* pour $\mathcal{T}$ si $\mathcal{I}$ satisfait toutes les inclusions de $\mathcal{T}$.

Considérons par exemple la base terminologique suivante (tirée de [Buchheit *et al.*,1993]) :

```
(t1) 𝒯 = {(some enseignant COURS) ⊑
              (or PROFESSEUR (and ÉTUDIANT (some diplôme DEA))),
(t2)          PROFESSEUR ⊑ (some diplôme DOCTORAT),
(t3)          (some diplôme DOCTORAT) ⊑ (some diplôme DEA),
(t4)          (and DOCTORAT DEA) ⊑ ⊥}

(a1) 𝒜 = {enseignant(Jean, Cours-ia),
(a2)        (≤1 diplôme)(Jean),
(a3)        COURS(Cours-ia)}
```

Cette base terminologique est composée de quatre inclusions de concepts dans $\mathcal{T}$ et de trois assertions dans $\mathcal{A}$. Intuitivement, les quatre inclusions de concepts dans $\mathcal{T}$ signifient :

(t1) quelqu'un enseignant un cours est soit un professeur soit un étudiant ayant comme diplôme un DEA,

(t2) un professeur est quelqu'un ayant comme diplôme un doctorat,

(t3) quelqu'un ayant comme diplôme un doctorat a forcément comme diplôme un DEA,

(t4) les diplômes de doctorat et de DEA sont différents ; s'il existe une personne P ayant comme diplôme un doctorat, alors il doit exister une instance de DEA qui est associée à P *via* le rôle diplôme.

La base terminologique $\Sigma = (\mathcal{T},\mathcal{A})$ admet un modèle $\mathcal{I}$ qui satisfait les inclusions de concepts de $\mathcal{T}$ et les assertions de $\mathcal{A}$ (en particulier, puisque Jean n'a qu'un diplôme au plus, il n'est donc pas professeur, et puisqu'il est enseignant, il est donc forcément un étudiant muni d'un DEA) :

$\Delta_{\mathcal{I}}$ = {Jean,Cours-ia,Dea-de-Jean},
$\texttt{Jean}^{\mathcal{I}}$ = Jean,
$\texttt{Cours-ia}^{\mathcal{I}}$ = Cours-ia,
$\texttt{ÉTUDIANT}^{\mathcal{I}}$ = {Jean},
$\texttt{PROFESSEUR}^{\mathcal{I}}$ = $\emptyset$,
$\texttt{COURS}^{\mathcal{I}}$ = {Cours-ia},
$\texttt{DEA}^{\mathcal{I}}$ = {Dea-de-Jean},
$\texttt{DOCTORAT}^{\mathcal{I}}$ = $\emptyset$,
$\texttt{enseignant}^{\mathcal{I}}$ = {(Jean,Cours-ia)},
$\texttt{diplôme}^{\mathcal{I}}$ = {(Jean,Dea-de-Jean)}

Seules les introductions de rôles primitifs ont été évoquées jusqu'à présent, mais il existe aussi des variantes sur ce point. Une première variante est donnée dans [Buchheit *et al.*,1994b], où sont autorisées des inclusions de rôles de la forme $\texttt{P} \sqsubseteq \texttt{A}_1 \times \texttt{A}_2$, où P est un nom d'*attribut* — un attribut est un rôle *mono-valué*, à l'image d'une fonction —, $\texttt{A}_1$ et $\texttt{A}_2$ sont deux noms de concepts primitifs. L'inclusion $\texttt{P} \sqsubseteq \texttt{A}_1 \times \texttt{A}_2$ signifie que l'attribut P a pour domaine $\texttt{A}_1$ et pour co-domaine $\texttt{A}_2$. Notons que dans certains systèmes comme LOOM (cf. § 4.3) ou BACK , les contraintes sur le domaines et le co-domaine d'un rôle sont données explicitement par les constructeurs `domain` et `range`. Une inclusion comme $\texttt{P} \sqsubseteq \texttt{A}_1 \times \texttt{A}_2$ est alors représentée par $\texttt{P} \doteq$ (and (domain $\texttt{A}_1$) (range $\texttt{A}_2$)), qui correspond à une définition de rôle.

Une autre variante, donnée dans [Buchheit *et al.*,1993], montre qu'il est possible de simuler certaines connaissances sur le domaine et le co-domaine d'un rôle par l'intermédiaire d'inclusions de concepts. Par exemple, restreindre le domaine d'un rôle r à un concept C et son co-domaine à un concept D peut s'écrire : $\exists\texttt{r}.\top \sqsubseteq \texttt{C}$ et $\top \sqsubseteq \forall\texttt{r}.\texttt{D}$. Une interprétation qui satisfait ces deux inclusions vérifie $\texttt{r}^{\mathcal{I}} \subseteq \texttt{C}^{\mathcal{I}} \times \texttt{D}^{\mathcal{I}}$. Toutefois, ces constructions alternatives ne remplacent pas complètement les déclarations explicites du domaine et du co-domaine d'un rôle.

2.9 Un bilan intermédiaire sur le raisonnement terminologique

2.9.1 Les opérations associées au raisonnement terminologique

Les principales opérations liées au raisonnement terminologique — ou raisonnement dans un système terminologique — sont les suivantes [Buchheit *et al.*,1993] [Donini *et al.*,1994] :

- Le *test de subsomption* permet de vérifier qu'un concept C subsume un concept D. Le test de subsomption est à la base de l'opération de *classification*, qui consiste à déterminer l'ensemble des ascendants directs d'un concept dans la hiérarchie des concepts, les subsumants les plus spécifiques ou SPS, et l'ensemble des descendants immédiats du concept, les subsumés les plus généraux ou SPG (voir § 4.1).

Vérifier qu'un concept C vérifie une propriété prop se ramène également à un test de subsomption : C vérifie la propriété prop si un des subsumants de C vérifie prop.

- Le *test de satisfiabilité d'un concept* C permet de vérifier qu'un concept C admet des instances (il existe au moins une interprétation $\mathcal{I}$ telle que $\mathtt{C}^{\mathcal{I}} \neq \emptyset$).
- Le *test de satisfiabilité d'une base terminologique* Σ permet de vérifier que Σ admet un modèle $\mathcal{I}$: tout concept et toute assertion sont satisfaits par $\mathcal{I}$.
- Le *test d'instanciation* permet de vérifier qu'un individu a est instance d'un concept C pour une base terminologique Σ : $\Sigma \models \mathtt{C(a)}$. Plus précisément, le test d'instanciation consiste à retrouver les concepts les plus spécifiques dont un individu a est instance.

Les quatre opérations de tests ne sont pas indépendantes les unes des autres :

(1) C est satisfiable $\Longleftrightarrow$ $\mathtt{C} \not\sqsubseteq \bot$,
(2) C est satisfiable $\Longleftrightarrow$ {C(a)} est satisfiable,
(3) $\mathtt{D} \sqsubseteq \mathtt{C} \Longleftrightarrow \mathtt{D} \sqcap \neg\mathtt{C}$ n'est pas satisfiable,
(4) $\mathtt{D} \sqsubseteq \mathtt{C} \Longleftrightarrow \{\mathtt{C(a)}\} \models \mathtt{D(a)}$,
(5) Σ est satisfiable $\Longleftrightarrow \Sigma \not\models \bot(\mathtt{a})$,
(6) $\Sigma \models \mathtt{C(a)} \Longleftrightarrow \Sigma \cup \{\neg\mathtt{C(a)}\}$ non satisfiable.

Le test de satisfiabilité d'un concept se ramène au problème complémentaire du test de subsomption (1) ou à un test d'instanciation (2). Le test de subsomption se ramène au test de satisfiabilité d'une conjonction de concepts (3) — qui suppose que le langage de description des concepts inclut la négation de concepts définis — ou à un test d'instanciation (4). Le test de satisfiabilité d'une base terminologique se ramène au problème complémentaire d'un test d'instanciation (5). Et finalement, le test d'instanciation se ramène au problème complémentaire du test de satisfiabilité d'une base terminologique (6).

Il est important de remarquer que les équivalences données ci-dessus ne sont plus valides dès que le langage de description des concepts et des rôles fait intervenir directement des instances dans la définition des concepts. Ce n'est pas le cas pour $\mathcal{ALCNR}$, mais ça l'est pour CLASSIC ou LOOM , où sont admis des constructeurs comme fills qui attache une valeur à un rôle, one-of qui permet de construire des domaines de valeurs par énumération, ou encore same-as qui indique qu'un rôle a les mêmes valeurs élémentaires qu'un autre rôle.

2.9.2 La complexité du raisonnement terminologique

Depuis le milieu des années 80 jusqu'à aujourd'hui, la complexité du raisonnement terminologique et plus spécialement la complexité de la subsomption[9]

9. Il faudrait dire plus précisément : « la complexité des algorithmes détectant les relations de subsomption entre descriptions ».

ont fait couler beaucoup d'encre et ont été des moteurs de la recherche sur les logiques de descriptions. Ces premiers travaux s'intéressent aux propriétés de correction, de complétude, et à la complexité algorithmique de la subsomption, en tant que procédure d'inférence d'un système terminologique considéré comme un système déductif.

Rappelons qu'un système déductif est *correct* si les inférences produites sont en accord avec la sémantique associée au système, autrement dit, ce qui est vrai sur le plan syntaxique l'est sur le plan sémantique ; un système déductif est *complet* si toutes les formules valides — vraies sur le plan sémantique — peuvent être démontrées sur le plan syntaxique [Loveland,1978].

Considérons à présent les concepts `C` pour « personne ayant au moins deux enfants » et `D` pour « personne ayant au moins une fille et un fils », qui se représentent en $\mathcal{ALCNR}$ par :

```
C ≐ (and PERSONNE
         (atleast 2 enfant))

D ≐ (and PERSONNE
         (atleast 1 fille)
         (atleast 1 fils))
```

Si le bon sens permet d'affirmer que toute instance de `D` est une instance de `C`, autrement dit que `C` subsume `D`, les règles algorithmiques données au paragraphe 2.7 ou dans l'annexe 7 ne permettent pas de détecter une telle relation de subsomption. Un système terminologique muni d'une procédure de subsomption s'appuyant sur un tel algorithme est alors *incomplet*, ce qui pose le problème suivant : si la procédure de subsomption retourne `vrai`, alors le résultat est exploitable, tandis qu'il ne l'est plus si la procédure retourne `faux` (une relation de subsomption peut être vraie et ne pas être détectée par la procédure).

Le problème de la non complétude de la subsomption a été soulevé une première fois pour KL-ONE dans [Schmolze and Lipkis,1983]. Ces travaux ont été repris et formalisés dans plusieurs articles dont [Brachman and Levesque,1984] [Levesque and Brachman,1987] [Nebel,1988] et [Nebel,1990a]. Il est montré dans ces articles que le test de subsomption est correct, complet et de complexité polynômiale pour $\mathcal{FL}$- $= \{\mathtt{C} \sqcap \mathtt{D}, \forall \mathtt{r.C}, \exists\, \mathtt{r}\}$ et $\mathcal{FLN} = \{\mathtt{C} \sqcap \mathtt{D}, \forall \mathtt{r.C}, \geq \mathtt{n\,r}, \leq \mathtt{n\,r}\}$; le test de subsomption devient de complexité exponentielle et les algorithmes (polynômiaux) proposés sont incomplets dès que des constructeurs portant sur les rôles comme `and` ou `restrict` sont ajoutés à $\mathcal{FL}$- ou à $\mathcal{FLN}$.

Au début des années 90, de nombreux articles traitent du classement des problèmes liés au raisonnement terminologique : subsomption, instanciation, test de satisfiabilité d'un concept ou d'une base terminologique. Des tables recensant ces résultats sont données dans [Woods and Schmolze,1992] [Heinsohn *et al.*,1994], [Donini *et al.*,1994] et [Donini *et al.*,1997]. Brièvement, le raisonnement terminologique est de complexité polynômiale pour des langages comme $\mathcal{AL}$, $\mathcal{ALN}$ et $\mathcal{FL}$- (voir aussi dans [Patel-Schneider,1989a] une étude donnée dans le cadre d'une logique à 4 valeurs) ; il l'est encore pour des langages comme $\mathcal{PL}_1 = \{\top, \bot, \neg\mathtt{A}, \mathtt{C} \sqcap \mathtt{D}, \forall \mathtt{r.C}, \geq \mathtt{n\,r}, \leq \mathtt{n\,r}, \mathtt{r}^{-1}\}$, où `A` désigne un concept primitif et $\mathtt{r}^{-1}$ dénote la relation inverse pour un rôle, et comme $\mathcal{PL}_2$

$= \{\mathtt{C} \sqcap \mathtt{D}, \forall \mathtt{r.C}, \exists\, \mathtt{r}, \mathtt{r}_1 \sqcap \mathtt{r}_2, \mathtt{r}^{-1}, \mathtt{r}_1 \circ \mathtt{r}_2\}$, où $\mathtt{r}_1 \circ \mathtt{r}_2$ dénote la composition de rôles [Donini *et al.*,1991b]. Les ensembles de constructeurs associés à $\mathcal{PL}_1$ et $\mathcal{PL}_2$ constituent des ensembles « maximaux » à ne pas étendre au risque de passer à une complexité exponentielle. En dehors de ces cas simples, les problèmes liés au raisonnement terminologique sont de complexité exponentielle [10], selon les cas NP-difficiles, co-NP-difficiles, NP-complets, ou dans la classe PSPACE [Hollunder *et al.*,1990] [Donini *et al.*,1991a] [Schmidt-Schauß and Smolka,1991] [Donini *et al.*,1997], voire indécidables [Schmidt-Schauß,1989] [Patel-Schneider,1989b] [Nebel,1990b].

Les travaux récents sur la subsomption, comme [Buchheit *et al.*,1993], [Donini *et al.*,1994] et [Donini *et al.*,1997], cherchent à exploiter la méthode des tableaux sémantiques pour mettre en valeur des algorithmes complets, comme cela est expliqué dans le paragraphe suivant. Outre leur intérêt théorique, tous les travaux sur la complexité de la subsomption ont permis de mettre en évidence les rapports qui existent entre la complexité du raisonnement terminologique et la richesse — en termes de nombre et de types de constructeurs — des langages de description de concepts et de rôles. Face à ces résultats, deux attitudes ont guidé les concepteurs de logiques de descriptions : avoir un langage de description plutôt pauvre mais une procédure d'inférence correcte, complète et de complexité polynômiale comme pour CLASSIC [Borgida and Patel-Schneider,1994], ou au contraire avoir un langage de description riche et une procédure d'inférence de complexité exponentielle, complète comme pour KRIS [Baader and Hollunder,1991a], ou non complète comme pour BACK [Peltason,1991] et LOOM [MacGregor,1991a]. Les deux attitudes se justifient et nous renvoyons le lecteur intéressé par ce débat aux articles déjà cités, mais aussi à [Doyle and Patil,1991].

3 La méthode des tableaux sémantiques dans les logiques de descriptions

Dans ce paragraphe, nous montrons comment la méthode des *tableaux sémantiques* peut être appliquée dans le cadre des logiques de descriptions. En vertu des correspondances exhibées au paragraphe 2.9.1, les quatre opérations de test se ramènent à l'une d'entre elles, en l'occurrence le test de satisfiabilité d'une base terminologique. La vérification de ce test de satisfiabilité s'appuie sur la méthode des tableaux sémantiques, dont les principes sont rappelés ci-dessous.

3.1 Brefs rappels sur la méthode des tableaux sémantiques

Ce bref rappel sur la méthode des tableaux sémantiques en logique classique est tiré de [Gochet and Gribomont,1991] (voir aussi [Fitting,1990]). Globalement, la méthode des tableaux sémantiques est une méthode de réfutation analytique : pour prouver une formule φ, il est supposé que φ est fausse et les conséquences de cette hypothèse sur les constituants de φ sont examinées. La

10. Voir [Garey and Johnson,1979] ou [Barthélémy *et al.*,1992] pour les définitions sur la complexité algorithmique.

méthode s'appuie sur un ensemble de *règles de décomposition* qui permettent d'assigner une valeur de vérité aux constituants de la formule φ, une fois fixée la valeur de vérité de la formule φ elle-même. Il existe deux règles de décomposition par connecteur logique, en l'occurrence pour $\neg$, $\vee$, $\wedge$, $\longrightarrow$, $\longleftrightarrow$. Par exemple, les règles associées au connecteur $\longrightarrow$ sont :

(R1) si φ contient « $\mathtt{a} \longrightarrow \mathtt{b} = \mathtt{faux}$ », alors φ contient « $\mathtt{a} = \mathtt{vrai}$ » et « $\mathtt{b} = \mathtt{faux}$ »,

(R2) si φ contient « $\mathtt{a} \longrightarrow \mathtt{b} = \mathtt{vrai}$ », alors φ contient soit « $\mathtt{a} = \mathtt{faux}$ », soit « $\mathtt{b} = \mathtt{vrai}$ », soit « $\mathtt{a} = \mathtt{faux}, \mathtt{b} = \mathtt{vrai}$ ».

Les règles de décomposition se divisent en *règles de prolongation* et *règles de ramification*, selon que le résultat est *déterministe* (R1) ou *indéterministe* (R2).

Il est possible de donner à une démonstration la forme d'un arbre fini étiqueté, qui est alors appelé *tableau sémantique*, où chaque branche du tableau mémorise une série d'évaluations possibles pour les énoncés testés ; les branches se divisent sous l'effet des règles de ramification et se prolongent grâce aux règles de prolongation. Dès qu'apparaît sur une même branche deux évaluations mutuellement contradictoires — c'est le cas par exemple lorsqu'une évaluation contient les deux équations $\{\mathtt{a} = \mathtt{faux}, \mathtt{a} = \mathtt{vrai}\}$ — la tentative d'évaluation à vrai échoue : la branche du tableau *fermée* et le nœud extrémité est un nœud *échec*. Une branche est *complète* si, pour chaque prolongation les deux constituants se trouvent sur la branche, et si, pour chaque ramification, l'un au moins des deux constituants se trouve sur la branche. Le développement d'une branche est arrêté lorsqu'il n'y a plus de formule à développer ou lorsqu'apparaît une contradiction. Un tableau dont chaque branche est complète ou fermée est un tableau *complet*. Une preuve d'une formule φ par la méthode des tableaux sémantiques est un tableau sémantique fermé pour $\neg\varphi$. Parallèlement, une formule ψ est satisfiable si et seulement si le tableau complet qui lui est associé est *ouvert* (une branche au moins n'est pas fermée).

Un ensemble de formules $\{\mathtt{F}_1, \mathtt{F}_2, ..., \mathtt{F}_\mathtt{n}\}$ est satisfiable s'il existe au moins une évaluation qui satisfait simultanément toutes les formules de l'ensemble (et cette évaluation apparaît lors de la construction du tableau associé). L'ensemble $\{\mathtt{F}_1, \mathtt{F}_2, ..., \mathtt{F}_\mathtt{n}\}$ est non satisfiable ou contradictoire s'il n'existe pas une telle évaluation. Enfin, pour montrer qu'un ensemble fini de formules $\{\mathtt{F}_1, \mathtt{F}_2, ..., \mathtt{F}_\mathtt{n}\}$ a pour conséquence logique une formule φ, il faut montrer que $\{\mathtt{F}_1, \mathtt{F}_2, ..., \mathtt{F}_\mathtt{n}, \neg\varphi\}$ n'est pas satisfiable, ou encore que le tableau qui lui est associé est fermé.

3.2 La satisfiabilité d'une base terminologique

Appliquer la méthode des tableaux sémantiques pour satisfaire une base terminologique Σ se fait en trois étapes principales :

- transformation de Σ en un *système de contraintes* $\mathtt{S}_\Sigma$,
- application systématique d'un ensemble de règles de décomposition jusqu'à *saturation* du système de contraintes ; plus aucune règle de décomposition n'est alors applicable et le système de contraintes est dit *complet*,
- évaluation : si le système complet obtenu ne contient pas de motif de contradiction, il est satisfiable ou non contradictoire.

Ci-dessous, nous explicitons une par une les trois étapes de la procédure de satisfaction d'une base terminologique.

3.2.1 Systèmes de contraintes

Avant toute chose, les concepts de la base terminologique considérée Σ sont mis sous *forme simple* [Schmidt-Schauß and Smolka,1991] [Donini *et al.*,1997] : les seules négations dans une description sont de la forme $\neg\mathtt{A}$, où $\mathtt{A}$ désigne un concept primitif. C'est après cette première transformation qu'est construit le système de contraintes $\mathtt{S}_\Sigma$ associé à Σ. Les règles du processus de normalisation sont détaillées ci-après :

$$\begin{aligned}
\neg\top &\longrightarrow \bot\\
\neg\bot &\longrightarrow \top\\
\neg(\mathtt{C}\sqcap\mathtt{D}) &\longrightarrow \neg\mathtt{C}\sqcup\neg\mathtt{D}\\
\neg(\mathtt{C}\sqcup\mathtt{D}) &\longrightarrow \neg\mathtt{C}\sqcap\neg\mathtt{D}\\
\neg\neg\mathtt{C} &\longrightarrow \mathtt{C}\\
\neg(\forall\mathtt{r}.\mathtt{C}) &\longrightarrow \exists\mathtt{r}.\neg\mathtt{C}\\
\neg(\exists\mathtt{r}.\mathtt{C}) &\longrightarrow \forall\mathtt{r}.\neg\mathtt{C}\\
\neg(\geq\mathtt{n}\,\mathtt{r}) &\longrightarrow \forall\mathtt{r}.\bot\ \textit{si}\ \mathtt{n}=1\\
\neg(\geq\mathtt{n}\,\mathtt{r}) &\longrightarrow (\leq\mathtt{n}-1\,\mathtt{r})\ \textit{si}\ \mathtt{n}>1\\
\neg(\leq\mathtt{n}\,\mathtt{r}) &\longrightarrow (\geq\mathtt{n}+1\,\mathtt{r})
\end{aligned}$$

La transformation d'une base terminologique Σ en un système de contraintes $\mathtt{S}_\Sigma$ nécessite l'utilisation de *variables* qui permettent de définir les *contraintes*. Une contrainte peut être de trois formes — $\mathtt{x}:\mathtt{C}$, $\mathtt{xry}$, $\mathtt{x}\neq\mathtt{y}$ — où $\mathtt{C}$ dénote un concept, $\mathtt{r}$ un rôle, $\mathtt{x}$ et $\mathtt{y}$ des instances ou des variables. Intuitivement, la contrainte $\mathtt{x}:\mathtt{C}$ exprime une instanciation, en l'occurrence que $\mathtt{x}$ est une instance de $\mathtt{C}$. La contrainte $\mathtt{xry}$ exprime que $\mathtt{x}$ est en relation avec $\mathtt{y}$ par l'intermédiaire du rôle $\mathtt{r}$; $\mathtt{y}$ est alors appelé un *r-successeur* de $\mathtt{x}$. La contrainte $\mathtt{x}\neq\mathtt{y}$ exprime que les individus $\mathtt{x}$ et $\mathtt{y}$ ont forcément une interprétation différente ; $\mathtt{x}$ et $\mathtt{y}$ sont alors dits *séparés*.

Un *système de contraintes* est un ensemble fini non vide de contraintes des trois formes $\mathtt{x}:\mathtt{C}$, $\mathtt{xry}$ et $\mathtt{x}\neq\mathtt{y}$. Les variables intervenant dans un système de contraintes sont supposées être ordonnées en fonction de leur ordre d'apparition dans le système [Buchheit *et al.*,1993]. De plus, deux variables $\mathtt{x}$ et $\mathtt{y}$ sont dites *équivalentes* dans $\mathtt{S}_\Sigma$ si elles vérifient les mêmes instanciations : $\{\mathtt{C}/\mathtt{x}:\mathtt{C}\in\mathtt{S}_\Sigma\}$ $=\{\mathtt{C}/\mathtt{y}:\mathtt{C}\in\mathtt{S}_\Sigma\}$.

Sur le plan pratique, une assertion $\mathtt{C}(\mathtt{a})$ est transformée en une contrainte $\mathtt{a}:\mathtt{C}$, et une assertion $\mathtt{r}(\mathtt{a},\mathtt{b})$ est transformée en une contrainte $\mathtt{arb}$ ou $\{\mathtt{ar_1b},\mathtt{ar_2b}\}$ selon que $\mathtt{r}$ est atomique ou que $\mathtt{r}=\mathtt{r_1}\sqcap\mathtt{r_2}$. Étant donné une base terminologique Σ, un premier système de contraintes $\mathtt{S}_\Sigma$ est construit en transformant les assertions comme indiqué ci-avant. Ensuite, de nouvelles contraintes — et donc de nouveaux systèmes de contraintes — sont produites par l'application de règles de décomposition aux contraintes existantes.

3.2.2 Les règles de décomposition

Les six règles de décomposition qui sont utilisées dans le cadre de $\mathcal{ALCNR}$ sont présentées à la figure 6 [Buchheit *et al.*,1993] [Donini *et al.*,1997]. Le système $\mathtt{S}_\Sigma$ y est noté plus simplement $\mathtt{S}$, et pour chaque règle sont décrites les conditions d'application et les conséquences de l'application de la règle sur le système de contraintes $\mathtt{S}$.

Les règles $\rightarrow_\exists$ et $\rightarrow_\geq$ sont dites *génératrices*, car elles introduisent de nouvelles variables dans le système de contraintes. En particulier, la règle $\rightarrow_\geq$ indique que, si le nombre minimal de r-successeurs de $\mathtt{x}$ n'est pas atteint, alors il peut être augmenté.

Les règles $\rightarrow_\sqcup$ et $\rightarrow_\leq$ sont, quant à elles, *non déterministes*, car elles peuvent être appliquées à un système de contraintes de façons différentes [11]. Dans la règle $\rightarrow_\leq$, $\mathtt{S[y/z]}$ dénote le système de contraintes obtenu à partir de $\mathtt{S}$ en substituant chaque occurrence de la variable $\mathtt{y}$ par la variable $\mathtt{z}$. Cette règle indique que certains des r-successeurs de $\mathtt{x}$, à condition qu'ils ne soient pas séparés, peuvent être *identifiés* pour que la contrainte sur la cardinalité maximale du rôle $\mathtt{r}$ soit respectée.

Les règles de décomposition sont appliquées jusqu'à ce que le système de contraintes $\mathtt{S}_\Sigma$ devienne complet et plus aucune règle n'est alors applicable. Deux cas peuvent se produire : soit le système est contradictoire et la base terminologique Σ associée n'est pas satisfiable, soit le système est non contradictoire ; la base Σ est alors satisfiable et un modèle de Σ peut être construit. La définition suivante précise les motifs de contradiction.

Définition 5 ***(Système de contraintes contradictoire et contradictions)***
Un système de contraintes est contradictoire *s'il renferme un des motifs de contradiction suivant :*

(1) $\{\mathtt{x} : \bot\}$.

(2) $\{\mathtt{x} : \mathtt{A}, \mathtt{x} : \neg\mathtt{A}\}$.

(3) $\{\mathtt{x} : (\leq \mathtt{m}\,\mathtt{r})\}$ *et le nombre de r-successeurs de* $\mathtt{x}$ *est strictement plus grand que* $\mathtt{m}$.

(4) $\{\mathtt{x} : (\geq \mathtt{m}\,\mathtt{r}), \mathtt{x} : (\leq \mathtt{n}\,\mathtt{r})\}$, *où* $\mathtt{n} < \mathtt{m}$.

De façon pratique, une base terminologique Σ est satisfiable si et seulement s'il existe un système de contraintes complet et non contradictoire $\mathtt{S}_\Sigma$ engendré à partir de Σ. Le problème est alors décidable dans le cadre de $\mathcal{ALCNR}$ et le résultat théorique est démontré dans [Schmidt-Schauß and Smolka,1991], [Buchheit *et al.*,1993] et [Donini *et al.*,1997]). Lorsque le système complet $\mathtt{S}_\Sigma$ est non contradictoire, il est possible de construire un modèle de Σ à partir de $\mathtt{S}_\Sigma$.

11. Ces règles correspondent aux règles de *ramification* des tableaux sémantiques en logique classique, par opposition aux règles de *prolongation* [Gochet and Gribomont,1991].

- Conjonction de concepts :
  $\mathtt{S} \rightarrow_{\sqcap} \mathtt{S} \cup \{\mathtt{x} : \mathtt{C_1}, \mathtt{x} : \mathtt{C_2}\}$,
  si 1. $\mathtt{x} : \mathtt{C_1} \sqcap \mathtt{C_2}$ est dans $\mathtt{S}$,
  2. les contraintes $\mathtt{x} : \mathtt{C_1}$ et $\mathtt{x} : \mathtt{C_2}$ ne sont pas toutes deux dans $\mathtt{S}$.
- Disjonction de concepts :
  $\mathtt{S} \rightarrow_{\sqcup} \mathtt{S} \cup \{\mathtt{x} : \mathtt{D}\}$,
  si 1. $\mathtt{x} : \mathtt{C_1} \sqcup \mathtt{C_2}$ est dans $\mathtt{S}$,
  2. ni $\mathtt{x} : \mathtt{C_1}$ ni $\mathtt{x} : \mathtt{C_2}$ ne sont dans $\mathtt{S}$,
  3. $\mathtt{D} = \mathtt{C_1}$ ou $\mathtt{C_2}$.
- Quantification universelle :
  $\mathtt{S} \rightarrow_{\forall} \mathtt{S} \cup \{\mathtt{z} : \mathtt{C}\}$,
  si 1. $\mathtt{x} : \forall \mathtt{r.C}$ est dans $\mathtt{S}$,
  2. $\mathtt{xrz}$ est dans $\mathtt{S}$,
  3. $\mathtt{z} : \mathtt{C}$ n'est pas dans $\mathtt{S}$.
- Quantification existentielle :
  $\mathtt{S} \rightarrow_{\exists} \mathtt{S} \cup \{\mathtt{xr_1y}, \ldots, \mathtt{xr_ky}, \mathtt{y} : \mathtt{C}\}$,
  si 1. $\mathtt{r} = \mathtt{r_1} \sqcap \mathtt{r_2} \ldots \sqcap \mathtt{r_k}$,
  2. $\mathtt{x} : \exists \mathtt{r.C}$ est dans $\mathtt{S}$,
  3. $\mathtt{y}$ est une nouvelle variable,
  4. il n'existe pas $\mathtt{z}$ tel que les contraintes $\mathtt{xrz}$ et $\mathtt{z} : \mathtt{C}$ soient toutes deux dans $\mathtt{S}$,
  5. si $\mathtt{x}$ est une variable, alors il n'existe aucune variable $\mathtt{z}$ introduite *avant* $\mathtt{x}$ dans $\mathtt{S}$ qui soit équivalente à $\mathtt{x}$ dans $\mathtt{S}$.
- Cardinalité minimale :
  $\mathtt{S} \rightarrow_{\geq} \mathtt{S} \cup \{\mathtt{xr_1y_i}, \ldots, \mathtt{xr_ky_i} | \mathtt{i} \in \mathtt{1..n}\} \cup \{\mathtt{y_i} \neq \mathtt{y_j} | \mathtt{i}, \mathtt{j} \in \mathtt{1..n}, \mathtt{i} \neq \mathtt{j}\}$,
  si 1. $\mathtt{r} = \mathtt{r_1} \sqcap \mathtt{r_2} \ldots \sqcap \mathtt{r_k}$,
  2. $\mathtt{x} : (\geq \mathtt{n}\ \mathtt{r})$ est dans $\mathtt{S}$,
  3. $\mathtt{y_1}, \ldots, \mathtt{y_n}$ sont des nouvelles variables,
  4. il n'existe pas dans $\mathtt{S}$ $\mathtt{n}$ r-successeurs de $\mathtt{x}$ séparés et deux à deux distincts,
  5. si $\mathtt{x}$ est une variable, alors il n'existe aucune variable $\mathtt{z}$ introduite *avant* $\mathtt{x}$ dans $\mathtt{S}$ qui soit équivalente à $\mathtt{x}$ dans $\mathtt{S}$.
- Cardinalité maximale :
  $\mathtt{S} \rightarrow_{\leq} \mathtt{S}[\mathtt{y}/\mathtt{z}]$
  si 1. $\mathtt{x} : (\leq \mathtt{n}\ \mathtt{r})$ est dans $\mathtt{S}$,
  2. $\mathtt{x}$ a plus de $\mathtt{n}$ r-successeurs dans $\mathtt{S}$,
  3. $\mathtt{y}$ et $\mathtt{z}$ sont deux r-successeurs de $\mathtt{x}$ dans $\mathtt{S}$ non séparés.

FIG. 6: *Les six règles de décomposition de la méthode des tableaux sémantiques pour $\mathcal{ALCNR}$.*

3.2.3 Un exemple d'application des règles de décomposition

Considérons la base terminologique Σ dont il faut tester la satisfiabilité ; cet exemple, où tous les concepts et les rôles employés sont primitifs, est tiré de [Donini *et al.*,1994] :

```
Σ = {enfant(Pierre,Marie),
     (all enfant (not MUSICIEN))(Pierre),
     (and (some enfant HOMME)
          (some ami (or CYCLISTE ALPINISTE)))(Marie)}
```

Intuitivement, cette base représente trois faits : `Marie` est l'enfant de `Pierre`, tous les enfants de `Pierre` ne sont pas musiciens, `Marie` a un garçon — un enfant de sexe masculin — et un ami qui est cycliste ou alpiniste. Cette base est satisfiable, ce qui va être confirmé formellement ci-dessous.

La notation allemande est généralement utilisée pour vérifier la satisfiabilité d'une base terminologique avec la méthode des tableaux sémantiques. La base Σ est donc présentée en notation allemande suivie du système de contraintes $\mathtt{S}_\Sigma$ associé.

$$\begin{aligned}\Sigma = \{&\mathtt{enfant(Pierre, Marie)},\\ &\forall\mathtt{enfant.}\neg\mathtt{MUSICIEN(Pierre)},\\ &\exists\mathtt{enfant.HOMME} \sqcap \exists\mathtt{ami.CYCLISTE} \sqcup \mathtt{ALPINISTE(Marie)}\}\end{aligned}$$

$$\begin{aligned}\mathtt{S}_\Sigma = \{&\mathtt{Pierre\ enfant\ Marie},\\ &\mathtt{Pierre} : \forall\mathtt{enfant.}\neg\mathtt{MUSICIEN},\\ &\mathtt{Marie} : \exists\mathtt{enfant.HOMME} \sqcap \exists\mathtt{ami.CYCLISTE} \sqcup \mathtt{ALPINISTE}\}\end{aligned}$$

Étant donné un système de contraintes Σ, une stratégie d'application des règles est la suivante :

- Appliquer une règle à une variable lorsqu'aucune règle n'est applicable à une instance.
- Appliquer une règle à une variable `x` lorsqu'aucune règle n'est applicable à une variable `y` introduite avant `x`.
- Appliquer une règle génératrice lorsqu'aucune règle non génératrice n'est applicable.

Voilà une suite possible d'applications de règles de décomposition :

$$\begin{aligned}
\text{Règle } \rightarrow_\forall : \mathtt{S}_1 &= \mathtt{S}_\Sigma \cup \{\mathtt{Marie} : \neg\mathtt{MUSICIEN}\}\\
\text{Règle } \rightarrow_\sqcap : \mathtt{S}_2 &= \mathtt{S}_1 \cup \{\mathtt{Marie} : \exists\mathtt{enfant.HOMME},\\
&\qquad\qquad \mathtt{Marie} : \exists\mathtt{ami.CYCLISTE} \sqcup \mathtt{ALPINISTE}\}\\
\text{Règle } \rightarrow_\exists : \mathtt{S}_3 &= \mathtt{S}_2 \cup \{\mathtt{Marie\ enfant\ x}, \mathtt{x} : \mathtt{HOMME}\}\\
\text{Règle } \rightarrow_\exists : \mathtt{S}_4 &= \mathtt{S}_3 \cup \{\mathtt{Marie\ ami\ y}, \mathtt{y} : \mathtt{CYCLISTE} \sqcup \mathtt{ALPINISTE}\}\\
\text{Règle } \rightarrow_\sqcup : \mathtt{S}_5 &= \mathtt{S}_4 \cup \{\mathtt{y} : \mathtt{CYCLISTE}\}
\end{aligned}$$

Le système de contraintes $\mathtt{S}_5$ est complet et non contradictoire : la base terminologique Σ est donc satisfiable. Un modèle de Σ est le suivant :

$$
\begin{array}{l}
\Delta_{\mathcal{I}} \ = \{\mathtt{Pierre}, \mathtt{Marie}, \mathtt{x}, \mathtt{y}\} \\
\mathtt{Pierre}^{\mathcal{I}} = \mathtt{Pierre} \\
\mathtt{Marie}^{\mathcal{I}} = \mathtt{Marie} \\
\mathtt{HOMME}^{\mathcal{I}} = \{\mathtt{x}\} \\
\mathtt{MUSICIEN}^{\mathcal{I}} = \emptyset \\
\mathtt{CYCLISTE}^{\mathcal{I}} = \{\mathtt{y}\} \\
\mathtt{ALPINISTE}^{\mathcal{I}} = \emptyset \\
\mathtt{enfant}^{\mathcal{I}} = \{(\mathtt{Marie}, \mathtt{x})\} \\
\mathtt{ami}^{\mathcal{I}} = \{(\mathtt{Marie}, \mathtt{y})\}
\end{array}
$$

3.2.4 Variante sur le test de satisfiabilité d'une base terminologique

Dans [Buchheit *et al.*,1993], les inclusions $\mathtt{D} \sqsubseteq \mathtt{C}$ de concepts primitifs ou définis et la présence de circuits terminologiques sont autorisés, ce qui nécessite une définition alternative de la notion de base terminologique (voir § 2.8.4). Dans ce cas, les contraintes peuvent avoir quatre formes : $\mathtt{x} : \mathtt{C}$, $\mathtt{xry}$, $\mathtt{x} \neq \mathtt{y}$ et $\forall \mathtt{x}.\mathtt{x} : \mathtt{C}$, où $\mathtt{C}$ dénote un concept, $\mathtt{r}$ un rôle, $\mathtt{x}$ et $\mathtt{y}$ des objets (variables ou instances). Les trois premières formes de contraintes ont déjà été expliquées. Intuitivement, la contrainte $\forall \mathtt{x}.\mathtt{x} : \mathtt{C}$ exprime que tout objet $\mathtt{x}$ est une instance du concept $\mathtt{C}$. Cette contrainte permet de traiter une inclusion comme $\mathtt{D} \sqsubseteq \mathtt{C}$, qui se transforme alors en $\forall \mathtt{x}.\mathtt{x} : \neg\mathtt{D} \sqcup \mathtt{C}$ (ce qui rappelle le traitement de l'implication en logique classique, où $\mathtt{D} \longrightarrow \mathtt{C} \ \equiv \ \neg\mathtt{D} \vee \mathtt{C}$).

Aux six règles de décomposition du paragraphe 3.2.2 doit être ajoutée une septième règle :

- Règle $\rightarrow_{\forall \mathtt{x}}$: $\mathtt{S} \rightarrow_{\forall \mathtt{x}} \mathtt{S} \cup \{\mathtt{z} : \mathtt{C}\}$,
  si 1. $\forall \mathtt{x}.\mathtt{x} : \mathtt{C}$ est dans $\mathtt{S}$,
  2. $\mathtt{z}$ est dans $\mathtt{S}$,
  3. $\mathtt{z} : \mathtt{C}$ n'est pas dans $\mathtt{S}$.

Intuitivement, $\forall \mathtt{x}.\mathtt{x} : \neg\mathtt{D} \sqcup \mathtt{C}$ signifie qu'un objet $\mathtt{x}$ est soit dans l'extension de $\neg\mathtt{D}$ soit dans l'extension de $\mathtt{C}$, puisque $\mathtt{D} \sqsubseteq \mathtt{C}$ et donc que la réunion $\neg\mathtt{D} \sqcup \mathtt{C}$ couvre l'ensemble du domaine des objets.

Le reste du développement sur le test de satisfiabilité d'une base terminologique demeure inchangé. L'exemple qui suit, tiré de [Buchheit *et al.*,1993], montre le traitement d'une base $\Sigma = (\mathcal{T},\mathcal{A})$, sa transformation en un système de contraintes et la vérification du système de contraintes à l'aide des sept règles de décomposition.

$$
\begin{array}{l}
\mathcal{T} = \{\mathtt{SPORTIF} \sqsubseteq \exists \mathtt{ami}.\mathtt{SPORTIF}\} \\
\mathcal{A} = \{\mathtt{ami}(\mathtt{Pierre}, \mathtt{Jean}), \\
\qquad (\forall \mathtt{ami}.\neg\mathtt{SPORTIF})(\mathtt{Pierre}), \\
\qquad (\exists \mathtt{ami}.\mathtt{SPORTIF})(\mathtt{Jean})\}
\end{array}
$$

Intuitivement, la base $\Sigma = (\mathcal{T},\mathcal{A})$ indique que tout sportif possède un ami sportif (un sportif est forcément quelqu'un ayant un ami sportif), que Pierre est un ami de Jean, que tous les amis de Pierre ne sont pas sportifs et que Jean a un ami sportif. Le système de contraintes correspondant est le suivant :

$$
\begin{aligned}
\mathtt{S}_\Sigma = \{ & \forall \mathtt{x}.\mathtt{x} : \neg\mathtt{SPORTIF} \sqcup \exists\mathtt{ami}.\mathtt{SPORTIF},\\
& \mathtt{Pierre\ ami\ Jean},\\
& \mathtt{Pierre} : \forall\mathtt{ami}.\neg\mathtt{SPORTIF},\\
& \mathtt{Jean} : \exists\mathtt{ami}.\mathtt{SPORTIF}\}\\
& \mathtt{Pierre} \not\doteq \mathtt{Jean}\}
\end{aligned}
$$

Voici une suite possible d'applications à $\mathtt{S}_\Sigma$ des règles de décomposition :

$$
\begin{aligned}
\text{Règle } \rightarrow_\forall : \mathtt{S}_1 &= \mathtt{S}_\Sigma \cup \{\mathtt{Jean} : \neg\mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_{\forall\mathtt{x}} : \mathtt{S}_2 &= \mathtt{S}_1 \cup \{\mathtt{Pierre} : \neg\mathtt{SPORTIF} \sqcup \exists\mathtt{ami}.\mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_{\forall\mathtt{x}} : \mathtt{S}_3 &= \mathtt{S}_2 \cup \{\mathtt{Jean} : \neg\mathtt{SPORTIF} \sqcup \exists\mathtt{ami}.\mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_\sqcup : \mathtt{S}_4 &= \mathtt{S}_3 \cup \{\mathtt{Pierre} : \neg\mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_\exists : \mathtt{S}_5 &= \mathtt{S}_4 \cup \{\mathtt{Jean\ ami\ x}, \mathtt{x} : \mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_{\forall\mathtt{x}} : \mathtt{S}_6 &= \mathtt{S}_5 \cup \{\mathtt{x} : \neg\mathtt{SPORTIF} \sqcup \exists\mathtt{ami}.\mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_\sqcup : \mathtt{S}_7 &= \mathtt{S}_6 \cup \{\mathtt{x} : \exists\mathtt{ami}.\mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_\exists : \mathtt{S}_8 &= \mathtt{S}_7 \cup \{\mathtt{x\ ami\ y}, \mathtt{y} : \mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_{\forall\mathtt{x}} : \mathtt{S}_9 &= \mathtt{S}_8 \cup \{\mathtt{y} : \neg\mathtt{SPORTIF} \sqcup \exists\mathtt{ami}.\mathtt{SPORTIF}\}\\
\text{Règle } \rightarrow_\sqcup : \mathtt{S}_{10} &= \mathtt{S}_9 \cup \{\mathtt{y} : \exists\mathtt{ami}.\mathtt{SPORTIF}\}
\end{aligned}
$$

Deux remarques peuvent être faites. La première concerne les règles $\rightarrow_\sqcup$ et la seconde le fait que le système $\mathtt{S}_{10}$ est complet. Lors de l'application des règles $\rightarrow_\sqcup$, deux choix sont possibles, et dans certains cas, l'un des choix n'est pas autorisé car il mène à une contradiction : c'est le cas par exemple dans l'application de la règle $\rightarrow_\sqcup$ au système $\mathtt{S}_5$, où l'assertion $\mathtt{x} : \neg\mathtt{SPORTIF}$ serait en contradiction avec l'assertion $\mathtt{x} : \mathtt{SPORTIF}$ du système $\mathtt{S}_5$. Ce cas se reproduit avec la variable $\mathtt{y}$ lors de la construction du système $\mathtt{S}_{10}$.

Le système $\mathtt{S}_{10}$ est complet non contradictoire. En outre, la règle $\rightarrow_\exists$ n'est pas applicable à $\mathtt{y}$ dans $\mathtt{S}_{10}$ car les variables $\mathtt{x}$ et $\mathtt{y}$ sont équivalentes dans $\mathtt{S}_{10}$ et, de ce fait, la condition 5 de la règle $\rightarrow_\exists$ n'est pas respectée. À partir de $\mathtt{S}_{10}$ il est possible de construire une interprétation $\mathcal{I}$ qui est un modèle de Σ :

$$
\begin{aligned}
& \Delta_\mathcal{I} = \{\mathtt{Pierre}, \mathtt{Jean}, \mathtt{x}, \mathtt{y}\}\\
& \mathtt{Pierre}^\mathcal{I} = \mathtt{Pierre},\\
& \mathtt{Jean}^\mathcal{I} = \mathtt{Jean},\\
& \mathtt{SPORTIF}^\mathcal{I} = \{\mathtt{x}, \mathtt{y}\},\\
& \mathtt{ami}^\mathcal{I} = \{(\mathtt{Pierre}, \mathtt{Jean})(\mathtt{Jean}, \mathtt{x})(\mathtt{x}, \mathtt{y})(\mathtt{y}, \mathtt{y})\}
\end{aligned}
$$

4 La pratique du raisonnement terminologique

Dans cette partie, qui se divise en trois paragraphes, nous présentons brièvement le processus de classification et donnons des détails sur les logiques de descriptions CLASSIC et LOOM : la première nous sert de cadre pour illustrer le processus de classification et la gestion de requêtes dans une base de concepts, tandis qu'est mis en valeur le processus d'instanciation de la seconde.

4.1 Le processus de classification

Le processus de classification est une des deux composantes principales du raisonnement terminologique. Il consiste à établir la position d'un concept dans la hiérarchie des concepts notée $\mathcal{T}$. Une technique de parcours standard de la hiérarchie $\mathcal{T}$ a été mise au point pour le système KL-ONE [Lipkis,1982], puis reprise et précisée dans [MacGregor,1988] et [Baader *et al.*,1994a]. Dans les logiques de descriptions, le processus de classification ne concerne que les concepts — et éventuellement les rôles — *définis*.

Deux cas se présentent lors de l'introduction d'un nouveau concept `X` dans $\mathcal{T}$, selon que `X` est primitif ou défini. Soit `X` est un concept primitif et la position de `X` dans $\mathcal{T}$ est directement donnée par l'expression qui introduit `X`. Par exemple, l'expression `X` $\sqsubseteq$ `(and Y Z)`, où `X`, `Y` et `Z` sont primitifs, indique que les *subsumants les plus spécifiques* ou SPS de `X` sont `Y` et `Z`. Un concept `C` fait partie des subsumants les plus spécifiques d'un concept `X` si `C` subsume `X` et si aucun subsumant de `X`, non équivalent à `C`, n'est subsumé par `C`. De façon duale, un concept `D` fait partie des *subsumés les plus généraux* de `X` si `X` subsume `D` et si aucun subsumé de `X`, non équivalent à `D`, ne subsume `D`. Lorsque `X` est un concept défini, la définition de `X` induit la position de `X` dans la hiérarchie.

Le processus de classification se décompose en trois étapes principales :

`(i)` la recherche des subsumants les plus spécifiques (SPS),

`(ii)` la recherche des subsumés les plus généraux (SPG),

`(iii)` la mise en place des nouvelles relations entre le concept à classer `X`, ses SPS et ses SPG.

Le parcours de $\mathcal{T}$ s'effectue en profondeur, sur l'ensemble des concepts de la hiérarchie, en partant de la racine `TOP`. Si le concept courant `C` ne subsume pas `X`, alors la sous-hiérarchie de racine `C`, notée $\mathcal{T}_{\texttt{C}}$, est élaguée. Si `C` subsume `X`, alors `C` devient temporairement le subsumant le plus spécifique de `X` dans $\mathcal{T}_{\texttt{C}}$, et le devient définitivement si aucun subsumant plus spécifique n'est trouvé dans $\mathcal{T}_{\texttt{C}}$.

L'obtention des subsumants les plus spécifiques permet de focaliser la recherche des subsumés les plus généraux de `X`. Il suffit alors de ne considérer que l'ensemble des descendants $\mathcal{D}$ des SPS. Un premier test est réalisé pour savoir si `X` subsume un des subsumants `SUP` trouvé lors de la première étape : si c'est le cas, `X` est équivalent à `SUP` et le processus est terminé. Si ce n'est pas le cas, la recherche des subsumés les plus généraux se fait comme suit : si `X` subsume un descendant `D` dans $\mathcal{D}$, alors `D` est un SPG et sa descendance est ignorée, sinon les descendants de `D` sont testés à leur tour, jusqu'à ce qu'un SPG soit trouvé ou bien qu'il n'y ait plus de descendant à tester ; le processus s'arrête lorsque tous les concepts de $\mathcal{D}$ ont été testés. Lorsque les SPS et les SPG de `X` ont été découverts, de nouveaux liens sont mis en place entre `X`, ses SPS et ses SPG.

Le processus de classification entre dans le cadre du *raisonnement par classification*, qui se décrit par une boucle *instanciation – classification – opérations* [Kaczmarek *et al.*,1986] [Napoli and Laurenço,1993]. Une classification a lieu chaque fois qu'une définition de concept est nouvellement créée ou modifiée. La donnée des SPS peut alors servir à déclencher certaines opérations en vue d'accomplir une certaine tâche. Ainsi, le processus de classification peut jouer un rôle dans (i) la mise en valeur des concepts en relation avec un nouveau concept, (ii) le traitement de requêtes, (iii) la construction de la hiérarchie $\mathcal{T}$.

Pour terminer, notons que la classification des rôles se calque sur celle des concepts tout en étant beaucoup plus simple, car le nombre de constructeurs est généralement nettement plus restreint. Cependant, il n'en est pas question dans ce rapport de recherche (voir [Royer and Quantz,1993] par exemple).

4.2 Règles et requêtes en CLASSIC

Pour illustrer le mécanisme de classification, nous allons montrer comment sont traitées les *règles* et les *requêtes* en CLASSIC [Borgida *et al.*,1989] [Brachman *et al.*,1991]. CLASSIC possède un langage de définition de concepts minimal et un certain nombre de caractéristiques qui lui sont propres.

Nous donnons à la figure 7 une grammaire simplifiée du langage de description des concepts [12]. Les expressions conceptuelles comprennent des concepts et des instances de concepts, et elles s'organisent dans une hiérarchie de racine `THING`. En dehors des constructeurs déjà rencontrés, `AND`, `ALL`, `AT-LEAST` et `AT-MOST`, le constructeur `ONE-OF` permet de définir un domaine de valeurs à la façon d'un type énuméré, en donnant la liste exhaustive des instances qui constituent le domaine de valeurs élémentaires possibles. Le constructeur `SAME-AS` permet de spécifier qu'un *attribut* — c'est-à-dire un rôle mono-valué — a la même valeur qu'un autre attribut (un enfant a le même nom que son père), et le constructeur `FILLS` associe une valeur élémentaire à un rôle dans une instance ou dans un concept. Les rôles et les attributs sont primitifs, mais il est possible de déclarer qu'un rôle est un sous-rôle d'un autre rôle ou encore qu'un rôle possède un rôle inverse.

Une autre particularité, et non des moindres, est la possibilité d'associer à un concept, par l'intermédiaire du constructeur `TEST`, des fonctions dites de *test*, qui retournent `T`, `NIL`, ou `?` pour *inconnu* (la dernière valeur existe en vertu de l'hypothèse du monde ouvert). Les fonctions (de) test servent à mettre en place des contraintes vérifiées à l'instanciation du concept, qui s'avèrent difficiles voire impossibles à exprimer d'une autre façon. Par exemple, l'expression conceptuelle `(AND INTEGER (TEST EVENP))` permet de définir le concept de nombre pair, où `INTEGER` désigne le type des entiers en LISP et `EVENP` la fonction LISP qui teste la parité d'un entier. Comme indiqué dans [Brachman,1992], les fonctions test ont notablement facilité l'utilisation de CLASSIC.

4.2.1 Les règles

Un ensemble de règles peut être attaché à un concept `C` pour fournir des informations additionnelles qui découlent de l'appartenance d'un individu à l'extension de `C`. La règle `R` « si `C` alors `D` » exprime que si un individu `i` est une

12. Bien que cela ne figure pas dans la grammaire simplifiée donnée ici, les objets du langage hôte, LISP en l'occurrence, sont aussi considérés comme des concepts et sont subsumés par le concept `HOST-THING`, qui est indépendant de `CLASSIC-THING`, le subsumant le plus général de tous les concepts introduits par l'utilisateur.

```
concept ⟶ identificateur |
           (AND concept+) |
           (ALL rôle expression) |
           (AT-LEAST entier rôle) |
           (AT-MOST entier rôle) |
           (ONE-OF instance*) |
           (SAME-AS attribut attribut) |
           (FILLS rôle instance+) |
           (TEST fonction argument*)
```

FIG. 7: *Une grammaire simplifiée du langage de description des concepts de* CLASSIC *[Resnick* et al.*,1995]. En* CLASSIC*, tous les rôles sont primitifs et les attributs sont des rôles particuliers mono-valués.*

instance du « concept prémisse » `C`, alors `i` est aussi une instance du « concept conclusion » `D`. La définition de la règle `R` n'est pas de même nature que l'adjonction de la conclusion `D` dans le corps du concept `C`. En effet, l'information exprimée dans la conclusion de la règle est associée à l'individu `i` s'il s'avère qu'à un certain moment cet individu est une instance de `C`, à la manière d'une condition nécessaire. En revanche, cette caractéristique n'est pas une condition suffisante et donc n'entre pas en ligne de compte dans le processus de classification pour reconnaître le concept instancié par `i` : ainsi, les conclusions des règles apparaissent comme des propriétés additionnelles ou *contingentes*.

Ci-dessous, nous donnons un exemple de concept défini et une règle associée. La syntaxe utilisée est celle de CLASSIC ; la primitive `cl-define-concept` permet d'introduire un concept défini, qui est automatiquement classé une fois que sa définition est donnée[13] :

```
(cl-define-concept 'MUSICIEN-DÉBUTANT
                   '(AND MUSICIEN (AT-MOST 1 instrument)
                                  (AT-MOST 1 professeur)
                                  (ALL pratique FAIBLE)))

(cl-add-rule 'IMD
             'MUSICIEN-DÉBUTANT
             (ALL instrument INSTRUMENT-D-ÉTUDE))
```

La règle nommée `IMD`, pour *instrument de musicien débutant*, précise que si un individu `i` est reconnu comme une instance de `MUSICIEN-DÉBUTANT`, alors `i` est subsumé par l'expression `(ALL instrument INSTRUMENT-D-ÉTUDE)`, ce qui signifie que la personne dénotée par `i` pratique un instrument d'étude. Cependant, jouer sur un instrument d'étude n'est pas une condition suffisante d'appartenance à l'extension du concept `MUSICIEN-DÉBUTANT`.

13. La primitive `cl-define-concept` correspond au symbole $\doteq$ employé dans les paragraphes précédents.

Les règles de CLASSIC sont un moyen pratique mais relativement limité d'exprimer des connaissances contingentes. Des études sont actuellement menées pour combiner $\mathcal{ALCNR}$ avec des systèmes de règles de production plus réalistes, faisant intervenir des variables et divers types de chaînage [Levy and Rousset,1996].

4.2.2 Les requêtes

Considérons à présent l'interrogation d'une hiérarchie de concepts et d'instances, à l'aide de *requêtes* [Beck *et al.*,1989] [Lenzerini and Schaerf,1991] [Buchheit *et al.*,1994b]. CLASSIC peut être considéré comme un système minimal de gestion de base de données relationnelle, où les concepts sont assimilés à des relations unaires, les rôles à des relations binaires, et la hiérarchie de concepts à une base de données. Toutefois, il n'existe qu'un seul et même langage pour décrire les relations et interroger la base, à savoir CLASSIC lui-même. Par suite, toute expression conceptuelle CLASSIC peut s'interpréter comme une requête : par exemple l'extension de l'expression conceptuelle `E = (AT-LEAST 2 enfant)` fournit l'ensemble des individus qui *satisfont* `E`.

Sur le plan pratique, une requête est représentée par un concept `Q`, et satisfaire la requête `Q` consiste à classifier `Q` dans la hiérarchie des concepts. L'ensemble des instances des SPG de `Q` fournit alors l'ensemble `Extension(SPG(Q))` des individus qui sont des réponses *sûres* à la requête `Q`, puisqu'ils sont dans l'extension de `Q`. Toutefois, certaines instances des SPS de `Q`, dont l'ensemble est noté `Extension(SPS(Q))`, peuvent aussi être des réponses à la requête. Pour trouver l'ensemble complet des réponses à la requête `Q`, il suffit d'ajouter à `Extension(SPG(Q))` les instances de `Extension(SPS(Q))` qui satisfont effectivement `Q`. Pour illustrer ce qui précède, considérons les définitions suivantes :

```
(cl-define-concept 'C1 '(AND PERSONNE (AT-LEAST 1 enfant)))
(cl-define-concept 'C2 '(AND PERSONNE (AT-LEAST 5 enfant)))
(cl-define-concept 'C3 '(AND PERSONNE (ALL enfant MUSICIEN)
                                      (AT-LEAST 4 enfant)))
(cl-define-concept 'Q '(AND PERSONNE (AT-LEAST 2 enfant)))
(cl-create-ind 'Daniel '(FILLS enfant Anne Aline))
```

Le concept `C1` subsume la requête `Q`, et `Q` subsume les concepts `C2` et `C3`. Les instances de `C2` et `C3` satisfont donc la requête `Q`. Par ailleurs, l'individu `Daniel`, qui est une instance de `C1` sans être une instance de `C2` et de `C3`, satisfait également la requête `Q`, et doit donc être ajouté à l'ensemble des solutions.

La méthode d'interrogation d'une hiérarchie de concepts que nous venons de voir est un bon exemple de ce que peut être l'intégration de techniques de classification dans l'univers des bases de données. Ainsi, certaines fonctionnalités associées aux logiques de descriptions, subsomption et classification par exemple, peuvent être reprises avec profit pour mettre en œuvre de nouveaux modèles et mécanismes de gestion de données [Bergamaschi and Sartori,1992] [Sheth *et al.*,1993] [Borgida,1995].

4.2.3 Pour conclure sur CLASSIC

CLASSIC est un langage relativement simple et facile à utiliser, et il a d'ailleurs été conçu comme tel, puisque des constructeurs comme `OR` ou `NOT` ont été volontairement écartés. De ce fait, le langage de description des concepts est moins riche, mais en contrepartie, les tests de subsomption, la classification et l'instanciation sont rapides et font de CLASSIC un langage ayant de bonnes performances (voir l'étude comparative réalisée dans [Baader *et al.*,1994a]). Toutefois, l'algorithme de subsomption n'est pas complet et certains cas où la subsomption n'est pas détectée sont discutés dans le manuel [Resnick *et al.*,1995], ainsi que dans [McGuinness and Borgida,1995] et [McGuiness,1996].

Parmi les applications qui ont été réalisées avec CLASSIC, citons le système de planification de tâches de programmation CLASP [Devanbu and Litman,1991] [Devanbu and Litman,1996], le système de gestion de réseaux téléphoniques KITSS [Nonnenmann and Eddy,1992], le système de configuration de réseaux de télécommunications PROSE [Wright *et al.*,1993], et le système de fouille de données IMACS [Brachman *et al.*,1993]. Parmi les travaux les plus intéressants réalisés autour de CLASSIC, il faut mentionner la gestion d'objets composites [Padgham and Lambrix,1994] et la production d'explications associées aux tests de subsomption (voir [McGuinness and Borgida,1995] [Hors,1995] et [Rousset and Hors,1996]).

4.3 L'instanciation en LOOM

Contrairement au parti pris de simplicité caractérisant CLASSIC, LOOM s'affiche comme une logique de descriptions riche, qui intègre des fonctionnalités nombreuses et diverses [MacGregor,1991a] [MacGregor and Burstein,1991]. Concepts et rôles peuvent être primitifs ou définis, et sont organisés en hiérarchies. Un extrait de la grammaire à partir de laquelle se construisent les descriptions de concepts et de rôles est donnée à la figure 8.

LOOM intègre notamment la disjonction et la négation de concepts définis, et donc contient $\mathcal{ALCNR}$, ce qui fait de LOOM une des logiques de descriptions offrant le plus de possibilités de représentation à l'heure actuelle. LOOM autorise les rôles définis ; `:domain` et `:range` spécifient respectivement le domaine et le co-domaine d'un rôle — l'expression `(:and r (:range C))` peut être vue comme une construction similaire à `(restrict r C)` —, `:filled-by` précise les valeurs élémentaires d'un rôle ; `:inverse` indique la relation inverse.

4.3.1 Le détail du processus d'instanciation en LOOM

La figure 9 montre un ensemble de descriptions de concepts et de rôles qui vont nous servir à illustrer le processus d'instanciation en LOOM (ce processus est détaillé dans les articles [MacGregor,1988] [MacGregor,1991b] [MacGregor and Brill,1992]). Le processus d'instanciation permet d'*ajuster* et de *maintenir* le *type* d'une instance chaque fois que ce type est modifié, et d'assurer le maintien de la cohérence globale de l'ensemble des instances lors de leur évolution. Le type d'une expression conceptuelle `E` est l'ensemble des subsumants les plus spécifiques de `E` ; par exemple, le type de l'expression `(:and MARIÉ PERSONNE`

```
concept ⟶ identificateur |
            (:and concept+) |
            (:or concept+) |
            (:not concept+) |
            (:all rôle concept) |
            (:at-least entier rôle) |
            (:at-most entier rôle) |
            (:exactly entier rôle) |
            (:one-of instance+) |
            (:filled-by instance+) |
            (:same-as rôle rôle)

rôle ⟶ identificateur |
         (:and rôle+) |
         (:domain concept) |
         (:range concept) |
         (:inverse rôle)
```

FIG. 8: *Une grammaire simplifiée du langage de description des concepts et des rôles de* LOOM *[*LOOM*,1991].*

FEMME) est le couple (MARIÉ FEMME). Le type d'un individu i est déterminé par l'ensemble des concepts les plus spécifiques dont i est une instance.

Trois opérations de modifications nécessitent la mise à jour du type d'une instance i :

(M1) la mise en place ou la suppression d'une assertion portant sur i,

(M2) la modification de la valeur d'un rôle attaché à i,

(M3) la modification du type d'une instance j en relation avec i par l'intermédiaire d'un rôle r (i et j sont alors dites *adjacentes*).

La stratégie sur laquelle s'appuie le processus d'instanciation s'articule autour de trois étapes, *propagation*, *abstraction* et *classification*. Lorsque se produit une modification de la forme (M1), (M2) ou (M3), les nouvelles informations apportées par cette modification sont *propagées* aux instances en relation avec i. Ensuite, est créé un concept A décrivant i, appelé l'*abstraction* de i, puis l'abstraction A est classifiée dans la hiérarchie des concepts, et l'ensemble des subsumants les plus spécifiques de A constitue le type de i.

Nous allons expliciter la mise en œuvre du processus d'instanciation en examinant un ensemble de 7 assertions tirées de [MacGregor,1988], qui sont données à la figure 10. Dans ces assertions, la primitive tell sert à mettre en place une valeur élémentaire, tandis que la primitive forget permet de supprimer une telle valeur.

L'assertion A1 provoque la construction de l'abstraction Pierre :

```
(defconcept Pierre :is (:and MARIÉ :primitive))
            TYPE : (MARIÉ)
            HITS : ()
```

```
(defconcept PERSONNE :is :primitive)
(defconcept HOMME :is (:and PERSONNE :primitive))
(defconcept FEMME :is (:and PERSONNE :primitive))
(defconcept MARIÉ :is (:and PERSONNE :primitive))

(defrelation enfant :is :primitive
             (:domain PERSONNE) (:range PERSONNE))
(defrelation fille :is (:and enfant (:range FEMME)))

(defconcept PÈRE :is (:and HOMME (:at-least 1 enfant)))
(defconcept PÈRE-CHANCEUX :is (:and PÈRE (:all fille MARIÉ)))
```

FIG. 9: *Un ensemble de descriptions de concepts et de rôles en* LOOM*, d'après [MacGregor,1988].*

```
(A1) : (tell (MARIÉ Pierre))
(A2) : (tell (enfant Pierre Marthe))
(A3) : (tell (HOMME Pierre))
(A4) : (tell (FEMME Marthe))
(A5) : (tell (MARIÉ Marthe))
(A6) : (forget (MARIÉ Marthe))
(A7) : (tell (PÈRE-CHANCEUX Pierre))
```

FIG. 10: *Une suite d'assertions en* LOOM*.*

```
MISSES : ()
```

La définition de l'abstraction se fait à partir de la définition des concepts qui subsument `i` au moment où commence la mise à jour du type de `i`. Les subsumants sont mémorisés dans une liste notée `TYPE`. Deux autres listes sont attachées à une abstraction `A`, `HITS` et `MISSES`, qui mémorisent respectivement les informations positives et négatives reçues lors du processus d'instanciation. Chaque fois qu'une nouvelle information apparaît, les listes `HITS` et `MISSES` sont consultées : si les réponses aux requêtes contenues dans `HITS` restent vraies et celles contenues dans `MISSES` restent fausses, alors rien ne se passe, sinon le type de `i` doit être ajusté. Ici, le type de `Pierre`, qui est `MARIÉ`, est mis en place dans la liste `TYPE`, et aucune question n'est posée au sujet de ce type, car `MARIÉ` ne possède aucun subsumé.

L'assertion `A2` provoque la mise en place du type de `Marthe`, en l'occurrence `PERSONNE`, sur la base de la définition du rôle `enfant`. La donnée de l'assertion `A3` provoque l'examen des subsumés immédiats de `HOMME`, afin d'y associer éventuellement `Pierre`. Ainsi, l'expression `(:at-least 1 enfant)` attachée au concept `PÈRE` fournit une première requête. La réponse à cette requête est positive grâce à `A2` ; le type de `Pierre` est ajusté et la valeur de `HITS` est mise à jour. La classification se continue pour essayer d'associer l'abstraction décrivant `Pierre` au subsumé immédiat de `PÈRE`, qui est `PÈRE-CHANCEUX`. L'expression `(:all fille MARIÉ)` fournit une seconde requête, qui reçoit cette fois une réponse négative. L'abstraction décrivant `Pierre` devient :

```
(defconcept Pierre :is (:and HOMME MARIÉ :primitive))
   TYPE : (PÈRE MARIÉ)
   HITS : ((:at-least 1 enfant))
   MISSES : ((:all fille MARIÉ))
```

La donnée de l'assertion `A4` provoque la mise à jour du type de `Marthe` qui devient `FEMME`, et la nouvelle information est propagée aux instances adjacentes. Toutefois, la modification du type de `Marthe` ne permet toujours pas de répondre à la question `(:all fille MARIÉ)`, et donc le type de `Pierre` demeure inchangé.

L'assertion `A5` provoque la mise à jour du type de `Marthe` en `(FEMME MARIÉ)`. Après propagation, la réponse à la requête `(:all fille MARIÉ)` est positive, ce qui, en cascade, implique une mise à jour du type de `Pierre` :

```
(defconcept Pierre :is (:and HOMME MARIÉ :primitive)
                        (:all fille MARIÉ))
   TYPE : (PÈRE-CHANCEUX MARIÉ)
   HITS : ((:at-least 1 enfant) (:all fille MARIÉ))
   MISSES : ()
```

L'assertion `A6` remet en cause les informations apportées par `(A5)`, ce qui provoque un retour des choses à l'état précédent (modifications des types de `Pierre` et de `Marthe`). La donnée de l'assertion `A7` modifie directement le type de `Pierre` et provoque la modification du type de `Marthe` qui devient `(FEMME MARIÉ)`, comme en `(A5)`. Aucune autre requête supplémentaire ne peut être posée.

Le traitement des assertions qui vient d'être décrit met en lumière une démarche qui peut se généraliser à d'autres problèmes, comme par exemple provoquer l'émission de requêtes pendant la classification d'un objet pour obtenir des informations supplémentaires et ainsi préciser davantage la représentation de l'objet étudié.

4.3.2 Pour conclure sur LOOM

Les concepteurs de LOOM ont voulu en faire un système de représentation de connaissances général et sophistiqué (avec, en contrepartie, une procédure d'inférence non complète). En particulier, l'environnement de LOOM a été enrichi d'un système à base de règles de production [Yen *et al.*,1991a] [Yen *et al.*,1991b], et le raisonnement peut également reposer sur un moteur d'inférences, ce qui renforce le côté hybride de LOOM.

Un bon nombre d'applications est développé en LOOM, où se signalent entre autres un système d'aide à la réutilisation de programmes [Mark *et al.*,1992], un système de gestion de bases de données hiérarchiques réparties [Chu *et al.*,1993], un système de gestion d'informations hétérogènes [Arens *et al.*,1993] et un système de raisonnement à partir de cas dans le domaine légal [Ashley and Aleven,1994].

5 Discussions sur les logiques de descriptions

Dans ce paragraphe, nous présentons la famille des logiques de descriptions dans sa globalité, et les systèmes les plus connus et les plus originaux. Ensuite, nous évoquons la place des logiques de descriptions en représentation des connaissances et les rapports qu'entretiennent les logiques de descriptions avec des formalismes de représentation comme les logiques classiques et les systèmes de représentation de connaissances par objets (ou systèmes de RCO).

5.1 La diversité des logiques de descriptions

Le formalisme des logiques de descriptions provient essentiellement des travaux entrepris par R.J. Brachman sur la représentation des connaissances, ainsi que sur le système KL-ONE et ses descendants [Brachman,1977] [Brachman,1979] [Brachman and Schmolze,1985]. Dans KRYPTON , un descendant de KL-ONE, apparaissent pour la première fois les notions de *TBox*, de *ABox*, et les fonctionnalités que doit recouvrir une *ABox* [Brachman *et al.*,1983] [Brachman *et al.*,1985]. La *ABox* de KRYPTON se compose d'un démonstrateur qui opère sur la base de la logique du premier ordre, ce qui fait à la fois sa force (généralité) mais aussi sa faiblesse (efficacité et indécidabilité). Une approche plus modeste est à l'origine de KL-TWO : la *TBox* est composée du langage NIKL et la *ABox* de PENNI [Vilain,1985], une adaptation du système de maintien de cohérence RUP [McAllester,1982]. Le fait que RUP ait été conçu indépendamment de toute *TBox* explique, au moins en partie, les problèmes posés par la coexistence des deux formalismes. Le système NIKL, pour *New Implementation of KL-ONE*, est une version épurée et améliorée de KL-ONE [Schmolze and Mark,1991], avec laquelle sont mis en valeur les principes du raisonnement par classification et les liens existant entre instanciation et maintien de cohérence [Kaczmarek *et al.*,1986] (toutefois, la subsomption dans le système NIKL reste indécidable [Patel-Schneider,1989b]).

KANDOR se présente comme un descendant de KRYPTON pour lequel les possibilités de représentation sont volontairement simplifiées, afin que les processus de classification et d'instanciation soient les plus efficaces possibles [Patel-Schneider,1984] [Patel-Schneider *et al.*,1984]. En particulier, le système KANDOR se caractérise par un langage de description de concepts et de rôles limité et un niveau factuel où seules les instanciations de concepts et de rôles sont autorisées, comme cela est présenté au paragraphe 2.8. Le niveau factuel des systèmes BACK et MESON , dont il est question plus loin, a été défini de la même façon ; ce type de niveau factuel a fini par devenir un standard pour les logiques de descriptions actuelles. Le gestionnaire de logiciels LaSSIE a notamment été développé en KANDOR [Devanbu *et al.*,1991] [Devanbu,1994].

Plus récemment, CLASSIC [Borgida *et al.*,1989] [Brachman *et al.*,1991] [Borgida and Patel-Schneider,1994], le dernier né de la famille KL-ONE, intègre l'ensemble de l'expérience acquise au cours de la décennie précédente. La simplicité d'utilisation de CLASSIC— voir à ce sujet [Brachman,1992] — la fiabilité du système et la qualité de sa documentation ont valu à CLASSIC d'être utilisé dans de nombreuses applications et d'être devenu une des références en matière de logiques de descriptions (voir 4.2).

À l'opposé, LOOM s'affiche comme un système de représentation de connaissances universel, malgré les problèmes de complexité et de complétude que pose un tel parti-pris. Comme pour CLASSIC, de nombreuses applications ont été réalisées avec LOOM, qui est également devenu une référence en matière de logiques de descriptions (voir § 4.3).

Après CLASSIC et LOOM, il faut mentionner BACK , un système qui a été développé à la Technische Universität de Berlin dans la deuxième moitié des années 80 [von Luck *et al.*,1987] [Peltason *et al.*,1989] [Hoppe *et al.*,1993]. Le système a eu une longue évolution, mais dès l'origine, ses concepteurs ont voulu faire de BACK un système fiable, utilisable dans des applications réalistes, notamment pour mettre en œuvre des systèmes d'information et de gestion de bases de données opérationnels. En dehors des études concernant la complexité de la subsomption, de nombreux problèmes relatifs aux logiques de descriptions ont été abordés par les membres du groupe travaillant sur BACK, comme les extensions temporelles pour une logique de descriptions [Schmiedel,1990], les relations entre rôles [Quantz,1990], les différentes possibilités d'implantation d'une *ABox* [Kindermann,1990], le retrait d'informations [Kindermann,1992] et la représentation de règles [Schild,1989].

Quelques applications ont été réalisées avec BACK : un système de représentation de connaissances médicales sur le cœur associé à un système de gestion de patients cardiaques [Schmiedel,1991b] et un système de traitement du langage naturel, toujours en cours de développement [Quantz,1992] [Quantz,1993] [Schmitz and Quantz,1993]. Cette dernière application a donné naissance à un descendant de BACK nommé FLEX [Bergmann and Quantz,1995] [Quantz *et al.*,1995]. Le système BACK a également servi de support au projet Esprit AIMS, pour *Advanced Information Management System*, dont l'objectif était de développer un environnement de programmation adapté à la réalisation de systèmes d'information hybrides intégrant représentation des connaissances et gestion de bases de données multimédia [Peltason,1991] [Volle,1992] [Blanco *et al.*,1994]. Deux domaines d'application ont été abordés pour le projet, l'évaluation des risques pour l'environnement liés aux activités d'une usine chimique et la gestion d'information médicales dans un centre de cardiologie [Schmiedel,1991a].

À côté des trois figures de proue que sont CLASSIC, LOOM et BACK, il faut mentionner KRIS [Baader and Hollunder,1991a] [Baader and Hollunder,1991c], qui possède un langage de description de concepts et de rôles contenant $\mathcal{ALCNR}$ et qui a servi de support à bon nombre d'études théoriques. Ensuite, il nous faut citer quelques systèmes dont l'influence est moins grande mais qui ont leur originalité sur les plans théoriques ou pratiques. Le système MESON [Owsnicki-Klewe,1988] a servi à concevoir un système de configuration, qui est une des applications privilégiés des systèmes terminologiques. Le système K-REP a servi de support à des études théoriques sur le point de vue intensionnel de la subsomption [Dionne *et al.*,1993], mais aussi à l'implantation de systèmes de planification [Mays *et al.*,1991] [Weida and Litman,1992]. En particulier, K-REP est pourvu d'une sémantique algébrique, et un système équationnel fournit les propriétés de la subsomption.

Il n'est bien sûr pas possible de mentionner tous les systèmes existants, mais plusieurs catalogues ont été dressés à des époques données, notamment dans [Woods and Schmolze,1992] et dans [Rich,1991], un numéro spécial du journal *SIGART Bulletin* qui a fait suite à un séminaire sur l'avenir des logiques de descriptions et leur place dans la conception d'applications [Patel-Schneider *et al.*,1990]. Depuis 1991, des séminaires ont lieu annuellement, mettant en avant la conception d'applications et les avancées de la théorie des logiques de descriptions [Nebel *et al.*,1991] [Peltason *et al.*,1991] [MacGregor *et al.*,1992] [Baader *et al.*,1994b] [Borgida *et al.*,1995] et [Padgham *et al.*,1996] [Rousset *et al.*,1997]. Notons, pour finir, que dans [Heinsohn *et al.*,1994], [Baader *et al.*,1994a], [Horrocks,1995] et [Speel,1995] sont données des comparaisons très instructives des principaux systèmes sur le plan des possibilités de représentation des connaissances et sur le plan de l'efficacité.

5.2 Les logiques de descriptions et la représentation de connaissances

Dans ce paragraphe, nous examinons d'abord les relations existant entre les logiques de descriptions et les deux formalismes de représentation que sont la logique classique et les systèmes de représentation de connaissances par objets. Ensuite, nous discutons de l'apport des logiques de descriptions en représentation des connaissances, de quelques limitations, et nous terminons en évoquant certaines perspectives de recherche sur les logiques de descriptions.

Logiques de descriptions et logique classique

Les éléments principaux qui caractérisent une logique de descriptions sont le langage de description des concepts et des rôles, l'interprétation associée aux expressions du langage et la relation de subsomption. Le parallèle avec un formalisme logique est alors naturel, puisqu'une logique s'appuie sur un langage de construction de formules (bien formées), une interprétation associée aux formules et des règles d'inférences. Les concepts d'une logique de descriptions peuvent être vus comme des prédicats unaires et les rôles comme des prédicats binaires, la subsomption comme une règle d'inférence ; la subsomption $\mathtt{D} \sqsubseteq \mathtt{C}$ est d'ailleurs quelquefois notée $\mathtt{D} \Rightarrow \mathtt{C}$ [Borgida,1996]. Toutefois, le parallèle est justifié jusqu'à un certain point : une expression conceptuelle et une assertion peuvent se voir comme des cas particuliers de formules logiques du premier ordre, où ne figurent que des variables instanciées. En dehors de l'absence de variables, la manipulation des formules se fait essentiellement de la même façon en logique classique et dans une logique de descriptions.

Une dernière question concerne le *pouvoir expressif* des logiques de descriptions. Cette question est étudiée en détail dans [Borgida,1996], qui montre qu'une logique de descriptions comme $\mathcal{ALCNR}$ a le même pouvoir expressif qu'un sous-ensemble de formules de la logique du premier ordre n'ayant que des prédicats unaires et binaires, et n'autorisant que l'usage de trois variables au plus (dont deux variables libres au plus). Nous n'avancerons pas plus dans ces considérations et renvoyons le lecteur intéressé à [Borgida,1996].

Logiques de descriptions et systèmes de RCO

Les logiques de descriptions présentent un certain nombre de similarités avec les systèmes de RCO. Brièvement, un système de RCO s'appuie sur des éléments de connaissances qui sont des *classes* — *classes* et *frames* sont considérés ici comme synonymes — qui encapsulent un état et un comportement [Masini *et al.*,1989]. L'état et le comportement d'une classe sont définis par un ensemble de propriétés *descriptives* et *fonctionnelles*. Les propriétés descriptives — les *attributs* — peuvent se voir associer des facettes introduisant un type, une valeur, et éventuellement des réflexes, qui se déclenchent lorsqu'un accès en lecture ou en écriture est effectué sur la propriété (*attachement procédural*). Les propriétés fonctionnelles — les *méthodes* — sont des procédures ou des fonctions activées par envoi de message. Les classes sont organisées en une hiérarchie d'héritage par l'intermédiaire d'une relation de *spécialisation* sur laquelle repose le mécanisme d'*héritage*. Les classes peuvent être instanciées pour fournir les instances qui décrivent les objets de base qui sont manipulés dans une application.

Un concept, considéré comme une conjonction de rôles, est comparable à une classe, considérée comme une conjonction de propriétés. Toutefois, les connecteurs `or` et `not`, qui font partie de $\mathcal{ALCNR}$, ne se rencontrent habituellement pas dans un système de RCO (pas plus que dans CLASSIC d'ailleurs). Cette absence rend les disjonctions et les incompatibilités difficiles à représenter (au moins de façon déclarative), bien que la sémantique d'un domaine de valeurs dans un système de RCO soit celle d'une disjonction de valeurs, les domaines spécialisants étant inclus dans les domaines spécialisés, tout comme c'est le cas pour le constructeur `one-of`, existant en CLASSIC (voir § 4.2) et en LOOM (voir § 4.3).

Un rôle peut se voir associer des restrictions qui portent sur son co-domaine et sa cardinalité, à l'instar des attributs d'une classe. En revanche, l'absence de caractère procédural, comme les méthodes ou les réflexes, différencie nettement logiques de descriptions et systèmes de RCO. Un système de RCO tire l'essentiel de son pouvoir expressif et déductif du mécanisme d'héritage, des réflexes associés aux attributs et des méthodes associées aux classes. De ce point de vue, l'opposition logiques de descriptions–systèmes de RCO s'apparente à la controverse déclaratif–procédural [Winograd,1975]. Pourtant, les choses ne sont pas aussi tranchées : dans CLASSIC par exemple, il existe des fonctions test et des règles qui jouent un rôle comparable à celui de certains réflexes [14] d'un système de RCO.

Malgré quelques tentatives d'intégration [Schmiedel,1992] [Kortüm,1993], les procédures et fonctions continuent à être écartées du formalismes des logiques de descriptions, car il n'existe actuellement aucun moyen satisfaisant de leur associer un statut sémantique convenable (bien que certaines études aillent dans ce sens [Balaban,1995] [Kifer *et al.*,1995]).

L'héritage est un mécanisme de partage de propriétés qui permet d'associer de nouvelles connaissances aux classes et à leurs instances, généralement en re-

14. Un réflexe si-possible est une procédure qui vérifie la validité d'une valeur devant être donnée à un attribut, ce qui est généralement la tâche d'une fonction test. Les réflexes si-ajout et si-enlève s'exécutent une fois qu'une valeur a été donnée à un attribut, tout comme une règle se déclenche après classification.

trouvant une valeur, une méthode ou un réflexe dans la hiérarchie d'héritage. Une nouvelle classe se construit en spécialisant une ou plusieurs classes existantes (par adjonction ou redéfinition de propriétés). La mise en place d'une nouvelle classe dans la hiérarchie est laissée à l'entière liberté du programmeur, tandis que la mise en place d'un nouveau concept défini est à la charge du processus de classification. De plus, la recherche de propriétés ne constitue qu'un aspect contingent du pouvoir déductif lié à la subsomption et à la classification. Plus précisément, les propriétés associées à une classe sont considérées comme des conditions nécessaires et jamais comme des conditions suffisantes : une telle sémantique est dite *descriptive* (voir par exemple [Nebel,1991] [Euzenat,1993] [Buchheit *et al.*,1994a] [Calvanese,1996a] [De Giacomo and Lenzerini,1997]). Dans les logiques de descriptions, les concepts primitifs ont aussi une sémantique descriptive, mais les concepts définis ont une sémantique *définitionnelle*, sur laquelle repose le mécanisme de classification. Ainsi, dans un système de RCO, si un objet `o` est une instance de la classe `C`, cela signifie que `o` a été défini comme tel et qu'il possède les propriétés de la classe `C` (*conditions nécessaires*). Réciproquement, si un objet `o` possède des propriétés qui pourraient en faire une instance de la classe `C`, il n'existe (généralement) pas de sémantique qui puisse assurer que `o` est une une instance de `C` (*conditions suffisantes*). Par suite, le processus de classification n'est pas directement et entièrement exploitable, ce qui peut poser des problèmes [Patel-Schneider,1990] [Chouvet *et al.*,1996]. Toutefois, dans certains cas, un processus de classification est tout de même exploitable et exploité, comme c'est le cas dans [Napoli and Laurenço,1993] et [Vismara,1996], où la classification se fait sur des propriétés décrivant une relation partie-tout (voir aussi [Ducournau,1996a]).

Nous terminerons en évoquant le *raisonnement par défaut.* Dans un système de RCO, tout attribut d'une classe `C` se voit généralement associer une valeur par défaut, qui peut être redéfinie dans toutes les sous-classes de `C`. Une sous-classe `D` hérite toutes les propriétés associées à ses super-classes, sauf *masquage* ou *exception* à l'héritage [Ducournau and Habib,1989]. Dans le premier cas, la valeur d'une propriété héritable, valeur effective, méthode ou réflexe, est redéfinie. Dans le second cas, un ensemble de propriétés héritables est écarté. Le masquage et les exceptions ont été beaucoup étudiés dans les systèmes de RCO [Ducournau *et al.*,1995], mais ces deux techniques posent de nombreux problèmes dans le cadre des logiques de descriptions, car elles conduisent à un raisonnement non monotone, qui va à l'encontre de la sémantique d'inclusion des extensions associée à la subsomption[Brachman,1985] : il ne peut y avoir d'exception dans l'inclusion ensembliste $\mathtt{D}^{\mathcal{I}} \subseteq \mathtt{C}^{\mathcal{I}}$ lorsque $\mathtt{D} \sqsubseteq \mathtt{C}$. Toutefois, un certain nombre de travaux traitent d'une possible intégration du raisonnement par défaut dans les logiques de descriptions : [Baader and Hollunder,1992] [Quantz and Royer,1992] [Baader and Hollunder,1993] [Padgham and Nebel,1993] [Padgham and Zhang,1993] [Straccia,1993]. Parmi les études d'intégration, celle qui est présentée dans [Coupey and Fouqueré,1997] et [Coupey and Salotti,1997] se démarque quelque peu : des exceptions et des valeurs par défauts sont directement associées à la structure des concepts dans la logique de descriptions $\mathcal{AL}_{\delta\epsilon}$. Une autre originalité de cette étude est de fournir à $\mathcal{AL}_{\delta\epsilon}$ une sémantique algébrique qui s'appuie sur un système équationnel, à l'image des travaux réalisés autour de K-REP [Dionne *et al.*,1993].

Il existe des comparaisons entre logiques de descriptions, modèles de données (dans les bases de données) et systèmes de RCO, qui ont été faites à l'initiative des chercheurs travaillant sur les logiques de descriptions [Borgida,1992] [Calvanese *et al.*,1994] [Borgida,1995] [Calvanese,1996b]. Toutefois, ces comparaisons ne tiennent pas compte de l'aspect procédural existant dans un système de RCO et peuvent donc apparaître comme incomplètes. À l'inverse, l'ensemble des études théoriques sur les logiques de descriptions a inspiré certains chercheurs travaillant sur les systèmes de RCO et les a amenés a mettre en place une théorie des systèmes classificatoires [Euzenat,1994] [Ducournau,1996a] [Ducournau,1996b].

Soulignons encore qu'un des intérêts principaux des travaux réalisés sur les logiques de descriptions est de fournir aux concepteurs de systèmes à bases de connaissances de bonnes indications sur ce qu'ils peuvent — et ne peuvent pas — représenter et traiter comme connaissances avec une logique de descriptions, à l'image de [Brachman *et al.*,1991], où sont examinées « les tâches pour lesquelles CLASSIC est et n'est pas approprié », et de [Baader,1996], où le pouvoir d'expression des logiques de descriptions est analysé formellement.

Quelques limitations des logiques de descriptions

Nous avons vu au paragraphe 2.9.2 que les travaux sur la complexité du raisonnement terminologique ont mis en évidence que plus un langage de description de concepts et de rôles est riche, plus la complexité du raisonnement est élevée. Le dilemme est alors le suivant : faut-il préférer un langage limité et concis, à l'image de CLASSIC, qui ne permet pas de tout représenter, mais qui a un comportement déductif contrôlable, ou bien faut-il préférer un langage riche, à l'image de BACK ou de LOOM , aux possibilités multiples et variées, mais au comportement déductif imprévisible? La dualité « langage pauvre » ou « langage riche » a fait l'objet de l'article [Doyle and Patil,1991], où est critiquée la tendance minimaliste associée à CLASSIC et l'« enfermement conceptuel » auquel aurait donné naissance les résultats des travaux sur la complexité de la subsomption[15]. Dans [Doyle and Patil,1991] sont également discutées les limitations des logiques de descriptions dans le cadre de la conception de systèmes d'intelligence artificielle réalistes. Ces limitations sont de plusieurs sortes, parmi lesquelles figurent :

- la difficulté de représenter des relations entre rôles, comme dans les exemples suivants : *deux personnes qui ont pour résidence des pays différents*, ou bien *deux enfants d'une même famille qui vont dans deux écoles différentes qui sont éloignées l'une de l'autre de 1 kilomètre* (ces exemples sont tirés de [Schmiedel,1992], sur les interactions entre rôles, voir aussi [Quantz,1990] et [Hanschke,1992]) ;

15. Le comportement déductif d'un sous-système de CLASSIC est examiné minutieusement dans [Borgida and Patel-Schneider,1994], tandis qu'une étude comparative concernant les langages « pauvres » et « riches » est donnée dans [Heinsohn *et al.*,1994].

- la représentation de quantifications sur les relations comme dans *tous les élèves assistent à tous les cours* [Woods,1991] ;
- les traitements numériques en général ;
- les définitions récursives ;
- les relations n-aires.

Il faut également mentionner le fait qu'une logique de descriptions ne peut pas être employée seule dans une application réaliste — comme tout formalisme de RCO d'ailleurs — et nécessite l'usage parallèle d'un langage de programmation, avec lequel sont écrites les fonctions de contrôle global de l'application.

À l'heure actuelle, les conséquences des travaux sur la complexité de la subsomption semblent avoir été assimilées et les réflexions se portent plutôt sur l'utilisation des logiques de descriptions dans des applications et domaines divers comme les bases de données et les systèmes d'information, la fouille de données, le traitement du langage naturel, etc.

Perspectives : niveaux de connaissances et systèmes multi-niveaux

Les apports des logiques de descriptions à la représentation de connaissances sont importants. L'un des principaux est d'avoir mis l'accent sur l'existence de différents niveaux de connaissances [Brachman,1979] et de proposer des moyens effectifs pour les prendre en compte, comme le font les processus de classification et d'instanciation. À côté des niveaux généraux que sont les niveaux terminologiques et factuels, les règles et implications — comme celles de CLASSIC et de LOOM — constituent un troisième niveau de représentation. D'autres niveaux de connaissances peuvent venir enrichir ces niveaux de base [MacGregor,1991a] et traiter divers besoins de représentation, comme le temps [Schmiedel,1990], les connaissances d'ordre numérique (*concrete domains* [16]) [Baader and Hanschke,1991], la relation de composition et les objets composites [Padgham and Lambrix,1994] et enfin les règles du premier ordre [Levy and Rousset,1996]. Dans le même genre d'idées, il est proposé dans [Buchheit *et al.*,1994a] (et repris dans [Calvanese,1996b]) de distinguer deux niveaux dans la description des concepts et des rôles : le niveau des *schémas*, qui se rapporte aux concepts et aux rôles primitifs, et le niveau des *vues*, qui se rapportent aux concepts définis (schéma et vue sont deux termes qui ont été choisis par analogie avec la terminologie employée dans les bases de données). Il est alors possible de travailler avec des circuits terminologiques au niveau des schémas et d'introduire des constructions soit au niveau des schémas soit au niveau des vues, ce qui conduit à une appréhension différente de la complexité du raisonnement terminologique.

Toutefois, différencier ces niveaux de connaissances est quelquefois difficile, certaines connaissances se décrivant aussi bien par des règles, des implications ou encore des expressions conceptuelles. Ce problème met en lumière l'absence de véritable méthodologie de représentation et de conception d'applications dans le cadre des logiques de descriptions [Brachman *et al.*,1991]. C'est également vrai

16. Une logique de descriptions s'appuyant sur une telle extension de $\mathcal{AL}$ a été utilisée dans le cadre d'un système de raisonnement à partir de cas [Kamp,1996].

pour les systèmes de RCO [Chouvet *et al.*,1996], en dehors de tous les travaux sur l'acquisition de connaissances.

En s'appuyant sur des travaux qui s'intéressent à l'intégration de plusieurs formalismes de représentation dans un même système [Carré *et al.*,1995], la mise au point de systèmes de représentation *multi-niveaux*, où chaque niveau est relatif à un langage de description de concepts et de rôles qui lui est propre, constitue une perspective d'utilisation des logiques de descriptions [MacGregor,1991a]. Un premier niveau peut correspondre à un langage de description de concepts et de rôles simple, comme $\mathcal{FL}$- , $\mathcal{PL}_1$ ou $\mathcal{PL}_2$ [Donini *et al.*,1991b], pour lequel la subsomption est correcte, complète et de complexité polynômiale. Les descriptions de premier niveau ne sont en relation qu'avec des descriptions de ce niveau. Au second niveau apparaît un langage de description de concepts et de rôles de l'ordre de $\mathcal{ALCNR}$, ce qui remet (forcément) en cause la complétude de la subsomption. Un troisième niveau peut être réservé à la représentation de connaissances procédurales, un autre aux méta-connaissances, et d'autres niveaux encore peuvent prendre en compte le temps, la composition, etc. — comme évoqué ci-dessus — sans oublier un niveau dédié à la programmation. Les niveaux peuvent être classés par ordre (partiel) de complexité croissant, et les descriptions de niveau n ne peuvent être en relation qu'avec des descriptions de niveau inférieur ou égal à n. L'utilisateur quant à lui peut choisir son niveau de représentation en toute connaissance de cause et tenir compte du comportement déductif du système au niveau où il travaille.

6 Conclusion

Dans ce rapport de recherche, nous avons présenté les logiques de descriptions, qui constituent un formalisme de représentation de connaissances caractérisé par les points suivants :

- Un langage permet de construire des descriptions conceptuelles qui sont génériques (concepts primitifs et définis) ou individuelles (instances).
- Une sémantique est associée à chaque construction syntaxique par l'intermédiaire d'une interprétation.
- Une relation de subsomption permet d'organiser les descriptions par niveau de généralité, et de procéder à des inférences ; cette relation est à la base des processus de classification et d'instanciation.

Sur le plan de la théorie de la représentation des connaissances, les logiques de descriptions ont apporté une certaine originalité dans le traitement de problèmes d'intelligence artificielle. Elles se sont avérées être un des premiers formalismes structurels — où sont manipulées des structures — à différencier et étudier explicitement le niveau syntaxique et le niveau sémantique dans une représentation, ainsi que les relations existant entre ces deux niveaux, comme c'est le cas en logique classique.

Les logiques de descriptions ne sont pas seulement un formalisme théorique et réservé aux théoriciens de la représentation des connaissances : la recherche autour des logiques de descriptions est très active et a des visées à la fois pratiques et théoriques. Ainsi, la construction de systèmes traitant de problèmes réels est au centre des préoccupations de nombreux travaux de recherches. Les logiques de descriptions ne sont pas des formalismes figés et sont suffisamment souples pour accepter l'introduction de nouveaux constructeurs, capable de répondre à des besoins particuliers, charge alors au concepteur d'évaluer et de tenir compte de la complexité de la subsomption qui en résulte.

Les études effectuées dans le cadre des logiques de descriptions constituent une base de première importance pour les recherches sur les systèmes de RCO, que ce soit pour les concepteurs ou les utilisateurs de ces systèmes. En particulier, le raisonnement par classification est devenu une méthode d'inférence essentielle, mais qui nécessite d'être intégrée et utilisée de façon correcte. Dans tous les cas, la recherche sur les systèmes de RCO peut se nourrir des recherches sur les logiques de descriptions, mais la réciproque est vraie aussi, et l'avenir des logiques de descriptions passe certainement par une intégration avec les systèmes de RCO.

Note sur les logiques de descriptions et le Web

Les lecteurs navigateurs pourront aller visiter l'adresse suivante sur le Web :
`http://dl.kr.org/dl`
où ils trouveront des informations concernant notamment la bibliographie et les séminaires annuels sur les logiques de descriptions, ainsi que les moyens de joindre les chercheurs du domaine et d'obtenir les systèmes disponibles.

7 Annexe : détails sur la subsomption dans $\mathcal{ALNR}$

Dans ce paragraphe, nous détaillons un algorithme NC pour le langage de description de concepts et de rôles $\mathcal{ALNR}$. Cet algorithme s'appuie sur celui que [Nebel,1990a] propose pour le langage noté $\mathcal{NTF}$, qui correspond à $\mathcal{ALNR}$, moyennant l'identification de la construction (`some r`) avec la construction (`atleast 1 r`). D'autres algorithmes NC sont détaillés dans [Patel-Schneider,1989a], [Borgida,1992] [Royer and Quantz,1993] et, pour le langage CLASSIC , dans [Borgida and Patel-Schneider,1994]. À cette occasion, signalons que [Royer and Quantz,1993] est un des documents les plus complets qui soient sur les algorithmes NC : plus de 150 règles de subsomption — pour les concepts et les rôles — y sont étudiées, règles qui ont été reprises dans le système FLEX [Bergmann and Quantz,1995] [Quantz *et al.*,1995].

7.1 La normalisation des concepts

Les règles de normalisation associées à l'algorithme de subsomption pour $\mathcal{ALNR}$ sont essentiellement de deux types : factorisation et traitement des incohérences.

[N1] Factorisation des conjonctions de concepts et de rôles :

- $\texttt{(and (and C}_1\texttt{ C}_2\texttt{) (and C}_3\texttt{ C}_4\texttt{)...)} \rightarrow \texttt{(and C}_1\texttt{ C}_2\texttt{ C}_3\texttt{ C}_4\texttt{...)}$.
- $\texttt{(and (and r}_1\texttt{ r}_2\texttt{) (and r}_3\texttt{ r}_4\texttt{)...)} \rightarrow \texttt{(and r}_1\texttt{ r}_2\texttt{ r}_3\texttt{ r}_4\texttt{...)}$.

[N2] Transformations et factorisation des co-domaines (`all`) :

- $\texttt{(all r C)} \rightarrow \texttt{(all r (and C))}$.
- $\texttt{(and (all r (and C}_1\texttt{ C}_2\texttt{)) (all r (and C}_3\texttt{ C}_4\texttt{))...)} \rightarrow$

  $\texttt{(and (all r (and C}_1\texttt{ C}_2\texttt{ C}_3\texttt{ C}_4\texttt{))...)}$.

[N3] Traitements des expressions incohérentes :

- $\texttt{(and A...(not A)...)} \rightarrow \texttt{BOTTOM}$, où `A` dénote un concept primitif.
- Si $\texttt{r}_2 \sqsubseteq \texttt{r}_1$ et si $\texttt{n}_1 < \texttt{n}_2$, alors

  $\texttt{(and (atmost n}_1\texttt{ r}_1\texttt{)...(atleast n}_2\texttt{ r}_2\texttt{)...)} \rightarrow \texttt{BOTTOM}$.
- Si $\texttt{r}_2 \sqsubseteq \texttt{r}_1$ et si $\texttt{n}_2 > 0$, alors

  $\texttt{(and (all r}_1\texttt{ BOTTOM)...(atleast n}_2\texttt{ r}_2\texttt{)...)} \rightarrow \texttt{BOTTOM}$.

7.2 La comparaison des formes normales

Voici les différentes règles de comparaison associées au test $\texttt{D} \sqsubseteq \texttt{C}$ dans l'algorithme de subsomption de $\mathcal{ALNR}$. Les données sont une description normalisée — concept ou rôle — $\texttt{C} = \texttt{(and C}_1\texttt{...C}_\texttt{m}\texttt{)}$ qui est comparée à une description normalisée $\texttt{D} = \texttt{(and D}_1\texttt{...D}_\texttt{n}\texttt{)}$. Le test « est-ce que la description `C` subsume la description `D` ? » retourne `vrai` si et seulement si, pour chaque $\texttt{C}_\texttt{i}$, $\texttt{i} = \texttt{1},..,\texttt{m}$, il existe $\texttt{D}_\texttt{j}$, $\texttt{j} = \texttt{1},..,\texttt{n}$, tel que $\texttt{D}_\texttt{j} \sqsubseteq \texttt{C}_\texttt{i}$.

Les règles de comparaison données ci-dessous examinent le test $\mathtt{D} \sqsubseteq \mathtt{C_i}$, où $\mathtt{C_i}$ est noté simplement `C`.

- `[bottom]` : si D = `BOTTOM` alors retourner `vrai`.
- `[concept primitif]` : si `C` et si `D` dénotent des concepts primitifs, alors tester C = D ou $\mathtt{D} \dot{\leq} \mathtt{C}$ (la relation $\mathtt{D} \dot{\leq} \mathtt{C}$ peut être vérifiée par transitivité).
- `[rôle primitif]` : si `C` et si `D` dénotent des rôles primitifs, alors tester C = D ou $\mathtt{D} \dot{\leq} \mathtt{C}$ (la relation $\mathtt{D} \dot{\leq} \mathtt{C}$ peut être vérifiée par transitivité).
- `[not]` : si C = `(not C')` et D = `(not D')`, où `C'` et `D'` dénotent des concepts primitifs, alors tester C' = D' ou $\mathtt{C}' \sqsubseteq \mathtt{D}'$.

  Les cas où les négations sont *croisées*, qui consistent à comparer « C = `(not C')` et D » ou « C et D = `(not D')` » ne sont pas traités.
- `[all]` : si C = $(\mathtt{all}\ \mathtt{r_C}\ \mathtt{C}')$ et si D = $(\mathtt{all}\ \mathtt{r_D}\ \mathtt{D}')$, alors tester $\mathtt{r_C} \sqsubseteq \mathtt{r_D}$ et $\mathtt{D}' \sqsubseteq \mathtt{C}'$
- `[some]` : si C = $(\mathtt{some}\ \mathtt{r_C})$ et si D = $(\mathtt{some}\ \mathtt{r_D})$, alors tester $\mathtt{r_D} \sqsubseteq \mathtt{r_C}$.
- `[atleast]` : si C = $(\mathtt{atleast}\ \mathtt{n_C}\ \mathtt{r_C})$ et si D = $(\mathtt{atleast}\ \mathtt{n_D}\ \mathtt{r_D})$, alors tester $\mathtt{n_C} \leq \mathtt{n_D}$ et $\mathtt{r_D} \sqsubseteq \mathtt{r_C}$.
- `[atmost]` : si C = $(\mathtt{atmost}\ \mathtt{n_C}\ \mathtt{r_C})$ et si D = $(\mathtt{atmost}\ \mathtt{n_D}\ \mathtt{r_D})$, alors tester $\mathtt{n_D} \leq \mathtt{n_C}$ et $\mathtt{r_C} \sqsubseteq \mathtt{r_D}$.
- `[D est une conjonction]` :

  si D = $(\mathtt{and}\ \mathtt{D_1}...\mathtt{D_n})$ dans `[concept primitif]`, `[all]`, `[some]`, `[atleast]` et `[atmost]`, alors tester $\mathtt{D_i} \sqsubseteq \mathtt{C}$ pour *un* $\mathtt{i},\ 1 \leq \mathtt{i} \leq \mathtt{n}$.

  Si `C` dénote un rôle et D = $(\mathtt{and}\ \mathtt{r_1}...\mathtt{r_n})$ dans `[rôle primitif]`, alors tester $\mathtt{r_i} \sqsubseteq \mathtt{C}$ pour *un* $\mathtt{i},\ 1 \leq \mathtt{i} \leq \mathtt{n}$.
- `[Autre cas]` : retourner `faux`.

La complexité de l'algorithme est en $\mathtt{O}((|\mathtt{C}| + |\mathtt{D}|)^2)$, où $|\mathtt{C}|$ et $|\mathtt{D}|$ dénotent respectivement les longueurs des descriptions `C` et `D` (le détail du calcul de la complexité est donné dans [Nebel,1990a], pages 78-80).

7.3 La relation de subsomption et le constructeur restrict

Dans ce paragraphe sont données les principales règles de normalisation et de comparaison associées au constructeur `restrict` (voir § 2.7). Ces règles viennent donc s'ajouter aux précédentes lorsque le langage de description des rôles est enrichi de `restrict`. Ce n'est bien sûr pas la seule extension qu'il soit possible de faire, et nous en présentons d'ailleurs quelques autres dans le paragraphe suivant.

Une règle de normalisation traite les expressions où figure `restrict` :

- $(\mathtt{and}\ (\mathtt{restrict}\ \mathtt{r_1}\ \mathtt{C_1})\ (\mathtt{restrict}\ \mathtt{r_2}\ \mathtt{C_2})) \rightarrow$

```
(1) (restrict fils DOCTEUR) ⊑ (restrict enfant BACHELIER)
(2) (restrict enfant BACHELIER) ⊑ enfant
(3) (restrict fils BACHELIER) ⊑ enfant
(4) (all (restrict enfant BACHELIER) GRAND) ⊑
    (all (restrict fils DOCTEUR) GRAND-OU-MOYEN)
(5) (all (restrict enfant (not DOCTEUR)) GRAND) ⊑
    (all (restrict fils (not BACHELIER)) GRAND-OU-MOYEN)
(6) (atleast 3 (restrict enfant BACHELIER)) ⊑ (atleast 2 enfant)
(7) (atmost 3 enfant) ⊑ (atmost 4 (restrict enfant DOCTEUR))
```

FIG. 11: *Exemples de relations de subsomption associées au constructeur* `restrict`.

(restrict (and r_1 r_2) (and C_1 C_2)).

Cette règle se décompose en deux sous règles :

- (and (restrict r_1 C) r_2) $\rightarrow$ (restrict (and r_1 r_2) C),
- (restrict (restrict r C_1) C_2) $\rightarrow$ (restrict r (and C_1 C_2)).

La règle de comparaison suivante traite les expressions où figure `restrict` :

- [restrict] Si $r_D \sqsubseteq r_C$ et $D \sqsubseteq C$, alors (restrict r_D D) $\sqsubseteq$ (restrict r_C C).
- Cette règle possède un cas particulier : si $r_D \sqsubseteq r_C$, alors (restrict r_D D) $\sqsubseteq$ r_C, ce qui se déduit de la règle précédente, car $r_C \equiv$ (restrict r_C TOP) et $D \sqsubseteq$ TOP.

Ainsi, les « fils docteurs » sont forcément des « enfants bacheliers », ou encore « enfant » est plus général que « enfant bachelier » ou que « fils bachelier » (figure 11, (1), (2), et (3)).

De même, le concept « personne dont tous les fils docteurs sont grands ou moyens » subsume le concept « personne dont tous les enfants bacheliers sont grands », où le concept `GRAND` est subsumé par le concept `GRAND-OU-MOYEN`. (figure 11, (4)). De la même façon, le concept « personne dont tous les fils non bacheliers sont grands ou moyens » subsume le concept « personne dont tous les enfants non docteurs sont grands » (figure 11, (5)).

En combinant les restrictions `atleast` et `atmost` avec `restrict`, nous obtenons les relations suivantes. Le concept « personne qui a au moins deux enfants » subsume le concept « personne qui a au moins trois enfants qui sont bacheliers » (figure 11, (6)), mais le concept « personne qui a au plus quatre enfants docteurs » subsume le concept « personne qui a au plus trois enfants » (figure 11, (7)).

Pour finir, nous allons examiner un exemple repris des articles [Brachman and Levesque,1984] et [Patel-Schneider,1990], qui est illustré par la figure 12, et qui met en valeur la règle :

- (and (all r D) (all (restrict r D) C)) $\sqsubseteq$ (all r C),

```
D ≐ (and PERSONNE
         (all enfant RICHE)
         (all (restrict enfant RICHE) DOCTEUR))

C ≐ (and PERSONNE
         (all enfant DOCTEUR))
```

FIG. 12: *Un exemple de subsomption qui est de l'ordre du* modus ponens, *d'après [Brachman and Levesque,1984] et [Patel-Schneider,1990].*

qui est une sorte de *modus ponens*. Ainsi, le concept défini D, qui dénote « une personne dont tous les enfants sont riches et dont tous les enfants riches sont docteurs », est subsumé par le concept défini C qui dénote « une personne dont tous les enfants sont docteurs ». L'inférence qui vient d'être explicitée est de l'ordre du *modus ponens* et peut se schématiser comme suit : si (enfant RICHE) et si (enfant RICHE) $\longrightarrow$ (enfant DOCTEUR), alors (enfant DOCTEUR).

La démonstration de cette règle, qui est brièvement esquissée ici, est détaillée dans [Brachman and Levesque,1984].

Posons $\mathtt{D}_1$ = (and (all r D) (all (restrict r D) C)) et $\mathtt{D}_2$ = (all r C). Soit $\Delta_{\mathcal{I}}$ un domaine d'interprétation, $\mathcal{I}$ une fonction d'interprétation sur $\Delta_{\mathcal{I}}$, et $\mathtt{x} \in \mathtt{D}_1^{\mathcal{I}}$. Dans ce cas, $\mathtt{x} \in (\mathtt{all\ r\ D})^{\mathcal{I}}$ (1) et $\mathtt{x} \in (\mathtt{all\ (restrict\ r\ D)\ C})^{\mathcal{I}}$ (2). Par (1), si $(\mathtt{x}, \mathtt{y}) \in \mathtt{r}^{\mathcal{I}}$ alors $\mathtt{y} \in \mathtt{D}^{\mathcal{I}}$. Considérons donc un couple $(\mathtt{x}, \mathtt{y}) \in \mathtt{r}^{\mathcal{I}}$ avec $\mathtt{y} \in \mathtt{D}^{\mathcal{I}}$: il vérifie $(\mathtt{x}, \mathtt{y}) \in (\mathtt{restrict\ r\ D})^{\mathcal{I}}$, et, par (2), $\mathtt{y} \in \mathtt{C}^{\mathcal{I}}$. D'où, $\forall \mathtt{x} \in \mathtt{D}_1$ tel que $(\mathtt{x}, \mathtt{y}) \in \mathtt{r}^{\mathcal{I}}$, il vient $\mathtt{y} \in \mathtt{C}^{\mathcal{I}}$; ce qui prouve que $\mathtt{x} \in \mathtt{D}_2^{\mathcal{I}}$, et donc que $\mathtt{D}_1^{\mathcal{I}} \subseteq \mathtt{D}_2^{\mathcal{I}}$, ou encore que $\mathtt{D}_1 \sqsubseteq \mathtt{D}_2$.

Il faut aussi remarquer que la combinaison de la règle [all] avec la règle [restrict] a les conséquences suivantes :

- (all r D) $\sqsubseteq$ (all (restrict r C) D), car (restrict r C) $\sqsubseteq$ r,
- (all r (and C D)) $\sqsubseteq$ (all (restrict r C) D), car (restrict r C) $\sqsubseteq$ r et (and C D) $\sqsubseteq$ D,
- (all r (and C D)) $\sqsubseteq$ (all (restrict r D) C), car (restrict r D) $\sqsubseteq$ r et (and C D) $\sqsubseteq$ C.

Par ailleurs, l'équivalence suivante montre les rapports existant entre la restriction de co-domaine, la négation et la disjonction de concepts [Schmidt-Schauß and Smolka,1991] :

- (all (restrict r D) C) $\equiv$ (all r (or (not D) C)).

7.4 Autres extensions de l'ensemble des constructeurs

Dans ce paragraphe, nous allons décrire plusieurs extensions possibles de l'ensemble des constructeurs.

- Le constructeur `exactly` précise que la cardinalité d'un rôle est exactement `n` : (`exactly n r`) $\equiv$ (`and` (`atleast n r`) (`atmost n r`)).
- Le constructeur (`c-some r C`) $\equiv$ (`some` (`restrict r C`)), qui correspond à $\exists$`r.C` de $\mathcal{ALE}$, est une généralisation de `some`.
- Le constructeur (`c-atleast n r C`) $\equiv$ (`atleast n` (`restrict r C`)) est une généralisation de `atleast`.
- Le constructeur (`c-atmost n r C`) $\equiv$ (`atmost n` (`restrict r C`)) est une généralisation de `atmost`.
- Le constructeur `exactly` peut être généralisé à son tour en utilisant `c-atleast` et `c-atmost` :

  (`c-exactly n r C`) $\equiv$ (`and` (`c-atleast n r C`) (`c-atmost n r C`)).

Les règles de subsomption associées à ces nouveaux constructeurs sont données ci-après ; elles proviennent d'une combinaison des règles de subsomption associées aux constructeurs de base `some`, `atleast`, `atmost` et `restrict`.

- `[exactly]` : (`exactly` $\mathtt{n_D}$ $\mathtt{r_D}$) $\equiv$ (`exactly` $\mathtt{n_C}$ $\mathtt{r_C}$) si et seulement si $\mathtt{n_D} = \mathtt{n_C}$ et $\mathtt{r_D} = \mathtt{r_C}$, puisque les règles `[atleast]` et `[atmost]` ont des conditions croisées.
- `[c-some]` : si $\mathtt{r_D} \sqsubseteq \mathtt{r_C}$ et $\mathtt{D} \sqsubseteq \mathtt{C}$, alors (`c-some` $\mathtt{r_D}$ `D`) $\sqsubseteq$ (`c-some` $\mathtt{r_C}$ `C`).

  Par exemple, le concept « personne ayant des enfants bacheliers » subsume le concept « personne ayant des fils docteurs » (voir `(1)`, figure 13).
- `[c-atleast]` : si $\mathtt{n_D} \geq \mathtt{n_C}$, $\mathtt{r_D} \sqsubseteq \mathtt{r_C}$ et $\mathtt{D} \sqsubseteq \mathtt{C}$, alors (`c-atleast` $\mathtt{n_D}$ $\mathtt{r_D}$ `D`) $\sqsubseteq$ (`c-atleast` $\mathtt{n_C}$ $\mathtt{r_C}$ `C`).

  Par exemple, le concept « personne ayant au moins un enfant bachelier » subsume le concept « personne ayant au moins deux filles docteurs » (`(2)`, figure 13).
- `[c-atmost]` : si $\mathtt{n_D} \leq \mathtt{n_C}$, $\mathtt{r_C} \sqsubseteq \mathtt{r_D}$ et $\mathtt{C} \sqsubseteq \mathtt{D}$, alors (`c-atmost` $\mathtt{n_D}$ $\mathtt{r_D}$ `D`) $\sqsubseteq$ (`c-atmost` $\mathtt{n_C}$ $\mathtt{r_C}$ `C`).

  Par exemple, le concept « personne ayant au plus deux filles docteurs » subsume le concept « personne ayant au plus un enfant bachelier » (`(3)`, figure 13).

Un algorithme s'appuyant sur la méthode des tableaux sémantiques et traitant les restrictions numériques ($\geq$`n r.C`) et ($\leq$`n r.C`) est présenté dans [Baader *et al.*,1996] — qui actualise [Baader and Hollunder,1991b] — et généralisé à des restrictions numériques variables dans [Baader and Sattler,1996].

```
(1) (c-some fils DOCTEUR) ⊑ (c-some enfant BACHELIER)
(2) (c-atleast 2 fille DOCTEUR) ⊑ (c-atleast 1 enfant BACHELIER)
(3) (c-atmost 1 enfant BACHELIER) ⊑ (c-atmost 2 fille DOCTEUR)
```

FIG. 13: *Exemples de relations de subsomption combinant* `restrict` *et les constructeurs* `some`*,* `atleast` *et* `atmost`*.*

Dans [Hors,1995] et [Hors and Rousset,1995], des relations de subsomption combinant les constructeurs `all`, `atleast`, `atmost` et `restrict` sont également données.

Il existe bien sûr beaucoup d'autres extensions possibles, et nous renvoyons le lecteur intéressé aux références citées, mais aussi à [Baader,1996] où sont discutés en termes formels les possibilités d'expression de différentes logiques de descriptions.

Références

[Aït-Kaci, 1986] H. Aït-Kaci. An algebraic semantics approach to the effective resolution of type equations. *Theoretical Computer Science*, 45(3):293–351, 1986.

[Arens *et al.*, 1993] Y. Arens, C.Y. Chee, C.N. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[Ashley and Aleven, 1994] K.D. Ashley and V. Aleven. A logical representation for relevance criteria. In S. Wess, K.D. Althoff, and M.M. Richter, editors, *Proceedings of the 1st European Workshop on Topics in Case-Based Reasoning, Kaiserslautern, Germany*, Lecture Notes in Artificial Intelligence 837, pages 338–352. Springer-Verlag, Berlin, 1994.

[Baader and Hanschke, 1991] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of IJCAI'91, Sydney, Australia*, pages 452–457, 1991.

[Baader and Hollunder, 1991a] F. Baader and B. Hollunder. KRIS: Knowledge Representation and Inference System. *ACM SIGART Bulletin*, 2(3):8–14, 1991.

[Baader and Hollunder, 1991b] F. Baader and B. Hollunder. Qualifying number restrictions in concept languages. In *Proceedings of KR'91, Cambridge (MA), USA*, pages 335–346, 1991.

[Baader and Hollunder, 1991c] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In H. Boley and M.M. Richter, editors, *Proceedings of the Conference on Processing Declarative Knowledge, Kaiserslautern, Germany*, Lecture Notes in Computer Science 567, pages 67–86. Springer-Verlag, Berlin, 1991.

[Baader and Hollunder, 1992] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Proceedings of KR'92, Cambridge (MA), USA*, pages 306–317, 1992.

[Baader and Hollunder, 1993] F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. In *Proceedings of IJCAI'93, Chambéry, France*, pages 669–674, 1993.

[Baader and Sattler, 1996] F. Baader and U. Sattler. Description logics with symbolic number restrictions. In *Proceedings of ECAI'96, Budapest, Hungary*, pages 283–287, 1996.

[Baader *et al.*, 1990] F. Baader, H.-J. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt, and H.-J. Profitlich. Terminological knowledge representation: A proposal for a terminological logic. Technical Memo TM-90-04, DFKI, Saarbrücken Universität, Germany, 1990.

[Baader *et al.*, 1994a] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems. *Journal of Applied Intelligence*, 4(2):109–132, 1994.

[Baader *et al.*, 1994b] F. Baader, M. Lenzerini, W. Nutt, and P.F. Patel-Schneider, editors. *Working Notes of the Description Logic Workshop, Bonn, Germany. Draft*, 1994.

[Baader *et al.*, 1996] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88:195–213, 1996.

[Baader, 1990] F. Baader. Terminological cycles in Kl-One-based knowledge representation languages. In *Proceedings of AAAI'90, Boston (MA), USA*, pages 621–626, 1990.

[Baader, 1996] F. Baader. A formal definition for the expressive power of terminological knowledge representation languages. *Journal of Logic and Computation*, 6(1):33–54, 1996.

[Balaban, 1995] M. Balaban. The F-Logic approach for description logics. *Annals of Mathematics and Artificial Intelligence*, 15(1):19–60, 1995.

[Barbut and Monjardet, 1970] M. Barbut and B. Monjardet. *Ordre et classification — Algèbre et combinatoire (2 tomes)*. Hachette, Paris, 1970.

[Barthélémy *et al.*, 1992] J.-P. Barthélémy, G. Cohen, and A. Lobstein. *Complexité algorithmique et problèmes de communications*. Masson, Paris, 1992.

[Beck *et al.*, 1989] H.W. Beck, S.K. Gala, and S.B. Navathe. Classification as a query processing technique in the Candide semantic data model. In *Proceedings of the 5th International Conference on Data Engineering, Los Angeles (CA), USA*, pages 572–581, 1989.

[Bergamaschi and Sartori, 1992] S. Bergamaschi and C. Sartori. On taxonomic reasoning in conceptual design. *ACM Transactions on Database Systems*, 17(3):385–422, 1992.

[Bergmann and Quantz, 1995] F.W. Bergmann and J.J. Quantz. Parallelizing description logics. In I. Wachsmuth, C.-R. Rollinger, and W. Brauer, editors, *KI-95: Advances in Artificial Intelligence, Bielefield, Germany*, Lecture Notes in Computer Science 981, pages 137–148. Springer-Verlag, Berlin, 1995.

[Blanco *et al.*, 1994] J.M. Blanco, A. Illarramendi, and A. Goñi. Building a federated database system: An approach using a knowledge base system. *International Journal of Intelligent and Cooperative Information Systems*, 3(4):415–455, 1994.

[Borgida and Patel-Schneider, 1994] A. Borgida and P.F. Patel-Schneider. A semantics and complete algorithm for subsumption in the Classic description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[Borgida *et al.*, 1989] A. Borgida, R.J. Brachman, D.L. McGuinness, and L.A. Resnick. Classic: A structural data model for objects. In *Proceedings of the ACM/SIGMOD International Conference on the Management of Data, Portland (OR), USA*, SIGMOD Record, 18(2), pages 59–67, 1989.

[Borgida *et al.*, 1995] A. Borgida, M. Lenzerini, D. Nardi, and B. Nebel, editors. *Proceedings of the International Workshop on Description Logics, Roma, Italy*. Technical Report 07.95, Università degli studi di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica, 1995.

[Borgida, 1992] A. Borgida. From type systems to knowledge representation: Natural semantics specifications for description logics. *International Journal of Intelligent and Cooperative Information Systems*, 1(1):93–126, 1992.

[Borgida, 1995] A. Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[Borgida, 1996] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[Brachman and Levesque, 1984] R.J. Brachman and H.J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of AAAI'84, Austin (TX), USA*, pages 34–37, 1984.

[Brachman and Schmolze, 1985] R.J. Brachman and J.G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[Brachman *et al.*, 1983] R.J. Brachman, R.E. Fikes, and H.J. Levesque. KRYPTON: A functional approach to knowledge representation. *Computer*, 16(10):67–73, 1983.

[Brachman *et al.*, 1985] R.J. Brachman, V.P. Gilbert, and H.J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON. In *Proceedings of IJCAI'85, Los Angeles (CA), USA*, pages 532–539, 1985.

[Brachman *et al.*, 1991] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L.A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE language. In J.F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[Brachman *et al.*, 1993] R.J. Brachman, P.G. Selfridge, L.G. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D.L. McGuinness, and L.A. Resnick. Integrated support for data archaeology. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):159–185, 1993.

[Brachman, 1977] R.J. Brachman. What's in a concept: Structural foundations for semantic networks. *International Journal of Man-Machine Studies*, 9:127–152, 1977.

[Brachman, 1979] R.J. Brachman. On the epistemological status of semantic networks. In N.V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, New York (NY), USA, 1979.

[Brachman, 1985] R.J. Brachman. "I lied about the trees" or defaults and definitions in knowledge representation. *The AI Magazine*, 6(3):80–93, 1985.

[Brachman, 1992] R.J. Brachman. "Reducing" CLASSIC to practice: Knowledge representation theory meets reality. In *Proceedings of KR'92, Cambridge (MA), USA*, pages 247–258, 1992.

[Buchheit *et al.*, 1993] M. Buchheit, F.-M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[Buchheit *et al.*, 1994a] M. Buchheit, F.-M. Donini, W. Nutt, and A. Schaerf. Refining the structure of terminological systems: Terminology = schema + views. In *Proceedings of AAAI'94, Seattle (WA), USA*, pages 199–204, 1994.

[Buchheit *et al.*, 1994b] M. Buchheit, M.A. Jeusfeld, W. Nutt, and M. Staudt. Subsumption between queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994.

[Calvanese *et al.*, 1994] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class-based representation formalisms. In *Proceedings of KR'94, Bonn, Germany*, pages 109–120, 1994.

[Calvanese, 1996a] D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proceedings of ECAI'96, Budapest, Hungary*, pages 303–307, 1996.

[Calvanese, 1996b] D. Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* Dottorato di ricerca in informatica, Università degli Studi di Roma "La Sapienza", Italia, 1996.

[Carré *et al.*, 1995] B. Carré, R. Ducournau, J. Euzenat, A. Napoli, and F. Rechenmann. Classification et objets : programmation ou représentation ? In *Actes des 5es Journées Nationales PRC-GDR Intelligence Artificielle, Nancy*, pages 213–237. Teknea, Toulouse, 1995.

[Chouvet *et al.*, 1996] M.-P. Chouvet, F. Le Ber, J. Lieber, L. Mangelinck, A. Napoli, and A. Simon. Analyse des besoins en représentation et raisonnement dans une représentation à objets — L'exemple de Y3. In Y. Dennebouy, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'96), Leysin, Suisse*, pages 150–169. Ecole Polytechnique Fédérale de Lausanne, 1996.

[Chu *et al.*, 1993] W.W. Chu, M.A. Merzbacher, and L. Berkovich. The design and implementation of CoBase. In *Proceedings of the ACM/SIGMOD International Conference on the Management of Data, Washington (DC), USA*, SIGMOD Record, 22(2), pages 517–522, 1993.

[Cohen *et al.*, 1992] W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In *Proceedings of AAAI'92, San Jose (CA), USA*, pages 754–760, 1992.

[Coupey and Fouqueré, 1997] P. Coupey and C. Fouqueré. Extending conceptual definitions with default knowledge. *Computational Intelligence*, 13(2):258–299, 1997.

[Coupey and Salotti, 1997] P. Coupey and S. Salotti. Une logique de descriptions comme cadre formel d'un système de raisonnement à partir de cas. *Revue d'Intelligence Artificielle*, 11(2):127–177, 1997.

[Coupey *et al.*, 1997] P. Coupey, C. Fouqueré, and S. Salotti. Formalizing partial matching and similarity in Cbr with a description logic. *Applied Artificial Intelligence*, 1997. À paraître.

[Davey and Priestley, 1990] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order.* Cambridge University Press, Cambridge, UK, 1990.

[De Giacomo and Lenzerini, 1997] G. De Giacomo and M. Lenzerini. A uniform framework for concept definitions in description logics. *Journal of Artificial Intelligence Research*, 6:87–110, 1997.

[Devanbu and Litman, 1991] P.T. Devanbu and D.J. Litman. Plan-based terminological reasoning. In *Proceedings of KR'91, Cambridge (MA), USA*, pages 128–138, 1991.

[Devanbu and Litman, 1996] P.T. Devanbu and D.J. Litman. Taxonomic plan reasoning. *Artificial Intelligence*, 84:1–35, 1996.

[Devanbu *et al.*, 1991] P. Devanbu, R.J. Brachman, P.G. Selfridge, and B.W. Ballard. LaSSIE: A knowledge-based software information system. *Communications of the ACM*, 34(5):35–49, 1991.

[Devanbu, 1994] P.T. Devanbu. Software information systems. The State University of New Jersey, Rutgers, 1994.

[Dionne *et al.*, 1993] R. Dionne, E. Mays, and F.J. Oles. The equivalence of model-theoretic and structural subsumption in description logics. In *Proceedings of IJCAI'93, Chambéry, France*, pages 710–716, 1993.

[Donini *et al.*, 1991a] F.-M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of KR'91, Cambridge (MA), USA*, pages 151–162, 1991.

[Donini *et al.*, 1991b] F.-M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of IJCAI'91, Sydney, Australia*, pages 458–463, 1991.

[Donini *et al.*, 1992] F.-M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. Marchetti Spaccamela. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53:309–327, 1992.

[Donini *et al.*, 1994] F.-M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation*, 4:1–30, 1994.

[Donini *et al.*, 1996] F.-M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, pages 191–236. CSLI Publications, Stanford (CA), USA, 1996.

[Donini *et al.*, 1997] F.-M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.

[Doyle and Patil, 1991] J. Doyle and R.S. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48(3):261–297, 1991.

[Ducournau and Habib, 1989] R. Ducournau and M. Habib. La multiplicité de l'héritage dans les langages à objets. *Technique et Science Informatiques*, 8(1):41–62, 1989.

[Ducournau *et al.*, 1995] R. Ducournau, M. Habib, M. Huchard, M.L. Mugnier, and A. Napoli. Le point sur l'héritage multiple. *Technique et Science Informatiques*, 14(3):309–345, 1995.

[Ducournau, 1996a] R. Ducournau. Des langages à objets aux logiques terminologiques : les systèmes classificatoires. Rapport de Recherche 96-030, LIRMM, Montpellier, 1996. `ftp://ftp.lirmm.fr/pub/LIRMM/papers/1996/RRI-Ducournau-96.ps`.

[Ducournau, 1996b] R. Ducournau. Les incertitudes de la classification incertaine. In Y. Dennebouy, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'96), Leysin, Suisse*, pages 183–200. Ecole Polytechnique Fédérale de Lausanne, 1996.

[Euzenat, 1993] J. Euzenat. Définition abstraite de la classification et son application aux taxonomies d'objets. In M. Habib and M. Oussalah, editors, *Actes de la Conférence Représentations Par Objets, La Grande Motte, France*, pages 235–246. EC2, Nanterre, 1993.

[Euzenat, 1994] J. Euzenat. Classification dans les représentations par objets : produits de systèmes classificatoires. In *Actes du 9e Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, Paris*, pages 185–196, 1994.

[Fitting, 1990] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, Berlin, 1990.

[Gaines, 1993] B. Gaines. A class library implementation of a principled open architecture knowldege representation server with plug-in data types. In *Proceedings of IJCAI'93, Chambéry, France*, pages 504–509, 1993.

[Garey and Johnson, 1979] M.R. Garey and D.S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, San Francisco (CA), USA, 1979.

[Gochet and Gribomont, 1991] P. Gochet and P. Gribomont. *Logique*, volume 1. Hermès, Paris, 1991.

[Hanschke, 1992] P. Hanschke. Specifying role interaction in concept languages. In *Proceedings of KR'92, Cambridge (MA), USA*, pages 318–329, 1992.

[Heinsohn *et al.*, 1994] J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68(2):367–397, 1994.

[Hollunder *et al.*, 1990] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proceedings of ECAI'90, Stockholm, Sweden*, pages 348–353, 1990.

[Hoppe *et al.*, 1993] T. Hoppe, C. Kindermann, J.J. Quantz, A. Schmiedel, and M. Fischer. Back V5 — Tutorial & manual. Kit-Report 100, Technische Universität, Berlin, 1993.

[Horrocks, 1995] I.R. Horrocks. A comparison of two terminological knowledge representation systems. Master of science, University of Manchester, UK, 1995.

[Hors and Rousset, 1995] P. Hors and M.-C. Rousset. Consistency of structured knowledge: A formal framework based on description logics. *Expert Systems With Applications*, 8(3):371–380, 1995.

[Hors, 1995] P. Hors. *Représentation et vérification de connaissances de structuration à l'aide de systèmes terminologiques.* Thèse, Université de Paris-Sud, Centre d'Orsay, 1995.

[Kaczmarek *et al.*, 1986] T.S. Kaczmarek, R. Bates, and G. Robins. Recent developments in Nikl. In *Proceedings of AAAI'86, Philadelphia (PA), USA*, pages 978–985, 1986.

[Kamp, 1996] G. Kamp. Using description logics for knowledge intensive case-based reasoning. In B. Faltings and I. Smith, editors, *Third European Workshop on Case-Based Reasoning (EWCBR'96), Lausanne, Switzerland*, Lecture Notes in Artificial Intelligence 1168, pages 204–218. Springer-Verlag, Berlin, 1996.

[Kietz and Morik, 1994] J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2):193–217, 1994.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[Kindermann, 1990] C. Kindermann. Class instances in a terminological framework — An experience report. Kit-Report 83, Technische Universität, Berlin, 1990.

[Kindermann, 1992] C. Kindermann. Retraction in terminological knowledge bases. In *Proceedings of ECAI'92, Vienna, Austria*, pages 420–424, 1992.

[Kortüm, 1993] G. Kortüm. How to compute 1 + 1? Kit-Report 103, Technische Universität, Berlin, 1993.

[Lenzerini and Schaerf, 1991] M. Lenzerini and A. Schaerf. Querying concept-based knowledge bases. In H. Boley and M.M. Richter, editors, *Proceedings of the Conference on Processing Declarative Knowledge (PDK'91), Kaiserslautern, Germany*, Lecture Notes in Computer Science 567, pages 107–123. Springer-Verlag, Berlin, 1991.

[Levesque and Brachman, 1987] H.J. Levesque and R.J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3(2):78–93, 1987.

[Levy and Rousset, 1996] A.Y. Levy and M.-C. Rousset. CARIN: A representation language combining Horn rules and description logics. In *Proceedings of ECAI'96, Budapest, Hungary*, pages 323–327, 1996.

[Lipkis, 1982] T. Lipkis. A KL-ONE classifier. In J.G. Schmolze and R.J. Brachman, editors, *Proceedings of the KL-ONE Workshop, Jackson, New Hampshire*, BBN Report 4842/Fairchild Technical Report 618, pages 126–143, 1982.

[LOOM, 1991] LOOM users guide (version 1.4). Information Science Institute, University of Southern California, Marina del Rey (CA), USA, 1991.

[Loveland, 1978] D.W. Loveland. *Automated Theorem Proving: A Logical Basis*. North-Holland, Amsterdam, 1978.

[MacGregor and Brill, 1992] R. MacGregor and D. Brill. Recognition algorithms for the LOOM classifier. In *Proceedings of AAAI'92, San Jose (CA), USA*, pages 774–779, 1992.

[MacGregor and Burstein, 1991] R. MacGregor and M.H. Burstein. Using a description classifier to enhance knowledge representation. *IEEE Expert*, 6(3):41–46, 1991.

[MacGregor *et al.*, 1992] R. MacGregor, D.B. Mcguiness, E. Mays, and T. Russ, editors. *Working Notes of the AAAI Fall Symposium on Issues in Description Logics: Users Meets Developers, Cambridge (MA), USA*, 1992.

[MacGregor, 1988] R. MacGregor. A deductive pattern matcher. In *Proceedings of AAAI'88, St Paul (MN), USA*, pages 403–408, 1988.

[MacGregor, 1991a] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In J.F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 385–400. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[MacGregor, 1991b] R. MacGregor. Inside the LOOM description classifier. *ACM SIGART Bulletin*, 2(3):88–92, 1991.

[Mark *et al.*, 1992] W. Mark, S. Tyler, J. McGuire, and J. Schlossberg. Commitment-based software development. *IEEE Transactions on Software Engineering*, 18(10):870–885, 1992.

[Masini *et al.*, 1989] G. Masini, A. Napoli, D. Colnet, D. Léonard, and K. Tombre. *Les langages à objets*. InterEditions, Paris, 1989. Traduit en anglais dans [Masini *et al.*,1991].

[Masini *et al.*, 1991] G. Masini, A. Napoli, D. Colnet, D. Léonard, and K. Tombre. *Object-Oriented Languages*. Academic Press, London, 1991. Traduction anglaise de [Masini *et al.*,1989].

[Mays *et al.*, 1991] E. Mays, R. Dionne, and R. Weida. K-REP system overview. *ACM SIGART Bulletin*, 2(3):93–97, 1991.

[McAllester, 1982] D.A. McAllester. Reasoning Utility Package user's manual. AI Memo 667, AI Lab, MIT, Cambridge (MA), USA, 1982.

[McGuiness, 1996] D.L. McGuiness. *Explaining Reasoning in Description Logics*. PhD thesis, Rutgers, The State University of New Jersey, USA, 1996.

[McGuinness and Borgida, 1995] D.L. McGuinness and A.T. Borgida. Explaining subsumption in description logics. In *Proceedings of IJCAI'95, Montréal, Canada*, pages 816–821, 1995.

[Napoli and Laurenço, 1993] A. Napoli and C. Laurenço. Représentations à objets et classification. conception d'un système d'aide à la planification de synthèses organiques. *Revue d'Intelligence Artificielle*, 7(2):175–221, 1993.

[Napoli and Volle, 1993] A. Napoli and P. Volle. Une introduction aux logiques terminologiques. Rapport de Recherche 93-R-033, Centre de Recherche en Informatique de Nancy, 1993.

[Nebel *et al.*, 1991] B. Nebel, C. Peltason, and K. von Luck, editors. *Proceedings of the International Workshop on Terminological Logics, Schloss Dagstuhl*. Document D-91-13, DFKI, Saarbrücken Universität, Germany, 1991.

[Nebel, 1988] B. Nebel. Computational complexity of terminological reasoning in Back. *Artificial Intelligence*, 34(3):371–383, 1988.

[Nebel, 1990a] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Lecture Notes in Artificial Intelligence 422. Springer-Verlag, Berlin, 1990.

[Nebel, 1990b] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, 1990.

[Nebel, 1991] B. Nebel. Terminological cycles: Semantics and computational properties. In J.F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[Nonnenmann and Eddy, 1992] U. Nonnenmann and J.K. Eddy. Kitss — A functional sotftware testing system using a hybrid domain model. In *Proceedings of the 8th IEEE Conference on Artificial Intelligence for Applications, Monterey (CA), USA*, pages 136–142, 1992.

[Owsnicki-Klewe, 1988] B. Owsnicki-Klewe. Configuration as a consistency maintenance task. In W. Hoeppner, editor, *Künstliche Intelligenz. Proceedings of the 12th German Workshop on Artificial Intelligence (GWAI'88), Eringerfeld, Germany*, pages 77–87. Springer-Verlag, Berlin, 1988.

[Padgham and Lambrix, 1994] L. Padgham and P. Lambrix. A framework for part-of hierarchies in terminological logics. In *Proceedings of KR'94, Bonn, Germany*, pages 485–496, 1994.

[Padgham and Nebel, 1993] L. Padgham and B. Nebel. Combining classification and nonmonotonic inheritance reasoning: A first step. In J. Komorowski and Z.W. Raś, editors, *Methodologies for Intelligent Systems*, Lecture Notes in Computer Science 689, pages 132–141. Springer-Verlag, Berlin, 1993.

[Padgham and Zhang, 1993] L. Padgham and T. Zhang. A terminological logic with defaults: A definition and an application. In *Proceedings of IJCAI'93, Chambéry, France*, pages 662–668, 1993.

[Padgham *et al.*, 1996] L. Padgham, E. Franconi, M. Gehrke, D.L. McGuinness, and P.F. Patel, editors. *Collected Papers of the 1996 International Description Logics Workshop, Cambridge (MA), USA*. AAAI Technical Report, 1996.

[Patel-Schneider and Swartout, 1993] P.F. Patel-Schneider and B. Swartout. Description logic specification from the Krss effort. Working version (draft), June 1993.

[Patel-Schneider *et al.*, 1984] P.F. Patel-Schneider, R.J. Brachman, and H.J. Levesque. ARGON: Knowledge representation meets information retrieval. In *Proceedings of the 1st IEEE Conference on Artificial Intelligence Applications, Denver (CO), USA*, pages 280–286, 1984.

[Patel-Schneider *et al.*, 1990] P.F. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. MacGregor, W.S. Mark, D.L. McGuinness, B. Nebel, A. Schmiedel, and J. Yen. Report on the workshop on term subsumption languages in knowledge representation. *The AI Magazine*, 11(2):16–22, 1990.

[Patel-Schneider, 1984] P.F. Patel-Schneider. Small can be beautiful in knowledge representation. Technical Report 660, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto (CA), USA, 1984.

[Patel-Schneider, 1989a] P.F. Patel-Schneider. A four-valued semantics for terminological logics. *Artificial Intelligence*, 38(3):319–351, 1989.

[Patel-Schneider, 1989b] P.F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2):263–272, 1989.

[Patel-Schneider, 1990] P.F. Patel-Schneider. Practical, object-based knowledge representation for knowledge-based systems. *Information Systems*, 15(1):9–19, 1990.

[Peltason *et al.*, 1989] C. Peltason, A. Schmiedel, C. Kindermann, and J. Quantz. The BACK system revisited. KIT-REPORT 75, Technische Universität, Berlin, 1989.

[Peltason *et al.*, 1991] C. Peltason, K. von Luck, and C. Kindermann, editors. *Proceedings of the Terminological Logic Users Workshop, Berlin, Germany.* KIT-REPORT 95, Department of Computer Science, Technische Universität, Berlin, 1991.

[Peltason, 1991] C. Peltason. The BACK system — An overview. *ACM SIGART Bulletin*, 2(3):114–119, 1991.

[Quantz and Royer, 1992] J. Quantz and V. Royer. A preference for defaults in terminological logics. In *Proceedings of KR'92, Cambridge (MA), USA*, pages 294–305, 1992.

[Quantz *et al.*, 1995] J.J. Quantz, G. Dunker, F. Bergmann, and I. Keller. The FLEX system. KIT-REPORT 124, Technische Universität, Berlin, 1995.

[Quantz, 1990] J. Quantz. Modeling and reasoning with defined roles in BACK. KIT-REPORT 84, Technische Universität, Berlin, 1990.

[Quantz, 1992] J. Quantz. How to fit generalized quantifiers into terminological logics. In *Proceedings of ECAI'92, Vienna, Austria*, pages 543–547, 1992.

[Quantz, 1993] J.J. Quantz. Interpretation as exception minimization. In *Proceedings of IJCAI'93, Chambéry, France*, pages 1310–1315, 1993.

[Resnick *et al.*, 1995] L.A. Resnick, A. Borgida, R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, and K.C. Zalondek. CLASSIC. *Description and Reference Manual for the Common Lisp Implementation (Version 2.3).* AT&T Bell Laboratories, Murray Hill (NJ), USA, 1995.

[Rich, 1991] C. Rich, editor. *SIGART Bulletin, (2)3, Special issue, "Implemented Knowledge Representation and Reasoning Systems"*, 1991.

[Rousset and Hors, 1996] M.-C. Rousset and P. Hors. Modeling and verifying complex objects: A declarative approach based on description logics. In *Proceedings of ECAI'96, Budapest, Hungary*, pages 328–332, 1996.

[Rousset *et al.*, 1997] M.-C. Rousset, R.J. Brachman, F.-M. Donini, E. Franconi, I. Horrocks, and A. Levy, editors. *Proceedings of the International Workshop on Description Logics (DL'97), Gif sur Yvette, France*. Laboratoire de Recherche en Informatique, Université de Paris-Sud, Centre d'Orsay, 1997.

[Royer and Quantz, 1992] V. Royer and J.J. Quantz. Deriving inference rules for terminological logics. In D. Pearce and G. Wagner, editors, *Proceedings of the European Workshop JELIA'92, Berlin, Germany*, Lecture Notes in Computer Science 633, pages 84–105. Springer-Verlag, Berlin, 1992.

[Royer and Quantz, 1993] V. Royer and J.J. Quantz. Deriving inference rules for description logics: A rewriting approach into sequent calculi. KIT-REPORT 111, Technische Universität, Berlin, 1993.

[Schild, 1989] K. Schild. Towards a theory of frames and rules. KIT-REPORT 76, Technische Universität, Berlin, 1989.

[Schmidt-Schauß and Smolka, 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Schmidt-Schauß, 1989] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R.J. Brachman, H.J. Levesque, and R. Reiter, editors, *Proceedings of KR'89, Toronto, Canada*, pages 421–431. Morgan Kaufmann Publishe, San Mateo (CA), USA, 1989.

[Schmiedel, 1990] A. Schmiedel. A temporal terminological logic. In *Proceedings of AAAI'90, Boston (MA), USA*, pages 640–645, 1990.

[Schmiedel, 1991a] A. Schmiedel. Modeling the medical data of the heart catheter report. Esprit Project Report 5210 (AIMS), Technische Universität, Berlin, 1991.

[Schmiedel, 1991b] A. Schmiedel. Using BACK in a medical domain. In C. Peltason, K. von Luck, and C. Kindermann, editors, *Proceedings of the Terminological Logic Users Workshop, Berlin, Germany*, pages 32–36. KIT-REPORT 95, Department of Computer Science, Technische Universität, Berlin, 1991.

[Schmiedel, 1992] A. Schmiedel. For a more expressive query language. In *AAAI Fall Symposium Series — Issues in Description Logics: Users Meets Developers, Cambridge (MA), USA*, pages 98–102, 1992.

[Schmitz and Quantz, 1993] B. Schmitz and J.J. Quantz. Defaults in machine translation. KIT-REPORT 106, Technische Universität, Berlin, 1993.

[Schmolze and Lipkis, 1983] J.G. Schmolze and T.A. Lipkis. Classification in the KL-ONE knowledge representation system. In *Proceedings of IJCAI'83, Karlsruhe, West Germany*, pages 330–332, 1983.

[Schmolze and Mark, 1991] J.G. Schmolze and W.S. Mark. The NIKL experience. *Computational Intelligence*, 6:48–69, 1991.

[Sheth *et al.*, 1993] A.P. Sheth, S.K. Gala, and S.B. Navathe. On automatic reasoning for schema integration. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):23–50, 1993.

[Speel, 1995] P.H. Speel. *Selecting Knowledge Representation Systems*. PhD thesis, Universiteit Twente, The Netherlands, 1995.

[Straccia, 1993] U. Straccia. Default inheritance reasoning in hybrid KL-ONE-style logics. In *Proceedings of IJCAI'93, Chambéry, France*, pages 676–681, 1993.

[Vilain, 1985] M. Vilain. The restricted language architecture of a hybrid representation system. In *Proceedings of IJCAI'85, Los Angeles (CA), USA*, pages 547–551, 1985.

[Vismara, 1996] P. Vismara. Appariements dirigés pour le raisonnement par classification sur des hiérarchies de graphes. In Y. Dennebouy, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'96), Leysin, Suisse*, pages 201–214. Ecole Polytechnique Fédérale de Lausanne, 1996.

[Volle, 1992] P. Volle. Présentation du projet ESPRIT P5210 — AIMS. *Bulletin de l'AFIA*, 8:29–33, 1992.

[von Luck *et al.*, 1987] K. von Luck, B. Nebel, C. Peltason, and A. Schmiedel. The anatomy of the BACK system. KIT-REPORT 41, Technische Universität, Berlin, 1987.

[Weida and Litman, 1992] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In *Proceedings of KR'92, Cambridge (MA), USA*, pages 282–293, 1992.

[Winograd, 1975] T. Winograd. Frame representation and the declarative/procedural controversy. In D.G. Bobrow and A. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 185–210. Academic Press, New York (NY), USA, 1975.

[Woods and Schmolze, 1992] W.A. Woods and J.G. Schmolze. The KL-ONE family. *Computers & Mathematics With Applications*, 23(2–5):133–177, 1992.

[Woods, 1991] W.A. Woods. Understanding subsumption and taxonomy: A framework for progress. In J.F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 45–94. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[Wright *et al.*, 1993] J.R. Wright, E.S. Weixelbaum, G.T. Vesonder, K.E. Brown, S.R. Palmer, J.I. Berman, and H.H. Moore. A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T Network Systems. *The AI Magazine*, 14(3):69–80, 1993.

[Yen *et al.*, 1991a] J. Yen, H.-L. Juang, and R. MacGregor. Using polymorphism to improve expert system maintainability. *IEEE Expert*, 6(2):48–55, April 1991.

[Yen *et al.*, 1991b] J. Yen, R. Neches, and R. MacGregor. CLASP: Integrating term subsumption systems and production systems. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):25–32, 1991.

Index

ABox, 20, 24, 44
$\mathcal{AL}$, 10, 27, 50
    grammaire, 10
$\mathcal{ALC}$, 11
$\mathcal{ALCNR}$, 12, 16, 26, 31, 39, 40, 45–47, 51
    base terminologique, 23, 24
    interprétation, 13
    règles de décomposition, 31, 34
$\mathcal{AL}_{\delta\epsilon}$, 48
$\mathcal{ALE}$, 11, 57
algorithme de classification, 36
algorithme de subsomption
    $\mathcal{ALNR}$, 20, 53
    $\mathcal{FL}$-, 18
    complétude, 18
    complexité, 26
    correction, 18, 26
    NC, 18, 53, 57
$\mathcal{ALN}$, 11, 27
$\mathcal{ALNR}$
    algorithme de subsomption, 20, 53
$\mathcal{ALR}$, 11
$\mathcal{ALU}$, 11
apprentissage, 17
assertion, 20, 41
    satisfiabilité, 23
attachement procédural, 47
attribut, 25, 47

BACK, 25, 28, 44, 45, 49
base de données, 39, 43, 45, 49, 50
base terminologique, 20
    modèle, 22, 24
    satisfiabilité, 23, 26, 29, 34

cardinalité (rôle), 8, 11, 57
circuit terminologique, 17, 24, 34
classe, 47
CLASSIC, 17, 26, 28, 37, 44, 47, 49, 50, 53
    fonction test, 37
    grammaire, 37
    règles, 37
    traitement des requêtes, 39
classification, 25, 36
    algorithme, 36
co-domaine
    rôle, 8, 11, 25
    restriction, 12, 54
complétude
    algorithme de subsomption, 18
    subsomption, 26
complexité
    algorithme de subsomption, 26
    raisonnement terminologique, 26, 49, 50
concept, 8
    défini, 8
    extension, 8, 20
    forme simple, 30
    intension, 20
    primitif, 8
    satisfiabilité, 14, 26
conjonction
    de concepts, 9
    de rôles, 12
constructeur, 8–10
contrainte, 30, 34
correction
    algorithme de subsomption, 18
    subsomption, 26

décidabilité (subsomption), 28
disjonction de concepts, 11
domaine (rôle), 25

équation terminologique, 17
équivalence de concepts, 14
exception (à l'héritage), 48
expression conceptuelle, 10
extension (concept), 8, 20

$\mathcal{FL}$, 12, 15
$\mathcal{FL}$-, 12, 15, 27, 51
    algorithme de subsomption, 18
FLEX, 45, 53
$\mathcal{FLN}$, 27
forme simple (concept), 30

frames, 47

grammaire
  $\mathcal{AL}$, 10
  CLASSIC, 37
  LOOM, 40

héritage, 47
hiérarchie, 9, 16, 25, 36
  d'héritage, 47
hypothèse
  du monde clos, 21
  du monde ouvert, 21
  du nom unique, 21

inclusion
  concept, 24, 34
  rôle, 25
incompatibilité entre concepts, 14
incomplétude (subsomption), 27
instance, 8
instanciation, 21, 26
  LOOM, 40
intension (concept), 20
interprétation, 13
introduction
  concept, 24
  rôle, 25
Italie, 8

KANDOR, 44
KL-ONE, 7, 27, 36, 44
K-REP, 45
KRIS, 28, 45
KRSS, 12
KRYPTON, 44

logique classique, 46
LOOM, 25, 26, 28, 40, 45, 47, 49, 50
  grammaire, 40
  instanciation, 40
  traitement des assertions, 41

masquage (héritage), 48
MESON, 44, 45
modèle
  base terminologique, 22, 24
modus ponens, 56

négation de concepts, 10, 11
NIKL, 44
normalisation des concepts, 18, 53
$\mathcal{NTF}$, 53

$\mathcal{PL}_1$, 27, 51
$\mathcal{PL}_2$, 28, 51

quantification existentielle, 10, 11
quantification universelle, 10

raisonnement
  à partir de cas, 17, 43, 50
  par classification, 36
  par défaut, 48
raisonnement terminologique, 25, 35
  complexité, 26, 49, 50
réflexe (RCO), 47
règles
  CLASSIC, 37
  de décomposition (tableaux sémantiques), 28, 31, 34
requête (CLASSIC), 39
restriction
  co-domaine de rôle, 12, 54
  numérique (rôle), 11, 57
rôle, 8
  cardinalité, 8, 11
  co-domaine, 8, 11, 25
  domaine, 25
  inclusion, 25
  introduction, 25
  restriction de co-domaine, 12, 54
  restriction numérique, 11
  valeur, 8

satisfiabilité
  assertion, 23
  base terminologique, 23, 26, 29, 34
  concept, 14, 26
sémantique, 13
  algébrique, 45, 49
  descriptive, 48
Site Web sur les logiques de descriptions, 52
spécialisation, 47
subsomption, 15, 17, 25

$\mathcal{ALNR}$, 20, 53
$\mathcal{FL}$-, 18
algorithme NC, 18, 53, 57
complétude, 18, 26
correction, 18, 26
décidabilité, 28
incomplétude, 27
système équationnel, 45, 49
système équationnel
subsomption, 45, 49
système de contraintes, 29
système de RCO, 47

tableaux sémantiques, 18, 28, 57
règles de décomposition, 28, 31, 34
TBox, 24, 44
treillis des descriptions, 16

valeur
par défaut, 48
rôle, 8
variable, 30

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399