Implication textuelle et logiques de description

Paul Bedaride

Université Henri Poincaré – Nancy I

Implication textuelle et logiques de description

MÉMOIRE

30 juin 2006

pour l'obtention du

DEA de l'Université Henri Poincaré – Nancy I

(Master Informatique
Spécialité : Traitement Automatique des Langues)

par

Paul BÉDARIDE

Composition du jury

MERY Dominique
GALMICHE Didier
GODART Claude
MARI Jean-François
PERRIER Guy

Encadrants : ARECES Carlos
GARDENT Claire

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Table des matières

Remerciements

Je tiens à remercier profondément Carlos ARECES pour son aide sur les logiques de description et sur Racer, et Claire GARDENT pour son aide sur les problèmes linguistiques propres à la représentation sémantique des textes. Je tiens aussi à les remercier plus généralement pour l'attention qu'ils ont su m'accorder malgré leur charge de travail. Je remercie aussi Patrick BLACKBURN et l'équipe *Langue et Dialogue* pour leur acceuil et leur sympathie.

1 Introduction

De nombreuses applications du traitement automatique des langues (TAL) font intervenir un traitement sémantique de la langue naturelle : ces applications ont besoin de *"comprendre"* le sens d'un texte traité (même si elles le font souvent de manière très superficielle).

Malheureusement, les méthodes utilisées dans ces applications sont généralement des méthodes ad hoc qui ne permettent pas une modélisation générique du phénomène de compréhension. Le but de ce mémoire est d'explorer dans quelle mesure les logiques de description (LDs) peuvent être utilisées pour représenter le sens d'un texte puis pour raisonner sur ce sens. Plus spécifiquement, ce mémoire explore dans quelle mesure les LDs peuvent être utilisées pour modéliser la reconnaissance d'implications textuelles, c'est-à-dire pour décider étant donnés deux fragments textuels T_1 et T_2 si,ou non, T_1 implique T_2.

Ce mémoire est structuré comme suit. Nous commençons (section 2) par introduire la tâche de détection d'implications textuelles, les motivations derrière cette tâche ainsi que les résultats obtenus à ce jour. Nous présentons ensuite (section 3) les LDs ainsi que le prouveur que nous utiliserons pour mettre en oeuvre notre algorithme, c'est-à-dire RACER. Dans les sections 4 et 5, nous proposons une méthode pour associer de façon systématique une représentation dans les LDs à un texte, et nous montrons comment encoder dans les LDs les connaissances lexicales nécessaires à la détection d'implications textuelles. La section 6 propose un algorithme permettant de détecter l'implication textuelle, tandis que la section 7 résume les capacités ets limites du système proposé. Les sections 8 et 9 concluent en identifiant les questions ouvertes et en indiquant les directions de recherche qu'il serait naturel de poursuivre pour compléter le travail entamé.

2 La reconnaissance d'implications textuelles

Initié en 2005, le challenge RTE (Recognising Textual Entailment `http://www.pascal-network.org/Challenges/RTE2`) a pour but de comparer des systèmes sur une tâche sémantique de base à savoir, la reconnaissance d'implications textuelles. Nous allons donc voir quelles sont les motivations derrière ce nouveau challenge (section 2.1), puis nous expliquerons ce qu'est l'implication textuelle (section 2.2). Enfin nous ferons le point sur les travaux et les résultats existants et situerons notre approche par rapport à ces résultats (section 2.3).

2.1 Motivation

Un phénomène fondamental des langues naturelles est la variabilité dans la verbalisation d'un même contenu sémantique : le même sens peut être exprimé ou impliqué par des textes très différents. Beaucoup d'applications dans le traitement automatique des langues, comme les systèmes de Questions-Réponses, la recherche d'information, l'extraction d'information, ou encore la synthétisation automatique de textes ont besoin de traiter cette variabilité afin de pouvoir reconnaître différentes formulations d'un même sens. Néanmoins, il est difficile de comparer, sous la forme d'une évaluation générique, les méthodes sémantiques qui ont été développées dans des applications différentes. Le challenge RTE vise à pallier ce problème en définissant un cadre d'évaluation permettant de mesurer les capacités sémantiques d'un système sur la base d'une tâche précise, à savoir la reconnaissance d'implication textuelle.

2.2 L'implication textuelle

La reconnaissance d'implications textuelles est une tâche permettant de décider, étant donné deux fragments de texte T_1 et T_2, si la signification de T_1 peut être déduite de celle de T_2. Cette tâche capture génériquement une large gamme d'inférences qui sont pertinentes pour de multiples applications. Par exemple, un système de Question-Réponse doit identifier les textes qui impliquent la réponse attendue. Étant donnée la question *"Qui a tué Kennedy ?"*, le texte *"L'assassinat de Kennedy par Oswald"* implique la réponse attendue *"Oswald a tué Kennedy"*. De la même façon, dans la recherche d'information, les concepts dénotés par une question doivent être impliqués par les documents réponses pertinents. Dans la synthétisation de documents, une phrase ou une expression superflue, pour être omise du résumé, doit être

impliquée par une autre phrase du résumé. Dans l'extraction d'information, l'implication existe entre les différentes variations textuelles qui expriment la même relation cible. Et dans la traduction automatique, une traduction correcte doit être sémantiquement équivalente au texte de base. Ainsi, reconnaitre les implications textuelles permet de consolider et de promouvoir la recherche sur le traitement sémantique de la langue naturelle et de poser des bases génériques au développement de ces applications.

2.3 Le challenge RTE : méthodes et résultats

Le but du challenge RTE est de procurer des moyens de présenter et de comparer des approches possibles pour modéliser l'implication textuelle. A cette fin, deux corpus sont mis à la disposition des participants : un corpus de test (fourni trois semaines avant la soumission des résultats) et un corpus de développement. Chacun de ces corpus consiste en un ensemble de paires de petits extraits de textes, les éléments d'une paire étant surnommés T (texte) et H (hypothèse). La tâche des systèmes participants est alors de décider pour chaque paire présente dans le corpus, si oui ou non le texte T est une implication textuelle du texte H. Les résultats de ces systèmes sont ensuite comparés avec les annotations manuelles produites par les organisateurs du challenge. On mesurera en particulier la précision des systèmes, c'est-à-dire le pourcentage de réponses correctes produites par le système par rapport au nombre total de réponses produites. Notons que la notion d'implication textuelle est une notion intuitive qui reflète la compréhension humaine (et non logique) : T_2 est une implication textuelle de T_1 si et seulement si une majorité de personnes s'accordent à dire que T_2 est une implication naturelle de T_1. Comme nous le verrons en section 4, cette notion peut être différente de celle d'implication logique.

Pour le second Challenge RTE (en 2006), une vingtaine de travaux ont été présentés. Les méthodes utilisées pour traiter la reconnaissance d'implications textuelles sont diverses (logique, n-grammes, ...) et les résultats obtenus ont une précision variant entre 52 et 75%. Comme les corpus contiennent autant de paires fausses que de paires vraies, la ligne de base (baseline) est de 50% si bien que les performances obtenues peuvent être caractérisées comme étant assez moyennes. Par ailleurs, il est intéressant de voir que les meilleurs systèmes utilisent des méthodes basées sur la sémantique, la syntaxe, et la logique (méthode symbolique) ; plutôt que des méthodes basées sur les mots et les n-grammes (méthodes statistiques). L'approche proposée dans ce mémoire s'inscrit dans le cadre des approches symboliques adoptées par les systèmes gagnant : elle vise à explorer l'apport potentiel d'un système de reconnaissance d'implications textuelles qui soit basés sur les logiques de description et sur des bases de connaissances linguistiques telles que FrameNet et WordNet.

3 Logiques de description

Nous allons maintenant introduire les logiques de description, pour voir d'où elles viennent, comment elles sont définies et ce que'on peux faire avec. Nous présenterons ensuite le prouveur Racer qui utilise la logique de description $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D}^-)$.

3.1 Origines et applications des logiques de description

Les logiques de description ont été conçues à partir des réseaux sémantiques de Quillian [Qui88] qui sont des graphes orientés étiquetés dans lesquels on associe des concepts aux noeuds et des relations aux arcs. Elles sont également influencées par la sémantique des cadres de Minsky [Min74] dans laquelle les concepts sont représentés par des cadres caractérisés par un certain nombre d'attributs (appelés aussi slots).

Les logiques de description forment une famille de langages de représentation de connaissances qui peuvent être utilisées pour représenter la connaissance terminologique d'un domaine d'application d'une façon structurée et formelle. Le nom "logique de description" peut être interprété de deux manières. D'une part ces langages ont étés élaborés pour écrire la "description" des concepts pertinents d'un domaine d'application. D'autre part, une caractéristique cruciale de ces langages est qu'ils ont une sémantique formelle définie en logique du première ordre (à la différence des propositions précédentes comme les cadres de Minsky). Dans ce sens, nous pouvons dire que les LDs ont une sémantique "descriptive" formelle.

Les logiques de descriptions sont utilisées pour de nombreuses applications (voir *International Workshop on Description Logics* [Hor05] et *Workshop on Applications of Description Logics* [Bec04]) comme par exemple :

- Le web sémantique (e.g., représentation d'ontologies et recherche d'information basée sur la logique)
- La médecine/bioinformatique (e.g., représenter et gérer des ontologies biomédicales)
- Le traitement automatique des langues
- L'ingénierie de processus (e.g., représenter des descriptions de service)
- L'ingénierie de la connaissance (e.g., représenter des ontologies)
- L'ingénierie logicielle (e.g., représenter la sémantique des diagrammes de classe UML)

3.2 Définition des Logiques de Description

La plupart des logiques de descriptions divisent la connaissance en deux parties contenants :

- *les informations terminologiques* : définition des notions basiques ou dérivées et des relations entre ces notions. Ces informations sont "génériques" ou "globales", vraies dans tous les modèles et pour tous les individus.
- *les information sur les assertions* : ces informations sont "spécifiques" ou "locales", vraies pour certains individus particuliers.

L'ensemble des informations est modélisé par un couple $\langle T, A\rangle$, où T est un ensemble de formules relatives aux informations terminologiques (la T-Box) et A est un ensemble de formules relatives aux informations sur les assertions (la A-Box).

Une autre manière de voir la séparation entre ces informations, est d'associer la T-Box aux règles qui régissent notre monde (e.g., la physique, la chimie, la biologie, ...), et d'associer les individus de notre monde à la A-Box (e.g., Jean, Marie, un chat, ...).

Nous allons maintenant voir comment sont définies la sémantique et la base de connaissances des logiques de description puis nous parlerons des différentes logiques de descriptions existantes.

3.2.1 La sémantique

Les logiques de description utilisent les notions de *concept*, de *rôle* et d'*individu*. Les concepts correspondent à des "classes d'éléments" et sont interprétés comme un ensemble dans un univers donné. Les rôles correspondent aux "liens entre les éléments" et sont interprétés comme des relations binaires sur un univers donné. Les individus correspondent aux éléments d'un univers donné. La sémantique des logiques de description est définie comme suit :

Définition 1 *Soit* $\mathsf{CON} = \{C1, C2, \ldots\}$ *un ensemble fini de concepts atomiques,* $\mathsf{ROL} = \{R1, R2, \ldots\}$ *un ensemble fini de rôles atomiques et* $\mathsf{IND} = \{a1, a2, \ldots\}$ *un ensemble fini d'individus. Pour* CON,ROL,IND *disjoint deux à deux,* $\mathcal{S} = \langle \mathsf{CON}, \mathsf{ROL}, \mathsf{IND}\rangle$ *est une* signature. *Une fois qu'une signature* $\mathcal{S}$ *est fixée, une* interprétation $\mathcal{I}$ *pour* $\mathcal{S}$ *est un tuple* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$*, ou :*

- $\Delta^{\mathcal{I}}$ *est un ensemble non-vide.*
- $\cdot^{\mathcal{I}}$ *est une fonction assignant un élément* $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ *à chaque individu* $a_i \in \mathsf{IND}$*; un sous-ensemble* $C_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ *à chaque concept atomique* $C_i \in \mathsf{CON}$*; et une relation* $R_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *à chaque rôle atomique* $R_i \in \mathsf{IND}$.

En d'autres mots, une interprétation de la logique de description n'est rien de plus qu'un modèle pour un type particulier de signature du première ordre, ou seulement les prédicats unaires et binaires sont autorisés et l'ensemble des symboles de fonctions est vide.

3.2.2 La base de connaissances

Typiquement la base de connaissances standard utilisée par les logiques de descriptions est définie de la manière suivante :

Définition 2 *Étant donné un langage de description* $\mathcal{L}$ *et une signature* $\mathcal{S}$*, une base de connaissances* Σ *dans* $\mathcal{L}$ *est une paire* $\Sigma = \langle T, A\rangle$ *tel que :*

- *T est la T(erminologique)-Box, un ensemble fini, qui peut être vide, d'expressions appelées GCI*[1] *de la forme* $C_1 \sqsubseteq C_2$ *où* C_1,C_2 *sont des concepts sans restriction.* $C_1 \doteq C_2$ *est une notation pour* $C_1 \sqsubseteq C_2$ *et* $C_2 \sqsubseteq C_1$. *Les formules de T sont appelées des axiomes terminologiques.*
- *A est la A(ssertion)-Box, un ensemble fini, qui peut être vide, d'expressions de la forme* $a : C$ *ou* $(a, b) : R$ *où C est un concept sans restriction, R un rôle qui n'est pas forcément atomique et a,b appartiennent à IND. Les formules de A sont appelées des assertions.*

Les axiomes terminologiques ont été pensés à l'origine comme une définition, et nombre de conditions plus restrictives ont étés imposées. Les deux restrictions les plus importantes sont les suivantes :

- *axiome terminological simple* : dans tous les axiomes terminologiques $C_1 \sqsubseteq C_2$, C_1 est un concept atomique dans CON, et chaque concept atomique dans CON apparaît au plus une fois dans la partie gauche d'un axiome terminologique de la T-Box.
- *définition acyclique* : le graphe obtenue en assignat un noeud n_A à chaque concept atomique A de la T-Box, et un arc orienté entre deux noeuds n_A et n_B si il existe un axiome terminologique $C_1 \sqsubseteq C_2$ dans T tel que A apparaît dans C_1 et B dans C_2, ne contienne pas de cycles.

Ces restrictions sont liées à l'idée de considérer les axiomes terminologiques comme des définitions de concepts.

3.2.3 Les différentes logiques de description

Les logiques de descriptions ont à peu près toutes la même base à laquelle elles ajoutent différentes extensions (voir tableau 1). On peut dés lors avoir des concepts complexes composés de concepts atomiques, et pareillement pour les rôles.

Constructeur	Syntaxe	Sémantique
nom de concept	C	$C^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
négation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjonction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
disjonction ($\mathcal{U}$)	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
quantificateur universel	$\forall R.C$	$\{d_1 \mid \forall d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \rightarrow d_2 \in C^{\mathcal{I}})\}$
quantificateur existentiel ($\mathcal{E}$)	$\exists R.C$	$\{d_1 \mid \exists d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \wedge d_2 \in C^{\mathcal{I}})\}$
restriction de nombre ($\mathcal{N}$)	$(\geq n\ R)$	$\{d_1 \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\| \geq n\}$
	$(\leq n\ R)$	$\{d_1 \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\| \leq n\}$
restriction de nombre qualifié ($\mathcal{Q}$)	$(\geq n\ R.C)$	$\{d_1 \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2), d_2 \in C^{\mathcal{I}}\}\| \geq n\}$
	$(\leq n\ R.C)$	$\{d_1 \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2), d_2 \in C^{\mathcal{I}}\}\| \leq n\}$
un-de ($\mathcal{O}$)	$\{a_1, \ldots, a_n\}$	$\{d \mid d = a_i^{\mathcal{I}}$ pour un $a_i\}$
role filler ($\mathcal{B}$)	$\exists R.\{a\}$	$\{d \mid R^{\mathcal{I}}(d, a^{\mathcal{I}})\}$
nom de rôle	R	$R^{\mathcal{I}}$
conjonction de rôle ($\mathcal{R}$)	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
rôles inverses ($\mathcal{I}$)	R^{-1}	$\{(d_1, d_2) \mid R^{\mathcal{I}}(d_2, d_1)\}$
hiérarchie des rôles ($\mathcal{H}$)	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
transitivité des rôles ($\mathcal{R}^+$)	$Trans(R)$	$\forall a_1, a_2, a_3(R^{\mathcal{I}}(a_1, a_2) \wedge R^{\mathcal{I}}(a_2, a_3) \rightarrow R^{\mathcal{I}}(a_1, a_3))$

TAB. 1 – Extensions des logiques de description

[1] General Concept Inclusion

L'une des premières logiques de description est le langage $\mathcal{FL}^-$ [Brachman and Levesque, 1984], qui est défini comme une logique de description permettant les quantificateurs universels, la conjonction, et les quantificateurs existentiels de la forme $\exists R.\top$. Le langage $\mathcal{FL}^-$ a été proposé comme un formalisme pour la sémantique des cadres de Minsky. La conjonction de concepts est implicite dans la structure d'un cadre, qui requiert un ensemble de conditions pour être satisfait. La quantification des rôles permet de caractériser les slots.

La logique $\mathcal{AL}$ [Schmidt-Schauss and smolka, 1991], a étendue la logique $\mathcal{FL}^-$ en y ajoutant la négation des concepts atomiques. Cette logique peut-être considérée comme la logique de base des autres logiques de descriptions.

Les logiques de descriptions qui existent, sont des combinaisons des différents éléments du tableau 1. Par exemple si on rajoute la négation complète $\mathcal{C}$ à la logique $\mathcal{AL}$ on obtient la logique $\mathcal{ALC}$.

3.2.4 Inférences

En LD, la notion d'inférence est décrite comme ci-dessous :

Définition 3 *Soit $\mathcal{I}$ une interprétation et φ un axiome terminologique ou une assertion. Alors $\mathcal{I}$ modélise φ (notation $\mathcal{I} \models \varphi$) si :*
- $\varphi = C_1 \sqsubseteq C_2$ *et* $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, *ou*
- $\varphi = a : C$ *et* $a^{\mathcal{I}} \in C^{\mathcal{I}}$, *ou*
- $\varphi = (a, b) : R$ *et* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Soit $\Sigma = \langle T, A\rangle$ une base de connaissances et $\mathcal{I}$ une interprétation, alors $\mathcal{I}$ est un modèle de Σ (notation, $\mathcal{I} \models \Sigma$) si pour tous $\varphi \in T \cup A, \mathcal{I} \models \varphi$. Nous disons dans ce cas que $\mathcal{I}$ est un modèle de la base de connaissances Σ. Étant donné une base de connaissances Σ est un axiome terminologique ou une assertion φ, $\Sigma \models \varphi$ si pour tout modèle $\mathcal{I}$ de Σ nous avons $\mathcal{I} \models \varphi$.

3.3 Les tâches de raisonnement

En LDs, l'expression *raisonnement sur la T-Box* fait référence à la capacité de réaliser des inférences depuis une base de connaissances $\Sigma = \langle T, A\rangle$ ou T est non-vide, et similairement, *raisonnement sur la A-Box* est l'implication pour une A-Box non vide.

Définition 4 *Soit Σ une base de connaissances, $C_1, C_2 \in \mathsf{CON}, R \in \mathsf{ROL}$ et $a, b \in \mathsf{IND}$, nous définissons les tâches de déduction suivantes :*
- *Subsomption,* $\Sigma \models C_1 \sqsubseteq C_2$
  Vérifie si pour toutes les interprétations $\mathcal{I}$ tel que $\mathcal{I} \models \Sigma$, nous avons $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- *Vérification d'instance,* $\Sigma \models a : C$
  Vérifie si pour toutes les interprétations $\mathcal{I}$ tel que $\mathcal{I} \models \Sigma$, nous avons $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- *Vérification de relations* $\Sigma \models (a, b) : R$
  Vérifie si pour toutes les interprétations $\mathcal{I}$ tel que $\mathcal{I} \models \Sigma$, nous avons $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.
- *Cohérence de concept,* $\Sigma \not\models C \doteq \bot$
  Vérifie si pour toutes les interprétations $\mathcal{I}$ tel que $\mathcal{I} \models \Sigma$, nous avons $C^{\mathcal{I}} \neq \{\}$.
- *Cohérence de la base de connaissances,* $\Sigma \not\models \bot$
  Vérifie si il existe $\mathcal{I}$ tel que $\mathcal{I} \models \Sigma$.

Les tâches de déduction de base, peuvent être utilisées pour définir des tâches plus complexes. En particulers :
- *Recherche* : étant donné un concept, trouver les individus mentionnés dans la base de connaissances qui sont des instances de ce concept.
- *Réalisation* : étant donné un individu mentionné dans la base de connaissances, trouver le concept le plus spécifique, en accord avec les relations de subsomption, duquel l'individu est une instance.

La saturation de la A-Box sert à compléter les informations de la A-Box en accord avec les connaissances de la T-Box, on obtient donc :

Définition 5 *Étant donné une base de connaissances $\langle T, A\rangle$, nous disons que A est saturé si pour chaque individu $a \in \mathsf{IND}$, concept atomique $C \in \mathsf{CON}$ et rôle $R \in \mathsf{REL}$:*
- *l'assertion $a : C$ si et seulement si $\langle T, A\rangle \models a : C$*
- *l'assertion $(a, b) : R$ si et seulement si $\langle T, A\rangle \models (a, b) : R$*

Exemple 1 *Soit Σ une base de connaissances $\langle T, A\rangle$ où*[2] *:*

$$T = \{\mathit{ÉTALON} \doteq \mathit{CHEVAL} \sqcap \forall \mathit{Sexe}.\mathit{MASCULIN}\}$$
$$A = \{\mathit{shadowfax} : \mathit{ÉTALON}\}$$

La formule de T dit que les chevaux de sexe masculin sont des étalons, et la formule de A dit que le cheval *shadowfax* est un étalon. La sémantique formelle que nous donnons dans la définition 3 nous permet de vérifier que Σ a au moins un modèle (i.e., il est cohérent). Et à partir de Σ nous pouvons déduire plusieurs informations comme, par exemple, que le concept `CHEVAL` est cohérent avec Σ (il existe une certaine interprétation satisfaisant Σ qui assigne une extension non-vide à `CHEVAL` :

$$\Sigma \not\models \texttt{CHEVAL} \doteq \bot$$

Notons qu'à cause des limitations syntaxiques dans la définition de base des assertions, il n'est pas possible de représenter les implications fortes (qui proviennent de $\langle T, A\rangle$) telle que par exemple le fait que dans tous les modèles de $\langle T, A\rangle$, l'extension de `CHEVAL` est non-vide :

$$\Sigma \models \neg(\texttt{CHEVAL} \doteq \bot)$$

Avec $\Sigma = \langle T, A\rangle$ comme connaissances de base on a la A-Box saturé :

$$A = \{\texttt{shadowfax} : \texttt{ÉTALON} \sqcap \texttt{CHEVAL}\}$$

3.4 Le prouveur Racer

RacerPro est un système de représentation des connaissances qui implémente un calcul des tableaux optimisé pour une logique de description très expressive. Il offre des services de raisonnement pour les T-Box et A-Box multiples. Le système implémente la logique de description $\mathcal{ALCQHI}_{\mathcal{R}^+}(\mathcal{D}^-)$ aussi connue comme $\mathcal{SHIQ}(\mathcal{D}^-)$ (voir [Hor00]). C'est la logique de base $\mathcal{ALC}$ étendue avec des restrictions qualifiées de nombres, des hiérarchies de rôle, des rôles inverses, et des rôles transitifs. En plus de ces dispositifs de base, RacerPro fournit également des facilités pour le raisonnement algébrique comprenant des domaines concrets ($\mathcal{D}^-$) pour traiter :
- des restrictions min/max sur les nombres entiers,
- des (in)équation linéaire polynomiales sur les réels ou les cardinaux avec des relations d'ordre, et
- des égalités et inégalités de chaînes de caractères.

RacerPro supporte la spécification d'axiomes terminologiques généraux. Une T-Box peut contenir des inclusions de concepts généraux (GCIs), qui spécifient la relation de subsomption entre deux concepts. RacerPro peut également manipuler des définitions multiples ou même les définitions cycliques des concepts.

Pour une T-Box donnée, on peut répondre à divers types de questions. Basés sur la sémantique logique, les différents types de questions sont définis comme des problèmes d'inférence (par conséquent, répondre à une question s'appelle fournir un service d'inférence). Sommairement, nous énumérons seulement les plus importants ici :
- le cohérence de concept en accord avec une T-Box : L'ensemble d'objets décrits par un concept est-il vide ?
- la subsomption de concepts en accord avec une T-Box : Y a-t-il un rapport de sous-ensemble entre l'ensemble d'objets décrits par deux concepts ?

[2] Durant tous le document, les `CONCEPTS` sont en majuscule, les `Rôles` sont en minuscule mais commencent par une majuscule et les `individus` sont en minuscule

- trouver tous les noms de concepts incohérents mentionnés dans une T-Box. Les noms de concepts incohérents proviennent des axiomes de la T-Box.
- déterminer les parents et les enfants d'un concept en accord avec une T-Box : Les parents d'un concept sont les noms de concepts les plus spécifiques mentionnés dans une T-Box qui subsument le concept. Les enfants d'un concept sont les noms des concepts les plus généraux mentionnés dans une T-Box que le concept subsume. Toutes les relations de parent (ou enfant) entre les noms de concepts d'une T-Box forment une structure de graphe qui est souvent appelé taxonomie. Notons que certains auteurs emploient le terme taxonomie comme un synonyme de ontologie.

Si nous voulons tester l'exemple vu précédemment avec Racer, il faut lui donner le fichier suivant en entrée :

```
(in-tbox racer-tbox)
(signature :atomic-concepts (etalon cheval masculin)
           :roles (sexe))

(equivalent etalon (and cheval (some sexe masculin)))

(in-abox racer-abox racer-tbox)
(signature :individuals (shadowfax))

(instance shadowfax etalon)
```

nous pouvons ensuite poser à Racer différentes questions comme cela :

```
query : (all-individuals racer-abox)
answer: (shadowfax)

query : (concept-instances etalon)
answer: (shadowfax)

query : (individual-direct-types shadowfax)
answer: ((etalon))

query : (individual-instance-p shadowfax etalon)
answer: T

query : (individual-instance-p shadowfax masculin)
answer: Nil
```

4 La représentation des phrases en LD

Dans cette section, nous présentons la façon dont le sens d'une phrase peut être (partiellement) représenté par une formule de la logique des descriptions. Parce que les LDs ont une expressivité limitée et parce que le sens d'un texte est une notion extrêmement complexe, la méthode proposée se limite à produire une représentation approximative du sens d'un texte. En particulier, certains éléments syntaxiques linguistiques ne sont pas pris en compte tels que les articles, les anaphores, les quantificateurs, le temps, et la modalité. Ainsi, la phrase *"le chat mange une pomme"* sera approximée par *"chat manger pomme"*.

4.1 Les représentations sémantiques existantes

La représentation de la sémantique de la langue naturelle est un problème qui a succité beaucoup d'intérêt depuis plus d'un siècle. Il existe donc un certain nombre de travaux sur ce sujet apportant différents éléments de réponses.

L'un des premiers éléments de réponse à ce problème est donné par Frege [Jan01] qui introduisit le principe de compositionnalité selon lequel :

> La signification d'une phrase est une fonction de la signification de ses parties et de la façon dont elles sont combinées.

Dans [Mon70], Montague reprend cette idée et montre comment associer de façon systématique une phrase avec une représentation sémantique formulée dans le cadre du lambda calcul. Pour cela, il établit une correspondance entre les catégories syntaxiques de la langue naturelle (noms communs, verbes transitifs, ...) et les catégories sémantiques de la langue formelle cible à savoir, le lambda calcul. Plus spécifiquement, le lexique associe chaque mot avec un lambda terme tandis que la grammaire associe chaque règle syntaxique avec une règle sémantique spécifiant la façon dont les représentations sémantiques des constituants combinés par la règle doivent être combinés pour déterminer la représentation sémantique du constituant décrit par la règle[3]. Les tableaux 2 et 3 donnent un lexique et une grammaire qui illustrent l'approche de Montague.

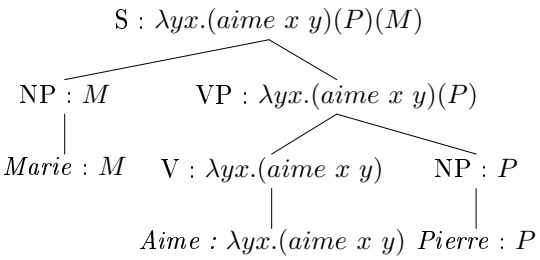
Verbe	**Catégorie**	**Sémantique**
Pierre	NP	P
Marie	NP	M
aime	V	$\lambda yx.(aime\ x\ y)$

TAB. 2 – Lexique

Règle grammaticale	**Règle sémantique**
S → NP VP	S' = VP'(NP')
VP → V NP	VP' = V'(NP')

TAB. 3 – Grammaire

La représentation sémantique d'une phrase est calculée par application récursive des règles de la grammaire (syntaxique et sémantique), par exemple pour l'analyse de la phrase *"Marie aime Pierre"* permet d'obtenir l'arbre ci-dessous avec pour chaque étiquette de l'arbre la syntaxe à gauche et la sémantique à droite :



Ainsi la représentation sémantique de la phrase *"Marie aime Pierre"* est le lambda terme $\lambda yx.(aime\ x\ y)(P)(M)$ qui se réduit en $(aime\ M\ P)$, ce qui correspond bien au sens de la phrase.

En bref, l'approche de Montague permet d'associer systématiquement à chaque expression de la langue naturelle un lambda terme représentant son sens. Cependant certains phénomènes linguistiques posent problème comme par exemple, les modificateurs. En effet, un même verbe peut en principe avoir plusieurs modificateurs du même type (lieu, temps, manière, etc.) :

> "Jean à fait ses devoirs gentiment et rapidement."
>
> "Jean a acheté une voiture le Lundi 8 mai à 17h."

Pour rendre compte du nombre arbitraire des modificateurs, plusieurs options sont possibles : soit représenter les verbes par des prédicats dont l'arité reflète le nombre maximum possible d'arguments et de modificateurs (e.g., $(acheter\ Jean\ voiture\ lundi\ 8\_mai\ 17h)$) ; soit utiliser des prédicats d'ordre supérieur prenant en argument une représentation verbale (e.g., $(date\ (date\ (date\ (acheter\ Jean\ voiture)\ lundi)\ 8\_mai)\ 17h)$). La première option n'en est pas vraiment une puisque au moins théoriquement, un verbe peut prendre un nombre infini de modificateurs. La deuxième est coûteuse informatiquement puisqu'elle exige l'utilisation de logiques d'ordre supérieur.

Pour pallier ces inconvénients, Davidson [Dav80] propose d'introduire des variables événementielles. Dans cette approche, un verbe introduit une prédication sur une variable événementielle, les arguments d'un verbe sont reliés par une relation thématique à cette variable et les modificateurs introduisent des prédications additionnelles sur cette même variable.

[3]Ici et dans le reste du rapport, C' dénote la représentation sémantique du constituent C

Avec cette approche on obtient pour la phrase *"Jean a acheté une voiture le Lundi 8 mai à 17h"* la représentation sémantique suivante :

$$(acheter\ e) \wedge (acheteur\ e\ Jean) \wedge (marchandise\ e\ voiture) \wedge (date\ e\ lundi) \wedge (date\ e\ 8\_mai) \wedge (date\ e\ 17h)$$

4.2 La base : les dépendances prédicat-arguments

Nous allons commencer par voir comment représenter le sens des verbes en LD.

Rappelons que la sémantique de Montague [Mon70] inclut dans sa représentation des prédicats d'ordre supérieur (e.g., les modificateurs par exemple) et des prédicats de plus de deux arguments (e.g., le verbe *vendre* qui prend en argument un acheteur, un vendeur et une marchandise). Or s'il est possible de retranscrire de telles représentations en LD, il faudrait pour ce faire, encoder les relations d'arité supérieures à deux en relations binaires. On aurait donc besoin d'une couche d'abstraction permettant de donner des relations d'arité arbitraire à Racer puis de récupérer ces relations quand on envoie des requêtes à Racer. Cela prendrait beaucoup de temps et on n'utiliserait pas pleinement la LD, car avec la couche d'abstraction, on perd de l'expressivité, comme par exemple la transitivité.

Nous allons donc baser notre approche sur la sémantique de Davidson [Dav80] et utiliser les correspondances indiquées ci-dessous. Pour notre représentation des textes en LDs, nous utilisons principalement la A-Box, la T-Box servant à représenter les connaissances de base et les contraintes sur les représentations sémantiques proposées. Notons cependant que, comme nous le verrons en section 8.1, il peut être nécessaire d'utiliser la T-Box pour la représentation de certains textes comme par exemple, les textes contenants des quantificateurs universels.

Sémantique de Davidson	**Représentation en LD (A-Box)**
variable/constante	individu
prédicat à un argument	concept atomique
prédicat à deux arguments	rôle atomique

Par exemple pour la phrase *"Jean mange une pomme"*, on aura la représentation suivante dans la sémantique de Davidson :

$$(manger\ e) \wedge (jean\ j) \wedge (pomme\ p) \wedge (mangeur\ e\ j) \wedge (mangé\ e\ p)$$

et la représentation en LD sera définie par la A-Box :

`e` : `MANGER`	(`e`, `j`) : `Mangeur`
`j` : `JEAN`	(`e`, `p`) : `Mangé`
`p` : `POMME`	

Une fois fixée la façon de représenter le sens d'un verbe et de ses arguments, il importe de déterminer la signature utilisée, c'est-à-dire l'ensemble des concepts et des relations servant à représenter ce sens. Pour ce faire, nous nous appuyons sur la base de données linguistiques FrameNet (`http://framenet.icsi.berkeley.edu`) qui vise à associer à chacune des foncteurs sémantiques de la langue (verbe, adjectif, nom), un cadre (i.e., un concept) et un ensemble d'éléments cadres (i.e., un ensemble de rôles thématiques). Par exemple dans FrameNet, les mots :

acheter vendre payer marchander coûter dépenser

évoquent tous le cadre *transaction commerciale* et les éléments cadres

acheteur vendeur marchandise prix

Pour spécifier la partie de la signature qui permet de représenter la sémantique des verbes, nous utilisons donc FrameNet et associons à chaque verbe :

- comme concept : le cadre spécifié pour ce verbe par FrameNet, et
- comme rôle thématique : les éléments cadres associés à un cadre par FrameNet.

Ainsi le verbe *vendre* sera représenté par le concept *transaction commerciale* et par les relations thématiques *acheteur*, *vendeur*, *marchandise* et *prix*. Pour la phrase *Jean achète du nutella au supermarché pour 2 euros"* on aura donc la représentation suivante :

t : TRANSACTION_COMMERCIALE	
j : JEAN	(t, j) : Acheteur
s : SUPERMARCHÉ	(t, s) : Vendeur
n : NUTELLA	(t, n) : Marchandise
p : 2_EUROS	(t, p) : Prix

On peut également utiliser FrameNet pour spécifie les axiomes terminologiques supplémentaires qui permettent de s'assurer que les contraintes additionnelles sur le sens du verbe soient bien respectées. On aura par exemple des contraintes imposant que pour une *transaction commerciale* on n'a qu'un seul prix associé :

$$\texttt{TRANSACTION\_COMMERCIALE} \sqsubseteq (\leq\ 1\ \texttt{Prix})$$

que la relation *prix* a pour source une *transaction commerciale* :

$$\exists \texttt{Prix}.\top \sqsubseteq \texttt{TRANSACTION\_COMMERCIALE}$$

et pour cible un objet de type PRIX :

$$\exists \texttt{Prix}^{-}.\top \sqsubseteq \texttt{PRIX}$$

ou PRIX est un concept qui met en évidence le fait qu'un individu est un prix (e.g. l'individu p : 2_EUROS est un prix 2_EUROS $\sqsubseteq$ PRIX), alors que Prix est une relation qui sert à attacher un PRIX à une TRANSACTION_COMMERCIALE.

Avec cette base pour notre représentation, nous pouvons déjà représenter des phrases basiques comme *"Le technicien répare la voiture"*, *"Marie aime Jean"*, ou encore *"Jean mange une pomme"*.

i : INGESTION	
j : JEAN	(i, p) : Ingestioneur
p : POMME	(m, j) : Ingestioné

4.3 Les modificateurs

Le sens d'un verbe peut être altéré par des modificateurs (lieu, temps, manière, ...). Par exemple dans la phrase *"Le chien aboie bruyamment"*, *bruyamment* altère le sens du verbe *aboyer* en ajoutant que le volume sonore produit par l'aboiement est fort. La différence entre les arguments et les modificateurs des verbes, est que les arguments sont dépendants du verbe, alors que les modificateurs peuvent s'ajouter à n'importe quel verbe. On peut ajouter, qu'un verbe peut avoir plusieurs modificateurs du même type alors qu'il n'a qu'un seul argument de chaque type. Ainsi, on peut avoir des phrases comme :

"Jean a acheté une voiture le Lundi 8 mai à 17h."

où on a trois modificateurs de temps : *Lundi*, *8 mai*, *17h*. Comme nous l'avons vu précédemment, l'approche de Davidson [Dav80] pour résoudre ce problème est la plus appropriée pour une représentation en LD, car elle ne demande pas d'avoir une logique d'ordre supérieur, et n'utilise que des prédicats à moins de deux arguments. Nous allons donc utiliser cette approche, et pour chaque modificateur que l'on ajoutera au verbe on ajoutera un concept représentant le sens du modificateur à l'individu de la A-Box correspondant au verbe.

On aura donc pour la phrase *"Jean a acheté une voiture le Lundi 8 mai à 17h"* que nous avons représentée précédemment dans la sémantique de Davidson :

$$(transaction\_commerciale\ e) \wedge (acheteur\ e\ Jean) \wedge (marchandise\ e\ voiture) \wedge (date\ e\ lundi) \wedge (date\ e\ 8\_mai) \wedge (date\ e\ 17h)$$

la représentation suivante en LD :

$$\texttt{e} : \texttt{TRANSACTION\_COMMERCIALE} \sqcap \texttt{LUNDI} \sqcap \texttt{8\_MAI} \sqcap \texttt{17H}$$

$$\texttt{j} : \texttt{JEAN} \qquad (\texttt{e}, \texttt{j}) : \texttt{Acheteur}$$

$$\texttt{v} : \texttt{VOITURE} \qquad (\texttt{e}, \texttt{v}) : \texttt{Marchandise}$$

4.4 Les adjectifs

Dans les approches de Montague et de Davidson, les adjectifs sont représentés comme des prédicats à un argument appliqués à la variable représentant le mot auquel l'adjectif s'applique. Cette représentation pourrait facilement être traduite en LD, mais il ne faut pas oublier que notre but final est de reconnaître les implications textuelles, et donc il faut que l'on soit capable de dire que *"le grand chat"* est équivalent à *"le chat est grand"*. Il nous sera plus facile d'identifier l'implication si dans notre représentation les éléments équivalents au plan sémantique sont représentés de la même façon. Nous allons donc représenter les adjectifs de la même façon que si l'on avait le verbe être entre le mot et l'adjectif. On aura donc pour *"le grand chat"* et *"le chat est grand"* la même représentation qui sera :

$$\texttt{c} : \texttt{CHAT} \sqcap \texttt{GRAND}$$

On considére donc le verbe *être* comme un verbe à part, on ne crée aucun individu de la A-Box pour lui, mais on ajoute à l'individu représentant le sujet du verbe, le concept représentant l'objet du verbe.

4.5 La négation

La représentation de la négation n'est pas triviale, car une négation peut avoir différentes portées possibles. Par exemple pour la phrase *"le chien n'aboie pas bruyamment"* en sémantique de Davidson on aurait les deux interprétations suivantes :

$$(chien\ c) \wedge (locuteur\ a\ e) \wedge (aboyer\ a) \wedge \neg(bruyamment\ e)$$

ou :

$$(chien\ c) \wedge (locuteur\ a\ e) \wedge \neg((aboyer\ a) \wedge (bruyamment\ e))$$

La première représentation qui a une portée courte est la plus évidente, elle signifie que le chien aboie mais qu'il n'aboie pas bruyamment. La seconde qui a une portée longue signifie qu'on ne sait pas s'il aboie, mais que s'il aboie alors il ne le fait pas bruyamment.

Il nous faut choisir quelle portée privilégier mais le problème est que selon le contexte la portée d'une négation peut changer. Nous allons donc choisir arbitrairement la portée longue en espérant que cela soit celle qui représente l'interprétation voulue dans la majorité des cas.

5 Représentation des connaissances utilisées par le moteur d'inférence

Maintenant que nous avons vu comment représenter les phrases en LD en les encodant dans la A-Box, nous allons voir comment représenter les connaissances de base que nous utiliserons pour détecter les implications textuelles telle que :

$$\textit{"Un chat mange"} \Rightarrow \textit{"Un animal mange"}$$

Les connaissances nécessaires pour détecter l'implication textuelle englobent des connaissances dites *"lexicales"* ainsi que des connaissances encyclopédiques. Or la représentation et la structuration de ces connaissances et en particulier, des connaissances encyclopédiques est un problème difficile. Aussi, nous nous limiterons ici à la représentation des connaissances lexicales pour lesquelles des ressources linguistiques peuvent être utilisées.

5.1 La sémantique lexicale

La sémantique lexicale vise à déterminer le sens d'un mot. L'approche dite "relationnelle" introduite par D.A. Cruse [Cru86], définit les différents types de relations que l'on peut avoir entre les unités lexicales, à savoir :

- synonymie : La synonymie est un rapport de proximité sémantique entre des mots d'une même langue.
- antonymie complémentaire : L'antonymie complémentaire concerne l'application ou la non-application d'une propriété, c'est-à-dire que deux mots sont antonymes compémentaires quand l'application de l'un implique la non-application de l'autre et inversement.
- antonymie scalaire : L'antonymie scalaire concerne une propriété affectant une valeur étalonnable (valeur élevée, valeur faible), si on a deux mots qui sont antonymes scalaires, le fait d'avoir la négation de l'un n'implique pas qu'on a l'autre (i.e., *pas froid* n'implique pas *chaud*).
- hyperonymie : L'hyperonymie est un rapport de généralisation entre des mots.
- hyponymie : L'hyponymie est un rapport de spécification entre des mots.
- méronymie : La méronymie est une relation partitive hiérarchisée : une relation de partie à tout.
- holonymie : L'holonymie est une relation partitive hiérarchisée : une relation de tout à partie.

Nous allons donc utiliser la sémantique lexicale pour encoder les connaissances de base, et nous ferons correspondre à chaque relation lexicale une formule de LD qui appartiendra à la T-Box, car le but de la connaissances de base est de compléter l'information des textes que l'on analyse contenue dans la A-Box. Voici donc le tableau de correspondance entre les relations de la sémantique lexicale et les formules en LD :

WordNet	**Description Logic**	**X**	**Y**
X est un synonyme de Y	$X \doteq Y$	*"Chat"*	*"Matou"*
X est un antonyme complémentaire de Y	$X \doteq \neg Y$ $Y \doteq \neg X$	*"Présent"*	*"Absent"*
X est un antonyme scalaire de Y	$\neg X \sqsubseteq Y$ $\neg Y \sqsubseteq X$	*"Grand"*	*"Petit"*
X est un hyperonyme de Y Y est un hyponyme de X	$X \sqsubseteq Y$	*"Animal"* *"Chat"*	*"Chat"* *"Animal"*
X est un méronyme de Y Y est un holonyme de X	$Y \sqsubseteq \exists compose\_de.X$	*"Bras"* *"Corps"*	*"Corps"* *"bras"*

5.2 WordNet

Maintenant que nous savons de quelle manière nous allons représenter notre connaissance de base, il nous faut une connaissance de base fondée sur la sémantique lexicale à transformer pour avoir notre connaissance de base en LD. Nous avons pris WordNet [Lin98] comme source pour notre connaissance de base. Wordnet est une base de données lexicale reliant de diverses manières les concepts existants dans la langue naturelle. Elle contient des relations analogues à celles de la sémantique lexicale et nous pouvons donc facilement extraire les informations de WordNet de la manière que nous avons définie dans la section précédente. WordNet est une grosse source de connaissances comme peut le montrer le tableau ci-dessous :

	chaîne de caractères	ensemble de synonymes	unités lexicales
Nom	117097	81426	145104
Verbe	11488	13650	24890
Adjectif	22141	18877	31302
Adverbe	4601	3644	5720
Total	155327	117597	207016

5.3 FrameNet

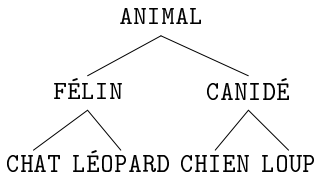
Comme indiqué en section 4.2, nous utilisons également FrameNet comme source des connaissances pour spécifier des axiomes terminologiques imposants des contraintes sur la représentation. Moins exhaustive que WordNet, FrameNet à néanmoins une couverture relativement large avec 792 cadres conceptuels et 9894 unités lexicales.

5.4 Un exemple de base de connaissances

Maintenant que nous savons comment concevoir notre base de connaissances, nous allons voir ce que cela donne si l'on extrait une partie de WordNet en une T-Box et que l'on applique cette T-Box a une A-Box représentant une phrase.

Pour notre exemple nous allons encoder une partie de l'information sur les animaux, on a ci-dessous, à droite l'arbre mettant en évidence les liaisons de hyperonymie entre les concepts et à gauche la T-Box représentant ces relations :

$$\begin{aligned} \texttt{FÉLIN} &\sqsubseteq \texttt{ANIMAL} \\ \texttt{CANIDÉ} &\sqsubseteq \texttt{ANIMAL} \\ \texttt{CHAT} &\sqsubseteq \texttt{FÉLIN} \\ \texttt{LÉOPARD} &\sqsubseteq \texttt{FÉLIN} \\ \texttt{CHIEN} &\sqsubseteq \texttt{CANIDÉ} \\ \texttt{LOUP} &\sqsubseteq \texttt{CANIDÉ} \end{aligned}$$



Maintenant si nous avons la phrase *"Un chat mange"* représentée sous la forme de la A-Box suivante :

$$\begin{aligned} &\texttt{m} : \texttt{MANGER} \\ &\texttt{c} : \texttt{CHAT} \qquad\qquad (\texttt{m}, \texttt{c}) : \texttt{Mangeur} \end{aligned}$$

alors on obtient la A-Box saturée suivante[4] en combinant cette A-Box à notre T-Box définie plus tôt :

$$\begin{aligned} &\texttt{m} : \texttt{MANGER} \\ &\texttt{c} : \texttt{CHAT} \sqcap \texttt{FÉLIN} \sqcap \texttt{ANIMAL} \qquad (\texttt{m}, \texttt{c}) : \texttt{Mangeur} \end{aligned}$$

Cela nous permettra donc de détecter l'implication entre *"Un chat mange"* et *"Un animal mange"*.

6 L'algorithme

Pour savoir s'il existe une implication entre deux textes, nous devons tout d'abord savoir une chose. En LD, la A-Box saturée (base de faits), peut être représentée sous la forme d'un ou plusieurs graphes orientées. Il peut y avoir plusieurs modèles pour une même base de faits, par exemple si on a des $\sqcup$ sur un $\exists$, la relation peut exister ou non selon le cas. Donc pour une A-Box de la forme :

$$\texttt{a} : \exists \texttt{Relation1}.\texttt{CONCEPT1} \sqcup \exists \texttt{Relation2}.\texttt{CONCEPT2}$$

[4] Tout au long du document les éléments rajoutés par saturation sont grisés

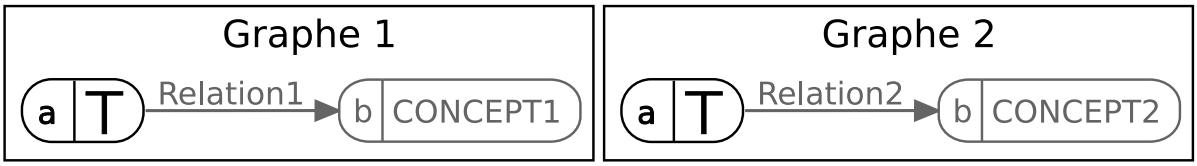


FIG. 1 –

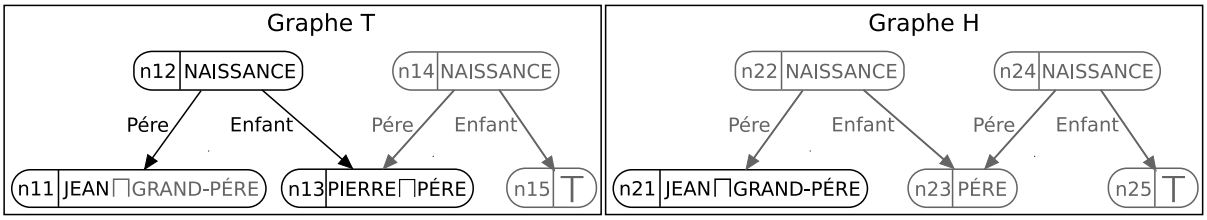


FIG. 2 – Le graphe T implique le graphe H

on a le graphe 1 de la figure 1 ayant un noeud $\texttt{a} : \top$ lié à un noeud $b : \texttt{CONCEPT1}$ par la relation `Relation1` ou le graphe 2 ayant un noeud $\texttt{a} : \top$ lié à un noeud $b : \texttt{CONCEPT2}$ par la relation `Relation2`.

Maintentant que nous savons cela il nous suffit de tester la correspondance entre les graphes représentants les textes. Nous utiliserons l'exemple de la figure 2 qui vise à montrer l'implication entre la phrase *"Jean est le père de Pierre et Pierre est père"* et la phrase *"Jean est grand-père"* pour illustrer l'algorithme. Cet exemple sera utilisé avec la connaissance de base suivante :

$$\texttt{PÈRE} \doteq \exists \texttt{Père}^{-}.(\texttt{NAISSANCE} \sqcap \exists \texttt{Enfant}.\top)$$
$$\texttt{GRAND-PÈRE} \doteq \exists \texttt{Père}^{-}.(\texttt{NAISSANCE} \sqcap \exists \texttt{Enfant}.\texttt{PÈRE})$$

On aura donc une fois les A-Box saturés par la T-Box, les graphes de la figure 2.

La méthode que nous proposons pour vérifier l'implication textuelle n'est pas toujours correcte. Pour commencer, nous travaillons déjà avec une semantique qui est une approximation du sens du texte. Cela peut produire des résultats incorrects (dire qu'une implication fausse est vraie) ou incomplets (dire qu'une implication vraie est fausse) Cependant, le prolème reste même si l'on prend une sémantique non approximative, parceque la saturation de la A-Box utilise seulement les concepts définis explicitements.

6.1 La correspondance de graphes

Pour savoir si l'implication textuelle est vraie, il nous faut maintenant tester qu'il existe bien une liaison de correspondance entre le graphe T représentant la base de faits du texte impliquant et le graphe H représentant la base de faits du texte impliqué. Dans notre cas, la correspondance de graphe revient à tester que le graphe impliqué est un sous-graphe du graphe impliquant. Nous allons maintenant étudier les différentes étapes nécessaires pour vérifier qu'un graphe est un sous-graphe d'un autre graphe.

6.1.1 Vérifier les noeuds

Pour qu'un graphe H soit un sous-graphe d'un autre graphe T, il faut tout d'abord que l'ensemble des noeuds qui compose le graphe H soit un sous-ensemble des noeuds qui compose le graphe T. On obtient donc la propriété suivante :

Propriété 1 *Soit Tn l'ensemble des noeuds de T et Hn l'ensemble de ceux de H. Une condition nécessaire pour que H soit un sous-graphe de T, est qu'il faut qu'il existe une fonction f qui à chaque élément x de Hn fait correspondre un élément y de Tn, tel que l'ensemble des concepts associés à x soit un sous-ensemble de ceux associés à y.*

Dans notre exemple, nous avons donc Tn qui contient les noeuds n11 : JEAN $\sqcap$ GRAND-PÈRE, n12 : NAISSANCE, n13 : PIERRE $\sqcap$ PÈRE, n14 : NAISSANCE et n15 : $\top$, et Hn qui contient des noeuds identiques n21 : JEAN $\sqcap$ GRAND-PÈRE, n22 : NAISSANCE, n23 : PÈRE, n24 : NAISSANCE et n25 : $\top$ après avoir saturé les deux A-Box avec la T-Box. On peut donc immédiatement voir qu'il existe bien une fonction f qui est définie comme cela :

$$f(\texttt{n21}) = \texttt{n11}$$
$$f(\texttt{n22}) = \texttt{n12}$$
$$f(\texttt{n23}) = \texttt{n13}$$
$$f(\texttt{n24}) = \texttt{n14}$$
$$f(\texttt{n25}) = \texttt{n15}$$

6.1.2 Vérifier les arcs

Ensuite, pour que le graphe H soit un sous-graphe du graphe T, il faut que tous les arcs existants dans H, le soit aussi dans T via la fonction f. On obtient donc la propriété suivante :

Propriété 2 *Soit Ta l'ensemble des arcs de T, et Ha l'ensemble des arcs de H. Les arcs sont représentés par des triplets (noeud source, noeud cible, nom de l'arc). On a alors : H est un sous graphe de T, si $(x, y, R) \in Ha$ implique que $(f(x), f(y), R) \in Ta$.*

Dans notre exemple, nous avons donc Ta qui contient Père(n12, n11), Enfant(n12, n13), Père(n14, n13) et Enfant(n14, n15), et Ha qui contient Père(n22, n21), Enfant(n22, n23), Père(n24, n23) et Enfant(n24, n25). Nous devons donc voir si les relations de Ha existent bien dans Ta, via la fonction f que nous avons définie précédemment. Pour Père(n22, n21) via f on obtient Père(n12, n11), pour Enfant(n12, n23) on obtient Enfant(n12, n13), pour Père(n24, n23) on obtient Père(n14, n13) et enfin pour Enfant(n24, n25) on obtient Enfant(n14, n15). Ces quatre relations existent bien dans Ta donc l'implication est bonne.

6.2 Implémentation en utilisant Racer

Pour mettre en oeuvre cet algorithme, nous avons utilisé le langage python, car il est plus important pour nous d'avoir un programme facile à modifier et compréhensible plutôt qu'un programme rapide. Nous avons crée trois classes différentes pour notre application, une première permettant de communiquer avec Racer, une seconde représentant les noeuds et les arcs du graphe, et une dernière contenant le moteur sémantique.

L'algorithme décrit précédemment n'a pas pu être respecté à la lettre, car avec Racer on ne peut pas connaître les éléments en dehors de la A-Box. On a donc du faire quelques modifications que nous expliquerons plus tard.

6.2.1 Classe Racer

La première classe permet donc de communiquer avec Racer, en envoyant des requêtes formulées en Lisp, par le biais d'une connexion TCP. Cette classe contient une fonction de communication permettant d'envoyer et de recevoir des messages de Racer, des fonctions correspondants aux diverses requêtes que l'on peut faire à Racer qui font appel à la fonction de communication et des fonctions permettant de parser les différents résultats. Pour la requête `(concept-instances Concept ABox)`, on a donc une fonction `concept_instances` qui prend en arguments un concept et une A-Box et qui renvoie une liste contenant les éléments de la A-Box étant des instances du concept.

6.2.2 Classe Entity

La classe permettant de représenter les noeuds et les arcs, contient le nom du noeud et des listes contenants :

- les arc normaux et inverses que le noeud a avec les autres noeuds,
- les concepts associés au noeud,

– les noeuds du second graphe auxquels le noeud peut être associé.
Et il contient aussi une fonction permettant d'afficher proprement toutes ces informations.

6.2.3 Classe Matching

C'est donc grâce à cette classe que l'on peut savoir si un texte en implique un autre. Cette classe est composée de plusieurs listes contenants :
– les différentes entités du graphe impliqué,
– les noeuds n'ayant qu'un seul noeud associé,
– les noeuds ayant plusieurs noeuds associés,
– les noeuds n'ayant aucun noeud associé.
Nous avons décomposé l'algorithme d'inférence en six fonctions qui sont expliquées ci-dessous.

La première fonction qui se nomme `create_entities_list`, sert à récupérer les différents individus de la A-Box impliquée, puis de crée un objet Entity pour chacun d'eux. Ensuite pour chacun de ces noeuds on récupère ses relations normales et inverses, ses concepts associés et ses attributs associés.

La fonction `get_associated_entities` permet d'associer aux objets Entity crées avec la fonction `create_entitys_list`, la liste les noeuds du graphe impliquant auxquels ils peuvent être associés.

La fonction `part_entities` permet de partitionner la liste des objets Entity en trois listes, une contenant les noeuds n'ayant aucun noeud associé, un autre contentant les noeuds n'ayant qu'un seul noeud associé, et une dernière contenant les noeuds ayant plusieurs noeuds associés.

La fonction `test_f` permet de vérifier les arcs à l'aide d'une fonction f . C'est à cet endroit que nous avons un différence avec l'algorithme présenté précédemment. La différence vient du fait que l'on ne peut pas accéder aux entités qui ne sont pas dans la A-Box. La fonction f n'est donc pas obligée d'avoir une noeud associé à chaque noeud du graphe impliqué, elle permet donc d'avoir des noeuds du graphe impliqué qui n'ont pas de noeuds associés dans le graphe impliquant. Pour chacun de ces noeuds on regarde si leurs arcs sont reliés à des noeuds appartenant à l'ensemble de définition de f. S'ils le sont on demande alors à Racer de vérifier que l'arc existe dans l'autre sens. Dans notre exemple, nous avons n14 qui n'est pas dans la A-Box et donc n24 n'a pas de correspondance dans f. Par contre on sait que n24 est relié par l'arc inverse *Temperature* à n23 qui est associé à n13 dans f. On demande donc à Racer de tester si le noeud 13 est bien relié par l'arc *Temperature* à un noeud ayant les mêmes concepts que n24.

La fonction `test_fs` permet de tester pour chacune des fonctions f possibles entre les entités de la A-Box impliqué et ceux de la A-Box impliquante si la fonction `test_f` est satisfaite.

Enfin la fonction `infAA` permet d'assembler les fonctions `create_entitys_list`, `get_associated_entitys`, `part_entitys` et `test_fs` pour pouvoir tester si un texte en implique un autre.

7 Les tests et les limites du système

Afin de pouvoir tester les limites du système, nous avons établi une base de tests sous la forme d'un fichier XML contenant un ensemble de paires de textes, nous pouvons d'ailleurs voir un échantillon de ce corpus dans le tableau 4. Pour chaque paire, on a deux textes T et H et on sait si $T \Rightarrow H$ et si $H \Rightarrow T$. De plus on a la représentation en LD de chaque texte qui est faite à partir de la représentation que nous avons définie et que l'on pourrait obtenir à l'aide d'un parseur. Voilà ci dessous un exemple de fichier xml que l'on peut avoir en entrée :

```
<?xml version="1.0" encoding="UTF-8"?>
<pairs>
 <pair id="t2" value="impl">
  <t>
```

Phrase 1	Phrase 2	Réel		Appli	
		⇒	⇐	⇒	⇐
Les tests qui fonctionnent					
1 Les synonymes, antonymes, hyperonymes, ...					
Jean a un vélo rouge.	Jean a une bicyclette rouge.	T	T	T	T
Le chat est grand.	Le chat n'est pas petit.	T	F	T	F
Le chat est grand.	Le chat est petit.	F	F	F	F
Le sac est plein.	Le sac n'est pas vide.	T	F	T	F
Le sac est plein.	Le sac est vide.	F	F	F	F
Un chat mange.	Un animal mange.	T	F	T	F
2 Les modificateurs					
L'homme court rapidement.	L'homme court.	T	F	T	F
Le chien aboie bruyamment.	Le chien aboie.	T	F	T	F
3 La négation des verbes					
Les ventes augmentent.	Les ventes ne déclinent pas.	T	F	T	F
L'homme ne court pas.	L'homme ne court pas rapidement.	T	F	T	F
Le chien n'aboie pas gentiment.	Le chien n'aboie pas bruyamment et gentiment.	T	F	T	F
4 Les individus ayant des concepts et des relations identiques					
Un chat monte sur le pommier et l'autre sur l'oranger.	Un animal monte sur le pommier et l'autre sur l'oranger	T	F	T	F
Il y a une bicyclette sur deux chemins, et il y a une autre bicyclette sur l'un de ces chemins.	Il y a un vélo sur deux routes, et il y a un autre vélo sur l'une de ces routes.	T	T	T	T
5 Les phrases contenant des entités implicites					
Le technicien baisse la température de la pièce.	Le technicien rafraîchi la pièce.	T	T	T	T
Jean est père.	Jean a un fils.	T	T	T	T
Les tests qui posent problème					
6 Les phrases contenant des entités implicites					
Jean est grand-père.	Jean a un enfant qui a un enfant.	T	T	F	T
Jean est un oncle.	Jean a un frère qui a un enfant.	T	T	F	T
7 Les phrases ne pouvant être représentées					
Quelques étudiants sont allés a l'école en voiture.	Quelques étudiants sont allés à l'école.	T	F	-	-
Aucun étudiant n'est allé à l'école en voiture.	Quelques étudiants sont allés à l'école.	F	F	-	-
Le diplomate a quitté Bagdad la semaine dernière.	Le diplomate a été à Bagdad.	T	F	-	-
Depuis qu'il fait froid, il a fermé la fenêtre.	Il fait froid.	T	F	-	-
Le journal a conclu que l'élection avait été truquée.	L'élection a été truquée.	F	F	-	-
L'homme avait 20 $ dans sa poche.	L'homme avait 10 $ dans sa poche.	T	F	-	-

TAB. 4 – Tableau représentant différentes paires de tests, ainsi que les implications entre ces paires, et les implications trouvées par notre application

```
   <fr>Le chat est grand.</fr>
   <dl>
    <individuals>
     <ind name="e1">chat</ind>
     <ind name="e2">grand</ind>
     <ind name="v1">etat</ind>
    </individuals>
    <relations>
     <rel name="Sujet" from="v1" to="e1"/>
     <rel name="Taille" from="v1" to="e2"/>
    </relations>
   </dl>
  </t>
  <h>
   <fr>Le chat n'est pas petit.</fr>
   <dl>
    <individuals>
     <ind name="e1">chat</ind>
     <ind name="e2">(not petit)</ind>
     <ind name="v1">etat</ind>
    </individuals>
    <relations>
     <rel name="Sujet" from="v1" to="e1"/>
     <rel name="Taille" from="v1" to="e2"/>
    </relations>
   </dl>
  </h>
 </pair>
</pairs>
```

On a aussi un fichier XML contenant les axiomes terminologiques de la connaissance de base, qui ressemble à :

```
<?xml version="1.0" encoding="UTF-8"?>
<knowledge>
 <s>(signature  :atomic-concepts (chien aboyer bruyamment gentillement ventes pomme sac plein vide gouv
 <r>(equivalent (some (inv Objet) rafraichir) (some temperature baisser))</r>
 <r>(equivalent rafraichir baisser-temperature)</r>
 <r>(equivalent velo bicyclette)</r>
 <r>(equivalent route chemin)</r>
 <r>(implies grand (not petit))</r>
 <r>(implies petit (not grand))</r>
 <r>(implies plein (not vide))</r>
 <r>(implies vide (not plein))</r>
 <r>(implies augmenter (not decliner))</r>
 <r>(implies decliner (not augmenter))</r>
 <r>(implies jean homme)</r>
 <r>(implies devorer manger)</r>
</knowledge>
```

Il nous suffit donc de lancer l'application avec en entrée ces deux fichiers XML et on obtient un rapport affichant pour chaque paire ces deux phrases et disant si l'implication fonctionne dans un sens puis dans l'autre en vrai (entre parenthèses) et d'après l'application. On obtient donc une sortie qui ressemble à ce que l'on a ci-dessous :

```
##################################################
T:Le chat est grand.
H:Le chat n'est pas petit.
T->H (True): True
H->T (False): False
##################################################
T:Le technicien baisse la temperature de la piece.
H:Le technicien rafraichi la piece.
T->H (True): True
H->T (True): True
##################################################
T:Le chien n'aboie pas gentillement.
H:Le chien n'aboie pas bruyamment et gentillement.
T->H (True): True
H->T (False): False
##################################################
```

Nous allons commencer par voir les exemples qui fonctionnent puis nous regarderons ceux qui posent problème. Pour ces exemples, notre connaissance de base est définie par la signature de la figure 3 et les assertions terminologiques de la figure 4.

7.1 Les test qui fonctionnent

7.1.1 Les phénomènes linguistiques

Les synonymes, antonymes, hyperonymes, ... Ces exemples sont ceux qui utilisent les règles simples de la T-Box dans lesquelles on a un concept ou un ensemble de concepts qui impliquent un autre concept ou ensemble de concepts. Notre corpus de tests contient plusieurs paires de ce type comme celle en section 1 du tableau 4. Nous n'avons aucune difficulté pour détecter ce type d'implication, car il suffit de saturer les A-Box pour que cela fonctionne.

Par exemple l'implication entre *"le chat est grand"* et *"le chat n'est pas petit"* on a les deux A-Box saturées suivantes :

$$\mathtt{c} : \mathtt{CHAT} \sqcap \mathtt{GRAND} \sqcap \neg\mathtt{PETIT} \qquad\qquad \mathtt{c} : \mathtt{CHAT} \sqcap \neg\mathtt{PETIT}$$

"le chat est grand" (A1) *"le chat n'est pas petit"* (A2)

on obtient bien une correspondance entre A2 et A1, donc l'implication est bonne.

Pour les phrases *"le chien court"* et *"le chien bouge"*, on a une implication due à l'hyperonymie entre *courir* et *bouger*. Cette liaison d'hyperonymie est mise en valeur par FrameNet qui représente le verbe *courir* comme un `MOUVEMENT` lié par une relation `Vitesse` à un individu `RAPIDE`, alors que le verbe *bouger* est seulement un `MOUVEMENT`. On obtient les représentations suivantes :

$$\begin{array}{ll} \mathtt{m1} : \mathtt{MOUVEMENT} & \\ \mathtt{c1} : \mathtt{CHIEN} & (\mathtt{m}, \mathtt{c}) : \mathtt{Agent} \\ \mathtt{v1} : \mathtt{RAPIDE} & (\mathtt{m}, \mathtt{v}) : \mathtt{Vitesse} \end{array} \qquad\qquad \begin{array}{ll} \mathtt{m1} : \mathtt{MOUVEMENT} & \\ \mathtt{c1} : \mathtt{CHIEN} & (\mathtt{m}, \mathtt{c}) : \mathtt{Agent} \end{array}$$

"le chien court" (A1) *"le chien bouge"* (A2)

on obtient bien une correspondance entre A2 et A1, donc l'implication est bonne.

Les modificateurs Les modificateurs sont les mots qui modifient le sens des verbes ou des noms. Avec la représentation que nous avons utilisé, nous n'avons aucun problème pour trouver les implications sur ce genre d'exemples, comme nous pouvons le voir dans la section 2 du tableau 4.

$$\mathsf{CON} = \{\texttt{ANIMAL}, \texttt{BAS}, \texttt{BICYCLETTE}, \texttt{BRUYAMMENT}, \texttt{CHANGEMENT\_DE\_POSITION\_SUR\_UNE\_ÉCHELLE}, \texttt{CHAT}, \texttt{CHEMIN}, \texttt{CHIEN}, \texttt{ENFANT}, \texttt{FAIRE\_DU\_BRUIT}, \texttt{FRÈRE}, \texttt{GENTIMENT}, \texttt{GRAND}, \texttt{GRAND-PÈRE}, \texttt{HAUT}, \texttt{HOMME}, \texttt{INGESTION}, \texttt{JEAN}, \texttt{MOUVEMENT}, \texttt{MOUVEMENT\_DIRECTIONNEL}, \texttt{NAISSANCE}, \texttt{ONCLE}, \texttt{ORANGER}, \texttt{PETIT}, \texttt{PLEIN}, \texttt{POMMIER}, \texttt{POSSÉDER}, \texttt{RAFRAÎCHIR}, \texttt{RELATION\_LOCATIVE}, \texttt{RAPIDE}, \texttt{RAPIDEMENT}, \texttt{ROUGE}, \texttt{ROUTE}, \texttt{SAC}, \texttt{TECHNICIEN}, \texttt{TEMPÉRATURE}, \texttt{VÉLO}, \texttt{VENTES}, \texttt{VIDE}\}$$

$$\mathsf{ROL} = \{\texttt{Agent}, \texttt{Attribut}, \texttt{Chemin}, \texttt{Destination}, \texttt{Direction}, \texttt{Élément}, \texttt{Enfant}, \texttt{Figure}, \texttt{Ingestionneur}, \texttt{Ingestionné}, \texttt{Mère}, \texttt{Père}, \texttt{Possession}, \texttt{Propriétaire}, \texttt{Sol}, \texttt{Source}, \texttt{Thème}, \texttt{Vitesse}\}$$

FIG. 3 – Signature de la connaissance de base

$$\begin{aligned}
\texttt{BICYCLETTE} &\doteq \texttt{VÉLO}\\
\texttt{CHEMIN} &\doteq \texttt{ROUTE}\\
\texttt{GRAND} &\sqsubseteq \neg\texttt{PETIT}\\
\texttt{PETIT} &\sqsubseteq \neg\texttt{GRAND}\\
\texttt{PLEIN} &\sqsubseteq \neg\texttt{VIDE}\\
\texttt{VIDE} &\sqsubseteq \neg\texttt{PLEIN}\\
\texttt{BAS} &\sqsubseteq \neg\texttt{HAUT}\\
\texttt{HAUT} &\sqsubseteq \neg\texttt{BAS}\\
\texttt{CHAT} &\sqsubseteq \texttt{ANIMAL}\\
\texttt{RAFRAÎCHIR} &\doteq \texttt{CHANGEMENT_DE_POSITION_SUR_UNE_ÉCHELLE} \sqcap\\
&\quad \exists\ \texttt{Attribut}.\texttt{TEMPÉRATURE} \sqcap \exists\ \texttt{Chemin}.\texttt{BAS}\\
\texttt{PÈRE} &\doteq \exists\ \texttt{Père}^{-}.(\texttt{NAISSANCE} \sqcap \exists\ \texttt{Enfant}.\mathit{ENFANT})\\
\texttt{GRAND-PÈRE} &\doteq \exists\ \texttt{Père}^{-}.(\texttt{NAISSANCE} \sqcap \exists\ \texttt{Enfant}.\texttt{PÈRE})
\end{aligned}$$

$$\begin{aligned}
\texttt{CHANGEMENT_DE_POSITION_SUR_UNE_ÉCHELLE} &\sqsubseteq (\leq\ 1\ \texttt{Agent}) \sqcap (\leq\ 1\ \texttt{Attribut}) \sqcap\\
&\quad (\leq\ 1\ \texttt{Chemin}) \sqcap (\leq\ 1\ \texttt{Élément})\\
\texttt{FAIRE_DU_BRUIT} &\sqsubseteq (\leq\ 1\ \texttt{Source})\\
\texttt{INGESTION} &\sqsubseteq (\leq\ 1\ \texttt{Ingestionneur}) \sqcap (\leq\ 1\ \texttt{Ingestionné})\\
\texttt{MOUVEMENT} &\sqsubseteq (\leq\ 1\ \texttt{Agent}) \sqcap (\leq\ 1\ \mathit{Vitesse})\\
\texttt{MOUVEMENT_DIRECTIONNEL} &\sqsubseteq (\leq\ 1\ \texttt{Destination}) \sqcap (\leq\ 1\ \texttt{Direction}) \sqcap\\
&\quad (\leq\ 1\ \texttt{Source}) \sqcap (\leq\ 1\ \texttt{Thème})\\
\texttt{NAISSANCE} &\sqsubseteq (\leq\ 1\ \texttt{Enfant}) \sqcap (\leq\ 1\ \texttt{Mère}) \sqcap (\leq\ 1\ \texttt{Père})\\
\texttt{POSSÉDER} &\sqsubseteq (\leq\ 1\ \texttt{Possession}) \sqcap (\leq\ 1\ \texttt{Propriétaire})\\
\texttt{RAFRAÎCHIR} &\sqsubseteq (\leq\ 1\ \texttt{Agent}) \sqcap (\leq\ 1\ \texttt{Élément})\\
\texttt{RELATION_LOCATIVE} &\sqsubseteq (\leq\ 1\ \texttt{Figure}) \sqcap (\leq\ 1\ \texttt{Sol})
\end{aligned}$$

FIG. 4 – Assertions terminologiques de la connaissance de base

Ainsi, si on a *"Le chien court rapidement"* implique que *"Le chien court"*, on obtient les A-Box saturés :

m1 : MOUVEMENT ⊓ RAPIDEMENT			m2 : MOUVEMENT		
c1 : CHIEN		(m, c) : Agent	c2 : CHIEN		(m, c) : Agent
v1 : RAPIDE		(m, v) : Vitesse	v2 : RAPIDE		(m, v) : Vitesse

"le chien court rapidement" (A1) *"le chien court"* (A2)

et donc le concept de m2 est un sous-concept du concept de m1.

La négation des verbes La négation des verbes peut avoir plusieurs effets sur l'implication textuelle. Le premier est la négation même du verbe qui fait que si un individu a le concept associé à un verbe alors il a la négation des concepts associés aux antonymes de ce verbe (e.g. AUGMENTER ⊑ ¬DÉCLINER). Le second est due aux modificateurs, qui fait que si on a une négation sur un verbe alors on a la négation sur le verbe plus n'importe quel modificateur (car on prend la porté longue pour la négation). Nous allons donc voir quelques exemples représentatifs qui se situent dans la section 3 du tableau 4 qui montrent que notre application gère bien ces implications.

Pour les phrases *"le chien ne court pas"* et *"le chien ne court pas rapidement"*, on a les A-Box saturées suivantes :

m1 : ¬MOUVEMENT			m2 : ¬(MOUVEMENT ⊓ RAPIDEMENT)		
c1 : CHIEN		(m, c) : Agent	c2 : CHIEN		(m, c) : Agent
v1 : RAPIDE		(m, v) : Vitesse	v2 : RAPIDE		(m, v) : Vitesse

"le chien ne court pas" (A1) *"le chien ne court pas rapidement"* (A2)

on obtient bien que A1 implique A2 vu que m1 est une instance du concept ¬(MOUVEMENT⊓RAPIDEMENT), mais l'implication n'est pas vérifiée dans l'autre sens car m1 n'est pas une instance de ¬MOUVEMENT.

7.1.2 Restrictions techniques dues au choix du langage pour la représentation sémantique

Les individus ayant des concepts et des relations identiques Pour ce cas, le problème, est que deux individus d'un graphe peuvent avoir des concepts et des relations identiques, il est donc difficile de les différencier. Notre méthode pour résoudre ce problème est de trouver toutes les fonctions f possibles puis de tester la correspondance de graphe sur chacune d'elle. Comme nous pouvons le voir dans la section 4 du tableau 4, ce type d'implication ne pose pas de problème à notre application.

Pour l'implication entre *"Un chat monte sur le pommier et un autre sur l'oranger"* et *"Un animal monte sur le pommier et un autre sur l'oranger"*, on a les A-Box saturés :

m11 : MOUVEMENT_DIRECTIONNEL			m21 : MOUVEMENT_DIRECTIONNEL		
m12 : MOUVEMENT_DIRECTIONNEL			m22 : MOUVEMENT_DIRECTIONNEL		
c11 : CHAT ⊓ ANIMAL		(m11,c11) : Thème	a21 : ANIMAL		(m21,a21) : Thème
p1 : POMMIER		(m11,p1) : Destination	p2 : POMMIER		(m21,p2) : Destination
c12 : CHAT ⊓ ANIMAL		(m12,c12) : Thème	a22 : ANIMAL		(m22,a22) : Thème
o1 : ORANGER		(m12,o1) : Destination	o2 : ORANGER		(m22,o2) : Destination

"Un chat monte sur le pommier et un autre sur l'oranger" (A1) *"Un animal monte sur le pommier et un autre sur l'oranger"* (A2)

et voila deux des fonctions f que l'on peut avoir :

$$f_1(\texttt{m11}) = \texttt{m21} \qquad f_2(\texttt{m11}) = \texttt{m21}$$
$$f_1(\texttt{m12}) = \texttt{m22} \qquad f_2(\texttt{m12}) = \texttt{m22}$$
$$f_1(\texttt{a21}) = \texttt{a11} \qquad f_2(\texttt{a21}) = \texttt{a12}$$
$$f_1(\texttt{a22}) = \texttt{a12} \qquad f_2(\texttt{a22}) = \texttt{a11}$$
$$f_1(\texttt{p2}) = \texttt{p2} \qquad f_2(\texttt{p2}) = \texttt{p1}$$
$$f_1(\texttt{o2}) = \texttt{o2} \qquad f_2(\texttt{o2}) = \texttt{o1}$$

Les phrases contenant des entités implicites Il se peut que dans certains cas une phrase fasse référence à des individus implicites. Par exemple dans la phrase *"Jean est père"*, on sait qu'il existe un individu qui est l'enfant de Jean. Le problème est que dans Racer, on ne peut pas accéder à ces données implicites qui sont créées par saturation de la A-Box avec la T-Box.

Si on veut montrer que T implique H, mais qu'il y a un individu explicite `i` de H qui n'a pas de correspondance dans T via f, alors on regarde dans H à quels individus $\{\texttt{i}_1, \ldots, \texttt{i}_n\}$ il est rattaché et par quelles relations $\{\texttt{Rel}_1, \ldots, \texttt{Rel}_n\}$ il l'est, puis on teste dans T si les équivalents des individus $\{\texttt{i}_1, \ldots, \texttt{i}_n\}$ via f sont liés via les relations inverses $\{\texttt{Rel}_1^-, \ldots, \texttt{Rel}_n^-\}$ à un individu ayant les mêmes concepts que `i`.

Nous allons donc tester l'existence de ces individus implicites en regardant à quels noeuds explicites ils sont rattachés. Le seul problème est que pour le moment nous allons seulement tester l'existence des individus implicites directement liés à un individu explicite. Ce genre de cas ne pose aucun problème dans la plupart des cas (voir section 5 du tableau 4). Cependant, il est possible d'avoir un individu implicite qui n'est pas directement lié à un individu explicite comme on peut le voir dans la section 6 du tableau 4, mais nous en reparlerons plus tard.

Si on prend l'exemple où *"Le technicien rafraîchi la pièce"* implique que *"Le technicien baisse la température de la pièce"* avec les A-Box saturés suivantes :

cp1 : RAFRAÎCHIR			cp2 : CHANGEMENT_DE_POSITION_SUR_UNE_ÉCHELLE		
⊓ CHANGEMENT_DE_POSITION_SUR_UNE_ÉCHELLE			⊓ RAFRAÎCHIR		
tc1 : TECHNICIEN	(cp1,tc1) : Agent		tc2 : TECHNICIEN	(cp2,tc2) : Agent	
tp1 : TEMPERATURE	(cp1,tp1) : Attribut		tp2 : TEMPERATURE	(cp2,tp2) : Attribut	
cm1 : BAS	(cp1,cm1) : Chemin		cm2 : BAS	(cp2,cm2) : Chemin	
el1 : PIÈCE	(cp1,el1) : Élément		el2 : PIÈCE	(cp2,el2) : Élément	
"Le technicien rafraîchi la pièce" (A1)			*"Le technicien baisse la température de la pièce"* (A2)		

on obtient donc que les individus `tp2` et `cm2` de A2 n'ont pas de correspondance explicite dans A1. On regarde alors à quels individus de A2 ils sont rattachés. On obtient que `tp2` est lié à `cp2` par $\texttt{Attribut}^-$ et que `cm2` est lié à `cp2` par $\texttt{Chemin}^-$. Il faut maintenant tester que dans A1 l'individu $f(cp2) = cp1$ est lié à un individu ayant le concept `TEMPERATURE` par la relation `Attribut`, et que l'individu $f(cp2) = cp1$ est lié à un individu ayant le concept `BAS` par la relation `Chemin`. On obtient une réponse affirmative donc l'implication est bonne.

7.2 Les tests qui posent problème

7.2.1 Les phrases contenant des entités implicites

Comme nous l'avons vu précédemment, si une phrase contient une entité implicite reliée directement à une entité explicite notre algorithme fonctionne bien, mais dans le cas contraire cela pose problème comme nous pouvons le voir dans les exemples de la section 6 du tableau 4.

Ce problème se manifeste quand on a des entités explicites dans la phrase impliquée qui correspondent à des entités implicites dans la phrase impliquante. L'autre sens ne pose pas de problème car on ne traite que les entités explicites de la phrase impliquée et on n'a pas a chercher des entités implicites.

Le problème est que pour le moment nous ne récupérons que les individus implicites directement reliés à un individu explicite. Il faudrait pouvoir les récupérer à plus d'un niveau, par exemple si on a la phrase *"Jean est grand-père"* alors on a l'individu `j` : JEAN ⊓ GRAND-PÈRE qui est rattaché à un individu

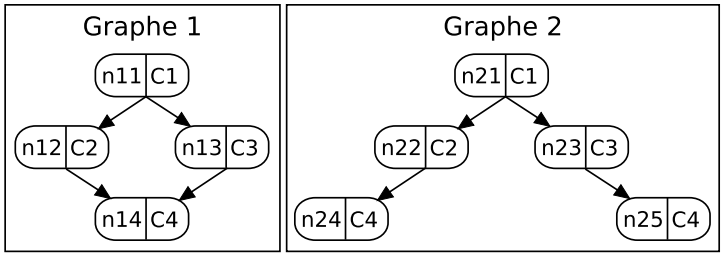


FIG. 5 –

`e1` : `ENFANT` lui même rattaché à un individu `e2` : `ENFANT` avec `e1` et `e2` qui sont des individus implicites. Si on veut montrer que cette phrase implique *"Jean a un enfant qui a un enfant"*, il faudra récupérer `e1` et `e2`, mais pour le moment nous ne savons que récupérer `e1`.

Il est possible de récuperer les individus implicites qui ne sont pas directement rattaché a un individu explicite, mais cela signifie que l'on ne veut plus tester un noeud mais un bout de graphe, et il faut pouvoir vérifier que l'on bien notre graphe et non quelque chose qui y ressemble vue que l'on ne peut voir ce graphe que point de vue d'un noeud. Ainsi dans la figure 5, on voit la même chose du point de vue du noeud `n11` que du point de vue du noeud `n21`. En effet on voit deux noeuds ayant pour concept `C2` et `C3`, et chacun de ces noeuds est reliés à un noeud ayant pour concept `C4`. On ne peut pas résoudre se problème vu qu'il représente un problème qui ne peut pas être modélisé en LD, c'est-à-dire de pouvoir dire que l'on veut pouvoir dire que deux individus font référence au même noeud.

7.2.2 Les phrases ne pouvant être représentées

Pour le moment, il existe encore beaucoup de phrases qui ne peuvent pas être représentées. Parmi ces phrases nous avons celles ayant des quantificateurs, celles jouant sur les temps et les suites d'événements ou encore celles utilisant les mondes possibles, ou imaginaires (*"Il se pourrait que ..."*, *"Jean pense que ..."*, ...). Nous avons donc plusieurs exemples qui ne peuvent même pas passer dans notre application car ils n'ont pas de représentation en LD qui se trouvent dans la section 7 du tableau 4.

8 Perspectives

8.1 Travaux en progrès : Les quantificateurs

La représentation des quantificateurs n'est pas une chose simple, car avec les quantificateurs on obtient des notions de groupes et de généralisation.

8.1.1 Généralisation

Quand on a une généralisation comme *"tous les humains travaillent bien"*, on a une information qui ne peut pas être représenté par des assertions (A-Box), mais qui est facilement représentable sous la forme d'un axiome terminologique (T-Box) :

$$\texttt{HUMAIN} \sqsubseteq \exists \texttt{Agent}^{-}.(\texttt{TRAVAILLER} \sqcap \texttt{BIEN})$$

On pourra alors avoir une implication entre *"tous les humains travaillent bien"* et *"Jean travaille bien"*. Mais comme on mélange la connaissance de base avec l'information ajoutée par le texte, on a un problème de pertinence et on peut inférer des textes de la connaissance de base. En effet on peut avoir une implication logique mais pas textuelle, comme :

$$\textit{"tous les humains travaillent bien"} \Rightarrow \textit{"un chat est un félin"}$$

Mais on ne peut pas séparer complètement la connaissance de base de l'information ajoutée par la phrase, car il faut que ces deux éléments puissent interagir entre eux afin de pouvoir avoir des implications comme :

"tous les animaux mangent" ⇒ "tous les chats mangent"

Un moyen de résoudre ces problèmes est de définir l'implication de la manière suivante, lorsque l'on a des quantificateurs :

Propriété 3 *On a $e_1 \Rightarrow_T e_2$, avec e_1 et e_2 des représentations sémantiques de deux textes, et $\Rightarrow_T$ qui représente l'implication textuelle et T la T-Box représentant notre connaissance de base, si et seulement si il existe T' tel que $T' \subseteq T$, et que $T' \not\models e_2$ et $T' + e_1 \models e_2$ (et donc que $T' \not\models e_1$ qui peut être déduit des deux formules précédentes).*

Le problème est alors de savoir comment trouver la T-Box T' si elle existe, sans avoir à tester que $T' \not\models e_2$ et que $T' + e_1 \models e_2$ pour toutes les T-Box incluses dans T. Pour le moment nous n'avons pas encore de solutions, mais nous travaillons dessus.

8.1.2 Groupes

Il faut différencier la notion de groupe et la notion de généralisation, car certains verbes peuvent avoir en argument un groupe d'invididus. Par exemple dans *"Les étudiants se sont rassemblés dans la cour"*, *Les étudiants* fait référence a un groupe d'étudiants qui est dans la cour et cela n'impliquera pas que *"Un étudiant s'est rassemblé dans la cour"*. Par contre pour *"Les étudiants parlent allemand"*, on a une généralisation qui dit que si quelqu'un est étudiant alors il parle allemand, et on pourra inférer que *"Un étudiant parle allemand"*. L'implication suivante est donc fausse :

les Allemands ont envahi la France et Jean est allemand ⇒ Jean a envahi France

8.2 Travaux futurs

Le système permettant de représenter toutes les phrases de la langue française, et de détecter les implications entre elles est encore très loin d'être atteint, car il reste encore beaucoup de problémes courants à résoudre comme la disjonction, le temps et la modalité.

8.2.1 Disjonction (A-Box booléenne)

Le problème de la disjonction est que dans la langue francaise elle est exclusive, et donc il faut parfois faire le choix sur l'existence d'individus.

Par exemple dans la phrase *"Jean mange une poire ou Pierre mange une pomme"* on ne peut pas savoir si il faut mettre un individu `j` :`JEAN` ou un individu `p` :`PIERRE` dans la A-Box, et on ne peut pas mettre les deux car on ne veut pas que cela implique la phrase *"Jean mange une poire et Pierre mange une pomme"*.

Une solution pour résoudre ce problème, est d'utiliser les A-Box booléennes [Are03] qui permettent justement de représenter ce type de disjonction en LD. Cela permet aussi de représenter les phrases du type *si ...alors ...sinon*. Les A-Box booléennes permettent d'avoir de représenter des conditions sur les assertions, comme dire que si `j` est `JEAN` alors il existe un individu `c` qui est `CAROLINE`.

8.2.2 Temps et modalité

Le temps et la modalité sont des problèmes complexes qui compliquent nettement la représentation et la détection de l'implication.

Pour le temps, on peut avoir un individu qui a des états différents au cours du temps, et ces changement d'état peuvent impliquer une action et inversement. Par exemple les phrases *"Jean était assis, mais maintenant il est debout"* et *"Jean s'est mis debout"* sont équivalentes.

Pour la modalité, c'est un peu le même problème que pour le temps car on a aussi plusieurs mondes différents mais qui sont liés d'une manière différente. En effet pour le temps, les mondes sont liés d'une manière assez linéaire, le monde n+1 est égal au monde n avec les changements qui ont été faits entre n et n+1, alors que pour la modalité c'est plus une liaison d'inclusion que l'on a (e.g. le monde réel est inclus dans l'ensemble des mondes possibles). Mais après il y a aussi tous les mondes imaginaires qui eux ne sont liés à rien et qui peuvent même outrepasser la connaissance de base (e.g. on peut avoir un chien qui conduit une voiture).

Certains travaux ont étés réalisés pour représenter ces éléments, comme [Wol99] pour la modalité [Art99] pour le temps, mais aucun n'a encore été implémenté.

9 Conclusion

Pour conclure, nous pouvons dire que le système mis en place est pour le moment limité par l'expressivité de la représentation qui ne permet pour le moment de représenter qu'un petit fragment des phrases de la langue francaise. Mais pour les phrases que l'on peut traiter, le mécanisme d'implication fonctionne plutôt bien à part pour quelques exceptions (e.g. les individus implicites). Pour les fragments restant de la langue, nous avons lancé quelques idées pour les intégrer à notre représentation, et pour le moment nous ne voyons pas d'éléments de la langue qui seraient impossible à représenter ou qui seraient en contradiction avec la représentation actuelle. Il parait donc pensable de pouvoir représenter pratiquement toutes les phrases de la langue francaise avec une représentation basée sur les logiques de descriptions. Les travaux qui pourraient être effectués pour compléter cette approche, seraient :

- de développer un analyseur syntaxique prenant en entrée un texte et renvoyant en sortie la représentation sémantique associée au texte,
- d'automatiser la transformation de WordNet, et de FrameNet en logique de description,
- d'utiliser les A-Box booléennes afin de pouvoir représenter la disjonction,
- d'utiliser les logiques hybrides, pour pouvoir mieux représenter certaines problèmes de représentation comme de pouvoir dire que si deux individus ont le même père alors ils sont frères.

Bibliographie

[Are03] C. Areces, P. Blackburn, B. Martinez Hernandez et M. Marx. *Handling Boolean ABoxes*. Dans D. Calvanese, G. De Giacomo et E. Franconi, rédacteurs, *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, tome 81 de *CEUR - Workshop Proceedings*. Rome, Italy, September 2003.

[Art99] A. Artale et E. Franconi. *Introducing Temporal Description Logics*. Dans *TIME*, pages 2–5. 1999.

[Baa03] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi et P. F. Patel-Schneider, rédacteurs. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2003. ISBN 0-521-78176-0.

[Bec04] S. Bechhofer, V. Haarslev, C. Lutz et R. Möller, rédacteurs. *KI-2004 Workshop on Applications of Description Logics (ADL'04)*. 2004. `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-115/` Available at CEUR Workshop Proceedings.

[Cru86] D. A. Cruse. *Lexical Semantics*. Cambridge University Press,, Cambridge, 1986.

[Dav80] D. Davidson. *Essays on Actions and Events*. Clarendon, Oxford, 1980.

[Hor00] I. Horrocks, U. Sattler et S. Tobies. *Reasoning with Individuals for the Description Logic SHIQ*. Dans D. MacAllester, rédacteur, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, numéro 1831. Springer Verlag, Germany, 2000.

[Hor05] I. Horrocks, U. Sattler et F. Wolter, rédacteurs. *Proceedings of the 2005 International Workshop on Description Logics (DL2005), Edinburgh, Scotland, UK, July 26-28, 2005*, tome 147 de *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[Jan01] T. M. V. Janssen. *Frege, Contextuality and Compositionality*. *J. of Logic, Lang. and Inf.*, *tome 10*(1) :115–136, 2001. ISSN 0925-8531.

[Lin98] D. Lin. *Review of WordNet An Electronic Lexical Database*, 1998.

[Low97] J. Lowe, C. Baker et C. Fillmore. *A frame-semantic approach to semantic annotation*, 1997.

[Min74] M. L. Minsky. *A Framework for Representing Knowledge. Report A. I. MEMO 306*, Massachusetts Institute of Technology, A.I. Lab., Cambridge, Massachusetts, juin 1974. McGraw-Hill, P. H. Winston (Ed.), "Psychology of Computer Vision", 1975.

[Mon70] R. Montague. *English as a Formal Language*, chapitre Linguaggi nella Societa e nella Tecnica, B. Visentini et al eds, pages 189–224. Edizioni di Communita, Milan, 1970.

[Par97] B. H. Partee. *Montague semantics*. Dans J. van Benthem et A. ter Meulen, rédacteurs, *Handbook of Logic and Language*, pages 5–91. MIT Press and North-Holland, Cambridge, MA and Amsterdam, 1997.

[Qui88] M. R. Quillian. *Semantic Memory*. Dans A. Collins et E. E. Smith, rédacteurs, *Readings in Cognitive Science : A Perspective from Psychology and Artificial Intelligence*, pages 80–101. Kaufmann, San Mateo, CA, 1988.

[Wol99] F. Wolter et M. Zakharyaschev. *Modal Description Logics : Modalizing Roles*. *Fundamenta Informaticae*, *tome 39*(4) :411–438, 1999.