TECHNISCHE UNIVERSITÄT DRESDEN

Dresden University of Technology
Computer Science Institution
International Master Programme in Computational Logic

Master's Thesis

Matching in Description Logics with Existential Restrictions and Terminological Cycles

Hongkai Liu

Overseeing Professor: Prof. Dr. Franz Baader
Supervisor: Sebastian Brandt

March, 2005

Author's e-mail: s9549913@mail.inf.tu-dresden.de

TECHNISCHE UNIVERSITÄT DRESDEN

Author:	**Hongkai Liu**
Matrikel-Nr.:	**2990570**
Title:	**Matching in Description Logics with Existential Restrictions and Terminological Cycles**
Degree:	**Master of Science**
Date of submission:	**17 March 2005**

Declaration
Hereby I certify that the thesis has been written by me. Any help that I have received in my research work has been acknowledged. Additionally, I certify that I have not used any auxiliary sources and literature except those I cited in the thesis.

Signature of Author

献给亲爱的母亲，愿母亲永远健康，幸福。

Abstract

Matching of concepts against patterns is a so-called non-standard inference problem [Küs01] in Description Logics. For the small description language $\mathcal{EL}$, matching problems without terminological cycles have been investigated in [BK00a]. In the present thesis we introduce $\mathcal{EL}$-matching problems allowing terminological cycles. Among the three different semantics defined by Nebel in [Neb91] for the interpretation of cyclic TBoxes we will argue that gfp-semantics is the appropriate one to define matching problems. Based on deciding subsumption [Baa02b] and computing the least common subsumers [Baa02a], a matching algorithm is provided whose soundness and completeness is shown. Moreover, the matching algorithm is implemented and tested in the programming language LISP.

Acknowledgements

I would like to first express gratitude and appreciation to Sebastian Brandt. He helps me not only in this master's thesis but also in my seminar and project, not only in the scientific research but also in technique of programming in LISP and LaTeX. Without his supervision, I could not have finished this thesis. I feel grateful with his patience for his providing answers of my questions and criticisms of my mistakes.
Thank Professor Franz Baader for giving me the topic of the thesis and trust my research ability. Also many thanks should go to Doctor Carsten Lutz who gave the lectures of Logic-based Knowledge Representation. These lectures introduced me to the world of Description Logics. I would like to thank Boontawee Suntisrivaraporn as well for implementing algorithm of subsumption which is used in the implementation of my matching algorithm.
I am thankful to my family for giving me spiritual and financial support on my study. They keep encouraging me all the time. I love you so much.

Hongkai Liu

Contents

Abstract **i**

Acknowledgements **iii**

1 Motivation **1**

1.1 Why do we do matching? 1

1.2 Why do we focus on $\mathcal{EL}$ interpreted with gfp-semantics? 2

1.3 The structure of this thesis 3

2 Cyclic $\mathcal{EL}$-TBoxes **5**

2.1 The description logic $\mathcal{EL}$ 5

2.2 Subsumption w.r.t. gfp-semantics 7

2.3 The least common subsumers 9

3 Matching in Cyclic $\mathcal{EL}$-TBoxes **11**

3.1 Introducing variables 11

3.2 $\mathcal{EL}$-matching problems 12

4 Solving $\mathcal{EL}$-matching problems **15**

4.1 Matching algorithm 15

4.2 Termination 17

5 Soundness and completeness **19**

5.1 Soundness 19

5.2 Completeness 24

6 Implementation **29**

6.1 Input $C \equiv^?_{\text{gfp},\mathcal{T}} D$ and normalization of TBox 29

6.2 Computation of simulation relations 31

6.3 Computation of TBox $\mathcal{T}'$ 32
6.4 Subsumption testing and output 33
6.5 Testing 35

7 Conclusion 37

Bibliography 38

Chapter 1

Motivation

Description Logics (DLs) are a family of knowledge representation languages which use concept descriptions to represent knowledge. In DLs, concept descriptions are built from atomic concepts and roles with the help of constructors. The constructors determine the expressive power of the DL. Using concept descriptions and the symbol "≡" we can introduce concept definitions. For example (from [Baa02b]), tigers and lions can be defined in DL by following concept definitions:

$$\text{Tiger} \equiv \text{Animal} \sqcap \exists\text{parent}.\text{Tiger}$$
$$\text{Lion} \equiv \text{Animal} \sqcap \exists\text{parent}.\text{Lion}$$

Tigers (lions) are viewed as the animals whose parents are also tigers (lions). A set of such concept definitions is called terminology (TBox). We rule out that the same concept name is defined more than once (multiple definitions). In this thesis, we will consider the DL language $\mathcal{EL}$ which allows for the top-concept ($\top$), conjunction ($\sqcap$), and existential restriction ($\exists r.C$).

1.1 Why do we do matching?

DL-systems consist of two components. First, a knowledge base, which can further be divided into the TBox and the ABox. Second, a reasoning engine, which implements the various inference services. Matching in $\mathcal{EL}$, the central problem concerned in this thesis, is one of these inference services on TBoxes.
Research in the field of DLs has mainly been driven by inventing decision algorithms for so-called standard inference problems, such as subsumption and instance checking. Subsumption is used to describe the relationship between two concept descriptions. In the above example, we can see that every tiger has to be an animal, in other words, an object that is a tiger implies that it is also an animal. In this case, we say that 'Tiger' is subsumed by 'Animal'. If two concept descriptions subsume each other, then we say that they are equivalent. Deciding subsumption and equivalence are included in standard inference problems (see [Küs01]). However, building and maintaining large DL

knowledge bases requires additional support beyond the set of standard inference services (see [MPS98]). Matching is one of non-standard inference services which originally has been motivated by the problem of pruning large concept descriptions, i.e., only printing the relevant aspects under current circumstances [BKBM99]. In fact, matching has already been used successfully in some real applications of pruning concept descriptions (see [Küs01]). Matching can also be applied to help detect and avoid redundancies [BK00b] and to integrate knowledge bases [BS96]. Another application is that matching can be seen as a way to implement query in knowledge bases [BT01].

1.2 Why do we focus on $\mathcal{EL}$ interpreted with gfp-semantics?

Although $\mathcal{EL}$ is a relatively inexpressive DL, it appears to be adequate for some real applications. The Gene Ontology (see `http://www.geneontology.org`) can be represented in $\mathcal{EL}$ with an acyclic TBox. Some other examples of $\mathcal{EL}$'s applications are in the field of medical terminologies: SNOMED (see [KAS97]) and GALEN (see [RNG93]).
TBoxes without terminological cycles (concept definitions depending on themselves) are called acyclic TBoxes. Matching problems w.r.t. acyclic TBoxes in $\mathcal{EL}$ have already been studied in [BK00a]. The matching algorithm in [BK00a] is based on finding homomorphisms between description trees, computing the least common subsumers (see [BKM98]), and testing subsumption.
In this thesis, we will deal with matching problems in the case that we allow for cyclic definitions in TBoxes. We have three choices of semantics to interpret cyclic TBoxes introduced by Nebel [Neb91], namely descriptive semantics, gfp-semantics, and lfp-semantics. The descriptive semantics is defined as the usual semantics for acyclic TBoxes. Subsumption in cyclic TBoxes w.r.t. these three semantics are proven to be decidable in [Baa02b]. For lfp-semantics, deciding subsumption in a cyclic TBox can be reduced to subsumption in an acyclic one, where gfp-semantics, lfp-semantics, and descriptive semantics coincide (see [Neb91]). In this sense, using gfp-semantics for cyclic TBoxes in $\mathcal{EL}$ is more interesting than lfp-semantics.
For cyclic TBoxes, testing subsumption depends on finding simulation relations on the description graph of TBox as it is proven in [Baa02b]. Moreover, the least common subsumers in $\mathcal{EL}$ with cyclic terminologies interpreted with gfp-semantics always exist and can be computed (see [Baa02a]). For the computation of least common subsumers we might need to extend the original TBox. It is also stated in [Baa02a] if we choose descriptive semantics, the least common subsumers need not always exist.
All of these good computational properties of gfp-semantics make solving matching problems to be possible in cyclic $\mathcal{EL}$-TBoxes based on the same intuition as acyclic ones.
The objective of this thesis: a suitable definition for matching problems w.r.t. cyclic terminologies in the DL $\mathcal{EL}$ is to be examined. Soundness and completeness of an appropriate matching algorithm is to be shown formally. A prototype implementation of this algorithm has to be done in the programming language LISP.

1.3 The structure of this thesis

Chapter 2 introduces the language $\mathcal{EL}$ and cyclic TBoxes in $\mathcal{EL}$. We also give definitions and characterizations of two inference problems (subsumption and the least common subsumers) in $\mathcal{EL}$.
In Chapter 3 we extend the notion of concept descriptions to concept patterns in order to introduce variables into cyclic $\mathcal{EL}$-TBoxes. Then we define $\mathcal{EL}$-matching problems formally. In Chapter 4, our matching algorithm is introduced. Moreover, we show termination of this algorithm.
We concentrate on proving soundness and completeness of our matching algorithm in Chapter 5. In order to prove completeness, we will restrict our attention to "interesting" solutions.
In Chapter 6, we illustrate the implementation of our matching algorithm in the programming language LISP. Main data structures and intuitions of important functions are discussed. Then we explain the strategy of testing our matching algorithm.
In the last chapter, we give a summary of this thesis and briefly consider the complexity of the matching algorithm.

Chapter 2

Cyclic $\mathcal{EL}$-TBoxes

In this chapter, we introduce the description logic language $\mathcal{EL}$. All the definitions in this thesis are based on this language. Then we define TBoxes in $\mathcal{EL}$. The notion of "matching problems" is constructed on these definitions.

2.1 The description logic $\mathcal{EL}$

At first, we fix some sets which are necessary to define $\mathcal{EL}$-concept descriptions. Those sets will be used throughout this thesis.

- N_C is set of concept names,
- N_R is set of role names,
- $\mathcal{X}$ is set of variables.

Any two of these sets have no common elements, i.e., $N_C \cap N_R = \emptyset$, $N_C \cap \mathcal{X} = \emptyset$, and $N_R \cap \mathcal{X} = \emptyset$. Starting from these sets we can define concept descriptions inductively with the help of constructors.

Definition 1 [Syntax]
The *set of all concept descriptions* over N_C and N_R is inductively defined as following:

- Every concept name $A \in N_C$ (atomic concept) and $\top$ (top-concept) are concept descriptions;
- If C and D are concept descriptions, then $C \sqcap D$ (conjunction) is a concept description;
- If $r \in N_R$ is a role name and C is a concept description, then $\exists r.C$ (existential restriction) is a concept description.

A *terminology* (or *TBox* for short) is a finite set of concept definitions of the form $A \equiv D$, where A is a concept name and D is a concept description. In addition, we require that TBoxes do not contain *multiple definitions*, i.e., there cannot be two distinct concept descriptions D_1 and D_2 such that both $A \equiv D_1$ and $A \equiv D_2$ belong to the TBox. Concept names occurring on the left-hand side of a definition are called *defined concepts*. All other concept names occurring in the TBox are called *primitive concepts*. $\diamond$

Note: We allow for cyclic dependencies between the defined concepts, i.e., the definition of A may refer (directly or indirectly) to A itself.
The model-theoretic semantics of $\mathcal{EL}$ is defined by specifying a domain and an interpretation function.

Definition 2 [Semantics]
Let Δ^I be a non-empty set. An *interpretation* $\mathcal{I}$ is defined by its domain $\Delta^{\mathcal{I}}$ and its interpretation function $\cdot^{\mathcal{I}}$ which assigns $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each $A \in N_C$ and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each $r \in N_R$. The interpretation function is extended to concept descriptions in the following way.

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$;
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
- $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ if and only if it satisfies all its concept definitions, i.e., $A^{\mathcal{I}} = D^{\mathcal{I}}$ for all definitions $A \equiv D$ in $\mathcal{T}$. This semantics of TBoxes is called *descriptive semantics* by Nebel (see [Neb91]). $\diamond$

Nebel introduced three different semantics to cyclic TBoxes in description logic: descriptive semantics, least fixpoint (lfp) semantics, and greatest fixpoint (gfp) semantics. Before we define the lfp- and gfp-semantics, we recall some definitions given in [Baa02b].

Definition 3
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox containing the roles N_R, the primitive concepts N_{prim}, and the defined concepts $N_{\text{def}} := \{A_1, \ldots, A_k\}$.

- A *primitive interpretation* $\mathcal{J}$ for $\mathcal{T}$ is given by a domain $\Delta^{\mathcal{J}}$, an interpretation of the primitive concepts $P \in N_{\text{prim}}$ by subsets $P_{\mathcal{J}}$ of $\Delta^{\mathcal{J}}$, and an interpretation of the roles $r \in N_R$ by binary relations $r^{\mathcal{J}}$ on $\Delta^{\mathcal{J}}$.
- The interpretation $\mathcal{I}$ is *based on* the primitive interpretation if and only if it has the same domain as $\mathcal{J}$ and coincides with $\mathcal{J}$ on N_R and N_{prim}.
- We define
$$Int(J) := \{\mathcal{I} \mid \mathcal{I} \text{ is an interpretation based on } \mathcal{J}\}.$$
- If $\mathcal{I}_1, \mathcal{I}_2 \in Int(\mathcal{J})$, then
$$\mathcal{I}_1 \preceq_{\mathcal{J}} \mathcal{I}_2 \text{ if and only if } A_i^{\mathcal{I}_1} \subseteq A_i^{\mathcal{I}_2} \text{ for all } i, 1 \leq i \leq k.$$

$\diamond$

Using Tarski's fixpoint theorem (see [Tar55]), it is shown in [Neb91] that for a given primitive interpretation $\mathcal{J}$, there is always a greatest and a least (w.r.t. $\preceq_{\mathcal{J}}$) model of $\mathcal{T}$ based on $\mathcal{J}$. We call these models respectively the *greatest fixpoint model* (*gfp-model*) and the *least fixpoint model* (*lfp-model*) of $\mathcal{T}$. *Greatest* (*least*) *fixpoint semantics* considers only gfp-models (lfp-models) as admissible models.
In this thesis, we consider only greatest fixpoint semantics. In Section 2.2, we define subsumption between defined concepts. We can restrict the attention to subsumption between defined concepts since subsumption between arbitrary concept descriptions can be reduced to this problem by introducing definitions for descriptions.

2.2 Subsumption w.r.t. gfp-semantics

We first define relationships named "subsumption" and "equivalence" between defined concepts.

Definition 4 [Subsumption and equivalence]
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and A, B be defined concepts occurring in $\mathcal{T}$. Then

- A is *subsumed* by B w.r.t. gfp-semantics ($A \sqsubseteq_{\mathrm{gfp},\mathcal{T}} B$) iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all gfp-models $\mathcal{I}$ of $\mathcal{T}$.
- A is *equivalent* to B w.r.t. gfp-semantics ($A \equiv_{\mathrm{gfp},\mathcal{T}} B$) iff $A \sqsubseteq_{\mathrm{gfp},\mathcal{T}} B$ and $B \sqsubseteq_{\mathrm{gfp},\mathcal{T}} A$.

$\diamond$

According to this definition, the equivalence problem is decidable if the subsumption problem is decidable. Before we give the theorem of deciding subsumption, we introduce the notion of the normal form of $\mathcal{EL}$-TBoxes.

Definition 5 [Normal form of $\mathcal{EL}$-TBoxes]
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox, N_{def} the set of the defined concepts of $\mathcal{T}$, and N_{prim} the set of primitive concepts of $\mathcal{T}$. Then $\mathcal{T}$ is called *in formal form* if and only if $A \equiv D \in \mathcal{T}$ implies that D is of the form

$$P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \cdots \sqcap \exists r_l.B_l$$

for $m, l \geq 0$, $P_1, \ldots, P_m \in N_{\mathrm{prim}}$, $r_1, \ldots, r_l \in N_R$, and $B_1, \ldots, B_l \in N_{\mathrm{def}}$. If $m = l = 0$, then $D = \top$. $\diamond$

We use the TBoxes in normal form to generate the description graphs.

Definition 6 [$\mathcal{EL}$-description graphs]
An *$\mathcal{EL}$-description graph* is a graph $G = (V, E, L)$ where

- V is a set of nodes;

- $E \subseteq V \times N_R \times V$ is a set of edges labeled by role names;
- $L : V \to 2^{N_{\text{prim}}}$ is a function that labels nodes with sets of primitive concepts.

◇

The normalized TBox $\mathcal{T}$ can be translated into the following $\mathcal{EL}$-description graph $\mathcal{G}_\mathcal{T} = (N_{\text{def}}, E_\mathcal{T}, L_\mathcal{T})$:

- the nodes of $\mathcal{G}_\mathcal{T}$ are the defined concepts in $\mathcal{T}$;
- if A is defined concept and

$$A \equiv P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \cdots \sqcap \exists r_l.B_l$$

  is its definition in $\mathcal{T}$, then

  - $L_\mathcal{T}(A) = \{P_1, \ldots, P_m\}$, and
  - A is the source of the edges $(A, r_1, B_1), \ldots, (A, r_l, B_l) \in E_\mathcal{T}$.

Simulations are binary relations between nodes of two $\mathcal{EL}$-description graphs.

Definition 7 [Simulation]
Let $\mathcal{G}_i = (V_i, E_i, L_i)$ $(i = 1, 2)$ be two $\mathcal{EL}$-description graphs. The binary relation $Z \subseteq V_1 \times V_2$ is a *simulation* from $\mathcal{G}_1$ to $\mathcal{G}_2$ iff

(S1) $(v_1, v_2) \in Z$ implies $L_1(v_1) \subseteq L_2(v_2)$; and

(S2) if $(v_1, v_2) \in Z$ and $(v_1, r, v_1') \in E_1$, then there exists a node $v_2' \in V_2$ such that $(v_1', v_2') \in Z$ and $(v_2, r, v_2') \in E_2$.

We write $Z : \mathcal{G}_1 \rightleftarrows \mathcal{G}_2$ to express that Z is a simulation from $\mathcal{G}_1$ to $\mathcal{G}_2$. ◇

From [Baa02b], we know that the subsumption problem w.r.t. gfp-semantics can be decided by test the existence of simulation relations on the description graph.

Theorem 8
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and A, B defined concepts in $\mathcal{T}$. Then the following are equivalent:

1. $A \sqsubseteq_{\text{gfp},\mathcal{T}} B$.
2. *There is a simulation $Z : \mathcal{G}_\mathcal{T} \rightleftarrows \mathcal{G}_\mathcal{T}$ with $(B, A) \in Z$.*

To obtain description graph of TBox, we employ Nebel's approach (see [Neb90]) to normalize TBox.
In next section, we will give the definition of the least common subsumers (lcs) w.r.t. gfp-semantics in $\mathcal{EL}$. Both deciding subsumption and computing lcs are preparations for solving $\mathcal{EL}$-matching problems.

2.3 The least common subsumers

Before we introduce the definition of the least common subsumers, we define the notion of conservative extension which will be used for computing lcs.

Definition 9 [Conservative extension]
Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes. Then we say that $\mathcal{T}_2$ is a *conservative extension* of $\mathcal{T}_1$ if and only if

- $\mathcal{T}_1 \subseteq \mathcal{T}_2$ and
- $\mathcal{T}_1$ and $\mathcal{T}_2$ have the same primitive concepts and roles.

◇

The lcs w.r.t. gfp-semantics in $\mathcal{EL}$ is formally defined as follows:

Definition 10 [Least common subsumers]
Let $\mathcal{T}_1$, $\mathcal{T}_2$ be $\mathcal{EL}$-TBoxes such that $\mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ containing new defined concept E. Then E in $\mathcal{T}_2$ is a *least common subsumer of A and B in $\mathcal{T}_1$ w.r.t. gfp-semantics* (*gfp-lcs*) iff the following two conditions are satisfied:

1. $A \sqsubseteq_{\text{gfp},\mathcal{T}_2} E$ and $B \sqsubseteq_{\text{gfp},\mathcal{T}_2} E$.
2. If $\mathcal{T}_3$ is a conservative extension of $\mathcal{T}_2$ and F a defined concept in $\mathcal{T}_3$ such that $A \sqsubseteq_{\text{gfp},\mathcal{T}_3} F$ and $B \sqsubseteq_{\text{gfp},\mathcal{T}_3} F$, then $E \sqsubseteq_{\text{gfp},\mathcal{T}_3} F$.

◇

By this definition we know that the lcs computation is associative and commutative. When we compute gfp-lcs, the product of description graphs is used.

Definition 11 [Product of description graphs]
Let $\mathcal{G}_1 = (V_1, E_1, L_1)$ and $\mathcal{G}_2 = (V_2, E_2, L_2)$ be two description graphs. Their *product* is the description graph $\mathcal{G}_1 \times G_2 := (V, E, L)$ where

- $V := V_1 \times V_2$;
- $E := \{((v_1, v_2), r, (v_1', v_2')) \mid (v_1, r, v_1') \in E_1 \land (v_2, r, v_2') \in E_2\}$;
- $L((v_1, v_2)) := L_1(v_1) \cap L_2(v_2)$.

◇

The following lemma in [Baa02a] shows the relation between the lcs and the graph product. In principle, the lcs of A, B in $\mathcal{T}$ is defined in a TBox whose description graph is the product of $\mathcal{G}_\mathcal{T}$ with itself.

Lemma 12
Let $\mathcal{T}$ be a normalized $\mathcal{EL}$-TBox containing defined concepts A and B. Then (A, B) in $\mathcal{T}'$ is the gfp-lcs of A and B in $\mathcal{T}$, where $\mathcal{T}' := \mathcal{T}_P \cup \mathcal{T}$, $\mathcal{G}_{\mathcal{T}_P} := \mathcal{G}_\mathcal{T} \times \mathcal{G}_\mathcal{T}$.

Chapter 3

Matching in Cyclic $\mathcal{EL}$-TBoxes

In this Chapter, we will define $\mathcal{EL}$-matching problems, the mainly considered problems in this thesis.

3.1 Introducing variables

In order to define matching problems, we need concept patterns to introduce variables to our concerning DL language $\mathcal{EL}$.

Definition 13 [Concept patterns]
The *set of all concept patterns over N_C, N_R, $\mathcal{X}$* is inductively defined as follows:

- Every concept variable $X \in \mathcal{X}$ is a concept pattern.
- Every $\mathcal{EL}$-concept description over N_C and N_R is a concept pattern.
- If D_1 and D_2 are concept patterns, then $D_1 \sqcap D_2$ is a concept pattern.
- If D is a concept pattern and $r \in N_R$ is a role name, then $\exists r.D$ is a concept pattern.

◇

We can extend the notion of concept patterns to $\mathcal{EL}$-TBoxes.

Definition 14 [Pattern TBox]
A *pattern terminology* (or *pattern TBox* for short) is a finite set of concept definitions of the form $A \equiv D$, where A is a concept ($A \in N_C$) name or a variable ($A \in \mathcal{X}$) and D is a concept pattern over N_C, N_R, $\mathcal{X}$. In addition, we require that TBoxes do not contain *multiple definitions*, i.e., there cannot be two distinct concept descriptions D_1 and D_2 such that both $A \equiv D_1$ and $A \equiv D_2$ belong to the TBox. Concept names occurring on the left-hand side of a definition are called *defined concepts*. All other concept names in the TBox are called *primitive concepts*. The set of defined concepts and the set of primitive concepts are respectively denoted by N_{def} and N_{prim}. We have

that $N_C = N_{\text{def}} \cup N_{\text{prim}}$ with $N_{\text{def}} \cap N_{\text{prim}} = \emptyset$. Treating variables as concept names, we can define $\mathcal{X}_{\text{def}}$ and $\mathcal{X}_{\text{prim}}$ also. Analogously, we have that $\mathcal{X} = \mathcal{X}_{\text{def}} \cup \mathcal{X}_{\text{prim}}$ with $\mathcal{X}_{\text{def}} \cap \mathcal{X}_{\text{prim}} = \emptyset$. ◇

Since every description concept is also a concept pattern, the usual TBox (containing no variables) is also a pattern TBox. In this thesis pattern TBoxes are also called TBoxes for abbreviation.
Introducing variables to TBoxes does not impact the normalization of TBoxes if we treat variables as concept names. When extending the notion of a simulation relation to description graphs containing concept patterns, we simply ignore the concept variables, i.e., (S1) is changed into that $(v_1, v_2) \in Z$ implies $(L_1(v_1) \setminus \mathcal{X}) \subseteq L_2(v_2)$.
Let $\mathcal{T}$ be a normalized $\mathcal{EL}$-TBox and C be a defined concept in $\mathcal{T}$. Let $\mathcal{G}_\mathcal{T}$ be the $\mathcal{EL}$-description graphs for $\mathcal{T}$ Then we defined:

- $Var_\mathcal{T}(C) = \{X \in \mathcal{X} \mid$ there exists a path in $\mathcal{G}_\mathcal{T}$ from C to D for some $D \in N_{\text{def}}$ such that X is the element of label of $D\}$.

- $Var_\mathcal{T} = \bigcup\limits_{D \in N_{\text{def}}} Var_\mathcal{T}(D)$.

All of these definitions and notations in this section will be used to define matching problems with terminological cycles.

3.2 $\mathcal{EL}$-matching problems

We now have the $\mathcal{EL}$-TBox containing variables in which we allow for cyclic definitions. Giving some constraints to variables makes it possible that some defined concepts are equivalent. Based on this intuition, we define the matching problems formally as following:

Definition 15 [$\mathcal{EL}$-matching problems]
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox containing the defined concepts C and D. $C \equiv^?_{\text{gfp},\mathcal{T}} D$ is an $\mathcal{EL}$-*matching problem modulo equivalence w.r.t. gfp-semantics* iff the following conditions hold:

- $Var_\mathcal{T}(C) = \emptyset$ and $Var_\mathcal{T} = Var_\mathcal{T}(D)$.

- There are no concept definitions for any variable in $\mathcal{T}$.

◇

Similarly, we can define matching problems w.r.t. lfp- and descriptive semantics. If we change $\equiv$ to $\sqsubseteq$, then we define matching problems modulo subsumption.
A solution of $C \equiv^?_{\text{gfp},\mathcal{T}} D$ is a TBox obtained by adding definitions to some conservative extension of $\mathcal{T}$ for variables occurring in $\mathcal{T}$. This precessing is called instantiation.

Definition 16 [Instantiation]
Let $C \equiv^{?}_{\mathrm{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem modulo equivalence w.r.t. gfp-semantics and $\mathcal{T}'$ be a conservative extension of $\mathcal{T}$. Then,

$$\mathcal{T}'' = T' \cup \{X \equiv D_X \mid X \in Var_{\mathcal{T}}(D)\}$$

is an *instantiation of* $\mathcal{T}$ *w.r.t.* $\mathcal{T}'$ iff every D_X is a concept pattern defined using concept names, role names, and variables in $\mathcal{T}'$. $\diamond$

Definition 17 [Matchers]
Let $C \equiv^{?}_{\mathrm{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem modulo equivalence w.r.t. gfp-semantics, $\mathcal{T}'$ be a conservative extension of $\mathcal{T}$ and $\mathcal{T}''$ be an instantiation of $\mathcal{T}$ w.r.t. $\mathcal{T}'$. We called T'' is a *matcher* (or *solution*) of $C \equiv^{?}_{\mathrm{gfp},\mathcal{T}} D$ iff $C \equiv_{\mathrm{gfp},\mathcal{T}''} D$. $\diamond$

We will show later how to find matchers of a given matching problem. Before that we use the following example to explain the definitions in this section.

Example 18
Let $C \equiv^{?}_{\mathrm{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem and $\mathcal{T} := \{C \equiv A, D \equiv X\}$. Then $\mathcal{T}' := \mathcal{T}$ is a conservative extension of $\mathcal{T}$ and $\mathcal{T}'' := T' \cup \{X \equiv A\}$ is a matcher of $C \equiv^{?}_{\mathrm{gfp},\mathcal{T}} D$.

Chapter 4

Solving $\mathcal{EL}$-matching problems

We now show how to solve $\mathcal{EL}$-matching problems modulo equivalence w.r.t. gfp-semantics.

4.1 Matching algorithm

W.l.o.g., we consider only normalized $\mathcal{EL}$-TBoxes, otherwise we can transform it into normalized one (see [Baa02b]). Sometimes we need to compute the lcs of more than two concept descriptions, for example, $lcs(C, D, E)$. To this end, we calculate the n-ary lcs by means of an $(n-1)$-fold binary lcs computation. Based on this intuition, we introduce the following definition:

Definition 19 Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox. Then we define

$$\begin{aligned}\mathcal{T}^1 &:= \mathcal{T};\\ \mathcal{T}^{i+1} &:= \mathcal{T}^i \times \mathcal{T} \text{ where } \mathcal{G}_{\mathcal{T}^i \times \mathcal{T}} = \mathcal{G}_{\mathcal{T}^i} \times \mathcal{G}_{\mathcal{T}} \text{ and } i \in \mathbb{N} \setminus \{0\}.\end{aligned}$$

$\diamond$

Let $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem. For some simulation relation $Z : \mathcal{G}_\mathcal{T} = (N_{\mathrm{def}}, E_\mathcal{T}, L_\mathcal{T}) \rightleftarrows \mathcal{G}_\mathcal{T}$ and some variable $X \in Var_\mathcal{T}(D)$, we define the following set:

$$Z(X) := \{N \in N_{\mathrm{def}} \mid \exists M \in N_{\mathrm{def}}.(M, N) \in Z \wedge X \in L_\mathcal{T}(M)\}.$$

These notations above will be used in the matching algorithm. We prove now some properties holding on the product of TBoxes.

Observation 20
Let $\mathcal{T}$ be an $\mathcal{EL}$-Tbox, $i, j \in \mathbb{N} \setminus \{0\}$ with $i \neq j$ and $N^{\mathcal{T}^i}_{\mathrm{def}}$, $N^{\mathcal{T}^j}_{\mathrm{def}}$ be the corresponding set of defined concepts occurring in $\mathcal{T}^i$, $\mathcal{T}^j$. Then

$$N^{\mathcal{T}^i}_{\mathrm{def}} \cap N^{\mathcal{T}^j}_{\mathrm{def}} = \emptyset.$$

Proof: This is an immediate consequence of the definition of the product of Tboxes.$\square$

Lemma 21
Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox. Then, $\mathcal{T} \cup \mathcal{T}^i$ is a conservative extension of $\mathcal{T}$ for all $i \in \mathbb{N} \setminus \{0\}$.

Proof: Let $i \in \mathbb{N} \setminus \{0\}$. $\mathcal{T} \cup \mathcal{T}^i$ is a conservative extension of $\mathcal{T}$ since

- $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{T}^i$
- $\mathcal{T}$ and $\mathcal{T} \cup \mathcal{T}^i$ have the same primitive concepts and roles (from construction of $\mathcal{T}^i$).

□

We are now ready to define the algorithm of solving $\mathcal{EL}$-matching problems. The input of the algorithm is an $\mathcal{EL}$-matching problem $C \equiv^?_{\text{gfp},\mathcal{T}} D$ (w.l.o.g., we assume $\mathcal{T}$ is in normal form). The output of the algorithm is a set of matchers to $C \equiv^?_{\text{gfp},\mathcal{T}} D$.

Algorithm 22 [Matching algorithm]
Input: $C \equiv^?_{\text{gfp},\mathcal{T}} D$.
Output: a set $\mathcal{S}$ of matchers to $C \equiv^?_{\text{gfp},\mathcal{T}} D$.

```
$\mathcal{S} := \emptyset$;
For all simulation relations $Z : \mathcal{G}_\mathcal{T} \overset{\rightarrow}{\sim} \mathcal{G}_\mathcal{T}$ with $(D, C) \in Z$ do
        $\mathcal{T}' := T \cup \bigcup_{i \in \{|Z(X)| \mid X \in Var_\mathcal{T}(D)\} \setminus \{1\}} \mathcal{T}^i$;
        $\mathcal{T}'' := \mathcal{T}' \cup \{X \equiv (A_1, \ldots, A_n) \mid Z(X) = \{A_1, \ldots, A_n\} \wedge |Z(X)| = n \wedge X \in Var_\mathcal{T}(D)\}$;
If $C \sqsupseteq_{\text{gfp},\mathcal{T}''} D$ then $\mathcal{S} := \mathcal{S} \cup \{\mathcal{T}''\}$;
return $\mathcal{S}$.
```

In this matching algorithm, $(A_1, \ldots, A_n)$ is an abbreviation of $(\ldots(A_1, A_2), \ldots, A_n)$ and $(\ldots(A_1, A_2), \ldots, A_n)$ in $\mathcal{T}'$ is $lcs(A_1, \ldots, A_n)$ of $A_1, \ldots, A_n$ in $\mathcal{T}$ (By Lemma 12). Moreover, for the special case $n = 1$, (A_1) means $lcs(A_1)$ which can be replaced by A_1. Using the following example, we show that to check $C \sqsupseteq_{\text{gfp},\mathcal{T}''} D$ is necessary after obtaining the candidate solutions.

Example 23
Let $C \equiv^?_{\text{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem with $\mathcal{T}$ containing the following definitions:

$$
\begin{aligned}
C &\equiv \exists r.A \sqcap \exists s.B \\
D &\equiv \exists r.E \sqcap \exists s.F \\
A &\equiv P_1, B \equiv P_2 \\
E &\equiv X \\
F &\equiv X
\end{aligned}
$$

For the simulation relation $Z = \{(D, C), (E, A), (F, B)\}$, we have $X \equiv lcs(A, B) = \top$. However, $\mathcal{T} \cup \{X \equiv \top\}$ is not a matcher of $C \equiv^?_{\text{gfp},\mathcal{T}} D$.

4.2 Termination

At the end of this chapter, we show termination of our matching algorithm.

Lemma 24
Algorithm 22 is always terminating.

Proof: Let $C \equiv^{?}_{\text{gfp},\mathcal{T}} D$ be the inputing $\mathcal{EL}$-matching problem. Since $\mathcal{T}$ is finite, there are only finitely many simulation relations between $\mathcal{G}_{\mathcal{T}}$ and itself. For all simulation relations Z and for all $X \in Var_{\mathcal{T}}(D)$, $|Z(X)| \leq |Z|$ is finite. Then $\mathcal{T}'$ is finite. Since $Var_{\mathcal{T}}(D)$ is finite, $\mathcal{T}''$ is finite. From [Baa02b], we know that subsumption w.r.t. gfp-semantics in $\mathcal{EL}$ can be decided in polynomial time. Thus, Algorithm 22 is always terminating. □
In next chapter, we will show Algorithm 22 is sound and complete.

Chapter 5

Soundness and completeness

In this chapter, we will show that our matching algorithm defined in the previous chapter is sound and complete. Every TBox in the output of Algorithm 22 is a solution of the relevant input matching problem (soundness). Moreover, if the input matching problem is solvable, then Algorithm 22 can find all so-called minimal matchers (s-completeness).

5.1 Soundness

We first show some auxiliary lemma for proving soundness and completeness of our matching algorithm.

Lemma 25
Let $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem and $\mathcal{S}$ be the output set generated by the matching algorithm. Then, for every $\mathcal{T}'' \in \mathcal{S}$ obtained from the corresponding $\mathcal{T}'$ and for every variable $X \in Var_\mathcal{T}(D)$, we have that

$$X \equiv (A_1, \ldots, A_n) \in \mathcal{T}'' \implies L_{\mathcal{T}'}((A_1, \ldots, A_n)) \cap Var_\mathcal{T}(D) = \emptyset$$

where $G_{\mathcal{T}'} = (V_{\mathcal{T}'}, E_{\mathcal{T}'}, L_{\mathcal{T}'})$.

Proof: Consider $X \equiv (A_1, \ldots, A_n) \in \mathcal{T}''$ for some $X \in Var_\mathcal{T}(D)$. By construction of $\mathcal{T}''$ in Algorithm 22 we know that there are defined concepts $M_1, \ldots, M_n$ in $\mathcal{T}$ such that for all $i \in \{1, \ldots, n\}$, $X \in L_\mathcal{T}(M_i)$ and $(M_i, A_i) \in Z$ for the corresponding simulation relation Z. For all $i \in \{1, \ldots, n\}$, there exists a path from D to M_i, since $X \in L_\mathcal{T}(M_i)$ and $X \in Var_\mathcal{T}(D)$. Together with $(D, C) \in Z$, we have that there exists an index $j \in \{1, \ldots, n\}$ such that

- $(M_j, A_j) \in Z$ and
- there is a path from C to A_j.

$L_\mathcal{T}(A_j)$ contains no variables by definition of matching problems. Hence,

$$L_{\mathcal{T}'}((A_1, \ldots, A_n)) = L_\mathcal{T}(A_1) \cap \cdots \cap L_\mathcal{T}(A_n)$$

by the construction of $\mathcal{T}'$. Thus, $L_{\mathcal{T}'}((A_1, \dots, A_n))$ contains no variables, i.e.,

$$L_{\mathcal{T}'}((A_1, \dots, A_n)) \cap Var_{\mathcal{T}}(D) = \emptyset.$$

□

This lemma tells us that no variables occur in the concept definition of $(A_1, \dots, A_n)$ in $\mathcal{T}'$ if $X \equiv (A_1, \dots, A_n)$ in $\mathcal{T}''$ for some $X \in Var_{\mathcal{T}}(D)$. This makes it possible that by replacing every variable X in $\mathcal{T}'$ by its definition in $\mathcal{T}''$ we obtain a variable-free TBox (to be explained). We now show that every Z considered in the matching algorithm is also a simulation relation from $\mathcal{G}_{\mathcal{T}'}$ to $\mathcal{G}_{\mathcal{T}'}$.

Lemma 26
Let $C \equiv^?_{\text{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem and $Z : \mathcal{G}_{\mathcal{T}} \rightleftarrows \mathcal{G}_{\mathcal{T}}$ with $(D, C) \in Z$ be a simulation relation and $\mathcal{T}'$ be a TBox as described in the matching algorithm. Then Z is a simulation relation on $\mathcal{G}'_{\mathcal{T}}$, i.e., $Z : \mathcal{G}_{\mathcal{T}'} \rightleftarrows \mathcal{G}_{\mathcal{T}'}$.

Proof: The conditions (S1) and (S2) in Definition 7 hold:

(S1) For all $(v_1, v_2) \in Z$ we have $L_{\mathcal{T}'}(v_1) \subseteq L_{\mathcal{T}'}(v_2)$ since $Z : \mathcal{G}_{\mathcal{T}} \rightleftarrows \mathcal{G}_{\mathcal{T}}$ and $\mathcal{T}' := T \cup \bigcup_{i \in \{|Z(X)| | X \in Var_{\mathcal{T}}(D)\} \setminus \{1\}} \mathcal{T}^i$.

(S2) Consider some $(v_1, v_2) \in Z$ and $(v_1, r, v'_1) \in E_{\mathcal{T}'}$. Then, $(v_1, v_2) \in Z$ implies that v_1 and v_2 are defined concepts in $\mathcal{T}$ since $Z : \mathcal{G}_{\mathcal{T}} \rightleftarrows \mathcal{G}_{\mathcal{T}}$. Hence,

$$v_1 \equiv \dots \sqcap \exists r.v'_1 \sqcap \dots \in \mathcal{T}.$$

Thus, v'_1 is also a defined concept in $\mathcal{T}$. So there exists a $v'_2 \in V_{\mathcal{T}} \subseteq V_{\mathcal{T}'}$ such that $(v'_1, v'_2) \in Z$ and $(v_2, r, v'_2) \in E_{\mathcal{T}} \subseteq E_{\mathcal{T}'}$ since $Z : \mathcal{G}_{\mathcal{T}} \rightleftarrows \mathcal{G}_{\mathcal{T}}$.

□

Let $C \equiv^?_{\text{gfp},\mathcal{T}} D$ be a matching problem. We view variables in $\mathcal{T}$ as primitive concepts since there is no concept definition for any a variable. $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$ by the construction of $\mathcal{T}'$ and $\mathcal{T}'$ is in normal form since it is obtained from a description graph. However, $\mathcal{T}''$ is not in normal form since we add concept descriptions for every variable $X \in Var_{\mathcal{T}}(D)$ and X occurs on the right-hand side of some concept definitions in $\mathcal{T}'$. We construct a variable-free TBox T''_Z from $\mathcal{T}''$ by placing every occurrence of variables with the corresponding variable's definition. Then, we prove that T''_Z is in normal form and $C \sqsubseteq_{\text{gfp},\mathcal{T}''} D$ iff $C \sqsubseteq_{\text{gfp},\mathcal{T}''_Z} D$.
$\mathcal{T}''_Z$ is constructed by the following definition:

Definition 27
Let $C \equiv^?_{\text{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem. Let $\mathcal{T}''$ be a TBox in the set $\mathcal{S}$, the output of Algorithm 22 giving $C \equiv^?_{\text{gfp},\mathcal{T}} D$ as input, and $E \equiv D_E$ be a concept definition in $\mathcal{T}''$. Then, we define:

- $sub(D_E)$ is a concept pattern obtained by replacing every occurrence of $X \in Var_{\mathcal{T}}(D)$ by $P_1 \sqcap \dots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \dots \sqcap \exists r_l.B_l$, where $X \equiv (A_1, \dots, A_n)$ and $(A_1, \dots, A_n) \equiv P_1 \sqcap \dots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \dots \sqcap \exists r_l.B_l$ are concept definitions in $\mathcal{T}''$.

- $\mathcal{T}''_Z := \{E \equiv sub(D_E) \mid E \equiv D_E \in \mathcal{T}'' \text{ and } E \notin Var_{\mathcal{T}}(D)\}$;

$\diamond$

By Lemma 25, $\mathcal{T}''_Z$ obtained by this construction is variable-free, i.e., containing no variables.

Lemma 28
$\mathcal{T}''_Z$ generated by above definition is in normal form.

Proof: The only difference between $\mathcal{T}'$ (obtained in the intermediate step when computing $\mathcal{T}''$) and $\mathcal{T}''_Z$ is that we substitute all variables in $\mathcal{T}'$ by the corresponding concept descriptions. As defined in the above definition, if $X \equiv (A_1, \ldots, A_n)$ in $\mathcal{T}''$, X is substituted by the definition of $(A_1, \ldots, A_n)$ in $\mathcal{T}'$. Since $\mathcal{T}'$ is in normal form, so is $\mathcal{T}''_Z$. $\square$
Since $\mathcal{T}''_Z$ is in normal form, we can translate it into the description graph $\mathcal{G}_{\mathcal{T}''_Z}$ by the method defined in Chapter 2. Moreover, $\mathcal{T}''$ and $\mathcal{T}''_Z$ are equivalent in the sense of subsumption relations between C and D.

Lemma 29

1. $C \sqsubseteq_{\text{gfp},\mathcal{T}''} D$ iff $C \sqsubseteq_{\text{gfp},\mathcal{T}''_Z} D$;

2. $C \sqsupseteq_{\text{gfp},\mathcal{T}''} D$ iff $C \sqsupseteq_{\text{gfp},\mathcal{T}''_Z} D$.

Proof: In the construction of $\mathcal{T}''_Z$ substitution of variables does not change semantics of $\mathcal{T}''$ since substitution happens between equivalent concepts. Removing definitions for variables does not change semantics of $\mathcal{T}''$, since variables do not occur in $\mathcal{T}''_Z$ after substitution. $\square$
To prove $C \sqsubseteq_{\text{gfp},\mathcal{T}''_Z} D$, it is enough to construct a simulation relation $Z_{\mathcal{T}''_Z} : \mathcal{G}_{\mathcal{T}''_Z} \rightleftarrows\!\!\!\sim \mathcal{G}_{\mathcal{T}''_Z}$ with $(D, C) \in Z$. This proves $C \sqsubseteq_{\text{gfp},\mathcal{T}''} D$ as well by Lemma 29. In the matching algorithm, we have the simulation relation $Z : \mathcal{G}_{\mathcal{T}} \rightleftarrows\!\!\!\sim \mathcal{G}_{\mathcal{T}}$ with $(D, C) \in Z$. Now we add some necessary tuples to Z and obtain $Z_{\mathcal{T}''_Z}$. Then we prove that $Z_{\mathcal{T}''_Z}$ is a simulation relation on $\mathcal{G}_{\mathcal{T}''_Z}$.

Definition 30
Let $C \equiv^?_{\text{gfp},\mathcal{T}} D$ be a $\mathcal{EL}$-matching problem, Z be a simulation relation on $\mathcal{G}_{\mathcal{T}}$ containing (D, C), and $\mathcal{T}''$ be the TBox obtained by Algorithm 22 using Z. $\mathcal{T}''_Z$ is obtained by Definition 27 corresponding to $\mathcal{T}''$. Then,

$$\begin{aligned} Z_{\mathcal{T}''_Z} :=& Z \cup \{((A_{j,1}, \ldots, A_{j,n}), A_{j,i}) \mid \exists i \in \{1, \ldots, n\}.\exists j \in \{1, \ldots, l\}.\exists X \in Var_{\mathcal{T}}(D). \\ & X \equiv (A_1, \ldots, A_n), (A_1, \ldots, A_n) \equiv \\ & P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.(A_{1,1}, \ldots, A_{1,n}) \sqcap \ldots \sqcap \exists r_l.(A_{l,1}, \ldots, A_{l,n}) \in \mathcal{T}''\}. \end{aligned}$$

$\diamond$

Note: The concept definition of $(A_1, \dots, A_n)$ in $\mathcal{T}''$ in the above definition is of the form

$$P_1 \sqcap \dots \sqcap P_m \sqcap \exists r_1.(A_{1,1}, \dots, A_{1,n}) \sqcap \dots \sqcap \exists r_l.(A_{l,1}, \dots, A_{l,n}).$$

The reason is the following: Suppose that

$$(A_1, \dots, A_n) \equiv P_1 \sqcap \dots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \dots \sqcap \exists r_l.B_l.$$

$(A_1, \dots, A_n)$ is the least common subsumer of $A_1, \dots, A_n$ in $\mathcal{T}'$. By Observation 20 and the definition of product of TBoxes we know that B_j is of the form $(A_{j,1}, \dots, A_{j,n})$ where $A_{j,i}$ is a defined concept in $\mathcal{T}$ for all $j \in \{1, \dots, l\}$ and for all $i \in \{1, \dots, n\}$.

Lemma 31
Let $C \equiv^?_{\text{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem, $Z : \mathcal{G}_\mathcal{T} \precsim \mathcal{G}_\mathcal{T}$ be a simulation relation with $(D, C) \in Z$, and $\mathcal{T}$, $\mathcal{T}''$ be TBoxes as described in Algorithm 22. Then $Z_{\mathcal{T}''_Z}$ is a simulation relation on $G_{\mathcal{T}''_Z}$, i.e., $Z_{\mathcal{T}''_Z} : \mathcal{G}_{\mathcal{T}''_Z} \precsim \mathcal{G}_{\mathcal{T}''_Z}$.

Proof: Since $Z : \mathcal{G}_\mathcal{T} \precsim \mathcal{G}_\mathcal{T}$, by Lemma 26, $Z : \mathcal{G}_{\mathcal{T}'} \precsim \mathcal{G}_{\mathcal{T}'}$. The difference between $\mathcal{T}'$ and $\mathcal{T}''_Z$ is that we substitute variables by the corresponding concept definitions. Let $\mathcal{G}_{\mathcal{T}'}$ and $\mathcal{G}_{\mathcal{T}''_Z}$ be the description graphs of $\mathcal{T}'$ and $\mathcal{T}''_Z$ respectively. It holds that the labels of $\mathcal{G}_{\mathcal{T}''_Z}$ are obtained from $\mathcal{G}_{\mathcal{T}'}$ by removing all variables and adding primitive concepts corresponding to the definitions of variables in $\mathcal{T}''$. Consider the following concept definition in $\mathcal{T}''$:

$$X \equiv (A^X_1, \dots, A^X_{n_X})$$
$$(A^X_1, \dots, A^X_{n_X}) \equiv P^X_1 \sqcap \dots \sqcap P^X_{m_X} \sqcap \exists r^X_1.(A^X_{1,1}, \dots, A^X_{1,n_X}) \sqcap \dots \sqcap \exists r^X_{l_X}.(A^X_{l_X,1}, \dots, A^X_{l_X,n_X})$$

Then, $\forall M \in V_{\mathcal{T}''_Z}.L_{\mathcal{T}''_Z}(M) = L_{\mathcal{T}'}(M) \setminus \{X \mid X \in Var_\mathcal{T}(D)\} \cup \bigcup_{X \in L_{\mathcal{T}'}(M)} \{P^X_1, \dots, P^X_{m_X}\}$.
Moreover, by Algorithm 22, there are $M^X_1, \dots, M^X_{n_X} \in N_{\text{def}}$ such that

- $(M^X_1, A^X_1), \dots, (M^X_{n_X}, A^X_{n_X}) \subseteq Z$;
- $X \in L_\mathcal{T}(M^X_1) \cap \dots \cap L_\mathcal{T}(M^X_{n_X})$.

Comparing $\mathcal{G}_{\mathcal{T}''_Z}$ and $\mathcal{G}_{\mathcal{T}'}$, it is clear that for every edge in $\mathcal{G}_{\mathcal{T}''_Z}$ but not in $\mathcal{G}_{\mathcal{T}'}$ there exist $i \in \{1, \dots, n_X\}$ and $j \in \{1, \dots, l_X\}$ such that the edge has the form of

$$(M^X_i, r^X_j, (A^X_{j,1}, \dots, A^X_{j,n_X})).$$

Accordingly, in order to obtain $Z_{\mathcal{T}''_Z}$, we add $((A^X_{j,1}, \dots, A^X_{j,n_X}), A^X_{j,i})$ to Z for all $i \in \{1, \dots, n_X\}$ and for all $j \in \{1, \dots, l_X\}$. Now we will show $Z_{\mathcal{T}''_Z} : \mathcal{G}_{\mathcal{T}''_Z} \precsim \mathcal{G}_{\mathcal{T}''_Z}$ (enough to show the conditions (S1) and (S2) in Definition 7 hold).

(S1) Consider some $(v_1, v_2) \in Z_{\mathcal{T}''_Z}$. We have to show that $L_{\mathcal{T}''_Z}(v_1) \subseteq L_{\mathcal{T}''_Z}(v_2)$.

Case 1 : $(v_1, v_2) \in Z$ and $L_{\mathcal{T}'}(v_1) \cap Var_{\mathcal{T}}(D) = \emptyset$:

$$\begin{aligned} L_{\mathcal{T}''_Z}(v_1) &= L_{\mathcal{T}'}(v_1) \\ &= L_{\mathcal{T}'}(v_1) \setminus Var_{\mathcal{T}}(D) \\ &\subseteq L_{\mathcal{T}'}(v_2) \text{ since } Z : \mathcal{G}_{\mathcal{T}'} \precsim \mathcal{G}_{\mathcal{T}'} \\ &\subseteq L_{\mathcal{T}''_Z}(v_2) \end{aligned}$$

Case 2 : $(v_1, v_2) \in Z$ and $L_{\mathcal{T}'}(v_1) \cap Var_{\mathcal{T}}(D) = \{X_1, \ldots, X_k\}$:
$L_{\mathcal{T}''_Z}(v_1) = (L_{\mathcal{T}'}(v_1) \setminus Var_{\mathcal{T}}(D)) \cup \bigcup\limits_{i=1}^{k} \{P_1^{X_i}, \ldots, P_{m_{X_i}}^{X_i}\}$.

Case 2.1 : $\forall P \in L_{\mathcal{T}'}(v_1) \setminus Var_{\mathcal{T}}(D)$: $P \in L_{\mathcal{T}''_Z}(v_2)$ since $(v_1, v_2) \in Z : \mathcal{G}_{\mathcal{T}'} \precsim \mathcal{G}_{\mathcal{T}'}$.

Case 2.2 : $\forall P \in \bigcup\limits_{i=1}^{k} \{P_1^{X_i}, \ldots, P_{m_{X_i}}^{X_i}\}.\exists j \in \{1, \ldots, k\}.X_j \equiv (A_1^{X_j}, \ldots, A_n^{X_j})$ and $P \in L_{\mathcal{T}'}(A_1^{X_j}) \cap \cdots \cap L_{\mathcal{T}'}(A_n^{X_j})$. By Algorithm 22, $v_2 \in \{A_1^{X_j}, \ldots, A_n^{X_j}\}$. Thus, $P \in L_{\mathcal{T}''_Z}(v_2)$.

Case 3 : $(v_1, v_2) \in Z_{\mathcal{T}''_Z} \setminus Z$:
Then there is an $X \in Var_{\mathcal{T}}(D)$ such that $X \equiv (A_1, \ldots, A_n)$. There are $M_1, \ldots, M_n \in N_{\text{def}}$ such that $\{(M_1, A_1), \ldots, (M_n, A_n)\} \subseteq Z$ and $X \in L_{\mathcal{T}}(M_i)$ for $i \in \{1 \ldots, n\}$. Suppose that

$$(A_1, \ldots, A_n) \equiv P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.(A_{1,1}, \ldots, A_{1,n}) \sqcap \cdots \sqcap \exists r_l.(A_{l,1}, \ldots, A_{l,n})$$

in $\mathcal{T}'$. Then there exist $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, l\}$ such that

- $v_1 = (A_{j,1}, \ldots, A_{j,n})$ and
- $v_2 = A_{j,i}$.

Hence,

$$\begin{aligned} L_{\mathcal{T}''_Z}(v_1) =& L_{\mathcal{T}'}(A_{j,1}) \cap \cdots \cap L_{\mathcal{T}'}(A_{j,n}) \\ &\subseteq (L_{\mathcal{T}'}(A_{j,i}) \setminus Var_{\mathcal{T}}(D)) \\ &\subseteq L_{\mathcal{T}''_Z}(A_{j,i}) \\ &= L_{\mathcal{T}''_Z}(v_2) \end{aligned}$$

(S2) Consider some $(v_1, v_2) \in Z_{\mathcal{T}''_Z}$ and $(v_1, r, v_1') \in E_{\mathcal{T}''_Z}$. We have to show that there exists a node $v_2' \in V_{\mathcal{T}''_Z}$ such that $(v_1', v_2') \in Z_{\mathcal{T}''_Z}$ and $(v_2, r, v_2') \in E_{\mathcal{T}''_Z}$.

Case 1 $(v_1, v_2) \in Z$ and $(v_1, r, v_1') \in E_{\mathcal{T}'}$. Then there is a node $v_2' \in V_{\mathcal{T}'} = V_{\mathcal{T}''_Z}$ such that $(v_1', v_2') \in Z \subseteq Z_{\mathcal{T}''_Z}$ and $(v_1', r, v_1') \in E_{\mathcal{T}'} \subseteq E_{\mathcal{T}''_Z}$ since $Z : \mathcal{G}_{\mathcal{T}'} \precsim \mathcal{G}_{\mathcal{T}'}$.

Case 2 $(v_1, v_2) \in Z_{\mathcal{T}''_Z} \setminus Z$ and $(v_1, r, v_1') \in E_{\mathcal{T}'}$. Then there exist $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, l\}$ such that $v_1 = (A_{j,1} \ldots, A_{j,n})$ and $v_2 = A_{j,i}$. Moreover, v_1' is of the form $(E_1, \ldots, E_n)$ such that $\{(A_{j,1}, r, E_1), \ldots, (A_{j,n}, r, E_n)\} \subseteq E_{\mathcal{T}'}$. So $(A_{j,i}, r, E_i) \in E_{\mathcal{T}'} \subseteq E_{\mathcal{T}''_Z}$.

Case 3 $(v_1, v_2) \in Z$ and $(v_1, r, v_1') \in E_{\mathcal{T}_Z''} \setminus E_{\mathcal{T}'}$. Then there exist $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, l\}$ such that $v_1 = M_i$, $v_2 = A_i$, $r = r_j$, and $v_1' = (A_{j,1} \ldots, A_{j,n})$. Then Definition 27 tells us $((A_{j,1} \ldots, A_{j,n}), A_{j,i}) \in Z_{\mathcal{T}_Z''}$ and $(A_i, r_j, A_{j,i}) \in E_{\mathcal{T}'}$ since $((A_1, \ldots, A_n), r_j, (A_{j,1} \ldots, A_{j,n})) \in E_{\mathcal{T}_Z''}$.

Case 4 $(v_1, v_2) \in Z_{\mathcal{T}_Z''} \setminus Z$ and $(v_1, r, v_1') \in E_{\mathcal{T}_Z''} \setminus E_{\mathcal{T}'}$. $(v_1, v_2) \in Z_{\mathcal{T}_Z''} \setminus Z$ yields that v_1 is of the form $(A_{j,1}, \ldots, A_{j,n})$. $(v_1, r, v_1') \in E_{\mathcal{T}_Z''} \setminus E_{\mathcal{T}'}$ yields that $L_{\mathcal{T}'}(v_1) = L_{\mathcal{T}'}((A_{j,1}, \ldots, A_{j,n}))$ contains some variable. This case is not possible by Lemma 25.

□

Using the above lemma it is easy to show soundness of our matching algorithm.

Theorem 32
Let $C \equiv^?_{\text{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem. Every $\mathcal{T}'' \in \mathcal{S}$ is a matcher of $C \equiv^?_{\text{gfp},\mathcal{T}} D$.

Proof: By Lemma 21 we know that $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$. And $\mathcal{T}''$ is an instantiation of $\mathcal{T}$ w.r.t. $\mathcal{T}'$ (by construction of $\mathcal{T}''$). To show that $\mathcal{T}''$ is a matcher of $C \equiv^?_{\text{gfp},\mathcal{T}} D$, we have to show $C \equiv_{\text{gfp},\mathcal{T}''} D$. From the matching algorithm, for every $\mathcal{T}'' \in \mathcal{D}$ we know that $C \sqsupseteq_{\text{gfp},\mathcal{T}''} D$. Thus, it suffices to show that for all $\mathcal{T}''$ in $\mathcal{S}$ we have $C \sqsubseteq_{\text{gfp},\mathcal{T}''} D$. By Lemma 31, we can construct $Z_{\mathcal{T}_Z''} : \mathcal{G}_{\mathcal{T}_Z''} \rightleftarrows \mathcal{G}_{\mathcal{T}_Z''}$ with $(D, C) \in Z_{\mathcal{T}_Z''}$. Thus, by Theorem 8, we have $C \sqsubseteq_{\text{gfp},\mathcal{T}_Z''} D$. Then, Lemma 29 yields $C \sqsubseteq_{\text{gfp},\mathcal{T}''} D$. □

5.2 Completeness

In this section we want to show completeness of our matching algorithm. If there is more than one solution, we are interested in solutions that contain as much information as possible. So in the proof of completeness, we claim that whenever there exists a matcher $\mathcal{M}''$, our matching algorithm can find one matcher which is more specific w.r.t. subsumption than $\mathcal{M}''$. To compare among matchers, we introduce the following definition:

Definition 33
Let $C \equiv^?_{\text{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem, $\mathcal{T}_1'$, $\mathcal{T}_2'$ be conservative extensions of $\mathcal{T}$, $\mathcal{T}_1''$ be an instantiation of $\mathcal{T}$ w.r.t. $\mathcal{T}_1'$ and $\mathcal{T}_2''$ be an instantiation of $\mathcal{T}$ w.r.t. $\mathcal{T}_2'$. By renaming defined concepts and variables in $\mathcal{T}$ we can make $\mathcal{T}_1''$ and $\mathcal{T}_2''$ such that they do not contain common defined concepts and variables. Then,

- $\mathcal{T}_1''$ is *s-subsumed by* $\mathcal{T}_2''$ ($\mathcal{T}_1'' \sqsubseteq_s \mathcal{T}_2''$) iff for all $X \in Var_{\mathcal{T}}(D)$, $X^{\mathcal{T}_1} \sqsubseteq_{\text{gfp},\mathcal{T}_1'' \cup \mathcal{T}_2''} X^{\mathcal{T}_2}$, where $X^{\mathcal{T}_1}$ in $\mathcal{T}_1''$ and $X^{\mathcal{T}_2}$ in $\mathcal{T}_2''$ correspond to X in $\mathcal{T}$.
- $\mathcal{T}_1''$ is *s-equivalent to* $\mathcal{T}_2''$ ($\mathcal{T}_1'' \equiv_s \mathcal{T}_2''$) iff $\mathcal{T}_1'' \sqsubseteq_s \mathcal{T}_2''$ and $\mathcal{T}_2'' \sqsubseteq_s \mathcal{T}_1''$.

A matcher $\mathcal{M}''$ is the *least matcher* w.r.t. $\sqsubseteq_s$ iff $\mathcal{M}'' \sqsubseteq_s \mathcal{T}''$ holds for all matchers $\mathcal{T}''$. A matcher $\mathcal{M}''$ is a *minimal matcher* w.r.t. $\sqsubseteq_s$ iff for all matchers $\mathcal{T}''$, $\mathcal{T}'' \sqsubseteq_s \mathcal{M}''$ implies $\mathcal{M}'' \equiv_s \mathcal{T}''$. ◇

The next lemma tells us that the least matcher is unique up to s-equivalence. This is an immediate consequence of Definition 33.

Lemma 34
Let $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem. If $\mathcal{M}''_1$ and $\mathcal{M}''_2$ are two least matchers of $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ then $\mathcal{M}''_1 \equiv_s \mathcal{M}''_2$.

The following example illustrates that the least matcher of $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ need not exist even if $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ is solvable.

Example 35
$\mathcal{T} := \{C \equiv \exists r.A \sqcap \exists r.B, D \equiv \exists r.X \sqcap \exists r.Y\}$. It is easy to see that $\mathcal{T} \cup \{X \equiv A, Y \equiv B\}$ and $\mathcal{T} \cup \{X \equiv B, Y \equiv A\}$ are matchers of $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$. However, the least matcher does not exist.

We now define our notion of completeness formally. If a matching problem is solvable, then our matching algorithm can compute an s-complete set of matchers. It is defined as following:

Definition 36
Let $\mathcal{S}_\mathcal{M}$ be the set of matchers of $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$. Then $\mathcal{S} \subseteq \mathcal{S}_\mathcal{M}$ is called *s-complete* iff $\forall \mathcal{M}'' \in \mathcal{S}_\mathcal{M}.\exists \mathcal{T}'' \in \mathcal{S}.\mathcal{T}'' \sqsubseteq_s \mathcal{M}''$. $\diamond$

Next we want to show that the set $\mathcal{S}$ of matchers computed by Algorithm 22 is s-complete.

Theorem 37
Let $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ be an $\mathcal{EL}$-matching problem. If $\mathcal{M}''$ is a matcher of $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ then there exists $\mathcal{T}'' \in \mathcal{S}$ with $\mathcal{T}'' \sqsubseteq_s \mathcal{M}''$, where $\mathcal{S}$ is the output of Algorithm 22 upon input $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$.

Proof: Suppose that $\mathcal{M}''$ is an instantiation of $\mathcal{T}$ w.r.t. $\mathcal{M}'$. W.l.o.g., we assume that $\mathcal{M}''$ is in normal form. Otherwise, we can obtain the corresponding normalized TBox $\mathcal{M}''_1$ using Nebel's approach mentioned in [Baa02b]. Moreover, $\mathcal{M}'_1 := \mathcal{M}''_1 \setminus \{X \equiv D_X \mid X \in Var_\mathcal{T}(D)\}$. $\mathcal{M}'_1$ is a conservative extension of $\mathcal{T}$ since normalization adds concept definitions for new concept names. Normalization does not change the semantics of TBoxes. Thus, $\mathcal{M}'_1$ is a conservative extension of $\mathcal{T}$ and $\mathcal{M}''_1$ is an instantiation of $\mathcal{T}$ w.r.t. $\mathcal{M}'_1$.
Since $\mathcal{M}''$ is a matcher of $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$, we have that $C \equiv_{\mathrm{gfp},\mathcal{M}''} D$. Thus, $C \sqsubseteq_{\mathrm{gfp},\mathcal{M}''} D$ and $C \sqsupseteq_{\mathrm{gfp},\mathcal{M}''} D$. Hence, $C \sqsubseteq_{\mathrm{gfp},\mathcal{M}''} D$ implies that there exists a simulation relation $Z_{\mathcal{M}''} : \mathcal{G}_{\mathcal{M}''} \overset{\rightarrow}{\sim} \mathcal{G}_{\mathcal{M}''}$ with $(D, C) \in Z_{\mathcal{M}''}$.
Comparing $\mathcal{G}_\mathcal{T}$ with $\mathcal{G}_{\mathcal{M}''}$, we observe that adding definitions of all variables changes the state of variables from primitive to defined variables. In the normalization of $\mathcal{M}''$ we replace every occurrence of a variable on the right-hand side of definitions in $\mathcal{M}''$ by its "normalized" definition. Thus, in the corresponding graph, $\mathcal{G}_\mathcal{T}$ can be obtained from $\mathcal{G}_{\mathcal{M}''}$ by removing edges and some nodes (not only nodes of variables since new defined

concept names can be introduced by the normalization) in $\mathcal{G}_{\mathcal{M}''}$, adding some variables to the labels of some nodes and at the same time removing the corresponding primitive concepts from the labels of these nodes. Hence, we have

- $E_{\mathcal{T}} \subseteq E_{\mathcal{M}''}$ and
- $\forall v \in V_{\mathcal{T}}.(L_{\mathcal{T}}(v) \setminus Var_{\mathcal{T}}(D)) \subseteq L_{\mathcal{M}''}(v)$.

Let $N_{\text{def}}^{\mathcal{T}}$ be the set of defined concepts in $\mathcal{T}$. We define a binary relation Y by restricting $Z_{\mathcal{M}''}$ to defined names in $\mathcal{T}$:

$$Y := \{(v_1, v_2) \in Z_{\mathcal{M}''} \mid \{v_1, v_2\} \subseteq N_{\text{def}}^{\mathcal{T}}\}.$$

It holds that Y is finite since $Z_{\mathcal{M}''}$ is finite and $Y \subseteq Z_{\mathcal{M}''}$. Then we compute a binary relation Z from Y by means of the following algorithm:

Algorithm
Input: Y
Output: Z

```
$Z := \{(D, C)\}$;
repeat
  for every $(v_1, v_2) \in Z$ do
    begin
      if $(v_1, r, v_1') \in \mathcal{G}_{\mathcal{T}}$ then
      find “one” $v_2'$ such that $(v_2, r, v_2') \in \mathcal{G}_{\mathcal{T}}$ and $(v_1', v_2') \in Y$;
      $Z := Z \cup \{(v_1', v_2')\}$;
    end;
until $Z$ is not increased;
return $Z$.
```

This algorithm always stops since Y is finite. Moreover, we have that $Z \subseteq Y$ since $(D, C) \in Y$ and the added elements of Z are also in Y. When computing Z, if there is more than one v_2' satisfying the condition we select arbitrarily one of them.
Claim 1: For every $(v_1, v_2) \in Z$, if $(v_1, r, v_1') \in \mathcal{G}_{\mathcal{T}}$ then v_2' as required by the above algorithm always exists.
Proof of Claim 1: For every $(v_1, v_2) \in Z$, if $(v_1, r, v_1') \in \mathcal{G}_{\mathcal{T}}$ then there exists a v_2' such that $(v_2, r, v_2') \in \mathcal{G}_{\mathcal{M}''}$ and $(v_1', v_2') \in Z_{\mathcal{M}''}$ since

- $Z_{\mathcal{M}''} : \mathcal{G}_{\mathcal{M}''} \precsim \mathcal{G}_{\mathcal{M}''}$;
- $(v_1', v_2') \in Z \subseteq Y \subseteq Z_{\mathcal{M}''}$;
- $(v_1, r, v_1') \in E_{\mathcal{M}''}$ since $(v_1, r, v_1') \in E_{\mathcal{T}} \subseteq E_{\mathcal{M}''}$.

Moreover, $(v_2, r, v_2') \in \mathcal{G}_{\mathcal{M}''}$ implies $v_2 \equiv \cdots \sqcap \exists r.v_2' \sqcap \cdots$ in $\mathcal{M}''$.
In the computation of Z, some paths from C are generated according to the corresponding paths from D in $\mathcal{G}_{\mathcal{T}}$. From the definition of $\mathcal{EL}$-matching problems, we know that

$Var_{\mathcal{T}}(C)$ contains no variables. This implies $v_2 \equiv \cdots \sqcap \exists r.v_2' \sqcap \cdots$ in $\mathcal{T}$ for every node v_2 in the path generated path from C. Hence, we have that $v_2' \in N_{\text{def}}^{\mathcal{T}}$. Thus, there exists a v_2' such that $(v_2, r, v_2') \in \mathcal{G}_{\mathcal{T}}$ and $(v_1', v_2') \in Y$.

□ (Claim 1)

$Z : \mathcal{G}_{\mathcal{T}} \precsim \mathcal{G}_{\mathcal{T}}$ since the conditions (S1) and (S2) in Definition 7 (extended to concept patterns) hold.

(S1) $(v_1, v_2) \in Z$ implies $(L_{\mathcal{T}}(v_1) \setminus Var_{\mathcal{T}}(D)) \subseteq L_{\mathcal{T}}(v_2)$ since

$$(L_{\mathcal{T}}(v_1) \setminus Var_{\mathcal{T}}(D)) \subseteq L_{\mathcal{M}''}(v_1) \subseteq L_{\mathcal{M}''}(v_2) = L_{\mathcal{T}}(v_2).$$

(S2) immediate consequence of the construction of Z.

Then we compute $\mathcal{T}'$ and $\mathcal{T}''$ by Algorithm 22 starting from Z.
Claim 2: $\mathcal{T}''$ is a matcher of $C \equiv^?_{\text{gfp},\mathcal{T}} D$ and $\mathcal{T}'' \sqsubseteq_s \mathcal{M}''$.
Proof of Claim 2: To prove that $\mathcal{T}''$ is a matcher of $C \equiv^?_{\text{gfp},\mathcal{T}} D$, it remains to show that $C \sqsupseteq_{\text{gfp},\mathcal{T}''} D$. By renaming, we obtain TBoxes $\mathcal{T}''$ and $\mathcal{M}''$ satisfying the conditions in Definition 33. For every variable X we suppose that $X^{\mathcal{T}} \equiv (A_1^{\mathcal{T}}, \ldots, A_n^{\mathcal{T}})$ in $\mathcal{T}''$ and $X^{\mathcal{M}} \equiv E_X$ in $\mathcal{M}''$. $X^{\mathcal{T}} \equiv (A_1^{\mathcal{T}}, \ldots, A_n^{\mathcal{T}})$ implies that there are defined concepts $M_1^{\mathcal{T}}, \ldots, M_n^{\mathcal{T}}$ in $\mathcal{T}$ such that $(M_1^{\mathcal{T}}, A_1^{\mathcal{T}}), \ldots, (M_n^{\mathcal{T}}, A_n^{\mathcal{T}}) \in Z$ and $X \in L_{\mathcal{T}}(M_1^{\mathcal{T}}) \cap \cdots \cap L_{\mathcal{T}}(M_n^{\mathcal{T}})$. We show that

$$X_{\mathcal{T}} \sqsubseteq_{\text{gfp},\mathcal{T}''\cup\mathcal{M}''} X_{\mathcal{M}}.$$

The relation Z generated by the above algorithm has the property that every node occurring in the second component of tuples in Z is reachable from C. This implies that $L_{\mathcal{T}}(A_i^{\mathcal{T}})$ contains no variables for all $i \in \{1, \ldots, n\}$. From [Baa02a] we know that $(A_1^{\mathcal{T}}, \ldots, A_n^{\mathcal{T}})$ in $\mathcal{T}''$ is the least common subsumer of $A_1^{\mathcal{T}}, \ldots, A_n^{\mathcal{T}}$ in $\mathcal{T}''$. Here we view $\mathcal{T}''$ is an conservative extension of itself. Hence, for all $i \in \{1, \ldots, n\}$, we have

$$A_i^{\mathcal{T}} \sqsubseteq_{\text{gfp},\mathcal{T}''} (A_1^{\mathcal{T}}, \ldots, A_n^{\mathcal{T}}).$$

$\mathcal{T}''\cup\mathcal{M}''$ is a conservative extension of $\mathcal{T}''$ and of $\mathcal{M}''$ since $\mathcal{T}''$ and $\mathcal{M}''$ have the same set of primitive concepts and the same set of roles. For every $i \in \{1, \ldots, n\}$, $(M_i^{\mathcal{T}}, A_i^{\mathcal{T}}) \in Z$ yields $(M_i^{\mathcal{M}}, A_i^{\mathcal{M}}) \in Z_{\mathcal{M}''}$ by construction of Z. Thus, $A_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{M}''} M_i^{\mathcal{M}}$. This yields

$$A_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{T}''\cup\mathcal{M}''} M_i^{\mathcal{M}}.$$

Moreover, $M_i^{\mathcal{T}} \sqsubseteq_{\text{gfp},\mathcal{T}} X^{\mathcal{T}}$ since $X \in L_{\mathcal{T}}(M_i)$ implies $M_i \equiv \cdots \sqcap X \sqcap \cdots$ in $\mathcal{T}$. Thus, $M_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{M}} X^{\mathcal{M}}$ since $\mathcal{M}$ is obtained by renaming defined concepts in $\mathcal{T}$. Hence, $M_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{M}''} X^{\mathcal{M}}$ and thus

$$M_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{T}''\cup\mathcal{M}''} X^{\mathcal{M}}.$$

Together with $A_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{T}''\cup\mathcal{M}''} M_i^{\mathcal{M}}$ this yields $A_i^{\mathcal{M}} \sqsubseteq_{\text{gfp},\mathcal{T}''\cup\mathcal{M}''} X^{\mathcal{M}}$. For all $i \in \{1, \ldots, n\}$, we have

- $A_i^{\mathcal{T}} \sqsubseteq_{\text{gfp},\mathcal{T}''} (A_1^{\mathcal{T}}, \ldots, A_n^{\mathcal{T}})$ and

- $A_i^{\mathcal{T}} \sqsubseteq_{\mathrm{gfp},\mathcal{T}''\cup\mathcal{M}''} X^{\mathcal{M}}$.

By definition of the least common subsumers, we have

$$(A_1^{\mathcal{T}}, \ldots, A_n^{\mathcal{T}}) \sqsubseteq_{\mathrm{gfp},\mathcal{T}''\cup\mathcal{M}''} X^{\mathcal{M}}.$$

This implies

$$(*) \qquad \forall X \in Var_{\mathcal{T}}(D).X^{\mathcal{T}} \sqsubseteq_{\mathrm{gfp},\mathcal{T}''\cup\mathcal{M}''} X^{\mathcal{M}}$$

since $X^{\mathcal{T}} \equiv (A_1, \ldots, A_n)$ in $\mathcal{T}''$. Hence, $D^{\mathcal{T}} \sqsubseteq_{\mathrm{gfp},\mathcal{T}''\cup\mathcal{M}''} D^{\mathcal{M}}$. So, we have that

$$C^{\mathcal{T}} \equiv_{\mathrm{gfp},\mathcal{T}''\cup\mathcal{M}''} C^{\mathcal{M}} \sqsupseteq_{\mathrm{gfp},\mathcal{T}''\cup\mathcal{M}''} D^{\mathcal{M}} \sqsupseteq_{\mathrm{gfp},\mathcal{T}''\cup\mathcal{M}''} D^{\mathcal{T}}.$$

$C^{\mathcal{T}} \sqsupseteq_{\mathrm{gfp},\mathcal{T}''\cup\mathcal{M}''} D^{\mathcal{T}}$ implies $C^{\mathcal{T}} \sqsupseteq_{\mathrm{gfp},\mathcal{T}''} D^{\mathcal{T}}$. Then, $\mathcal{T}''$ is a matcher of $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$.
From $(*)$ we have $\mathcal{T}'' \sqsubseteq_s \mathcal{M}''$. □ (Claim 2)
Claim 2 proves Theorem 37. □

Chapter 6

Implementation

We will illustrate the implementation of Algorithm 22. The programming language is LISP. The programming environment is Allegro LISP system (see `http://www.franz.com`).

6.1 Input $C \equiv^?_{\text{gfp},\mathcal{T}} D$ and normalization of TBox

The input of Algorithm 22 is an $\mathcal{EL}$-matching problem $C \equiv^?_{\text{gfp},\mathcal{T}} D$. C and D are defined concepts in $\mathcal{T}$. In the implementation we store the TBox $\mathcal{T}$ as a file using standard LISP syntax. We use the constants (listed in Table 6.1) to represent the constructors and top-concept defined in $\mathcal{EL}$. The following example will be used in this chapter to illustrate the result of running the matching algorithm step by step.

Example 38
Let $\mathcal{T}$ contain the following concept definitions:

$$\begin{array}{rcl}
D & \equiv & P_1 \sqcap X_2 \sqcap \exists r_1.A_1 \sqcap \exists r_2.A_2 \\
A_1 & \equiv & P_2 \sqcap X_1 \sqcap \exists r_2.D \\
A_2 & \equiv & P_1 \sqcap X_1 \sqcap X_2 \sqcap \exists r_2.D \\
C & \equiv & P_1 \sqcap \exists r_1.A_3 \sqcap \exists r_2.A_4 \\
A_3 & \equiv & P_2 \sqcap P_3 \sqcap \exists r_2.C \\
A_4 & \equiv & P_1 \sqcap P_3 \sqcap \exists r_2.C
\end{array}$$

Then this TBox is stored in a file as

```
(DEFCONCEPT  D   (AND P1 var_x2 (SOME R1 A1) (SOME R2 A2)))
(DEFCONCEPT  A1  (AND P2 var_x1 (SOME R2 D)))
(DEFCONCEPT  A2  (AND P1 var_x1 var_x2 (SOME R2 D)))
(DEFCONCEPT  C   (AND P1 (SOME R1 A3) (SOME R2 A4)))
(DEFCONCEPT  A3  (AND P2 P3 (SOME R2 C)))
(DEFCONCEPT  A4  (AND P1 P3 (SOME R2 C)))).
```

An algorithm to normalized $\mathcal{EL}$-TBoxes has been implemented by Suntisrivaraporn (see [Sun04]). The normalized TBox is translated into the description graph. In the implementation, the function `my-start (file-name)` initializes two hash tables storing the

constant	value in LISP	value in $\mathcal{EL}$
`*defconcept-keyword*`	`'DEFCONCEPT`	$\equiv$
`*top-keyword*`	`'TOP`	$\top$
`*and-keyword*`	`'AND`	$\sqcap$
`*some-keyword*`	`'SOME`	$\exists$
`*varprefixS*`	`'VAR_`	

Table 6.1: Syntax in LISP

information about the description graph of the normalized TBox in the file `filename`. The two hash tables are named `*t-graph-label-hash*` and `*t-graph-out-edge-hash*`. The key of `*t-graph-label-hash*` is the nodes of the description graph and the value is the label of the corresponding node. The key of `*t-graph-out-edge-hash*` consists of a node and a role name. The value is the list of the corresponding node's successors. In the function `init-tbox-t ()`, we generate the following basic information about input TBox $\mathcal{T}$:

- The list `*t-defined-concepts*`: the set of defined concepts.
- The list `*t-var*`: the set of variables.
- The list `*t-n-role*`: the set of roles.

The following example shows the values after initialization.

Example 39
Let the input TBox $\mathcal{T}$ in Example 38 be stored in the file `input-el-tbox`. After running `(my-start "input-el-tbox")` and `(init-tbox-t)` we get two hash tables in Table 6.2 and three lists in Table 6.3.

`*t-graph-label-hash*`		`*t-graph-out-edge-hash*`	
key	value	key	value
C	(P1)	(C . R1)	(A3)
A1	(VAR_X1 P2)	(D . R2)	(A2)
A2	(VAR_X2 VAR_X1 P1)	(C . R2)	(A4)
A3	(P3 P2)	(A1 . R2)	(D)
A4	(P3 P1)	(A3 . R2)	(C)
D	(VAR_X2 P1)	(D . R1)	(A1)
		(A2 . R2)	(D)
		(A4 . R2)	(C)

Table 6.2: The hash tables for $\mathcal{T}$

name	value
`*t-defined-concepts*`	`(A4 A3 A2 A1 D C TOP)`
`*t-var*`	`(VAR_X2 VAR_X1)`
`*t-n-role*`	`(R2 R1)`

Table 6.3: The lists

Note that the top-concept $\top$ is viewed as a node whose label is the empty set and from whom there are no out-edges in the description graph of the underlying TBox. `TOP` is a member of the set of defined concepts `*t-defined-concepts*` since the nodes in a description graph are considered as defined concepts.
The function `init-tbox-t ()` is called by the function `matching-el (c d)` which corresponds to Algorithm 22.

6.2 Computation of simulation relations

Let $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$ be the input matching problem. In Algorithm 22, we consider all simulation relations containing (D, C). This makes some simulation relations contain redundant tuples, i.e., there are some simulation relations such that if we remove some tuples from them, the obtained relations are still simulation relations. For example, if Z_1 and Z_2 are simulation relations on $\mathcal{G}_\mathcal{T}$, then it is easy to show that $Z := Z_1 \cup Z_2$ is also a simulation relation on $\mathcal{G}_\mathcal{T}$. However, this implies that the TBox $\mathcal{T}''$ computed from Z by Algorithm 22 is more general than the TBoxes computed from Z_1 and Z_2. This is because for every variable $X \in Var_\mathcal{T}(D)$, the definition of X in $\mathcal{T}''$ depends on $Z(X)$ for the corresponding simulation relation Z and in this example $Z_1, Z_2 \subseteq Z$ implies $Z_1(X), Z_2(X) \subseteq Z(X)$. By Lemma 12, the definition of X is the lcs of the elements in $Z(X)$. Hence, $Z_1(X), Z_2(X) \subseteq Z(X)$ implies the definition of X computing from Z is more general than the ones from Z_1 and Z_2. Since we are only interested in the minimal matchers, we will only consider the simulation relations containing no redundant tuples, i.e., every tuple is necessary for being a simulation relation. So, in the implementation, we compute simulation relations in the following way:

- start from $Z := (D, C)$ (check condition (S1) on (D, C));
- check condition (S2);
  - yes: return Z;
  - no: add one possible (to satisfy condition (S1)) and necessary (to satisfy condition (S2)) tuple to Z and store this backchecking point;
- if find one tuple in the last step, then run the last step again, otherwise check another possible simulation relation at the backchecking point.

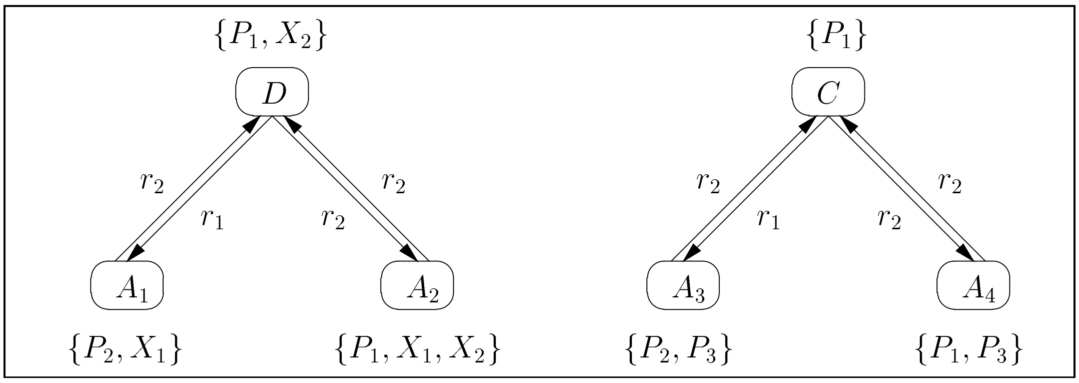

Figure 6.1: The description graph of $\mathcal{T}$

When storing the backchecking points we associated a hash table which labels the checked edges in the description graph $\mathcal{G}_\mathcal{T}$ to every possible simulation relation. Using this strategy, we can obtain all simulation relations which generate the minimal matchers. In the implementation, the function `find-s-r-containing-d-c-without-var (d c)` returns the list of such simulation relations.

Example 40
Consider the input matching problem from Example 38. The description graph of $\mathcal{T}$ is depicted in Figure 6.1. Then, after running `find-s-r-containing-d-c-without-var (d c)`, we get the following list of simulation relations:

```
(((A1 A3) (D C) (A2 A4)))
```

Note that certainly there exist other simulation relations on $\mathcal{G}_\mathcal{T}$ not listed above. However, the ones in the returned list above are more useful for the construction of minimal matchers.

6.3 Computation of TBox $\mathcal{T}'$

In our matching algorithm, the purpose of computing the TBox $\mathcal{T}'$ is to obtain a conservative extension of $\mathcal{T}$ where the new defined concepts is introduced for the computation of the least common subsumers. For a simulation relation Z, $\mathcal{T}'$ is union of the product of $\mathcal{T}$ whose power is decided by the maximal value of $|Z(X)|$ for all $X \in Var_\mathcal{T}(D)$. It should be noticed that if we compute $\mathcal{T}'$ using the formal product's definition, too many (the number of defined concepts in $\mathcal{T}$ to the power of $\max\{Z(X) \mid X \in Var_\mathcal{T}(D)\}$) defined concepts will be generated many of whom might be irrelevant for the computation of the result. In order to decrease the number of defined concepts in $\mathcal{T}'$, we sort the lists which are the names of the new defined concepts in $\mathcal{T}'$ in the implementation . For example, (A_1, A_2) and (A_2, A_1) are expressed by the same node in the description graph of $\mathcal{T}'$. We can do so because both of these two nodes represent the lcs of A_1 and A_2. The other advantage of sorting the list is that we can compare ordered lists faster than non-ordered lists when we check whether some node has already been generated in the product of the description graph. Moreover, we extend the description graph of $\mathcal{T}'$ only using the nodes those are reachable from sorted $(A_1, \dots, A_n)$ for some $X \in Var_{\mathcal{T}}(D)$ and $Z(X) = \{A_1, \dots, A_n\}$. This optimization is also used to decrease the number of defined concepts in $\mathcal{T}'$. The function `get-t-prime (z-x-s-r-hash)` computes $\mathcal{T}'$ according to the hash table `z-x-s-r-hash` storing $Z(X)$ for every $X \in Var_{\mathcal{T}}(D)$ and returns two hash tables `tp-graph-label-hash` and `tp-graph-out-edge-hash` which store the information about $\mathcal{T}'$ (similar to the data structures for $\mathcal{T}$).

Example 41
Let the input be the TBox $\mathcal{T}$ from Example 38. Then from the unique simulation relation in Example 40, we obtain $\mathcal{T}'$ stored in two hash tables (see Table 6.4).

tp-graph-label-hash		tp-graph-out-edge-hash	
key	value	key	value
C	(P1)	(C . R1)	(A3)
A1	(VAR_X1 P2)	(D . R2)	(A2)
A2	(VAR_X2 VAR_X1 P1)	(C . R2)	(A4)
A3	(P3 P2)	(A1 . R2)	(D)
A4	(P3 P1)	(A3 . R2)	(C)
D	(VAR_X2 P1)	(D . R1)	(A1)
		(A2 . R2)	(D)
		(A4 . R2)	(C)
lcs-A4-C	(P1)	(lcs-A3-A3 . R2)	(lcs-C-C)
lcs-C-C	(P1)	(lcs-C-C . R2)	(lcs-A4-A4)
lcs-A4-A3	(P3)	(lcs-C-C . R1)	(lcs-A3-A3)
lcs-A3-A3	(P2 P3)	(lcs-A4-A3 . R2)	(lcs-C-C)
lcs-A4-A4	(P1 P3)	(lcs-A4-C . R2)	(lcs-A4-C)
		(lcs-A4-A4 . R2)	(lcs-C-C)

Table 6.4: The hash tables for $\mathcal{T}'$

Note that the names of new defined concepts in $\mathcal{T}'$ are changed from a list to a string. For example, `(A4 C)` is expressed as `lcs-A4-C`. We do this for a mere technical reason: we have to make the concept names recognizable by Suntisrivaraporn's subsumption algorithm.

6.4 Subsumption testing and output

After the computation of the TBox $\mathcal{T}'$, the concept definitions to be assigned to all variables according to current simulation relation can be returned. For every $X \in Var_{\mathcal{T}}(D)$,

$$X \equiv (A_1, \dots, A_n)$$

where $Z(X) = \{A_1, \ldots, A_n\}$ and $|Z(X)| = n$. We use the function `get-z-x-s-r-hash (s-r)` to return a hash table `z-x-s-r-hash` storing $Z(X)$ for every variable $X \in Var_{\mathcal{T}}(D)$, where the argument `s-r` is the simulation relation Z. From this hash table, we can generate the definitions of variables in the TBox $\mathcal{T}''$. The function

```
write-t-double-prime-file(output-el-tbox-tpp
                          tp-graph-label-hash
                          tp-graph-out-edge-hash
                          z-x-s-r-hash)
```

writes the TBox $\mathcal{T}''$ to a file named `output-el-tbox-tpp`. The arguments `tp-graph-lab -el-hash` and `tp-graph-out-edge-hash` are the hash tables providing the information of the TBox $\mathcal{T}'$. Then we call the function `(my-start output-el-tbox-tpp)` to be ready for checking subsumption w.r.t. the TBox $\mathcal{T}''$. If calling the function `(subsumes? c d)` returns `'yes`, then we have found a solution to the matching problem. Hence, we call the function

```
(run-shell-command
 (format nil "cat begin-t-mark ~A >> ~A"
         output-el-tbox-tpp output-s-file))
```

to append the file named by the value of the variable `output-el-tbox-tpp` to the result file named by the value of the variable `output-s-file` where the file `begin-t-mark` stores one line to mark the beginning of every TBox $\mathcal{T}''$ in the result file. After checking all of the simulation relations containing (D, C), we get the result file storing matchers of $C \equiv^?_{\text{gfp},\mathcal{T}} D$. The function `subsumes? (c d)` is provided by the subsumption algorithm by Suntisrivaraporn (see [Sun04]).

Example 42
Consider the input TBox $\mathcal{T}$ from Example 38. After running our matching algorithm, the following contents are written into the result file:

```
***This is the beginning of TBox T''*******************


(DEFCONCEPT C (AND (SOME R2 A4) (SOME R1 A3) P1))
(DEFCONCEPT D (AND (SOME R1 A1) (SOME R2 A2) VAR_X2 P1))
(DEFCONCEPT A1 (AND (SOME R2 D) VAR_X1 P2))
(DEFCONCEPT A2 (AND (SOME R2 D) VAR_X2 VAR_X1 P1))
(DEFCONCEPT A3 (AND (SOME R2 C) P3 P2))
(DEFCONCEPT A4 (AND (SOME R2 C) P3 P1))
(DEFCONCEPT lcs-A4-C (AND (SOME R2 lcs-A4-C) P1))
(DEFCONCEPT lcs-C-C (AND (SOME R1 lcs-A3-A3) (SOME R2 lcs-A4-A4) P1))
(DEFCONCEPT lcs-A4-A3 (AND (SOME R2 lcs-C-C) P3))
(DEFCONCEPT lcs-A3-A3 (AND (SOME R2 lcs-C-C) P2 P3))
```

```
(DEFCONCEPT lcs-A4-A4 (AND (SOME R2 lcs-C-C) P1 P3))
(DEFCONCEPT VAR_X1 lcs-A4-A3)
(DEFCONCEPT VAR_X2 lcs-A4-C)
```

6.5 Testing

The testing of our $\mathcal{EL}$-matching algorithm is mainly based on the testing data of acyclic $\mathcal{EL}$-TBoxes produced by a random generator for $\mathcal{EL}$-matching problems. Instead of generating C and D independently of each other, we randomly generate a concept C and then construct a concept pattern D from C by randomly replacing sub-concepts of C by variables. The reason of doing this is that we want the generated matching problems to be more probably solvable. In [BL04], there is more information about the strategy of generating such matching problems. For cyclic $\mathcal{EL}$-TBoxes, testing matching problems are generated manually.
To test soundness of our matching algorithm, i.e., every TBox $\mathcal{T}''$ in $\mathcal{S}$ is really a matcher to the corresponding input $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$, it is enough to test whether $C \equiv_{\mathrm{gfp},\mathcal{T}''} D$ holds. Since

$$C \equiv_{\mathrm{gfp},\mathcal{T}''} D \iff C \sqsubseteq_{\mathrm{gfp},\mathcal{T}''} D \wedge D \sqsubseteq_{\mathrm{gfp},\mathcal{T}''} C,$$

we can reduce testing equivalence to testing subsumption.
For completeness testing, we compare the results with the output of an existing $\mathcal{ALE}$-matching algorithm (see [BK00a]) implemented by Brandt (see [Bra03]). The $\mathcal{ALE}$-matching algorithm can solve $\mathcal{ALE}$-matching problems without terminological cycles. $\mathcal{EL}$ is a sub-language of $\mathcal{ALE}$, so acyclic $\mathcal{EL}$-TBoxes are used as testing data. This $\mathcal{ALE}$-matching algorithm also generates an s-complete set of matchers to $C \equiv^?_{\mathrm{gfp},\mathcal{T}} D$. The testing strategy for completeness is that for each matcher $\mathcal{M}''$ generated by $\mathcal{ALE}$-matching algorithm, we check whether there exists a matcher $\mathcal{T}''$ to the same matching problem in the output of our $\mathcal{EL}$-matching algorithm such that $\mathcal{T}'' \sqsubseteq_s \mathcal{M}''$.
We run the matching algorithm on 100 acyclic $\mathcal{EL}$-matching problems of average size 23 costing average time 8.2 milliseconds on a standard PC.

Chapter 7

Conclusion

In this thesis, we have defined formally $\mathcal{EL}$-matching problems with terminological cycles and provided an algorithm for solving the problems w.r.t. the greatest fixpoint semantics. Our algorithm follows a strategy analogous to the algorithm in [BK00a] for the acyclic case. Soundness and completeness of the matching algorithm have been shown. Based on these results, matching, one of non-standard inference problems, can be applied to more powerful DLs.
The theoretical complexity of $\mathcal{EL}$-matching problems has not yet been discussed. Deciding $\mathcal{EL}$-matching problems is at least as hard as deciding $\mathcal{EL}$-matching problems only considering acyclic TBoxes. The results about complexity in acyclic case are listed as following [Küs01]:

1. Deciding the solvability of matching problems modulo equivalence in $\mathcal{EL}$ is an NP-complete problem.

2. The cardinality of s-complete sets of matchers may grow exponentially in the size of the matching problem.

3. The cardinality of s-complete sets of matchers can exponentially be bounded in the size of the matching problem.

Corresponding to 1, the complexity of deciding the solvability of matching problems in cyclic case is still an open problem. 2 and 3 lead to the fact that the algorithm of computing s-complete sets of matchers for matching problems in cyclic case is at least an exponential time algorithm. For our matching algorithm, transforming an $\mathcal{EL}$-TBox $\mathcal{T}$ into normal form can be done in time quadratic in $|\mathcal{T}|$ (see [Sun04]). Subsumption testing consumes polynomial time (see [Baa02b]). It takes also polynomial time to compute $Z(X)$ for every $X \in Var_{\mathcal{T}}(D)$ for some simulation relation Z. However, it takes exponential time to find all simulation relations in the worst case. Directly computing the product of TBoxes $\mathcal{T}$ leads to exponentially large TBox $\mathcal{T}'$ in the worst case. These two steps make our matching algorithm exponential.

Bibliography

[Baa02a] F. Baader. Least common subsumers, most specific concepts, and role-value-maps in a description logic with existential restrictions and terminological cycles. LTCS-Report LTCS-02-07, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2002. See http://lat.inf.tu-dresden.de/research/reports.html.

[Baa02b] F. Baader. Terminological cycles in a description logic with existential restrictions. LTCS-Report LTCS-02-02, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2002. See http://lat.inf.tu-dresden.de/research/reports.html.

[BK00a] F. Baader and R. Küsters. Matching in description logics with existential restrictions. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*, pages 261–272, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

[BK00b] A. Borgida and R. Küsters. What's not in a name: Some Properties of a Purely Structural Approach to Integrating Large DL Knowledge Bases. In F. Baader and U. Sattler, editors, *Proceedings of the 2000 International Workshop on Description Logics (DL2000)*, number 33 in CEUR-WS, Aachen, Germany, 2000. RWTH Aachen. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-33/.

[BKBM99] F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.

[BKM98] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. LTCS-Report LTCS-98-09, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1998. See http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[BL04] Sebastian Brandt and Hongkai Liu. Implementing matching in $\mathcal{ALN}$. In *Proceedings of the KI-2004 Workshop on Applications of Description Logics (KI-ADL'04)*, CEUR-WS, Ulm, Germany, September 2004.

[Bra03] Sebastian Brandt. Implementing matching in $\mathcal{ALE}$—first results. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, 2003.

[BS96] F. Baader and U. Sattler. Knowledge representation in process engineering. In *Proceedings of the International Workshop on Description Logics*, Cambridge (Boston), MA, U.S.A., 1996. AAAI Press/The MIT Press.

[BT01] S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (KIDLWS'01)*, number 44 in CEUR-WS, Vienna, Austria, September 2001. RWTH Aachen. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-44/.

[KAS97] Cote RA K. A. Spackman, K. E. Campbell. Snomed rt: A reference terminology for health care. In *Proceedings/AMIA Annual Fall Symposium*, pages 640–644, 1997.

[Küs01] R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 2001. Ph.D. thesis.

[MPS98] Deborah L. McGuinness and Peter F. Patel-Schneider. Usability issues in knowledge representation systems. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 608–614. American Association for Artificial Intelligence, 1998.

[Neb90] Bernhard Nebel. *Reasoning and revision in hybrid representation systems.* Springer-Verlag New York, Inc., 1990.

[Neb91] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[RNG93] A. Rector, W. Nowlan, and A. Glowinski. Goals for concept representation in the galen project. In *Proceedings of the 17th Annual Symposium on Computer Applications in Medical Care (SCAMC'93)*, pages 414–418, Washington DC, USA, 1993.

[Sun04] B. Suntisrivaraporn. Implementation and optimization of subsumption algorithms in the dl el with cyclic tboxes and general concept inclusion axioms. Master's thesis, Dresden University of Technology, Germany, 12 2004. See http://lat.inf.tu-dresden.de/research/reports.html.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.