A Framework for Integrating $\mathcal{ALN}$-Theories with Recursive Definitions

Aida Vitória and Margarida Mamede

Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2825–114 Caparica, Portugal
{av,mm}@di.fct.unl.pt

Abstract. In this paper we focus on the problem of integrating knowledge bases expressed in a description logic. To this end, we propose three basic operations: *union*, *intersection* and *renaming*. First, the semantics of these compositional operations is studied abstracting away from implementation details. Then, we present an implementation of the proposed operations, for knowledge bases expressed in the language $\mathcal{ALN}$ extended with recursive definitions of concepts, which transforms compositions of knowledge bases into knowledge bases. This transformation is sound and complete with respect to the semantics referred before.

Keywords: Description Logics, Knowledge Representation and Reasoning, Compositional Operators.

1 Introduction

The possibility of re-using and composing existing small-sized software units is becoming an increasingly important issue in the construction of new computing systems. Since knowledge bases are important components of many computing systems, the problem of combining knowledge bases arises naturally.

Suppose that the knowledge of several experts about some domain is encoded in separate knowledge bases. An interesting problem is how these knowledge bases can be combined. We might be interested, for instance, in obtaining the common conclusions at which the experts have arrived, or in combining their knowledge in order to obtain new knowledge that none of the experts had.

The main goal of this paper is to propose a basic set of compositional operators that can be used to combine separate concept-based knowledge bases, i.e., knowledge bases written in a concept description language (or description logic). In particular, we focus our attention on the language $\mathcal{ALN}$, extended with recursive definitions of concepts.

To the best of our knowledge, the problem of integrating concept-based knowledge bases has been addressed for the first time in [23, 22]. In that work, knowledge bases are expressed in a very simple, tractable language called $\mathcal{AL}$, which is usually considered as a core-language of a family of well-studied concept description languages [13]. This family of languages is obtained by extending $\mathcal{AL}$ with other constructs. Since $\mathcal{ALN}$ is one of the most powerful extensions of $\mathcal{AL}$ with tractable reasoning procedures, it seemed to us to be a suitable candidate to be studied from the point of view of integrating description logic theories. A closer look to the language $\mathcal{ALN}$ has revealed some interesting problems (described below) that are not raised with $\mathcal{AL}$.

Most of the research done in this area aims at characterizing description logics, either from the point of view of their expressive power [10, 16], or from the point of view of the computational complexity of the corresponding deductive system [5, 13, 14, 17, 19–21]. Recently, new directions in this field have been explored. In [12], for instance, a description logic is used to integrate information in the context of distributed databases and data warehousing. The authors propose a declarative methodology for information integration. A description logic with n-ary relations is used to model data from multiple sources. Thus, it is possible to build a unified representation of the data and exploit the reasoning techniques of the underlying description logic when querying the global information system.

In this paper, we are concerned with information integration in the area of knowledge representation. Concept-based knowledge bases are seen as logical theories that can be merged (or composed). To this end, we propose three basic operations for composing knowledge bases: *union* ($\vee$), *intersection* ($\wedge$) and *renaming* (RN). Let Σ_1 and Σ_2 be two knowledge bases. $\Sigma_1 \vee \Sigma_2$ represents a knowledge base containing the logical consequences of Σ_1 or Σ_2. The *intersection*, $\Sigma_1 \wedge \Sigma_2$, captures the set of common logical consequences of Σ_1 and Σ_2. Finally, *renaming* is a binary operator, $RN(\Sigma_1, \rho)$, where ρ is a renaming function, that replaces names of relations or concepts in Σ_1 by some other ones according to ρ.

We characterize the above operators in two distinct ways. Firstly, we present a semantics that abstracts away from implementation details. Secondly, we define a syntactic transformation τ, from knowledge bases expressions to knowledge bases, for representing the operational behaviour. We show that the intersection of two $\mathcal{ALN}$-knowledge bases may have an infinite number of logical consequences that can only be captured in a finite way by concepts defined recursively. Although the intersection of two knowledge bases expressed in $\mathcal{AL}$ may also have an infinite number of logical consequences, there is no need for recursive definitions (as it is shown in [23, 22]).

It is worth referring that we make use of notions such as "least common subsumer" and "most specific concept" (presented in [3, 15]) to define the operational semantics of the compositional operators, whereas the operational semantics defined in [23, 22], for knowledge bases expressed in $\mathcal{AL}$, relies on a completely different technique: completion rules.

Furthermore, it is important to mention that compositional operators have also been investigated in the context of logic programming [8, 11], for combining not only definite programs [9], but also general logic programs [6, 7].

The rest of the paper is organized as follows. Section 2 is dedicated to the concept description language: Section 2.1 is concerned with the syntactic aspects and Section 2.2 presents its semantics. Section 3 is devoted to the declarative semantics of the compositional operations and several examples are included to motivate the need and usefulness of the proposed operators. Since the operational semantics of the compositional operators relies on the notions of lcs and msc, we introduce these notions and some related theoretical results in Section 4: Section 4.1 and Section 4.2 present the lcs and the msc, respectively (see [3, 15] for a full description). Next, Section 5 is concerned with the syntactic transformation τ, from knowledge bases expressions to knowledge bases. Finally, Section 6 contains some concluding remarks and possible future extensions.

2 The Language $\mathcal{ALN}$: Basic Definitions

2.1 Syntax

The language used in this work is $\mathcal{ALN}$ [13] extended with cyclic concept descriptions.

Let (Conc, Rel, Ind) be a tuple of pairwise disjoint sets of *primitive concepts*, Conc, *primitive binary relations*, Rel, and *individuals*, Ind. The set of primitive concepts Conc has two disjoint subsets: the set of *atomic concepts*, Atom, and the set of *defined concepts*, Def, such that $\mathrm{Conc} = \mathrm{Atom} \cup \mathrm{Def}$. In the sequel, we assume that $P \in \mathrm{Conc}$, $A \in \mathrm{Atom}$, $D \in \mathrm{Def}$, $R \in \mathrm{Rel}$, and $i \in \mathrm{Ind}$.

An $\mathcal{ALN}$-*terminological axiom* χ is defined by the following grammar:

$$\begin{array}{ll} \chi & \longrightarrow D \doteq C_t \\ C_t & \longrightarrow \top \mid \bot \mid D \mid A \mid \neg A \mid C_{t1} \sqcap C_{t2} \mid \forall R.C_t \mid (\leq n_1\, R) \mid (\geq n_2\, R) \end{array} ,$$

where n_1 is a non-negative integer and n_2 is a positive integer.

A TBox $\mathcal{T}$ is a finite and nonempty sequence of terminological axioms such that

- each defined concept D appears at most once on the left-hand side of a terminological axiom (i.e., each definition is unique), and
- every defined concept that occurs on the right-hand side of a terminological axiom $\chi_1 \in \mathcal{T}$, occurs also on the left-hand side of a terminological axiom $\chi_2 \in \mathcal{T}$.

A concept D *is defined in a TBox* $\mathcal{T}$ iff it occurs on the left-hand side of a terminological axiom $\chi \in \mathcal{T}$.

An $\mathcal{ALN}$-*concept* C and an $\mathcal{ALN}$-*assertion* β are defined by the following grammar:

$$\begin{array}{ll} C & \longrightarrow \top \mid \bot \mid P \mid \neg A \mid C_1 \sqcap C_2 \mid \forall R.C \mid (\leq n_1\, R) \mid (\geq n_2\, R) \\ P & \longrightarrow A \mid (D, \mathcal{T}) \\ \beta & \longrightarrow i : C \mid R(i_1, i_2) \end{array} ,$$

where n_1 is a non-negative integer, n_2 is a positive integer, and D is a concept defined in the TBox $\mathcal{T}$. Concepts of the form $(\leq n_1\, R)$ or $(\geq n_2\, R)$ are usually called *numeric restrictions*.

Notice that we do not preclude the existence of cyclic terminological axioms. Assume that $\mathcal{T}$ is a TBox, $D \doteq C \in \mathcal{T}$, and P is a primitive concept. Then, D *directly uses* P iff P occurs in C. Now, let *uses* be the transitive closure of the relation *directly uses*. We say that D is *recursively defined* iff D uses D.

A *knowledge base* Σ is a finite set of assertions.

Each defined concept D occurring in a knowledge base Σ is coupled to its definition, i.e., is associated with a TBox $\mathcal{T}$ in which it is defined. We represent this concepts as a pair $(D, \mathcal{T})$. Thus, these definitions have a flavour of "local definitions".

2.2 Semantics

The semantics of description logic theories is defined model-theoretically: a concept represents a set of elements of a given domain, and a relation denotes a set of pairs of elements of that domain. Intuitively, a concept may denote: the domain of discourse ($\top$); the empty set ($\bot$); the set of elements that belong to a primitive concept (P) or to the complement of an atomic concept ($\neg A$); the intersection of two concepts ($C_1 \sqcap C_2$); the set of elements whose image by a relation is contained in a concept ($\forall R.C$); or the set of elements whose image by a relation contains at most n_1 elements ($(\leq n_1\, R)$, or at least n_2 elements ($(\geq n_2\, R)$).

An assertion β states that an individual belongs to a concept ($i : C$), or that two individuals are related by some relation ($R(i_1, i_2)$). The meaning of a terminological axiom $D \doteq C$ is that concepts D and C represent the same set.

Example 1 Let Σ_{fam} be a knowledge base, where `son`, `daughter`, and `child` are relations; `mother` is a defined concept; `woman` is an atomic concept; and `Joe`, `Fred`, `Ann`, and `Kate` are individuals.

Σ_{fam} :	$\mathtt{son}(\mathtt{Joe}, \mathtt{Fred})$	`Fred` is `Joe`'s `son`,
	$\mathtt{daughter}(\mathtt{Fred}, \mathtt{Ann})$	`Ann` is `Fred`'s `daughter`,
	$\mathtt{daughter}(\mathtt{Ann}, \mathtt{Kate})$	`Kate` is `Ann`'s `daughter`,
	$\mathtt{Fred} : \forall \mathtt{child}.(\mathtt{mother}, \mathcal{T}_m)$	All children of `Fred` are mothers,

where

$$\mathcal{T}_m = < \mathtt{mother} \doteq \mathtt{woman} \sqcap (\geq 1\mathtt{child}) >,$$

which can be read as "Mothers are women with children". $\diamondsuit$

To simplify the remaining of this section, let us assume that $(\mathrm{Conc}, \mathrm{Rel}, \mathrm{Ind})$ is a fixed alphabet. An *interpretation* of $\mathcal{ALN}$ is a pair $I = (\Delta^I, \cdot^I)$, where Δ^I is a nonempty set, called the *interpretation domain*, and $\cdot^I$ is the *interpretation function* that satisfies the following conditions:

– $i^I \in \Delta^I$, for every individual $i \in \mathrm{Ind}$;

– $i_1 \neq i_2 \Rightarrow i_1^I \neq i_2^I$, for every individuals $i_1, i_2 \in \text{Ind}$;
– $A^I \subseteq \Delta^I$, for every atomic concept $A \in \text{Atom}$; and
– $R^I \subseteq \Delta^I \times \Delta^I$, for every relation $R \in \text{Rel}$.

The interpretation function is extended to arbitrary concepts but the defined ones as follows, where $R^I(a) = \{b \in \Delta^I \mid (a,b) \in R^I\}$:

$$\begin{aligned}
\top^I &= \Delta^I \\
\bot^I &= \emptyset \\
(\neg A)^I &= \Delta^I \setminus A^I \\
(C_1 \sqcap C_2)^I &= C_1^I \cap C_2^I \\
(\forall R.C)^I &= \{a \in \Delta^I \mid R^I(a) \subseteq C^I\} \\
(\leq n_1\, R)^I &= \{a \in \Delta^I \mid \#R^I(a) \leq n_1\} \\
(\geq n_2\, R)^I &= \{a \in \Delta^I \mid \#R^I(a) \geq n_2\}\ .
\end{aligned}$$

There are different kinds of semantics, in the presence of recursive terminologies (c.f. [1, 18]): the descriptive semantics, the greatest fixed point semantics, and the least fixed point semantics. As it is usual in the cases similar to ours, we shall adopt the greatest fixed point semantics, in that it is the one that best captures the meaning of the recursive definitions.

The intuitive idea behind this semantics is to see a TBox

$$\mathcal{T} = < D_1 \doteq C_1, D_2 \doteq C_2, \ldots, D_n \doteq C_n >$$

as a system of equations, defining the concepts $D_1, D_2, \ldots, D_n$. Let I be an interpretation. We can look at the right-hand side of each terminological axiom, C_k, as a function $\mathcal{T}_{I,k}$ that, given n sets $X_1, X_2, \ldots, X_n$, acting as if they were $D_1^I, D_2^I, \ldots, D_n^I$, returns the set C_k^I. Note that if we apply simultaneously the same input $X_1, X_2, \ldots, X_n$ to the n functions $\mathcal{T}_{I,1}, \mathcal{T}_{I,2}, \ldots, \mathcal{T}_{I,n}$, we obtain n new sets $C_1^I, C_2^I, \ldots, C_n^I$ in the output. Therefore, $\mathcal{T}$ and I may also be seen as characterizing a function $\mathcal{T}_I$ whose fixed points are the solutions of the system of equations. Roughly speaking, the interpretation of $D_1, D_2, \ldots, D_n$ we are interested in is given by the greatest fixed point of this function $\mathcal{T}_I$. Now we will formalize these notions.

Let $\mathcal{T} = < D_1 \doteq C_1, \ldots, D_n \doteq C_n >$ be a TBox and I be an interpretation. For every n-tuple $\boldsymbol{x} = (X_1, ..., X_n)$ of subsets of Δ^I, let $I[\boldsymbol{x}]$ denote the interpretation I that associates the set X_k with the defined concept D_k, i.e., $D_k^{I[\boldsymbol{x}]} = X_k$, for every $k = 1, \ldots, n$. Then, the function

$$\begin{gathered}
\mathcal{T}_I : (\wp(\Delta^I))^n \longrightarrow (\wp(\Delta^I))^n \\
\mathcal{T}_I(\boldsymbol{x}) = (C_1^{I[\boldsymbol{x}]}, \ldots, C_n^{I[\boldsymbol{x}]})
\end{gathered}$$

is called the *function defined by $\mathcal{T}$ and I.*

We may ask if, for every TBox $\mathcal{T}$ and every interpretation I, the function defined by $\mathcal{T}$ and I has a greatest fixed point. Notice that the answer is yes if the functions $\mathcal{T}_I$, defined on complete lattices, are monotonic. Of course this property depends on the underlying concept language but, in our case, the result achieved by Nebel ([18]) for the language $\mathcal{NTF}$ allows us to conclude that monotonicity holds. The next definition makes use of this fact.

Let I be an interpretation and D be a concept defined in a TBox $\mathcal{T}$. The interpretation I is extended to D — in the context of $\mathcal{T}$ — and to $(D, \mathcal{T})$ — in the context of any assertion — as follows:

$$D^I = (D, \mathcal{T})^I = D^{I[\nu]},$$

where ν is the greatest fixed point of the function defined by $\mathcal{T}$ and I.

An interpretation I *satisfies* an assertion $i : C$ iff $i^I \in C^I$; and *satisfies* $R(i_1, i_2)$ iff $(i_1^I, i_2^I) \in R^I$.

A knowledge base Σ is *satisfiable* if there is an interpretation I that satisfies all of its assertions. In this case, I is said to be a *model* of Σ. When Σ is not satisfiable, we say that Σ is *unsatisfiable*. Two knowledge bases are said to be *equivalent* if and only if they have the same models.

Furthermore, an assertion β is a *logical consequence* of a knowledge base Σ, written $\Sigma \models \beta$, if every model of Σ satisfies β. Notice that any assertion is a logical consequence of any unsatisfiable knowledge base and that equivalent knowledge bases possess exactly the same logical consequences.

3 Combining Knowledge Bases: Three Basic Operations

To begin with, let us present the declarative semantics of the compositional operators: *union* ($\vee$), *intersection* ($\wedge$), and *renaming* (RN).

So far, the letter Σ has always stood for a set of assertions forming a knowledge base. Nevertheless, in the sequel, we shall also use Σ to denote the name of a knowledge base.

Besides, a *renaming* ρ is a total function from $\mathrm{Atom} \cup \mathrm{Rel}$ to a set of atomic concepts and primitive relations, $\mathrm{Atom}' \cup \mathrm{Rel}'$, such that:

- Atom', Def, Rel', and Ind are pairwise disjoint
  (thus $(\mathrm{Atom}' \cup \mathrm{Def}, \mathrm{Rel}', \mathrm{Ind})$ is a valid alphabet); and
- $\mathrm{Atom} \cap \mathrm{Rel}' = \mathrm{Atom}' \cap \mathrm{Rel} = \emptyset$
  (which means that changes in the word type are not permitted).

Notice that defined concepts are not worth renaming due to their local nature.

Starting from a collection of $\mathcal{ALN}$-knowledge bases Σ and a collection of renamings ρ, we define the language of *$\mathcal{ALN}$-knowledge base expressions* E as follows:

$$E \longrightarrow \Sigma \mid RN(E, \rho) \mid E_1 \vee E_2 \mid E_1 \wedge E_2 \ .$$

Intuitively, the effect of renaming a knowledge base Σ via a renaming ρ (i.e., $RN(\Sigma, \rho)$) is that every occurrence of an atomic concept $A \in \mathrm{Atom}$ (resp., primitive relation $R \in \mathrm{Rel}$) in Σ is replaced by $\rho(A)$ (resp., $\rho(R)$). Obviously, the renaming has no practical effect whenever ρ is the identity function.

We also write $\rho = [d_1/d_1', \ldots, d_n/d_n']$ to make explicit that

– $\rho(d_i) = d'_i$, for all $i = 1, \ldots, n$, and
– $\rho(d) = d$, for every $d \notin \{d_1, \ldots, d_n\}$,

where $n \geq 1$, $\{d_1, \ldots, d_n\} \subseteq \text{Atom} \cup \text{Rel}$, and $\{d'_1, \ldots, d'_n\} \subseteq \text{Atom}' \cup \text{Rel}'$. Renamings are extended to arbitrary concepts, assertions, terminological axioms, and knowledge bases in the usual way.

The next step is to provide a compositional declarative semantics for knowledge base expressions. An $\mathcal{ALN}$-knowledge base expression E is characterized in terms of a set of $\mathcal{ALN}$-assertions, by induction on the structure of E:

$$\begin{aligned} RN(E, \rho) &= \rho(\{\beta \mid E \models \beta\}) \\ E_1 \vee E_2 &= \{\beta \mid E_1 \models \beta \text{ or } E_2 \models \beta\} \\ E_1 \wedge E_2 &= \{\beta \mid E_1 \models \beta \text{ and } E_2 \models \beta\}. \end{aligned}$$

Notice that, according to the semantics given before, an $\mathcal{ALN}$-knowledge base Σ is equivalent to the set of its logical consequences, $\{\beta \mid \Sigma \models \beta\}$.

At first sight, it seems that this definition can be easily rephrased in terms of models. If $\mathcal{M}(\Sigma)$ denotes the set of models of an $\mathcal{ALN}$-knowledge base Σ, one could be lead to believe, in particular, that $\mathcal{M}(\Sigma_1 \vee \Sigma_2) = \mathcal{M}(\Sigma_1) \cap \mathcal{M}(\Sigma_2)$ and $\mathcal{M}(\Sigma_1 \wedge \Sigma_2) = \mathcal{M}(\Sigma_1) \cup \mathcal{M}(\Sigma_2)$. Nevertheless, as the next example shows, this is not the case.

Example 2 Let $\Sigma_1 = \{R(i_1, i_2)\}$ and $\Sigma_2 = \{R(i_1, i_3)\}$. The set of common logical consequences of Σ_1 and Σ_2 is $\{i_1 : (\geq 1\, R)\}$. So, in our definition, the set of models associated with $\Sigma_1 \wedge \Sigma_2$ will be $\mathcal{M}(\{i_1 : (\geq 1\, R)\})$, to which the interpretation M such that $\Delta^M = \{i_1^M, i_2^M, i_3^M, a\}$ and $R^M = \{(i_1^M, a)\}$ belongs. But, $M \notin \mathcal{M}(\Sigma_1)$ and $M \notin \mathcal{M}(\Sigma_2)$, which implies $M \notin \mathcal{M}(\Sigma_1) \cup \mathcal{M}(\Sigma_2)$. $\diamondsuit$

Example 3 Consider a knowledge base, $\Sigma_{disease}$, that contains information on people with diseases. An excerpt of $\Sigma_{disease}$ is:

$$\begin{aligned} \Sigma_{disease} :\ & \texttt{Joe} : (\texttt{diseaseA}, \mathcal{T}_d) \\ & \texttt{Tom} : (\texttt{diseaseA}, \mathcal{T}_d) \\ & \ldots \end{aligned}$$

where $\mathcal{T}_d$ stands for

$$< \texttt{diseaseA} \doteq \texttt{abnA} \sqcap \forall \texttt{child}.(\texttt{healthy} \sqcap \forall \texttt{child}.\texttt{diseaseA}) > .$$

To some extent, someone is infected with `diseaseA` if he has abnormality `abnA`, all his `children` are `healthy`, but all grandchildren suffer from `diseaseA`.

It could be desirable to combine knowledge bases Σ_{fam} and $\Sigma_{disease}$, in order to get information about other people infected with diseases. This could be achieved, for instance, in the following way:

$$RN(\Sigma_{fam}, \rho) \ \vee \ \Sigma_{disease},$$

where $\rho = [\texttt{son}/\texttt{child}, \texttt{daughter}/\texttt{child}]$. Then, we would conclude that `Fred` and `Kate` are `healthy`, whereas `Ann` is infected with `diseaseA` and has the abnormality `abnA`. Note that family relations, like `son` and `daughter`, would be destroyed (e.g., $RN(\Sigma_{fam}, \rho) \ \vee \ \Sigma_{disease} \not\models \texttt{son}(\texttt{Joe}, \texttt{Fred})$). $\diamondsuit$

The previous example shows that the renaming operator can be used for two distinct purposes: to eliminate vocabulary differences from independently developed knowledge bases, or to implement some form of information protection strategies.

Example 4 Let Σ_{bus} and Σ_{train} be two knowledge bases that contain information about cities directly connected by `bus` and by `train`, respectively. An excerpt of Σ_{bus} could be:

$$\begin{array}{l} \texttt{bus}(\texttt{Montijo}, \texttt{Lisboa}) \\ \texttt{bus}(\texttt{Lisboa}, \texttt{Setubal}) \\ \ldots \end{array}$$

If we want to know whether a city b can be reached from a city a, by some means of transport, then we only need to check if assertion $b : \texttt{reachable}$ is a logical consequence of the following expression:

$$RN(\Sigma_{bus}, [\texttt{bus/link}]) \ \vee RN(\Sigma_{train}, [\texttt{train/link}]) \ \vee \Sigma,$$

where Σ is defined as follows

$$\Sigma = \{a : (\texttt{rea}, < \texttt{rea} \doteq \texttt{reachable} \sqcap \forall \texttt{link.rea} >)\}.$$

Now, suppose that, given any two cities, a and b, we would like to know whether there are the two types of connections from a to b: one only by bus and another one by train. In that case, we could define

$$\begin{array}{c} (RN(\Sigma_{bus}, [\texttt{bus/link}]) \vee \Sigma) \\ \wedge \\ (RN(\Sigma_{train}, [\texttt{train/link}]) \vee \Sigma), \end{array}$$

which would allow to easily retrieve the desired information. $\diamondsuit$

The next example aims at showing an interesting problem that the intersection of $\mathcal{ALN}$-knowledge bases may raise.

Example 5 Let us consider the knowledge bases Σ_1 and Σ_2, which contain results on several chemical experiments.

Σ_1	Σ_2
$\texttt{decompose}(\texttt{subsA}, \texttt{subsB}_1)$	$\texttt{decompose}(\texttt{subsA}, \texttt{subsB}_2)$
$\texttt{subsA}$:$(\leq 1 \ \texttt{decompose})$	$\texttt{subsA}$:$(\leq 1 \ \texttt{decompose})$
$\texttt{decompose}(\texttt{subsB}_1, \texttt{subsA})$	$\texttt{decompose}(\texttt{subsB}_2, \texttt{subsA})$
$\texttt{subsB}_1 : (\leq 1 \, \texttt{decompose})$	$\texttt{subsB}_2 : (\leq 1 \, \texttt{decompose})$
$\texttt{subsB}_1 : \texttt{unknown}$	$\texttt{subsB}_2 : \texttt{unknown}$
$\vdots$	$\vdots$

The intersection $\Sigma_1 \wedge \Sigma_2$ captures the common conclusions we may take from these experiments. For example, we may conclude, from any of these knowledge bases, that all substances in which substance `subsA` has been `decompose`d are `unknown`. That is to say, for every $k = 1, 2$,

$$\Sigma_k \models \mathtt{subsA} : \forall \mathtt{decompose.unknown},$$

because the following propositions hold, for every model M of Σ_k:

- $\mathtt{decompose}^M(\mathtt{subsA}^M) = \{\mathtt{subsB_k}^M\}$;
- $\mathtt{decompose}^M(\mathtt{subsB_k}^M) = \{\mathtt{subsA}^M\}$; and
- $\mathtt{subsB_k}^M \in \mathtt{unknown}^M$.

In addition, it is easy to conclude that any assertion of the form

$$\mathtt{subsA} : \underbrace{\forall \mathtt{decompose}.\forall \mathtt{decompose}.\cdots.\forall \mathtt{decompose}.}_{\text{an odd number of times}}\,\mathtt{unknown}$$

is also a logical consequence of Σ_1 and Σ_2 and, consequently, of $\Sigma_1 \wedge \Sigma_2$. But then, there is an infinite number of "non-equivalent" logical consequences of $\Sigma_1 \wedge \Sigma_2$ and the obvious question is how to build the corresponding $\mathcal{ALN}$-knowledge base. As a matter of fact, recursive definitions are crucial for coping with these cases. $\Sigma_1 \wedge \Sigma_2$, for instance, can be represented by the two following assertions:

$$\mathtt{subsA} : (\mathtt{C_1}, < \mathtt{C_1} \doteq \forall \mathtt{decompose.unknown} \sqcap \forall \mathtt{decompose}.\forall \mathtt{decompose.C_1} >)$$
$$\mathtt{subsA} : (\mathtt{C_2}, < \mathtt{C_2} \doteq (\leq 1\,\mathtt{decompose}) \sqcap (\geq 1\,\mathtt{decompose}) \sqcap \forall \mathtt{decompose.C_2} >).$$

◊

4 Least Common Subsumer and Most Specific Concept

The goal of this section is to present the notions of least common subsumer and most specific concept, which play an important role in the implementation of the compositional operators.

4.1 Least Common Subsumer

Let C_1, C_2, and C_3 be $\mathcal{ALN}$-concepts. We say that C_3 *subsumes* C_1, which is represented by $C_1 \sqsubseteq C_3$, iff $C_1^I \subseteq C_3^I$, for all interpretations I. Moreover, C_3 is a *least common subsumer* of C_1 and C_2, written $\mathrm{lcs}(C_1, C_2)$, iff the following two propositions hold:

(i) $C_1 \sqsubseteq C_3$ and $C_2 \sqsubseteq C_3$;
(ii) C_3 is the least $\mathcal{ALN}$-concept satisfying (i), i.e., if C_3' is an $\mathcal{ALN}$-concept such that $C_1 \sqsubseteq C_3'$ and $C_2 \sqsubseteq C_3'$, then $C_3 \sqsubseteq C_3'$.

Example 6 Let us consider again the knowledge base of Example 1. It is easy to verify that:

- $\mathrm{lcs}((\geq 1\,\mathtt{son}), (\geq 1\,\mathtt{daughter})) = \top$; and
- $\mathrm{lcs}(\mathtt{mother}, \mathtt{woman} \sqcap \forall \mathtt{child}.\, \mathtt{mother}) = \mathtt{woman}$.

Now, assume that

$$\mathcal{T}_1 = < D_1 \doteq \forall R_2.A \sqcap \forall R_1.D_1 >, \text{ and}$$
$$\mathcal{T}_2 = < D_2 \doteq A \sqcap \forall R_1.\forall R_2.D_2 > .$$

Then, $\mathrm{lcs}((D_1, \mathcal{T}_1), (D_2, \mathcal{T}_2) = \forall R_1.\forall R_2.A$. $\diamondsuit$

In [3, 15], it is shown that the least common subsumer of two $\mathcal{ALN}$-concepts always exists and can be computed in exponential time. This is done by means of an automata-theoretic characterization of the least common subsumer problem, in three steps. One starts by building some finite automata (which accept the languages denoted by $V_{C_1}(C_2)$, below); then one combines them, obtaining a finite automaton that accepts the least common subsumer of two concepts; and one translates back the automata into a concept (possibly making use of recursively defined concepts).

In order to define those auxiliary regular languages, let us assume that, in the following, a concept of the form $\forall R_1.\forall R_2.\cdots.\forall R_n.C$, with $n \geq 0$, is abbreviated by $\forall R_1 R_2 \cdots R_n.C$, where $R_1 R_2 \cdots R_n$ is a word over the alphabet Rel. If I is an interpretation and $W = R_1 R_2 \cdots R_n \in \mathrm{Rel}^*$, W^I denotes the composition of the binary relations $R_1^I \circ \cdots \circ R_n^I$. The empty word ε (when $n = 0$) is interpreted as the identity relation. Thus, $\forall \varepsilon.C$ and C represent the same concept.

Let C_1 be an arbitrary concept and C_2 be an atomic concept or a number restriction. Then,

$$V_{C_1}(C_2) = \{W \in \mathrm{Rel}^* \mid C_1 \sqsubseteq \forall W.C_2\}\,.$$

Example 7 Recall Example 5 and the knowledge base that represents $\Sigma_1 \wedge \Sigma_2$. It follows from the terminological axiom

$$\mathtt{C}_1 \doteq \forall \mathtt{decompose.unknown} \sqcap \forall \mathtt{decompose.}\forall \mathtt{decompose.C}_1\,,$$

that $V_{C_1}(\mathtt{unknown}) = \mathtt{decompose}\,(\mathtt{decompose}\ \mathtt{decompose})^*$. $\diamondsuit$

It turns out that the sets $V_{C_1}(C_2)$ may contain an infinite number of words, even when there are no recursive definitions, as the next example illustrates.

Example 8 Consider concept $C = \forall R_1.(\geq 2\,R_2) \sqcap \forall R_1.(\leq 1\,R_2)$ and let P be an atomic concept. For any interpretation I,

$$a \in C^I \Longrightarrow R_1^I(a) \subseteq (\geq 2\,R_2)^I \cap (\leq 1\,R_2)^I\,.$$

Besides, since $(\geq 2\,R_2)^I \cap (\leq 1\,R_2)^I = \emptyset$, we conclude that $R_1^I(a) = \emptyset$, and $R_1^I(a) \subseteq {C'}^I$, for every concept C'. So, in particular,

$$\begin{array}{l} C \sqsubseteq \forall R_1.P \\ C \sqsubseteq \forall R_1.\forall R_1.P \\ \quad\vdots \\ C \sqsubseteq \forall R_1.\cdots.\forall R_1.P, \end{array}$$

which implies $R_1 R_1^* \subseteq V_C(P)$.

Another immediate consequence is that, if

$$\Sigma_1 = \{i : \forall R_1.(\geq 2R_2) \sqcap \forall R_1.(\leq 1R_2)\} \;\text{ and }\; \Sigma_2 = \{i : \forall R_1.\forall R_1.P\},$$

$\Sigma_1 \wedge \Sigma_2 = \{i : \forall R_1.\forall R_1.P\}$. ♢

The construction of the final automaton, which lead us to the $\mathrm{lcs}(C_1, C_2)$, is based on the finite automata that accept the languages $V_{C_1}(C) \cap V_{C_2}(C)$, for all atomic concepts or number restrictions C.

4.2 Most Specific Concept

Let Σ be an $\mathcal{ALN}$-knowledge base, C be an $\mathcal{ALN}$-concept, and i be an individual. C is said to be the *most specific concept* for i w.r.t. Σ, written $\mathrm{msc}_\Sigma(i)$, iff:

(i) $\Sigma \models i : C$; and
(ii) C is the least concept satisfying condition (i), i.e., if C' is an $\mathcal{ALN}$-concept such that $\Sigma \models i : C'$, then $C \sqsubseteq C'$.

Example 9 Let $\Sigma = \{R(i,i), i : (\leq 1\,R)\}$. It is easy to see that i is an instance of any concept of the form $\forall R.\cdots.\forall R.((\leq 1\,R) \sqcap (\geq 1\,R))$. In order to represent all these concepts in a finite way, we need to rely on a recursive definition, such as:

$$C \doteq (\leq 1\,R) \sqcap (\geq 1\,R) \sqcap \forall R.C\,.$$

Thus, $\mathrm{msc}_\Sigma(i) = (C, \; < C \doteq (\leq 1\,R) \sqcap (\geq 1\,R) \sqcap \forall R.C >)$.

It turns out that, without recursive definitions, the $\mathrm{msc}_\Sigma(i)$ could not have been expressed in $\mathcal{ALN}$. ♢

Most specific concept can also be found by building finite automata. In this case, the construction relies on the sets $V_\Sigma(i, C)$, which Baader and his colleagues [3, 15] proved to be regular.

Let Σ be an $\mathcal{ALN}$-knowledge base and C be an atomic concept or number restriction. We define:

$$V_\Sigma(i, C) = \{W \in \mathrm{Rel}^* \mid \Sigma \models i : \forall W.C\}\,.$$

We shall end this section mentioning that both the knowledge base satisfiability problem and the instance problem can also have an automata-theoretic characterization (c.f. [3]).

5 Operational Semantics

In this section, we discuss a possible implementation of the proposed operations. We define a syntactic transformation τ that, given a knowledge base expression, produces an $\mathcal{ALN}$-knowledge base. This technique is comparable with the compilation-oriented implementation of the language of definite logic program expressions presented in [8]. Another alternative (not explored in this paper) is to implement the language of $\mathcal{ALN}$-knowledge base expressions without building any new concrete knowledge base.

The definition of τ relies on the following notion.

$$\begin{aligned}\Sigma_1 \nabla \Sigma_2 = \{&R(i_1, i_2) \in \Sigma_1 \cap \Sigma_2\} \\ &\cup \\ \{&i : C \mid i \text{ occurs in } \Sigma_1 \text{ and } \Sigma_2,\ \ C = \mathrm{lcs}(\mathrm{msc}_{\Sigma_1}(i), \mathrm{msc}_{\Sigma_2}(i))\}\ .\end{aligned}$$

It is easy to prove that $\Sigma_1 \models \beta$ and $\Sigma_2 \models \beta$, for every assertion $\beta \in \Sigma_1 \nabla \Sigma_2$.

The transformation τ can now be defined, by induction on the structure of the knowledge base expressions:

$$\begin{aligned}\tau(\Sigma) &= \Sigma \\ \tau(RN(E,\rho)) &= \rho(\tau(E)) \\ \tau(E_1 \vee E_2) &= \tau(E_1) \cup \tau(E_2) \\ \tau(E_1 \wedge E_2) &= \begin{cases} \tau(E_1)\nabla\tau(E_2) & \text{if } \tau(E_1) \text{ and } \tau(E_2) \text{ are both satisfiable} \\ \tau(E_1) & \text{if } \tau(E_1) \text{ is satisfiable and } \tau(E_2) \text{ is not} \\ \tau(E_2) & \text{if } \tau(E_2) \text{ is satisfiable and } \tau(E_1) \text{ is not} \\ \{i : \bot\} & \text{otherwise.} \end{cases}\end{aligned}$$

The definition of $\tau(E_1 \wedge E_2)$ requires four different cases. If $\tau(E_1)$ is unsatisfiable, then every assertion is a logical consequence of E_1. Consequently, the logical consequences of E_2 are the common logical consequences of E_1 and E_2, and $\tau(E_2)$ is returned. If neither $\tau(E_1)$ nor $\tau(E_2)$ are satisfiable, then every assertion is a logical consequence of E_1 and E_2. So, every assertion should also be a logical consequence of $\tau(E_1 \wedge E_2)$. This is captured by any knowledge base containing an explicit contradiction, i.e., an assertion of the form $i : \bot$.

The next proposition states the correctness of the proposed implementation.

Proposition 1. *Let E be an $\mathcal{ALN}$-knowledge base expression and β be $\mathcal{ALN}$-assertion. Then, $E \models \beta \Leftrightarrow \tau(E) \models \beta$.*

6 Conclusions and Future Work

The purpose of the work described in this paper was to introduce three operators (union, intersection and renaming) for composing knowledge bases expressed in the description logic $\mathcal{ALN}$. To this end, knowledge base expressions have been defined with a compositional semantics. A computational interpretation of knowledge base expressions has also been formalized through a transformation that maps knowledge base expressions into knowledge bases. A novel aspect of this work is that we have applied the notions of "least common subsumer" and "most specific concept" [3, 15] to the problem of integrating description logic theories.

Concerning future work, we aim at investigating the following open problems.

On the one hand, it is possible to extend both the underlying description logic and the set of compositional operators. We are working on an extension of $\mathcal{ALN}$ with transitive relations, inverse relations, and relation hierarchies (i.e., a set of axioms of the form $R_1 \sqsubseteq R_2$). In this case, the results on least common subsumers and most specific concepts, presented in [3, 15], have to be extended.

We have also considered the possibility of extending the language with relation compositions and relation hierarchies. The problem, however, is that the sets $V_\Sigma(i, C)$ may not be regular. In order to illustrate this point, let us see an example.

Example 10 Consider the following knowledge base Σ:

$$\begin{array}{l} R_1 \circ R_2 \circ R_3 \sqsubseteq R_2 \\ i : \forall R_2.A \end{array}$$

Since $(R_1 \circ R_2 \circ R_3)^M \subseteq R_2^M$, for every model M of Σ,

$$V_\Sigma(i, A) = \{R_1^n R_2 R_3^n \mid n \geq 0\} ,$$

which is a context free language. Moreover, it is well-known that the intersection of context free languages may not be a context free language. Hence, the intersection of knowledge bases expressed in a language that contains relation compositions and relation hierarchies is not a trivial operation. $\diamondsuit$

On the other hand, it is interesting to study alternative implementations. In this work, we have proposed a compilation oriented approach. Another possible implementation could be interpretation oriented so as not to need a new knowledge base to be built. The main advantage of this setting is that we could cater for distributed concept-based knowledge bases.

Finally, we would like to explore the possibility of applying the recent results of Baader and his colleagues on matching in description logics [2, 4], in order to (automatically) build renamings that match concept names in arbitrary concepts.

References

1. F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. Research Report RR-90-01, DFKI, Postfach 2080, D-6750 Kaiserslautern, Germany, 1990.
2. F. Baader, A. Borgida, and D. L. MacGuiness. Matching in description logics: Preliminary results. In M.-L. Mugnier and M. Chein, editors, *Proc. of the Sixth International Conference on Conceptual Structures (ICCS'98)*, Lecture Notes in Artificial Intelligence, 1998.

3. F. Baader and R. Küsters. Computing the least commom subsumer and the most specific concept in the presence of cyclic ALN-concept descriptions. LTCS-Report 98-06, Aachen University of Technology, Research Group for Theoretical Computer Science, Ahornstr. 55, D-52074 Aachen, Germany, 1998.
4. F. Baader and P. Narendran. Unification of concepts in description logics. In *Proc. of DL'97*, 1997.
5. R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In R. J. Brachman, editor, *Proc. of the 4th National Conference on Artificial Intelligence (AAAI-84)*, pages 34–37. AAAI Press / MIT Press, 1984.
6. A. Brogi, S. Contiero, and F. Turini. Composing general logic programs. In J. Dix, U. Furbach, and A. Nerode, editors, *Proc. of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97)*, volume 1265 of *Lecture Notes in Artificial Intelligence*, pages 273–288. Springer-Verlag, 1997.
7. A. Brogi, S. Contiero, and F. Turini. The use of renaming in composing general programs. In P. Flener, editor, *Proc. of the 8th International Workshop on Logic Programming Synthesis and Transformation (LOPSTR'98)*, volume 1559 of *Lecture Notes in Computer Science*, pages 124–142. Springer-Verlag, 1999.
8. A. Brogi, P. Mancarella, D. Pedreschi, and F. Turini. Modular logic programming. *ACM Transactions on Programming Languages and Systems*, 16(4):1361–1398, 1994.
9. A. Brogi, C. Renso, and F. Turini. Dynamic composition of parameterised logic programs. In P. Hill and A. Brogi, editors, *Proc. of the Post Conference Workshop on Logic-Based Composition of Software for the ICLP'97 (LOCOS'97)*, 1997.
10. M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. In R. Bajcsy, editor, *Proc. of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 704–709. Morgan Kaufmann, 1993.
11. M. Bugliesi, E. Lamma, and P. Mello. Modularity in logic programming. *Journal of Logic Programming*, 19&20:443–502, 1994.
12. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Proc. of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13. Morgan Kaufmann, 1998.
13. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.
14. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: from subsumption to instance checking. *Journal of Logic and Computation*, 4(4):423–452, 1994.
15. R. Küsters. Characterizing the semantics of terminological cycles in $\dashv\updownarrow\backslash$ using finite automata. In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Proc. of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 499–510. Morgan Kaufmann, 1998.
16. M. Mamede and L. Monteiro. $\mathcal{ALC}$rn: A decidable terminological language with role negation. In C. Rowles, H. Liu, and N. Foo, editors, *Proc. of the 6th Australian Joint Conference on Artificial Intelligence (AI'93)*, pages 229–235. World Scientific Publishing, 1993.
17. B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988.

18. B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1990.
19. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, 1990.
20. A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. In H. J. Komorowski and Z. W. Raś, editors, *Proc. of the 7th International Symposium on Methodologies for Intelligence Systems (ISMIS'93)*, volume 689 of *Lecture Notes in Artificial Intelligence*, pages 508–517. Springer-Verlag, 1993.
21. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
22. A. Vitória and M. Mamede. Integrating concept-based knowledge bases (extended abstract). In *Proc. of DL'99*.
23. A. Vitória and M. Mamede. Integrating concept-based knowledge bases. Technical Report UNL-DI-02-99, Dep. Informática, Universidade Nova de Lisboa, Quinta da Torre, 2825–114 Caparica, Portugal, 1999.