



IDDL Formal Specifications Un état de l'art sur les logiques de description

—
Délivrable n°2

Salem BENFERHAT – benferhat@cril.univ-artois.fr
CRIL – Université d'Artois

Safa YAHY – yahi@cril.univ-artois.fr
CRIL – Université d'Artois

Benjamin MORIN – benjamin.morin@supelec.fr
Supélec Rennes



Projet PLACID
Livrable 2 : IDDL Formal Specifications
Un état de l'art sur les logiques de description

Salem BENFERHAT Safa YAHY
benferhat@cril.univ-artois.fr yahi@cril.univ-artois.fr

Benjamin MORIN
benjamin.morin@supelec.fr

5 janvier 2008

Table des matières

1	Introduction	2
2	Définition du formalisme de base	3
2.1	Langages de description	3
2.1.1	Le langage de description basique \mathcal{AL}	3
2.1.2	La famille des langages \mathcal{AL}	4
2.1.3	Les langages de description et la logique des prédicats du premier ordre	5
2.2	Les terminologies TBox	6
2.2.1	Axiomes terminologiques	6
2.2.2	Définitions	7
2.2.3	Terminologies définitoriales	7
2.2.4	Terminologies acycliques et terminologies cycliques	8
2.2.5	Terminologies avec axiomes d'inclusion	8
2.3	Descriptions du monde ABox	9
3	Algorithmes de raisonnement	10
3.1	Algorithmes de subsomption structurelle	11
3.2	Algorithmes de tableaux	12
4	Conclusion	15

1 Introduction

Les logiques de description forment une famille de langages de représentation de connaissance qui peuvent être utilisées pour représenter la connaissance terminologique d'un domaine d'application d'une façon structurée et formelle. Ces logiques sont issues de modèles graphiques de représentation de connaissances notamment les réseaux sémantiques qui dérivent à leur tour des modèles à mémoire sémantique de Quillian. En général, dans un réseau sémantique on distingue des concepts (dénotés par des noeuds génériques), des individus (dénotés par des noeuds d'individus), des liens classe-fille/classe-mère et des liens de propriétés.

En utilisant les liens classe-fille/classe-mère, les concepts peuvent être classés dans un hiérarchie et via les liens de propriétés, les propriétés peuvent être assignées aux concepts.

Toutefois, par manque de sémantique précise, les liens classe-fille/classe-mère peuvent être interprétés de différentes manières.

Afin de palier cette ambiguïté, d'autres formalismes ont été développés tels que les réseaux d'héritage structuré qui ont été implémentés dans le système KL-ONE [2]. Par la suite, KL-ONE a été doté d'une sémantique "Tarsky Style" ce qui a fixé de manière précise la signification des ses constructeurs graphiques et a conduit à la définition de la première logique de description [5], appelée aussi à cette époque langages terminologiques, langages conceptuels ou encore langage basés KL-ONE.

D'un point de vue pratique, les logiques de description sont assez prometteuses. En fait, elles ont été utilisées dans l'implémentation de nombreuses applications dans divers domaines.

Le génie logiciel, par exemple, a été un des premiers domaines d'application pour les logiques de description. L'idée de base était d'utiliser la logique de description afin d'implémenter un système qui puisse assister le développeur à retrouver des informations concernant un logiciel d'une taille importante. Le système LASSIE est une des applications les plus référencées dans ce contexte.

D'autre part, l'utilisation des logiques de description en configuration a connu un grand succès. La configuration revient à trouver un ensemble de composants pouvant être convenablement connectées en vue d'implémenter un système donné. Lorsque le nombre et la connectivité des composants deviennent importants, la configuration devient assez complexe. Dans ce cas, la logique de description permet de classer les composants et de les organiser dans une taxonomie.

Par ailleurs, et sans être exhaustif, on peut citer d'autres domaines tels que le web sémantique pour la représentation d'ontologies et la recherche d'information basée sur la logique, la médecine où l'objectif est la construction et la maintenance de très grandes ontologies médicales, les bibliothèques numériques, les systèmes d'information basés web, le traitement du langage naturel et la gestion de bases de données.

Enfin, on cite quelques moteurs d'inférences basés sur les DLs tels que FaCT [4], Racer [3], Pellet [8], FaCT++ [9], F-OWL [10].

2 Définition du formalisme de base

Une base de connaissances (KB pour Knowledge Representation) basée sur la logique de description comprend deux composant, la TBox et la ABox. La TBox introduit la terminologie, i.e, le vocabulaire du domaine d'une application, alors que la ABox contient des assertions quant à des individus nommés en termes de ce vocabulaire.

Le vocabulaire consiste en des concepts, qui dénotent des ensembles d'individus, et des rôles qui dénotent des relations binaires entre des individus. En plus des concepts atomiques et des rôles atomiques (les noms des concepts et des rôles), tous les systèmes DL permettent de construire des descriptions de concepts et de rôles plus complexes. Le langage de construction de descriptions constitue une caractéristique du système en question.

2.1 Langages de description

Les descriptions élémentaires sont les concepts atomiques et les rôles atomiques à partir desquels des descriptions complexes peuvent être générées via les constructeurs de concepts et les constructeurs de rôles. Dans ce qui suit, on utilisera les lettres A et B pour désigner des concepts atomiques, la lettre R pour un rôle atomique et les lettres C et D pour les descriptions de concepts. Les langages de description sont distingués par rapport aux constructeurs qu'ils fournissent. Le langage \mathcal{AL} (pour *Attributive Langage*) a été introduit dans [7] comme un langage minimal ayant un intérêt pratique. Les autres langages de la famille \mathcal{AL} en sont une extension.

2.1.1 Le langage de description basique \mathcal{AL}

Les descriptions de concepts dans le langage \mathcal{AL} sont générées selon la règle syntaxique suivante :

$C, D \rightarrow$	A		(concept atomique)
	\top		(concept universel)
	\perp		(concept bottom)
	$\neg A$		(negation atomique)
	$C \sqcap D$		(intersection)
	$\forall R.C$		(restriction de valeur)
	$\exists R.\top$		(quantificateur existentiel limité)

Il est à noter, que dans le langage \mathcal{AL} , la négation n'est appliquée qu'aux concepts atomiques et seulement le concept universel est permis à la portée du quantificateur existentiel. D'autre part, le sous langage de \mathcal{AL} obtenu en supprimant la négation atomique est appelé \mathcal{FL}^- alors que le sous langage de \mathcal{FL}^- obtenu en supprimant le quantificateur existentiel limité est appelé \mathcal{FL}_0 .

L'exemple suivant nous donne une idée sur ce qui peut être exprimé dans le langage \mathcal{AL} :

Exemple 1 Soient **Person** et **Female** deux concepts atomiques. Donc $\text{Person} \sqcap \text{Female}$ et $\text{Person} \sqcap \neg \text{Female}$ sont des concepts décrivant intuitivement les personnes qui sont femelle et les personnes qui ne sont pas femelle. En outre, étant donné un rôle atomique **hasChild**, on peut construire les concepts $\text{Person} \sqcap \exists \text{hasChild}.\top$ et $\text{Person} \sqcap \forall \text{hasChild}.\text{Female}$ dénotant les personnes qui ont un enfant et les personnes dont tous les enfants sont une femelle. Enfin, en utilisant le concept bottom, on peut également décrire les personnes qui n'ont pas d'enfants par $\text{Person} \sqcap \forall \text{hasChild}.\perp$

En vue de définir une sémantique formelle des concepts- \mathcal{AL} , on considère une interprétation \mathcal{I} définie par la donnée d'un ensemble non vide $\Delta^{\mathcal{I}}$ (le domaine d'interprétation) et d'une fonction d'interprétation, qui assigne à chaque concept atomique A un ensemble $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ et à chaque rôle atomique R une relation binaire $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Une fonction d'interprétation est étendue aux descriptions de concepts via la définition inductive suivante :

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\perp^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} - A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b, (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b, (a, b) \in R^{\mathcal{I}}\}
\end{aligned}$$

Deux concepts C et D sont dits équivalents, $C \equiv D$, si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$ pour toute interprétation \mathcal{I} . Par exemple, on peut aisément vérifier que les concepts $\forall \text{hasChild}.\text{Female} \sqcap \forall \text{hasChild}.\text{Student}$ et $\forall \text{hasChild}.\text{(Female} \sqcap \text{Student)}$ sont équivalents.

2.1.2 La famille des langages \mathcal{AL}

D'autres langages, plus expressifs, peuvent être définis en rajoutant d'autres constructeurs au langage \mathcal{AL} à savoir :

- L'union de concepts, désignée par la lettre \mathcal{U} , notée par $C \sqcup D$ et interprétée par :

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}.$$

- La quantification existentielle complète, désignée par la lettre \mathcal{E} , notée par $\exists R.C$ et interprétée par :

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b, (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}.$$

- Les restrictions de nombres, désignées par la lettre \mathcal{N} et notées par $\geq nR$ (restriction au moins) et par $\leq nR$ (restriction au plus) où n représente un entier positif. Ces restrictions de valeurs sont interprétées respectivement comme suit :

$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \geq n\},$$

et

$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \leq n\},$$

où " $|\cdot|$ " dénote le cardinal d'un ensemble donné.

- La négation de concepts arbitraires, désignée par la lettre (C) , notée par $\neg C$ et interprétée de la façon suivante :

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - (C)^{\mathcal{I}}.$$

Exemple 2 Avec ces nouveaux constructeurs, on peut par exemple décrire les personnes ayant soit pas plus d'un enfant soit au moins trois enfants dont un est une femelle comme suit :

$$\text{Person} \sqcap (\leq 1 \text{ hasChild} \sqcup (\geq 3 \text{ hasChild} \sqcap \exists \text{hasChild.Female})).$$

Etendre le langage \mathcal{AL} par n'importe quel sous ensemble des constructeurs précédents génère un langage \mathcal{AL} particulier désigné par une chaîne de caractères de la forme :

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}],$$

où une lettre dans l'appellation reflète la présence du constructeur correspondant. Par exemple, $\mathcal{AL}\mathcal{E}\mathcal{N}$ est l'extension de \mathcal{AL} par la quantification existentielle complète et les restrictions de nombres.

D'un point de vue sémantique, on a $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ et $\exists R.C \equiv \neg \forall R. \neg C$. Par conséquent, l'union et la quantification universelle complète peuvent être exprimées via la négation et vice versa. Ainsi, on utilisera la lettre \mathcal{C} au lieu des lettres \mathcal{UE} dans les noms de langages. Par exemple, on écrira $\mathcal{AL}\mathcal{C}\mathcal{N}$ au lieu de $\mathcal{AL}\mathcal{UE}\mathcal{N}$.

2.1.3 Les langages de description et la logique des prédicats du premier ordre

La sémantique des concepts identifie les langages de description comme étant des fragments de la logique des prédicats du premier ordre. En fait, puisqu'une interprétation \mathcal{I} assigne à chaque concept atomique et à chaque rôle une relation unaire et une relation binaire sur $\Delta^{\mathcal{I}}$ respectivement, les concepts et les rôles peuvent être vus comme étant des prédicats unaires et binaires. Donc, n'importe quel concept C peut être traduit par une formule logique $\phi_C(x)$ avec une seule variable libre x tel que pour toute interprétation \mathcal{I} , l'ensemble des éléments de $\Delta^{\mathcal{I}}$ qui satisfont $\phi_C(x)$ est exactement $C^{\mathcal{I}}$:

- Un concept atomique A est traduit par la formule $A(x)$;
- Les constructeurs intersection, union et négation sont traduits par la conjonction logique, la disjonction et la négation respectivement ;
- si $\phi_C(x)$ est la traduction de C et R est un rôle atomique, alors la restriction de valeurs et la quantification universelle sont capturées par les formules :

$$\phi_{\exists R.C}(y) = \exists x.R(y, x) \wedge \phi_C(x)$$

$$\phi_{\forall R.C}(y) = \forall x.R(y, x) \rightarrow \phi_C(x)$$

où y est une nouvelle variable ;

– Enfin, les restrictions de nombres sont exprimées par les formules :

$$\phi_{\geq n R}(x) = \exists y_1, \dots, y_n.R(x, y_1) \wedge \dots \wedge R(x, y_n) \wedge \bigwedge_{i < j} y_i \neq y_j$$

$$\phi_{\leq n R}(x) = \exists y_1, \dots, y_{n+1}.R(x, y_1) \wedge \dots \wedge R(x, y_{n+1}) \rightarrow \bigvee_{i < j} y_i = y_j$$

Etant donné que les concepts soient traductibles en logique des prédicats, on peut dire qu'une syntaxe spéciale est dénuée d'intérêts. Néanmoins, les translations précédentes montrent, en particulier pour la restriction de nombres, que la syntaxe à variable free des logiques de description est vraiment plus concise. Ceci facilite également le développement d'algorithmes.

2.2 Les terminologies TBox

On a vu comment former des descriptions de concepts complexes. Maintenant, on introduit les axiomes terminologiques. Après, on considère les définitions comme étant des axiomes spécifiques et on identifie les terminologies comme un ensemble de définitions par lequel on introduit des concepts atomiques comme des abréviations ou des noms pour des concepts complexes.

2.2.1 Axiomes terminologiques

En général, les axiomes terminologiques sont de la forme

$$C \sqsubseteq D(R \sqsubseteq S)$$

ou

$$C \equiv D(R \equiv S)$$

où C, D sont des concepts (et R, S son des rôles). Les axiomes du premier type sont appelés inclusions alors que ceux du second type sont dits égalités. Pour simplifier, nous traiterons par la suite uniquement les axiomes liés aux concepts.

Sémantiquement, une interprétation \mathcal{I} satisfait une inclusion $C \sqsubseteq D$ si et seulement si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ et satisfait une égalité $C \equiv D$ si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$. Si Σ est un ensemble d'axiomes, alors \mathcal{I} satisfait Σ si et seulement si \mathcal{I} satisfait chaque élément de Σ . Si \mathcal{I} satisfait un axiome (resp. un ensemble d'axiomes), alors \mathcal{I} est dite modèle de cet axiome (resp. l'ensemble d'axiomes). Deux axiomes ou deux ensembles d'axiomes sont équivalents si et seulement s'ils ont les mêmes modèles.

2.2.2 Définitions

Une égalité dont le membre gauche est un concept atomique est une définition. Les définitions sont utilisées pour affecter des noms symboliques à des descriptions complexes. Par exemple, par l'axiome :

$$\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild}.\text{Person}$$

on associe à la description $\text{Woman} \sqcap \exists \text{hasChild}.\text{Person}$ le nom **Mother**. Les noms symboliques peuvent être utilisés comme des abréviations dans d'autres descriptions. Si, par exemple, on a défini le concept **Father** par analogie au concept **Mother**, on peut définir le concept **Parent** comme suit :

$$\text{Parent} \equiv \text{Mother} \sqcup \text{Father}.$$

On appelle un ensemble de définitions Σ une terminologie ou une TBox si aucun nom symbolique n'est défini plus d'une fois, i.e, pour chaque concept atomique A , il existe au plus un axiome dans Σ dont le membre gauche est A .

Exemple 3 *La terminologie suivante exprime des concepts liés à des relations familiales :*

$$\begin{aligned} \text{Woman} &\equiv \text{Person} \sqcap \text{Female} \\ \text{Man} &\equiv \text{Person} \sqcap \neg \text{Woman} \\ \text{Mother} &\equiv \text{Woman} \sqcap \exists \text{hasChild}.\text{Person} \\ \text{Father} &\equiv \text{Man} \sqcap \exists \text{hasChild}.\text{Person} \\ \text{Parent} &\equiv \text{Mother} \sqcup \text{Father} \\ \text{GrandMother} &\equiv \text{Mother} \sqcap \exists \text{hasChild}.\text{Parent} \\ \text{MotherWithoutDaughter} &\equiv \text{Mother} \sqcap \forall \text{hasChild}.\neg \text{Woman} \end{aligned}$$

2.2.3 Terminologies définitoriales

Etant donnée une terminologie Σ . On divise les concepts atomiques y figurant en deux ensembles, les symboles de nom ou les symboles définis N_Σ qui figurent dans la partie gauche d'un certain axiome et les symboles de base ou les symboles primitifs B_Σ qui figurent uniquement dans les parties droites des axiomes. Normalement, une terminologie définit les symboles de nom en termes des symboles de base.

Une interprétation de base pour Σ est une interprétation qui interprète uniquement les symboles de base. Soit \mathcal{J} une telle interprétation. Une interprétation \mathcal{I} qui interprète également les symboles de nom est une extension de \mathcal{J} si elle a le même domaine que \mathcal{J} , i.e, $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, et interprète les symboles de base de la même manière que \mathcal{J} .

On dit que Σ est définitorial si chaque interprétation de base admet exactement une extension qu'est un modèle de Σ . Autrement dit, si on connaît l'interprétation des symboles de base, et Σ est définitoriale, alors le sens des concepts de nom est complètement défini.

2.2.4 Terminologies acycliques et terminologies cycliques

Voir si une terminologie est définitoriale ou non est lié à voir si ses définitions contiennent des cycles ou non. Par exemple, la définition

$$\text{Human} \equiv \text{Animal} \sqcap \forall \text{hasParent} . \text{Human}$$

contient un cycle. En général, les cycles dans une terminologie Σ sont définis comme suit. Soient A et B deux concepts atomiques dans Σ . On dit que A 'utilise directement' B si B apparaît dans le membre droit de la définition de A , et on appelle 'utilise' la fermeture transitive de la relation 'utilise directement'. Donc une terminologie Σ contient un cycle si et seulement s'il existe un concept atomique dans Σ qui utilise lui même. Autrement, Σ est dite acyclique.

Si Σ est acyclique, alors elle est définitoriale. La raison est qu'on peut effectuer itérativement l'expansion des définitions de Σ en remplaçant chaque occurrence d'un symbole de nom dans la partie droite d'une définition par les concepts qu'il représente. Comme il n'a pas de cycles, le processus se termine en générant une terminologie Σ' contenant des définitions de la forme $A \equiv C'$, où C' contient uniquement des symboles de base et non pas des symboles de nom. Σ' est dite expansion de Σ et sa taille peut être exponentielle par rapport à celle de Σ et elle lui est équivalente.

Par contre, les extensions uniques peuvent ne pas exister si une terminologie contient des cycles.

Néanmoins, il existe des terminologies avec cycles qui sont définitoriales. Par exemple, soit l'axiome

$$A \equiv \forall R . B \sqcup \exists R . (A \sqcap \neg A)$$

qui contient un cycle. Toutefois, comme $\exists R . (A \sqcap \neg A)$ est équivalent au concept bottom, l'axiome précédent est équivalent à l'axiome acyclique

$$A \equiv \forall R . B.$$

D'autre part, on montre que toute terminologie \mathcal{ALC} définitoriale est équivalente à une terminologie acyclique.

Selon la sémantique que nous avons considérée plus haut, qui est essentiellement celle des logiques des prédicats du premier ordre, les terminologies sont définitoriales seulement si elle sont essentiellement acycliques. Les sémantiques du point fixe sont motivées par le fait qu'il existe des situations où intuitivement des définitions cycliques sont significatives et l'intuition peut être capturée par les sémantiques du point fixe.

2.2.5 Terminologies avec axiomes d'inclusion

Des fois, on se trouve incapables de définir certains concepts d'une manière complète. Dans ce cas, on peut établir des conditions nécessaires pour l'appartenance de concepts en utilisant l'inclusion. On appelle une inclusion où le membre gauche est un concept atomique une spécialisation. Par exemple

Woman \sqsubseteq Person.

Si on permet les spécialisations dans une terminologie, cette dernière perd son caractère définitorial même si elle est acyclique. Une telle terminologie est dite généralisée.

Une terminologie généralisée Σ peut être transformée en une terminologie régulière Σ' contenant uniquement des définitions. La terminologie Σ' est obtenue à partir de Σ en attribuant pour chaque spécialisation $A \sqsubseteq C$ un nouveau concept de base \overline{A} et en remplaçant la spécialisation $A \sqsubseteq C$ par la définition $A \equiv \overline{A} \sqcap C$. La terminologie Σ' est appelée la normalisation de Σ .

La normalisation de la terminologie généralisée

Woman \sqsubseteq Person

est la terminologie

Woman $\equiv \overline{Woman} \sqcap Person$.

Intuitivement, le nouveau symbole de base \overline{Woman} correspond aux qualités qui différencient les femmes des autres personnes.

2.3 Descriptions du monde ABox

Outre la terminologie ou la TBox, le second composant d'une base de connaissance est la description du monde ou la ABox.

La ABox décrit un état spécifique du domaine d'application en termes de concepts et de rôles. En fait, dans une ABox, on introduit des individus (en leur donnant des noms) ainsi que leur propriétés. Les individus sont notés par a, b, c . Soit C un concept et R un rôle. On peut former des assertions comme suit : $C(a)$ et $R(b, c)$.

La première assertion est dite assertion de concept, elle signifie que a appartient à (l'interprétation de) C tandis que la deuxième, qui est appelée assertion de rôle, elle signifie que c est un remplisseur du rôle R pour b .

Par exemple, si PETER, PAUL et MARY sont des noms d'individus, alors **Father**(PETER) signifie que PETER est un père et **hasChild**(MARY, PAUL) signifie que PAUL est enfant de MARY. Une ABOX notée \mathcal{A} est un ensemble fini de telles assertions.

Une ABox peut être vue comme étant une instance d'une base de données relationnelle avec seulement des relations unaires et binaires. Toutefois, contrairement à la sémantique du monde clos des bases de données classiques, la sémantique des ABox est une sémantique monde ouvert étant donné que les systèmes de représentation de connaissances sont appliqués dans des situations où l'on peut supposer que l'information est incomplète.

La sémantique des ABox est définie par l'extension des interprétations aux noms d'individus. Donc une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ n'assigne pas uniquement des ensembles et des relations binaires aux concepts et aux rôles mais aussi

assigne à chaque nom d'individu a un élément $a^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$. On suppose que des noms d'individus distincts dénotent des objets différents. Par conséquent, l'interprétation doit satisfaire l'assomption de nom unique UNA (pour unique name assumption), i.e, si a et b sont des noms différents alors $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

D'autre part, une interprétation \mathcal{I} satisfait l'assertion de concept $C(a)$ si $a^{\mathcal{I}} \in C^{\mathcal{I}}$. Elle satisfait l'assertion de rôle $R(a, b)$ si $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ et elle satisfait une ABox \mathcal{A} si elle satisfait chaque assertion dans \mathcal{A} . Dans ce cas, \mathcal{I} est dite modèle de l'assertion ou de la ABox. Enfin, une interprétation \mathcal{I} satisfait une assertion ou une ABox par rapport à une terminologie Σ si en plus d'être modèle de l'assertion ou de la base, elle est également modèle de la terminologie Σ .

3 Algorithmes de raisonnement

Intuitivement, au lieu de concevoir de nouveaux algorithmes en DLs, on peut essayer de réduire le problème qu'on veut résoudre à un problème d'inférence connu en logique classique. Par exemple, la décidabilité du problème d'inférence en logique \mathcal{ALC} peut être obtenu en ayant la décidabilité du langage \mathcal{L}^2 (un fragment de la logique des prédicats du premier ordre à deux variables) en sachant que tout concept \mathcal{ALC} peut être traduit dans le langage \mathcal{L}^2 . Prenons l'exemple suivant :

Exemple 4 Une traduction directe du concept $\forall R.(\exists R.A)$ génère la formule

$$\forall y.(R(x, y) \rightarrow (\exists z.(R(y, z) \wedge A(z)))).$$

Comme la sous formule $\exists z.(R(y, z) \wedge A(z))$ ne contient pas la variable x , cette dernière peut être réutilisée à la place de z . Ce renommage génère la formule équivalente suivante :

$$\forall y.(R(x, y) \rightarrow (\exists x.(R(y, x) \wedge A(x))))$$

qui est utilisée uniquement deux variables.

Toutefois, la complexité des procédures de décision obtenues ainsi est souvent plus élevée qu'il en faut : par exemple, le problème de satisfiabilité de la logique \mathcal{L}^2 est NEXPTIME-complet alors que le problème de satisfiabilité des descriptions concepts \mathcal{ALC} est "uniquement" PSPACE-complet. Ceci montre l'intérêt de développer de nouveaux algorithmes spécifiques aux logiques de description.

D'autre part, tous les problèmes d'inférence peuvent être réduits au problème de satisfiabilité en sachant que le langage de description en question permet la négation et la disjonction.

Toutefois, certains langages de description ne le permettent pas. Pour de tels langages, la subsomption de concepts peut être calculée par des algorithmes de subsomption structurelle, i.e, des algorithmes qui comparent la structure syntaxique des descriptions de concepts.

Bien que ces algorithmes soient très efficaces, ils ne sont complets que pour des langages simples peu expressifs. En particulier, les langages de description

avec négation et disjonction ne peuvent être traités par ce type d'algorithmes. Pour de tels langages, les algorithmes basés-tableau se trouvent très utiles. Le premier algorithme basé-tableau en logique de description a été proposé par [7] dans l'optique de tester la satisfiabilité des concepts \mathcal{ALC} . Depuis, cette approche a été utilisée pour tester la satisfiabilité de nombreuses logiques de description qui étendent \mathcal{ALC} .

3.1 Algorithmes de subsomption structurelle

Ces algorithmes opèrent en général en deux phases :

1. la normalisation des descriptions en question ;
2. la comparaison des structures syntaxiques des formes normales.

En vue de simplifier les choses, on présente cette approche dans le cadre du langage simple \mathcal{FL}_0 . Ensuite, on montre comment traiter le concept bottom (\perp), la négation atomique ($\neg A$) et les restriction de nombres ($\leq nR$ et $\geq nR$).

La description d'un concept \mathcal{FL}_0 est sous forme normale si et seulement si elle est de la forme :

$$A_1 \sqcap \dots \sqcap A_m \sqcap R_1.C_1 \sqcap \dots \sqcap R_n.C_n$$

où $A_1 \dots A_m$ sont des concepts de noms distincts, $R_1.C_1 \dots R_n.C_n$ sont des noms de rôles distincts, et $C_1 \dots C_n$ sont des descriptions de concepts \mathcal{FL}_0 sous forme normale.

Il est à noter que toute description peut être transformée en une forme normale en utilisant l'associativité, la commutativité, l'idempotence de \sqcap et le fait que les descriptions $\forall R.(C \sqcap D)$ et $(\forall R.C) \sqcap (\forall R.D)$

Proposition 1 *Soient*

$$A_1 \sqcap \dots \sqcap A_m \sqcap R_1.C_1 \sqcap \dots \sqcap R_n.C_n$$

la forme normale de la description du concept \mathcal{FL}_0 C , et

$$B_1 \sqcap \dots \sqcap B_k \sqcap S_1.D_1 \sqcap \dots \sqcap S_l.D_l$$

la forme normale de la description du concept \mathcal{FL}_0 D .

Alors, $C \sqsubseteq D$ si et seulement si les deux conditions suivantes sont vérifiées :

1. *pour tout $i, 1 \leq i \leq k$, il existe $j, 1 \leq j \leq m$ tel que $A_j \sqsubseteq B_i$*
2. *pour tout $i, 1 \leq i \leq l$, il existe $j, 1 \leq j \leq n$ tel que $S_i = R_j$ et $C_j \sqsubseteq D_i$.*

Cette caractérisation de subsomption est saine et complète. Elle permet de définir un algorithme récursif polynomial [6].

Exemple 5 *Soient quatre concepts A, B, C et D où $A \sqsubseteq B$ et on veut montrer par l'algorithme de subsomption que :*

$$A \sqcap \forall R.C \sqcap \forall R.D \sqsubseteq B \sqcap \forall R.C$$

1. Normalisation :

$$A \sqcap \forall R.C \sqcap \forall R.D \equiv A \sqcap \forall R.(C \sqcap D)$$

Donc ça revient à montrer que :

$$A \sqcap \forall R.(C \sqcap D) \sqsubseteq B \sqcap \forall R.C$$

2. Comparaison structurelle, i.e, on vérifie que pour chaque sous expression X du membre droit $B \sqcap \forall R.C$, il existe une sous expression Y du membre gauche $A \sqcap \forall R.(C \sqcap D)$ tel que $Y \sqsubseteq X$:

- $X = B, Y = A$ avec $A \sqsubseteq B$
- $X = \forall R.C, Y = \forall R.(C \sqcap D)$: il faut montrer que $C \sqcap D \sqsubseteq C$.
 - donc on applique le même algorithme : $X = C, Y = C \sqcap D$ avec $C \sqsubseteq C$

Il est possible d'étendre cet algorithme afin de tester la subsomption de descriptions de concepts appartenant au langage \mathcal{FL}_0 enrichi par le concept bottom et la négation atomique. Enfin, on présente dans [1] un algorithme de subsomption structurelle pour le langage \mathcal{ALN} . Au delà de ce dernier, les algorithmes de subsomption structurelle se trouvent incomplets. En particulier, ils ne traitent pas la disjonction, la négation complète et le quantificateur existentiel complet d'où l'utilité des algorithmes de tableaux qui sont l'objet de la section suivante.

3.2 Algorithmes de tableaux

Les algorithmes de tableaux réduisent le problème de subsomption au problème de satisfiabilité. En fait, on sait que $C \sqsubseteq D$ si et seulement si $C \sqcap \neg D$ est insatisfiable.

Avant de présenter un algorithme de tableaux pour la logique $ALCN$, on illustre le principe correspondant via l'exemple suivant.

Exemple 6 Soient A et B deux noms de concepts, R un nom de rôle et supposons que l'on veut savoir si

$$(\exists R.A) \sqcap (\exists R.B) \sqsubseteq \exists R.(A \sqcap B).$$

Ceci revient à tester l'insatisfiabilité de

$$(\exists R.A) \sqcap (\exists R.B) \sqcap \neg(\exists R.(A \sqcap B)).$$

Dans un premier temps, on déplace la négation en utilisant les règles de Morgan et les règles usuelles de quantificateurs. Comme résultat on obtient la description

$$C = (\exists R.A) \sqcap (\exists R.B) \sqcap \forall R.(\neg A \sqcup \neg B)$$

qui est sous forme NNF (pour négation normal form), i.e, la négation se trouve uniquement devant les noms de concepts.

Ensuite, on essaye de construire une interprétation finie \mathcal{I} telle que $C^{\mathcal{I}} \neq \emptyset$. Ceci signifie qu'il doit exister un individu de $\Delta^{\mathcal{I}}$ qui soit un élément de $C^{\mathcal{I}}$. Soit b un tel élément et donc b doit satisfaire les contraintes $b \in (\exists R.A)^{\mathcal{I}}$, $b \in (\exists R.B)^{\mathcal{I}}$ et $b \in (\forall R.(\neg A \sqcup \neg B))^{\mathcal{I}}$.

De $b \in (\exists R.A)^{\mathcal{I}}$, on déduit qu'il doit exister un élément c tel que $(b, c) \in R^{\mathcal{I}}$ et $c \in A^{\mathcal{I}}$. Il est de même, $b \in (\exists R.B)^{\mathcal{I}}$ implique l'existence d'un individu d avec $(b, d) \in R^{\mathcal{I}}$ et $d \in B^{\mathcal{I}}$.

- Pour toute restriction existentielle, on introduit un nouvel individu comme remplisseur du rôle. En plus, cet individu doit satisfaire les contraintes correspondantes.

Comme b doit également satisfaire $b \in (\forall R.(\neg A \sqcup \neg B))^{\mathcal{I}}$ et c et d sont des R -remplisseurs de b , on tire que $c \in (\neg A \sqcup \neg B)^{\mathcal{I}}$ et $d \in (\neg A \sqcup \neg B)^{\mathcal{I}}$.

- On utilise les restrictions de valeurs afin d'imposer de nouvelles contraintes sur les individus.

D'autre part, $c \in (\neg A \sqcup \neg B)^{\mathcal{I}}$ signifie que $c \in (\neg A)^{\mathcal{I}}$ ou $c \in (\neg B)^{\mathcal{I}}$. Toutefois, $c \in (\neg A)^{\mathcal{I}}$ contredit l'autre contrainte $c \in (A)^{\mathcal{I}}$ si bien qu'on doit opter pour $c \in (\neg B)^{\mathcal{I}}$. D'une manière similaire, on prend $d \in (\neg A)^{\mathcal{I}}$.

- Pour les contraintes disjonctives, on choisit les possibilités successivement en faisant des retours arrière à chaque fois qu'une contradiction est découverte.

Donc là on vient de satisfaire toutes les contraintes en générant une interprétation \mathcal{I} telle que : $\Delta^{\mathcal{I}} = \{b, c, d\}$, $R^{\mathcal{I}} = \{(b, c), (b, d)\}$, $A^{\mathcal{I}} = \{c\}$ et $B^{\mathcal{I}} = \{d\}$, i.e, C est satisfiable et donc $(\exists R.A) \sqcup (\exists R.B)$ n'est pas subsumé par $\exists R.(A \sqcap B)$.

Maintenant, on rajoute une restriction de nombres au premier concept, i.e, on veut vérifier si

$$(\exists R.A) \sqcup (\exists R.B) \sqcap \leq 1R \sqsubseteq \exists R.(A \sqcap B)$$

qui l'est intuitivement.

En premier lieu, on procède exactement comme précédemment mais là pour satisfaire la nouvelle contrainte $b \in (\leq 1R)^{\mathcal{I}}$, les individus c et d doivent être indentiques.

- Donc si une restriction de nombres est violée on doit identifier les différents individus remplisseurs de rôle.

Dans ce cas, $c = d \in A^{\mathcal{I}}$, $c = d \in B^{\mathcal{I}}$ et $c = d \in (\neg A \sqcup \neg B)^{\mathcal{I}}$ produit toujours un conflit si bien que le concept C soit insatisfiable et on conclut que $(\exists R.A) \sqcup (\exists R.B) \sqcap \leq 1R \sqsubseteq \exists R.(A \sqcap B)$.

Afin de représenter les contraintes de type "a appartient à l'interprétation de C" et "b est un R-remplisseur de a", on peut prendre les assertions ABox comme structures de données. Mais dans ce cas on élimine l'hypothèse de nom unique (UNA) pour de telles ABox car comme on l'a vu dans l'exemple précédent, on peut être amenés à identifier des noms d'individus différents lorsqu'il s'agit de restriction de nombres. En revanche, on permet des assertions d'inégalité explicites de la forme $a \neq b$ qui est satisfaite par une interprétation \mathcal{I} si et seulement si $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

Plus formellement, soit C un concept \mathcal{ALCN} sous forme NNF. Afin de tester la satisfiabilité de ce dernier, l'algorithme de tableaux démarre avec une ABox $A_0 = \{C(x)\}$ et applique les règles de transformations qui préservent la consistance jusqu'à ce qu'aucune règle ne puisse être appliquée. Les règles de transformations sont résumées comme suit :

<p>The \rightarrow_{\sqcap}-rule Condition: \mathcal{A} contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$. Action: $\mathcal{A}' = \mathcal{A} \cup \{C_1(x), C_2(x)\}$.</p>
<p>The \rightarrow_{\sqcup}-rule Condition: \mathcal{A} contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$. Action: $\mathcal{A}' = \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' = \mathcal{A} \cup \{C_2(x)\}$.</p>
<p>The \rightarrow_{\exists}-rule Condition: \mathcal{A} contains $(\exists R.C)(x)$, but there is no individual name z such that $C(z)$ and $R(x, z)$ are in \mathcal{A}. Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y), R(x, y)\}$ where y is an individual name not occurring in \mathcal{A}.</p>
<p>The \rightarrow_{\forall}-rule Condition: \mathcal{A} contains $(\forall R.C)(x)$ and $R(x, y)$, but it does not contain $C(y)$. Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$.</p>
<p>The \rightarrow_{\geq}-rule Condition: \mathcal{A} contains $(\geq n R)(x)$, and there are no individual names z_1, \dots, z_n such that $R(x, z_i)$ ($1 \leq i \leq n$) and $z_i \neq z_j$ ($1 \leq i < j \leq n$) are contained in \mathcal{A}. Action: $\mathcal{A}' = \mathcal{A} \cup \{R(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n\}$, where y_1, \dots, y_n are distinct individual names not occurring in \mathcal{A}.</p>
<p>The \rightarrow_{\leq}-rule Condition: \mathcal{A} contains distinct individual names y_1, \dots, y_{n+1} such that $(\leq n R)(x)$ and $R(x, y_1), \dots, R(x, y_{n+1})$ are in \mathcal{A}, and $y_i \neq y_j$ is not in \mathcal{A} for some $i \neq j$. Action: For each pair y_i, y_j such that $i > j$ and $y_i \neq y_j$ is not in \mathcal{A}, the ABox $\mathcal{A}_{i,j} = [y_i/y_j]\mathcal{A}$ is obtained from \mathcal{A} by replacing each occurrence of y_i by y_j.</p>

FIG. 1 – Règles de transformations

Si la ABox "complète" obtenue à la fin ne contient pas de contradiction (appelée clash), alors A_0 est consistante (et donc C est satisfiable) sinon elle est inconsistante (insatisfiable).

Par ailleurs, les règles de transformations liées à la disjonction et à la restric-

tion de nombres ne sont pas déterministes. Pour cela, on considère des ensembles finis de ABox $S = \{A_1, \dots, A_k\}$ où un ensemble est consistant si et seulement s'il existe $i, 1 \leq i \leq k$, tel que A_i soit consistant.

Donc, une règle de transformation est appliquée à un ensemble fini de ABox S comme suit : elle prend un élément A de S et le remplace par une seule ABox A' , par deux ABox A' et A'' , ou bien par un nombre fini de ABox $A_{i,j}$.

Enfin, il est à noter que cet algorithme est sain, complet et se termine. Il est à noter aussi que le problème de satisfiabilité d'un concept \mathcal{ALCN} est décidable. En outre, il est PSPACE-complet.

4 Conclusion

Ce rapport a pour objet une introduction à la logique de description. Après une description du langage de base \mathcal{AL} et ses dérivés, nous avons présenté ce que sont les TBox et les ABox. Par la suite, on a décrit les algorithmes d'inférence tels que les algorithmes de subsumption et les algorithmes de tableaux.

L'étape suivante consiste en une définition d'une logique de description pour la corrélation d'alerte (IDDL pour Intrusion Detection Description Logic) permettant aux systèmes de sécurité de partager leur connaissances sur le système surveillé. En fait, le langage utilisé pour décrire les attaques détectées joue un rôle important pour l'efficacité du système. Dans ce contexte, a été développé le langage IDMEF (Intrusion Detection Message Exchange Format) qui est un IETF standard ayant pour but de définir un format d'alertes commun pour permettre aux systèmes de détection d'intrusion d'échanger leurs observations. Cependant, l'IDMEF présente plusieurs inconvénients. Par exemple, il est limité à une représentation syntaxique de quelques caractéristiques des alertes et donc il ne permet pas à deux systèmes de détection d'intrusion de comparer deux attaques sémantiquement.

Références

- [1] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *J. Artif. Intell. Res. (JAIR)*, 1 :277–308, 1994.
- [2] R. J. Brachman and J. G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2) :171–216, 1985.
- [3] Volker Haarslev and Ralf Möller. Description of the racer system and its applications. In *Description Logics*, 2001.
- [4] Ian Horrocks. The fact system. In *TABLEAUX*, pages 307–312, 1998.
- [5] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3 :78–93, 1987.

- [6] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3 :78–93, 1987.
- [7] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1) :1–26, 1991.
- [8] Evren Sirin and Bijan Parsia. Pellet : An owl dl reasoner. In *Description Logics*, 2004.
- [9] D. Tsarkov and I. Horrocks. Efficient reasoning with range and domain constraints. In *Description Logics*, 2004.
- [10] Y. Zou, T. W. Finin, and H. Chen. F-owl : An inference engine for semantic web. In *FAABS*, pages 238–248, 2004.