Non-Standard Inferences in Description Logics: The Story So Far

Franz Baader
Theoretical Computer Science
TU Dresden
baader@inf.tu-dresden.de

Ralf Küsters
Institut für Informatik
Christian-Albrechts-Universität zu Kiel
kuesters@ti.informatik.uni-kiel.de

Abstract

Description logics (DLs) are a successful family of logic-based knowledge representation formalisms that can be used to represent the terminological knowledge of an application domain in a structured and formally well-founded way. DL systems provide their users with inference procedures that allow to reason about the represented knowledge. Standard inference problems (such as the subsumption and the instance problem) are now well-understood. Their computational properties (such as decidability and complexity) have been investigated in detail, and modern DL systems are equipped with highly optimized implementations of these inference procedures, which—in spite of their high worst-case complexity—perform quite well in practice.

In applications of DL systems it has turned out that building and maintaining large DL knowledge bases can be further facilitated by procedures for other, non-standard inference problem, such as computing the least common subsumer and the most specific concept, and rewriting and matching of concepts. While the research concerning these non-standard inferences is not as mature as the one for the standard inferences, it has now reached a point where it makes sense to motivate these inferences within a uniform application framework, give an overview of the results obtained so far, describe the remaining open problems, and give perspectives for future research in this direction.

1 Introduction

Description logics (DLs) [12] are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way. The name *description logics* is motivated by the fact that, on the one hand, the important notions of the domain are described by *concept descriptions*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. For example, the concept of "a man that is married to a doctor, and has only happy children" can be expressed using the concept description

$$\mathsf{Man} \sqcap \exists \mathsf{married}.\mathsf{Doctor} \sqcap \forall \mathsf{child}.\mathsf{Happy}.$$

On the other hand, DLs differ from their predecessors, such as semantic networks and frames [84, 79], in that they are equipped with a formal, *logic*-based semantics, which can, e.g., be given by a translation into first-order predicate logic. For example, the above concept description can be translated into the following first-order formula (with one free variable x):

$$\mathsf{Man}(x) \wedge \exists y.(\mathsf{married}(x, y) \wedge \mathsf{Doctor}(y)) \wedge \forall y.(\mathsf{child}(x, y) \rightarrow \mathsf{Happy}(y)).$$

In addition to the formalism for describing concepts, DLs usually also provide their users with means for describing individuals by stating to which concepts they belong and in which role relationships they participate. For example, the assertions

$$\mathsf{Man}(\mathsf{JOHN}), \quad \mathsf{child}(\mathsf{JOHN}, \mathsf{MARY}), \quad \mathsf{Happy}(\mathsf{MARY})$$

state that the individual John has a child Mary, who is happy.

Knowledge representation systems based on description logics (DL systems or DL reasoners) [95, 81] provide their users with various inference capabilities that deduce implicit knowledge from the explicitly represented knowledge. Standard inference services are *subsumption* and *instance checking*. Subsumption allows the user to determine subconcept-superconcept relationships, and hence, compute a subconcept-superconcept hierarchy: C is subsumed by D iff all instances of C are also instances of D, i.e., the first description is always interpreted as a subset of the second description. Instance checking asks whether a given individual necessarily belongs to a given concept, i.e., whether this instance relationship logically follows from the descriptions of the concept and of the individual.

In order to ensure a reasonable and predictable behaviour of a DL reasoner, these inference problems should at least be decidable for the DL employed by the reasoner, and preferably of low complexity. Consequently, the expressive power of the DL in question must be restricted in an appropriate way. If the imposed restrictions are too severe, however, then the important notions of the application domain can no longer be expressed. Investigating this trade-off between the expressivity of DLs and the complexity of their inference problems has been one of the most important issues of DL research in the 1990ies. As a consequence of this research, the complexity of reasoning in various DLs of different expressive power is now well-investigated (see [49] for an overview of these complexity results). In addition, there are highly optimized implementations of reasoners for very expressive DLs [61, 54, 62], which—despite their high worst-case complexity—behave very well in practice [60, 53].

DLs have been applied in many domains, such as medical informatics, software engineering, configuration of technical systems, natural language processing, databases, and web-based information systems (see Part III of [12] for details on these and other applications). A recent success story is the use of DLs as ontology languages [15, 16] for the Semantic Web [33]. In particular, the W3C recommended ontology web language OWL [64] is based on an expressive description logic [67, 66].

Editors—such as OilEd [32] and the OWL plug-in of Protègè [69]—supporting the design of ontologies in various application domains usually allow their users to access a DL reasoner, which realizes the aforementioned *standard inferences* such as subsumption and instance checking. Reasoning is not only useful when working with "finished" ontologies, it can also support the ontology engineer while building an ontology, by pointing out inconsistencies and unwanted consequences. The ontology engineer can thus use reasoning to check whether the definition of a concept or the description of an individual makes sense.

However, these standard DL inferences—subsumption and instance checking—provide only little support for actually coming up with a first version of the definition of a concept. The non-standard inferences considered in this paper were introduced to overcome this deficit, by allowing the user to construct new knowledge from the existing one. Our own motivation for investigating these novel inferences comes from an application in chemical process engineering where a knowledge base has been built by different knowledge engineers over a rather long period of time [87, 71, 80, 44, 35, 77, 94].

The goal of this paper is (i) to motivate non-standard inferences by means of a simple application scenario, (ii) to provide an overview of the results that have been obtained for non-standard inferences so far, and (iii) to explain the main techniques employed for solving these novel inference problems. In order to be able to describe the latter in detail, the exposition of the techniques is mainly restricted to the DL $\mathcal{ALE}$. However, we also provide references to results for other DLs.

Structure of the paper. In *Section 2*, we introduce typical DL constructors and the most important standard inference problems. In addition, we give a brief review of the different approaches for solving these inference problems, and of their complexity in different DLs. In *Section 3*, we first motivate the need for non-standard inferences in a typical application scenario, and then formally define the most important non-standard inferences in description logics. Then, we briefly introduces the techniques used to solve these problems. Since these techniques depend on a syntactic characterization of the subsumption problem, Section 3 is followed by a section that describes such a characterization for the DL $\mathcal{ALE}$, which we use as a prototypical example (*Section 4*). The *next four sections* consider the four most important non-standard inference problems: computing the *least common subsumer* and the *most specific concept*, *rewriting*, and *matching*. Related non-standard inferences are briefly discussed in the respective sections as well. We explain the results on these four non-standard inferences in $\mathcal{ALE}$ in detail, whereas results for other DLs are reviewed only briefly. Finally, *Section 9* summarizes the results on non-standard inferences obtained so far, and gives perspectives for further research.

Name	Syntax	Semantics
top-concept	$\top$	$\Delta^{\mathcal{I}}$
bottom-concept	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
at-least restriction	$\geq n\, r$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$
at-most restriction	$\leq n\, r$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in r^{\mathcal{I}}\} \leq n\}$
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$r(a,b))$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Table 1: Syntax and semantics of concept descriptions, definitions, and assertions.

2 Description Logics and Standard Inferences

In order to define concepts in a DL knowledge base, one starts with a set N_C of concept names (unary predicates) and a set N_R of role names (binary predicates), and defines more complex *concept descriptions* using the *concept constructors* provided by the concept description language of the particular system. In this paper, we consider the DL $\mathcal{ALCN}$ and some of its sublanguages. *Concept descriptions* of $\mathcal{ALCN}$ are built using the constructors shown in the first part of Table 1. In this table, r stands for a role name, n for a nonnegative integer, A for a concept name, and C, D for arbitrary concept descriptions.

A *concept definition* $A \equiv C$ (as shown in the second part of Table 1) assigns a concept name A to a complex description C. A finite set of such definitions is called a *TBox* iff it is unambiguous, i.e., each name has at most one definition. The concept names occurring on the left-hand side of a concept definition are called *defined* concepts, and the others *primitive*. In many cases, one restricts the attention to *acyclic* TBoxes, where the definition of a defined concept A cannot (directly or indirectly) refer to A itself.

A (concept or role) *assertion* is of the form shown in the last part of Table 1. Here, a, b belong to an additional set N_I of individual names. A finite set of such assertions is called an *ABox*.

The *sublanguages* of $\mathcal{ALCN}$ that will be considered in this paper are shown in Table 2. The first column explains the naming scheme for the members of the $\mathcal{AL}$-family.

The *semantics* of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty set and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role name $r \in N_R$ to a binary relation

Symbol	Syntax	$\mathcal{ALC}$	$\mathcal{ALEN}$	$\mathcal{ALE}$	$\mathcal{ALN}$	$\mathcal{EL}$	$\mathcal{FL}_0$
$\mathcal{AL}$	$\top$	x	x	x	x	x	x
	$\bot$	x	x	x	x		
	$\sqcap$	x	x	x	x	x	x
	$\neg A$	x	x	x	x		
	$\forall r.C$	x	x	x	x		x
$\mathcal{C}$	$\neg C$	x					
$\mathcal{E}$	$\exists r.C$	x	x	x		x	
$\mathcal{U}$	$C \sqcup D$	x					
$\mathcal{N}$	$(\leq n\, r), (\geq n\, r)$		x		x		

Table 2: The relevant sublanguages of $\mathcal{ALCN}$.

$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name $a \in N_I$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1. In the rows treating at-least and at-most number restrictions, $\sharp M$ denotes the cardinality of a set M.

The interpretation $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ iff it satisfies all its concept definitions, i.e., $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all $A \equiv C$ in $\mathcal{T}$, and it is a *model* of the ABox $\mathcal{A}$ iff it satisfies all its assertions, i.e., $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all concept assertions $C(a)$ in $\mathcal{A}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all role assertions $r(a, b)$ in $\mathcal{A}$.

Based on this semantics, we can now formally introduce the standard inference problems in description logics.

Definition 1 *Let $\mathcal{A}$ be an ABox, $\mathcal{T}$ a TBox, C, D concept descriptions, and a an individual name.*

- *C is* satisfiable *w.r.t. $\mathcal{T}$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$.*
- *D* subsumes *C w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$.*
- *$\mathcal{A}$ is* consistent *w.r.t. $\mathcal{T}$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ that is also a model of $\mathcal{A}$.*
- *a is an* instance *of C in $\mathcal{A}$ w.r.t. $\mathcal{T}$ ($\mathcal{A}, \mathcal{T} \models C(a)$) iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$.*

In case the TBox $\mathcal{T}$ is empty, we omit the appendage "w.r.t. $\emptyset$." In particular, we say that D *subsumes* C and write this as $C \sqsubseteq D$. Two concept descriptions are *equivalent* ($C \equiv D$) if they subsume each other (w.r.t. the empty TBox), i.e., if $C \sqsubseteq D$ and $D \sqsubseteq C$. We write $C \sqsubset D$ to express that $C \sqsubseteq D$ but $D \not\sqsubseteq C$.

If the DL under consideration allows for full negation ($\mathcal{C}$), then subsumption and satisfiability are interreducable, and the same is true for the instance and the consistence problem. In addition, satisfiability (subsumption) can always be reduced to ABox-consistency (instance checking). This follows from the following equivalences:

- $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$;
- C is unsatisfiable w.r.t. $\mathcal{T}$ iff $C \sqsubseteq_{\mathcal{T}} \bot$;
- $\mathcal{A}, \mathcal{T} \models C(a)$ iff $\mathcal{A} \cup \{\neg C(a)\}$ is inconsistent w.r.t. $\mathcal{T}$;
- $\mathcal{A}$ is inconsistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}, \mathcal{T} \models \{\bot(a)\}$ where a is an arbitrary individual name;
- C is satisfiable w.r.t. $\mathcal{T}$ iff $\{C(a)\}$ is consistent where a is an arbitrary individual name;
- $C \sqsubseteq_{\mathcal{T}} D$ iff $\{C(a)\}, \mathcal{T} \models D(a)$ where a is an arbitrary individual name.

If the TBox $\mathcal{T}$ is acyclic, then reasoning w.r.t. $\mathcal{T}$ can be reduced to reasoning w.r.t. the empty TBox by expanding concept definitions, i.e., by replacing defined concept by their definitions until all defined concepts have been replaced. This can, however, result in an exponential blow-up of the problem [82].

Most of the early research on reasoning in DLs concentrated on the subsumption problem for concept descriptions (i.e., w.r.t. the empty TBox). For the DLs introduced above, the worst-case complexity of this problem is well-investigated. Subsumption in $\mathcal{ALCN}$, $\mathcal{ALC}$, and $\mathcal{ALEN}$ is PSPACE-complete, whereas subsumption in $\mathcal{ALE}$ is NP-complete. The subsumption problem for $\mathcal{ALN}$, $\mathcal{EL}$, and $\mathcal{FL}_0$ is polynomial (see [49] for references and additional complexity results for other DLs). In the presence of an acyclic TBox, the complexity of subsumption may increase, but not in all cases. For example, subsumption w.r.t. an acyclic TBox in $\mathcal{FL}_0$ is coNP-complete [82], but it remains polynomial in $\mathcal{EL}$ [8] and PSPACE-complete in $\mathcal{ALCN}$ [75]. Cyclic TBoxes may increase the complexity of the subsumption problem even further (e.g., for $\mathcal{FL}_0$ to PSPACE [4, 68]), but again not in all cases (e.g., for $\mathcal{EL}$, subsumption w.r.t. cyclic TBoxes remains polynomial [8]). In most cases, the complexity of the instance problem is the same as the complexity of the subsumption problem (e.g., in $\mathcal{ALCN}$ [57] and $\mathcal{EL}$ [7]), but in some cases it may be harder (e.g., in $\mathcal{ALE}$, where it is PSPACE-complete [51]).

The original KLone system [40] as well as its early successor systems (such as Back [83], KRep [78], and Loom [76]) employed so-called *structural subsumption algorithms*, which first normalise the concept descriptions, and then recursively compare the syntactic structure of the normalised descriptions. These algorithms are usually very efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e., for more expressive DLs they cannot detect all the existing subsumption relationships. The DL $\mathcal{ALN}$ is an example of a DL where this structural approach yields a polynomial-time subsumption algorithm (see [27] for a sketch of such an algorithm and [38] for a detailed description of a structural subsumption algorithm for an extension of $\mathcal{ALN}$). The syntactic characterization of subsumption in $\mathcal{EL}$ and $\mathcal{ALE}$ given in Section 4 can in principle also be used to obtain a structural subsumption algorithm for these DLs. It should be noted, however, that in the case of $\mathcal{ALE}$ the normalization phase is not polynomial. For $\mathcal{EL}$, the normalization phase is void, but a naive top-down structural comparison would not result in a deterministic polynomial-time algorithm. To obtain a polynomial subsumption algorithm, one must use a dynamic programming approach, i.e., work bottom-up. Overall, structural subsumption does not seem to be the right tool for solving standard inferences for expressive DLs. However, as we will see, structural subsumption plays an important role for solving non-standard inferences.

For expressive DLs (in particular, DLs allowing for disjunction and/or negation), for which the structural approach does not lead to complete subsumption algorithms, *tableau algorithms* have turned out to be useful: they are complete and often behave quite well in practice. The first such algorithm was proposed by Schmidt-Schauß and Smolka [89] for the DL $\mathcal{ALC}$.[1] It quickly turned out that this approach for deciding subsumption can be extended to various other DLs [59, 58, 13, 2, 55, 46, 11, 28, 65, 67, 29] and also to other inference problems such as the instance problem [56, 51, 57]. Early on, DL researchers started to call the algorithms obtained this way "tableau-based algorithms" since they observed that the original algorithm by Schmidt-Schauß and Smolka for $\mathcal{ALC}$, as well as subsequent algorithms for more expressive DLs, could be seen as specialisations of the tableau calculus for first-order predicate logic (the main problem to solve was to find a specialisation that always terminates, and thus yields a decision procedure). After Schild [88] showed that $\mathcal{ALC}$ is a syntactic variant of multi-modal K, it turned out that the algorithm by Schmidt-Schauß and Smolka was actually a re-invention of the tableau algorithm for K known from modal logics [34].

The first DL systems employing tableau-based algorithms (Kris [14] and Crack [45]) demonstrated that (in spite of the high worst-case complexity of the underlying DL $\mathcal{ALCN}$) such algorithms can be implemented in a practical way. The complexity barrier has been pushed even further back by the seminal system FaCT [61]. Although FaCT employs the very expressive DL $\mathcal{SHIQ}$, which has an EXPTIME-complete subsumption problem, its highly optimized tableau-based subsumption algorithm outperforms the early systems based on structural subsumption algorithms and Kris by several orders of magnitude [63]. The equally well-performing system Racer [54] also provides for a highly-optimized implementation of the ABox-consistency and instance test for an extension of $\mathcal{SHIQ}$.

3 Non-standard Inferences—Motivation and Definitions

In this section, we will first motivate the non-standard inferences considered in this paper within a uniform application scenario, in which these inferences are used to support the design of DL knowledge bases. Then, we give formal definitions of the relevant non-standard inferences, and briefly sketch different techniques for solving them. Each non-standard inference will be considered in more detail in a separate section, where we concentrate on

[1] Actually, at that time the authors were not aware of the close connection between their rule-based algorithm working on constraint systems and tableau procedures for modal and first-order predicate logics.

the DL $\mathcal{ALE}$.

3.1 Motivation

As mentioned in the introduction, the standard DL inferences introduced in Section 2 can already be employed during the design phase of a DL knowledge base since they allow the knowledge engineer to check whether the definition of a concept make senses (i.e., whether the defined concept is satisfiable) and whether it behaves as expected (i.e., whether the computed subsumption relationships are the ones intuitively expected). However, inferences such as subsumption provide no support for actually coming up with a first version of the definition of a concept. The non-standard inferences introduced in this section can be used to overcome this deficit, basically by providing two ways of re-using "old" knowledge when defining new one: (i) constructing concepts by generalizing from examples, and (ii) constructing concepts by modifying "similar" ones.

The first approach was introduced as *bottom-up construction* of description logic knowledge bases in [17, 22]. Instead of defining the relevant concepts of an application domain from scratch, this methodology allows the user to give typical examples of individuals belonging to the concept to be defined. These individuals are then generalized to a concept by first computing the most specific concept (msc) of each individual (i.e., the least concept description in the available description language that has this individual as an instance), and then computing the least common subsumer (lcs) of these concepts (i.e., the least concept description in the available description language that subsumes all these concepts). The knowledge engineer can then use the computed concept as a starting point for the concept definition. As a simple example, assume that the knowledge engineer has already defined the concept of a man and a woman as

$$\mathsf{Man} \equiv \mathsf{Human} \sqcap \mathsf{Male} \quad \text{and} \quad \mathsf{Woman} \equiv \mathsf{Human} \sqcap \mathsf{Female},$$

and now wants to define the concept of a parent, but does not know how to do this within the available DL (which we assume to be $\mathcal{EL}$ in this example). However, the available ABox

$$\begin{array}{lll} \mathsf{Man}(\mathsf{JACK}), & \mathsf{child}(\mathsf{JACK}, \mathsf{CAROLINE}), & \mathsf{Woman}(\mathsf{CAROLINE}), \\ \mathsf{Woman}(\mathsf{JACKIE}), & \mathsf{child}(\mathsf{JACKIE}, \mathsf{JOHN}), & \mathsf{Man}(\mathsf{JOHN}), \end{array}$$

contains the individuals JACK and JACKIE, of whom the knowledge engineer knows that they are parents. The most specific concepts of JACK and JACKIE in the given ABox are

$$\mathsf{Man} \sqcap \exists \mathsf{child}.\mathsf{Woman} \quad \text{and} \quad \mathsf{Woman} \sqcap \exists \mathsf{child}.\mathsf{Man},$$

respectively, and the least common subsumer (in $\mathcal{EL}$) of these two concepts w.r.t. the definitions of Man and Woman is

$$\mathsf{Human} \sqcap \exists \mathsf{child}.\mathsf{Human},$$

which looks like a good starting point for a definition of parent.

In contrast to standard inferences such as subsumption and instance checking, the output of the non-standard inferences we have mentioned until now (computing the msc and the lcs) is a concept description rather than a yes/no answer. In such a setting, it is important that the returned descriptions are as readable and comprehensible as possible. Unfortunately, the descriptions that are produced by the known algorithms for computing the lcs and the msc do not satisfy this requirement. The reason is that—like most algorithms for the standard inference problems—these algorithms work on expanded concept descriptions, i.e., concept descriptions that do not contain names defined in the underlying TBox. Consequently, the descriptions that the algorithms produce also do not use defined concepts, which makes them in many cases large and hard to read and comprehend. In the above example, this means that the definitions of Man and Woman are expanded before applying the lcs algorithm. If Human also had a definition, then it would also be expanded, and instead of the concept description containing Human shown above, the algorithm would return its expanded version.

This problem can be overcome by *rewriting* the resulting concept w.r.t. the given TBox. Informally, the problem of rewriting a concept given a terminology can be stated as follows: given an acyclic TBox $\mathcal{T}$ and a concept description C that does not contain concept names defined in $\mathcal{T}$, can this description be rewritten into an equivalent shorter description E by using (some of) the names defined in $\mathcal{T}$? For example, w.r.t. the TBox

$$\begin{array}{rcl} \mathsf{Woman} & \equiv & \mathsf{Human} \sqcap \mathsf{Female}, \\ \mathsf{Man} & \equiv & \mathsf{Human} \sqcap \mathsf{Male}, \\ \mathsf{Parent} & \equiv & \mathsf{Human} \sqcap \exists \mathsf{child}.\mathsf{Human}, \end{array}$$

the concept description

$$\mathsf{Human} \sqcap \forall \mathsf{child}.\mathsf{Female} \sqcap \exists \mathsf{child}.\top \sqcap \forall \mathsf{child}.\mathsf{Human}$$

can be rewritten to the equivalent concept $\mathsf{Parent} \sqcap \forall \mathsf{child}.\mathsf{Woman}$.

In order to apply the second approach of constructing concepts by modifying existing ones, one must first find the right candidates for modification. One way of doing this is to give a partial description of the concept to be defined as a concept pattern (i.e., a concept description containing variables), and then look for concept descriptions that match this pattern. For example, the pattern

$$\mathsf{Man} \sqcap \exists \mathsf{child}.(\mathsf{Man} \sqcap X) \sqcap \exists \mathsf{spouse}.(\mathsf{Woman} \sqcap X)$$

looks for descriptions of classes of men whose wife and son share some characteristic. An example of a concept description *matching* this pattern is

$$\mathsf{Man} \sqcap \exists \mathsf{child}.(\mathsf{Man} \sqcap \mathsf{Tall}) \sqcap \exists \mathsf{spouse}.(\mathsf{Woman} \sqcap \mathsf{Tall}).$$

We refer the reader to [71, 44, 24] for a description of other possible applications of non-standard inferences.

3.2 Definitions

In the following, we formally define the most important non-standard inferences.

Least Common Subsumer. Intuitively, the least common subsumer of a given collection of concept descriptions is a description that represents the properties that all the elements of the collection have in common. More formally, it is the most specific concept description that subsumes the given descriptions. How this most specific description looks like, whether it really captures the intuition of representing the properties common to the input descriptions, and whether it exists at all strongly depends on the DL under consideration.

Definition 2 *Let $\mathcal{L}$ be a DL. A concept description E of $\mathcal{L}$ is a* least common subsumer *(lcs) of the concept descriptions $C_1, \ldots, C_n$ in $\mathcal{L}$ ($lcs_{\mathcal{L}}(C_1, \ldots, C_n)$ for short) iff it satisfies*

1. *$C_i \sqsubseteq E$ for all $i = 1, \ldots, n$, and*
2. *E is the least $\mathcal{L}$ concept description with this property, i.e., if E' is an $\mathcal{L}$ concept description satisfying $C_i \sqsubseteq E'$ for all $i = 1, \ldots, n$, then $E \sqsubseteq E'$.*

As an easy consequence of this definition, the lcs is unique up to equivalence, which justifies talking about *the* lcs. In addition, the n-ary lcs as defined above can be reduced to the binary lcs (the case $n = 2$ above). Indeed, it is easy to see that $lcs_{\mathcal{L}}(C_1, \ldots, C_n) \equiv lcs_{\mathcal{L}}(C_1, \ldots, lcs_{\mathcal{L}}(C_{n-1}, C_n) \cdots)$. Thus, it is enough to devise algorithms for computing the binary lcs.

It should be noted, however, that the lcs does not always need to exist. This can have several reasons: (a) there may not exist a concept description in $\mathcal{L}$ satisfying (1) of the definition (i.e., subsuming $C_1, \ldots, C_n$); (b) there may be several subsumption incomparable minimal concept descriptions satisfying (1) of the definition; (c) there may be an infinite chain of more and more specific descriptions satisfying (1) of the definition. Obviously, (a) cannot occur for DLs containing the top concept. It is easy to see that, for DLs allowing for conjunction, (b) cannot occur. Case (c) is also rare to occur for DLs allowing for conjunction, but this is less obvious to see. Basically, for many DLs one can use the role depth of the concepts $C_1, \ldots, C_n$ to restrict the role depth of (relevant) common subsumers. The existence of the lcs then follows from the presence of conjunction and the fact that, up to equivalence, there are only finitely many concepts over a finite vocabulary having a fixed role depth (see [30] for more details). An example where case (c) actually occurs is the DL $\mathcal{EL}$ with cyclic terminologies interpreted with descriptive semantics [5] (see also Section 5.3).

It is clear that in DLs allowing for disjunction, the lcs of $C_1, \ldots, C_n$ is their disjunction $C_1 \sqcup \ldots \sqcup C_n$. In this case, the lcs is not of interest. In fact, as we have said above, the lcs is supposed to make explicit the properties that the input concepts have in common.

This is, of course, not achieved by writing down their disjunction. Hence, the lcs appears to be useful only in cases where the DL does not allow for disjunction.

Definition 2 is formulated for concept descriptions, i.e., it does not take a TBox into account. For acyclic TBoxes, this is not a real restriction since one can first expand the definitions before computing the lcs, and then apply rewriting to the lcs to obtain an equivalent shorter description containing defined concepts. For cyclic TBoxes, expansion is not possible. In addition, it may be advantageous to use cycles within the definition of the lcs, i.e., to allow the TBox to be extended by additional (possibly cyclic) concept definitions [17, 7]. The use of cyclic TBoxes in this context is also motivated by the most specific concept (see below).

Most Specific Concept. The most specific concept of a given ABox individual captures all the properties of the individual that are expressible by a concept description of the DL under consideration. Again, the form of the most specific concept and its existence strongly depend on this DL.

Definition 3 *Let $\mathcal{L}$ be a DL. The $\mathcal{L}$ concept description E is the* most specific concept *(msc) in $\mathcal{L}$ of the individual a in the $\mathcal{L}$ ABox $\mathcal{A}$ ($msc_{\mathcal{L}}(a)$ for short) iff*

1. *$\mathcal{A} \models E(a)$, and*
2. *E is the least $\mathcal{L}$ concept description satisfying (i), i.e., if E' is an $\mathcal{L}$ concept description satisfying $\mathcal{A} \models E'(a)$, then $E \sqsubseteq E'$.*

As with the lcs, the msc is unique up to equivalence, if it exists. In contrast to the lcs, which always exists for the DLs shown in Table 2, the msc does not always exist in these DLs. This is due to the presence of so-called role cycles in the ABox. For example, w.r.t. the ABox

$$\{\mathsf{loves}(\mathsf{NARCIS}, \mathsf{NARCIS}),\ \mathsf{Vain}(\mathsf{NARCIS})\},$$

the individual NARCIS does not have an msc in $\mathcal{EL}$. In fact, assume that E is the msc of NARCIS. Then E has a finite role depth, i.e., a finite maximal number of nestings of existential restrictions. If this role depth is smaller than n, then E is not subsumed by the $\mathcal{EL}$ concept description

$$E' := \underbrace{\exists \mathsf{loves}. \cdots \exists \mathsf{loves}.}_{n \text{ times}} \mathsf{Vain},$$

in spite of the fact that NARCIS is an instance of E'.

One way to overcome this problem is to allow for cyclic TBoxes interpreted with greatest fixpoint semantics. In the above example, the defined concept

$$\mathsf{Narcis} \equiv \mathsf{Vain} \sqcap \exists \mathsf{loves}.\mathsf{Narcis}$$

is an msc of the individual NARCIS in $\mathcal{EL}$ w.r.t. cyclic TBoxes with greatest fixpoint semantics. In order to employ this approach in the bottom-up construction of DL knowledge bases, the impact of such cyclic definitions on the subsumption problem and the problem of computing the lcs must also be dealt with. In [17] this is done for $\mathcal{ALN}$, and in [8, 7] for $\mathcal{EL}$. Another possibility is to approximate the msc by restricting the attention to concept descriptions whose role depth is bounded by a fixed number k [48, 73] (see Section 6 for details).

Rewriting. In [23], a very general framework for rewriting in DLs is introduced, which has several interesting instances. In order to introduce this framework, we fix a set N_R of role names and a set N_P of primitive concept names.

Definition 4 *Let $\mathcal{L}_s$, $\mathcal{L}_d$, and $\mathcal{L}_t$ be three DLs (the source-, destination, and TBox-DL, respectively). A* rewriting problem *is given by*

- *an $\mathcal{L}_t$ TBox $\mathcal{T}$ containing only role names from N_R and primitive concepts from N_P; the set of defined concepts occurring in $\mathcal{T}$ is denoted by N_D;*
- *an $\mathcal{L}_s$ concept description C using only the names from N_R and N_P;*
- *a binary relation ρ between $\mathcal{L}_s$ and $\mathcal{L}_d$ concept descriptions.*

An $\mathcal{L}_d$ rewriting of C using $\mathcal{T}$ *is an $\mathcal{L}_d$ concept description E built using role names from N_R and concept names from $N_P \cup N_D$ such that $C \rho E$. Given an appropriate ordering $\preceq$ on $\mathcal{L}_d$ concept descriptions, a rewriting E is called* $\preceq$-minimal *iff there does not exist a rewriting E' such that $E' \prec E$.*

In this paper, we consider one instances of this general framework in more detail, the minimal rewriting problem [23], and briefly discuss another instance, the approximation problem [42]. The *minimal rewriting problem* is the instance of the framework where

- all three DLs are the same language $\mathcal{L}$;
- the TBox $\mathcal{T}$ is acyclic;
- the binary relation ρ corresponds to equivalence modulo the TBox;
- $\mathcal{L}$ concept descriptions are ordered by size, i.e., $E \preceq E'$ iff $|E| \leq |E'|$, where the size $|E|$ of a concept description E is defined to be the number of occurrences of concept and role names in E.

The *approximation problem* is the instance of the framework where

- $\mathcal{T}$ is empty, and thus $\mathcal{L}_t$ is irrelevant;

- both ρ and $\preceq$ are the subsumption relation $\sqsubseteq$.

Given two DLs $\mathcal{L}_s$ and $\mathcal{L}_d$, an $\mathcal{L}_d$ *approximation* of an $\mathcal{L}_s$ concept description C is thus an $\mathcal{L}_d$ concept description D such that $C \sqsubseteq D$ and D is minimal (w.r.t. subsumption) in $\mathcal{L}_d$ with this property. Typically, $\mathcal{L}_d$ is a less expressive DL than $\mathcal{L}_s$, and hence, D is the best approximation from above of C in $\mathcal{L}_d$. One motivation for approximation is to be able to translate a knowledge base expressed in an expressive DL into a knowledge base expressed in a less expressive DL [23, 42].

Matching. Before we can define matching, we must define the notion of a pattern. *Concept patterns* are concept descriptions in which *concept variables* (usually denoted by X, Y, etc.) may occur in place of concept names. However, concept variables may not occur in the scope of a negation. The main difference between concept names and concept variables is that the latter can be replaced by concept descriptions when applying a substitution. For example,

$$D := P \sqcap X \sqcap \forall r.(Y \sqcap \forall r.X)$$

is a concept pattern containing the concept variables X and Y. By applying the substitution $\sigma := \{X \mapsto Q,\ Y \mapsto \forall r.P\}$ to it, we obtain the concept description

$$\sigma(D) = P \sqcap Q \sqcap \forall r.(\forall r.P \sqcap \forall r.Q).$$

Definition 5 *Let C be a concept description and D a concept pattern. Then $C \equiv^? D$ is called a* matching problem modulo equivalence *and $C \sqsubseteq^? D$ is called a* matching problem modulo subsumption*. The substitution σ is a* matcher *of the matching problem $C \equiv^? D$ ($C \sqsubseteq^? D$) iff $C \equiv \sigma(D)$ ($C \sqsubseteq \sigma(D)$).*

Since $C \sqsubseteq \sigma(D)$ iff $C \sqcap \sigma(D) \equiv C$, the matching problem modulo subsumption $C \sqsubseteq^? D$ can be reduced to the following matching problem modulo equivalence: $C \equiv^? C \sqcap D$. However, in many cases, matching modulo subsumption is simpler than matching modulo equivalence since it can be reduced to the subsumption problem. If the DL contains $\top$ and all its constructors are monotonic, then $C \sqsubseteq^? D$ has a matcher iff the substitution $\sigma_\top$ that replaces all variables by $\top$ is a matcher, i.e., if $C \sqsubseteq \sigma_\top(D)$. However, in the context of matching modulo subsumption, one is usually not interested in an arbitrary solution, but rather in certain "interesting" ones. One criterion for being interesting is that the matcher should bring D as close to C as possible, i.e., an interesting matcher σ of $C \sqsubseteq^? D$ should be minimal in that there does not exist another substitution δ such that $C \sqsubseteq \delta(D) \sqsubset \sigma(D)$ [37]. Other criteria for defining interesting matchers are discussed in Section 8.2.

In Section 8, we will briefly mention an extension of matching modulo equivalence, namely *unification*, where besides D also C may contain variables. Given a unification problem of the form $C \equiv^? D$, a substitution σ is a *unifier* of this problem iff $\sigma(C) \equiv \sigma(D)$.

3.3 Techniques

The approaches for solving non-standard inferences in DLs developed so far are based on appropriate structural characterizations of the subsumption or the instance problem. Based on these characterizations, the non-standard inferences can be characterized as well, and from these characterizations, approaches solving these inferences can be deduced. In the literature, two different approaches for developing structural characterizations of subsumption have been considered: the language-based and the tree-based approach.

In the *language based approach*, one first computes a normal form that is based on finite or regular sets of words over the alphabet of role names, and then characterizes subsumption by inclusion relationships between these languages (see, e.g., [3, 70]). In the *tree-based approach*, concept descriptions are turned into so-called description trees, and subsumption is then characterized via the existence of certain homomorphisms between these trees (see Section 4).

Since the tree-based approach to characterizing subsumption and solving non-standard inferences will be considered in detail in the next sections, we briefly illustrate the language-based approach for the simple DL $\mathcal{FL}_0$ and the non-standard inferences lcs and matching.

Using the equivalence $\forall r.(C \sqcap D) \equiv \forall r.C \sqcap \forall r.D$ as a rewrite rule from left to right, any $\mathcal{FL}_0$ concept description can be transformed into an equivalent description that is a conjunction of descriptions of the form $\forall r_1.\cdots\forall r_m.A$ for $m \geq 0$ (not necessarily distinct) role names $r_1, \dots, r_m$ and a concept name A. We abbreviate $\forall r_1.\cdots\forall r_m.A$ by $\forall r_1 \dots r_m.A$, where $r_1 \dots r_m$ is viewed as a word over the alphabet of all role names. In addition, instead of $\forall w_1.A \sqcap \dots \sqcap \forall w_\ell.A$ we write $\forall L.A$ where $L := \{w_1, \dots, w_\ell\}$ is a finite set of words over Σ. The term $\forall \emptyset.A$ is considered to be equivalent to the top concept $\top$, which means that it can be added to a conjunction without changing the meaning of the concept. Using these abbreviations, any pair of $\mathcal{FL}_0$ concept descriptions C, D containing the concept names $A_1, \dots, A_k$ can be rewritten as

$$C \equiv \forall U_1.A_1 \sqcap \dots \sqcap \forall U_k.A_k \quad \text{and} \quad D \equiv \forall V_1.A_1 \sqcap \dots \sqcap \forall V_k.A_k,$$

where U_i, V_i are finite sets of words over the alphabet of all role names. This normal form provides us with the following *characterization of subsumption* of $\mathcal{FL}_0$ concept descriptions [26]:

$$C \sqsubseteq D \quad \text{iff} \quad U_i \supseteq V_i \ \text{ for all } i, 1 \leq i \leq k.$$

Since the size of the language-based normal forms is polynomial in the size of the original descriptions, and since the inclusion tests $U_i \supseteq V_i$ can also be realized in polynomial time, this yields a polynomial-time decision procedure for subsumption in $\mathcal{FL}_0$.

As an easy consequence of this characterization we obtain that the lcs E of C, D is of the form

$$E \equiv \forall(U_1 \cap V_1).A_1 \sqcap \dots \sqcap \forall(U_k \cap V_k).A_k,$$

and thus can also be computed in polynomial time.

In order to treat matching in $\mathcal{FL}_0$ using the language-based approach, the language-based normal form of $\mathcal{FL}_0$ concept descriptions is extended in the obvious way to patterns. Let C be an $\mathcal{FL}_0$ concept description and D an $\mathcal{FL}_0$ concept pattern. We can write C, D in the form

$$C \equiv \forall S_{0,1}.A_1 \sqcap \ldots \sqcap \forall S_{0,k}.A_k,$$
$$D \equiv \forall T_{0,1}.A_1 \sqcap \ldots \sqcap \forall T_{0,k}.A_k \sqcap \forall T_1.X_1 \sqcap \ldots \sqcap \forall T_n.X_n,$$

where $A_1, \ldots, A_k$ are the concept names and $X_1, \ldots, X_n$ the concept variables occurring in C, D, and $S_{0,i}, T_{0,i}, T_j$ with $i = 1, \ldots, k, j = 1, \ldots, n$ are finite sets of words over the alphabet of all role names.

In [26] it is shown that the matching problem modulo equivalence $C \equiv^? D$ has a matcher iff for all $i = 1, \ldots, k$, the *linear language equation*

$$S_{0,i} = T_{0,i} \cup T_1 X_{1,i} \cup \cdots \cup T_n X_{n,i}$$

has a solution, i.e., we can substitute the variables $X_{j,i}$ by finite languages such that the equation holds. Solvability of this linear language equation can be decided in polynomial time since it is sufficient to check whether the following substitution θ is a solution:

$$\theta(X_{j,i}) := \bigcap_{u \in T_j} u^{-1} S_{0,i},$$

where $u^{-1}S_{0,i} = \{v \mid uv \in S_{0,i}\}$.

We have used $\mathcal{FL}_0$ to sketch how the language based approach for characterizing subsumption can be used to solve non-standard inferences. In the rest of this paper, we will concentrate on the tree based approach.

4 A Structural Characterization of Subsumption

As explained in the previous section, the basis for solving non-standard inferences is an appropriate structural characterization of subsumption. In this section, we present the characterization for the DL $\mathcal{ALE}$ given in [22] in detail. Characterizations for other DLs are discussed only briefly.

The idea underlying the characterization of subsumption between $\mathcal{ALE}$ concept description is as follows. First, the concept descriptions are presented as edge- and node-labeled trees—called description trees—in which certain implicit facts have been made explicit. Then, we show that subsumption between $\mathcal{ALE}$ concept descriptions corresponds to the existence of homomorphisms between description trees.

As a warming-up, in Section 4.1, we first present the characterization of subsumption for the sublanguage $\mathcal{EL}$ of $\mathcal{ALE}$, with the extension to $\mathcal{ALE}$ presented in Section 4.2. We then briefly discuss characterizations of subsumption for extensions of $\mathcal{ALE}$ and other families of DLs (Section 4.3).

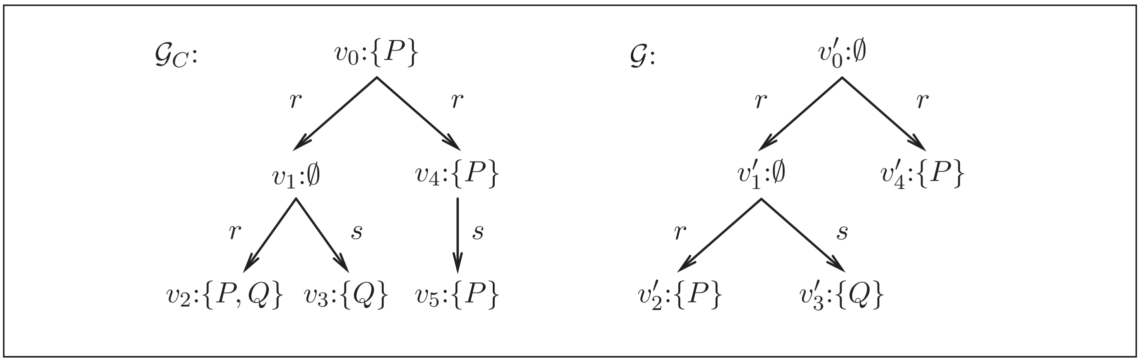

Figure 1: Two $\mathcal{EL}$ description trees.

4.1 Getting Started — The Characterization for $\mathcal{EL}$

We first introduce $\mathcal{EL}$ description trees, and then present the characterization of subsumption.

Definition 6 $\mathcal{EL}$ description trees *are of the form $\mathcal{G} = (V, E, v_0, \ell)$ where $\mathcal{G}$ is a tree with root v_0 whose edges $vrw \in E$ are labeled with role names $r \in N_R$, and whose nodes $v \in V$ are labeled with sets $\ell(v)$ of concept names from N_C. The empty label corresponds to the top-concept.*

Intuitively, such a tree is merely a graphical representation of the syntax of the concept description. More formally, every $\mathcal{EL}$ concept description C can be written (modulo equivalence) as $C \equiv P_1 \sqcap \ldots \sqcap P_n \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$ with $P_i \in N_C \cup \{\top\}$. This description can now be translated into an $\mathcal{EL}$ description tree $\mathcal{G}_C = (V, E, v_0, \ell)$ as follows. The set of all concept names occurring in the top-level conjunction of C yields the label $\ell(v_0)$ of the root v_0, and each existential restriction $\exists r_i.C_i$ in this conjunction yields an r_i-successor that is the root of the tree corresponding to C_i. For example, the $\mathcal{EL}$ concept description

$$C := P \sqcap \exists r.(\exists r.(P \sqcap Q) \sqcap \exists s.Q) \sqcap \exists r.(P \sqcap \exists s.P)$$

yields the tree $\mathcal{G}_C$ depicted on the left-hand side of Figure 1.

Conversely, every $\mathcal{EL}$ description tree $\mathcal{G} = (V, E, v_0, \ell)$ can be translated into an $\mathcal{EL}$ concept description $C_{\mathcal{G}}$. Intuitively, the concept names in the label of v_0 yield the concept names in the top-level conjunction of $C_{\mathcal{G}}$, and each r-successor v of v_0 yields an existential restriction $\exists r.C$ where C is the $\mathcal{EL}$ concept description obtained by translating the subtree of $\mathcal{G}$ rooted at v. For a leaf $v \in V$, the empty label is translated into the top-concept. For example, the $\mathcal{EL}$ description tree $\mathcal{G}$ in Figure 1 yields the $\mathcal{EL}$ concept description

$$C_{\mathcal{G}} = \exists r.(\exists r.P \sqcap \exists s.Q) \sqcap \exists r.P.$$

These translations preserve the semantics of concept descriptions in the sense that $C \equiv C_{\mathcal{G}_C}$ holds for all $\mathcal{EL}$ concept descriptions C.

Definition 7 *A* homomorphism *from an $\mathcal{EL}$ description tree $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ into an $\mathcal{EL}$ description tree $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ is a mapping $\varphi : V_H \longrightarrow V_G$ such that*

1. *$\varphi(w_0) = v_0$,*
2. *$\ell_H(v) \subseteq \ell_G(\varphi(v))$ for all $v \in V_H$, and*
3. *$\varphi(v) r \varphi(w) \in E_G$ for all $vrw \in E_H$.*

Subsumption in $\mathcal{EL}$ can be characterized in terms of homomorphisms between $\mathcal{EL}$ description trees.

Theorem 8 *Let C, D be $\mathcal{EL}$ concept descriptions and $\mathcal{G}_C, \mathcal{G}_D$ be the corresponding $\mathcal{EL}$ description trees. Then $C \sqsubseteq D$ iff there exists a homomorphism from $\mathcal{G}_D$ into $\mathcal{G}_C$.*

In our example (see Figure 1), the $\mathcal{EL}$ concept description $C_{\mathcal{G}}$ subsumes C. Indeed, mapping v_i' to v_i for all $0 \leq i \leq 4$ yields a homomorphism from $\mathcal{G} = \mathcal{G}_{C_{\mathcal{G}}}$ to $\mathcal{G}_C$.

Theorem 8 may look like a special case of the characterization of subsumption between simple conceptual graphs [47], and of the characterization of containment of conjunctive queries [1]. In the more general setting of simple conceptual graphs and conjunctive queries, one considers homomorphisms between *graphs*, and thus testing for the existence of a homomorphism is an NP-complete problem [52]. If one restricts the attention to graphs that are *trees*, then testing for the existence of a homomorphism can be realized in polynomial time using dynamic programming techniques [86]. Thus, Theorem 8 implies that subsumption between $\mathcal{EL}$ concept descriptions is a tractable problem, as already mentioned in Section 2. The fact that both subsumption in $\mathcal{EL}$ and subsumption of conceptual graphs (containment of conjunctive queries) corresponds to the existence of homomorphisms suggests a stronger connection between these problems than is actually the case. In fact, the nodes in conceptual graphs (the variables in conjunctive queries) stand for individuals whereas the nodes of $\mathcal{EL}$ description trees stand for concepts (i.e., sets of individuals). This semantic difference becomes relevant if one considers cyclic $\mathcal{EL}$ TBoxes, which can be represented by description graphs. In this case, however, subsumption no longer corresponds to the existence of a homomorphism, but to the existence of a so-called simulation relation [8]. Whereas the existence of a homomorphism is an NP-complete problem, the existence of a simulation relation can still be checked in polynomial time. It is only for trees that both problems are identical, i.e., on trees the existence of a simulation relation implies the existence of a homomorphism and vice versa, whereas this does not hold on general graphs.

4.2 Extending the Characterization to $\mathcal{ALE}$

To obtain a characterization of subsumption for $\mathcal{ALE}$, we must first extend $\mathcal{EL}$ description trees to $\mathcal{ALE}$ description trees. Since $\mathcal{ALE}$ concept descriptions may contain value restrictions in addition to existential restrictions, $\mathcal{ALE}$ description trees have two types of edges, namely those labeled with a role name $r \in N_R$, which correspond to existential restrictions of the form $\exists r.C$, and those labeled with $\forall r$, which correspond to value restrictions of the form $\forall r.C$. Also, we have to allow negated concept names $\neg P$ and the bottom concept $\bot$ in the labels of nodes, in addition to concept names $P \in N_C$. As in the case of $\mathcal{EL}$, there is a one-to-one correspondence between $\mathcal{ALE}$ concept descriptions and $\mathcal{ALE}$ description trees.

It might be tempting to think that the notion of a homomorphism can also be extended in such a straightforward way to $\mathcal{ALE}$ description trees as well by just adding the following requirement to Definition 7:

4. $\varphi(v)\,\forall r\,\varphi(w) \in E_G$ *for all* $v\,\forall r\,w \in E_H$.

Now, using this notion of a homomorphism between $\mathcal{ALE}$ description trees, one could try to characterize subsumption as before. However, this fails for several reasons.

First, we need to take into account implicit facts that are implied by interactions among value restrictions and among value restrictions and existential restrictions. Consider, for instance, the $\mathcal{ALE}$ concept descriptions and their translations into $\mathcal{ALE}$ description trees depicted in Figure 2. It is easy to see that $C \sqsubseteq D$ and $C' \sqsubseteq D'$, but that there exist neither a homomorphism from $\mathcal{G}_D$ to $\mathcal{G}_C$ nor one from $\mathcal{G}_{D'}$ to $\mathcal{G}_{C'}$. The problem is that C and D are actually equivalent to $\forall r.(P \sqcap Q)$ and that C' is equivalent to $\exists r.(P \sqcap Q) \sqcap \forall r.Q$, but that this is not reflected in the description trees.

To make such implicit facts explicit, we have to normalize the $\mathcal{ALE}$ concept descriptions before translating them into $\mathcal{ALE}$ description trees. For this purpose, the following *normalization rules* are exhaustively applied to the given $\mathcal{ALE}$ concept descriptions:

$$\begin{array}{rcl}
\forall r.E \sqcap \forall r.F & \longrightarrow & \forall r.(E \sqcap F), \\
\forall r.E \sqcap \exists r.F & \longrightarrow & \forall r.E \sqcap \exists r.(E \sqcap F), \\
\forall r.\top & \longrightarrow & \top, \\
E \sqcap \top & \longrightarrow & E.
\end{array}$$

Since each normalization rule preserves equivalence, the resulting $\mathcal{ALE}$ concept descriptions are equivalent to the original ones. The rules should be read modulo associativity and commutativity of conjunction. For instance, $\exists r.E \sqcap \forall r.F$ is also turned into $\exists r.(E \sqcap F) \sqcap \forall r.F$.

The above normalization rules are, however, not yet sufficient to make all implicit facts explicit. This is due to the fact that an $\mathcal{ALE}$ concept description may contain unsatisfiable subdescriptions. In addition to the above normalization rules, we need three more rules to handle this:

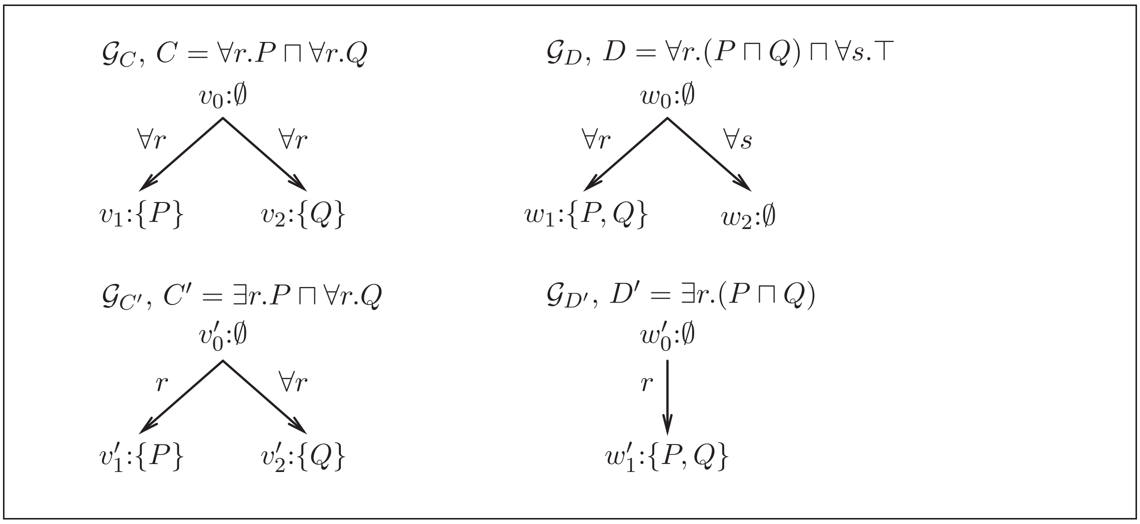

Figure 2: Examples illustrating that implicit facts induced by value and existential restrictions must be taken into account.

$$
\begin{array}{rcll}
P \sqcap \neg P & \longrightarrow & \bot & \text{for each } P \in N_C, \\
\exists r.\bot & \longrightarrow & \bot, & \\
E \sqcap \bot & \longrightarrow & \bot. &
\end{array}
$$

Starting with an $\mathcal{ALE}$ concept description C, the exhaustive application of (both groups of) rules yields an equivalent $\mathcal{ALE}$ concept description in *normal form*. Given such a normal form, the corresponding $\mathcal{ALE}$ description tree is obtain as in the case of $\mathcal{EL}$, with the obvious adaptations due to the existence of two different kinds of edges and the fact that the label of a node may be $\bot$. We refer to the $\mathcal{ALE}$ *description tree corresponding to the normal form* of C as $\mathcal{G}_C$.

Unfortunately, even after normalization, the straightforward adaptation of the notion of a homomorphism from $\mathcal{EL}$ description trees to $\mathcal{ALE}$ description trees sketched above does not yield a sound and complete characterization of subsumption in $\mathcal{ALE}$. As an example, consider the following $\mathcal{ALE}$ concept descriptions:

$$
\begin{array}{rcl}
C & := & (\forall r.\exists r.(P \sqcap \neg P)) \sqcap (\exists s.(P \sqcap \exists r.Q)), \\
D & := & (\forall r.(\exists r.P \sqcap \exists r.\neg P)) \sqcap (\exists s.\exists r.Q).
\end{array}
$$

The description D is already in normal form, and the normal form of C is

$$C' := \forall r.\bot \sqcap \exists s.(P \sqcap \exists r.Q).$$

The corresponding $\mathcal{ALE}$ description trees $\mathcal{G}_C$ and $\mathcal{G}_D$ are depicted in Figure 3. It is easy to see that there does not exist a homomorphism in the above sense from $\mathcal{G}_D$ into $\mathcal{G}_C$, although

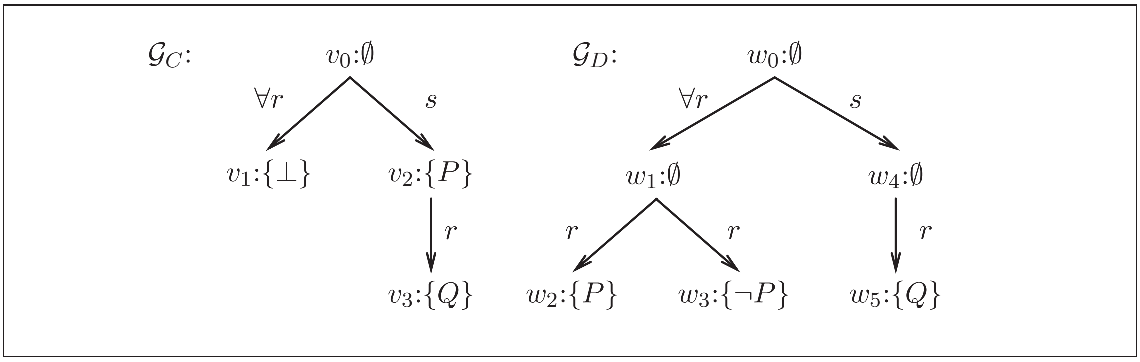

Figure 3: Example illustrating that the notion of a homomorphism must be adapted.

we have $C \sqsubseteq D$. In particular, the $\mathcal{ALE}$ concept description $\exists r.P \sqcap \exists r.\neg P$ corresponding to the subtree with root w_1 of $\mathcal{G}_D$ subsumes $\bot$, which is the concept description corresponding to the subtree with root v_1 in $\mathcal{G}_C$. Therefore, a homomorphism from $\mathcal{G}_D$ into $\mathcal{G}_C$ should be allowed to map the whole tree corresponding to $\exists r.P \sqcap \exists r.\neg P$, i.e., the nodes w_1, w_2, w_3, onto the tree corresponding to $\bot$, i.e., onto v_1.

The example suggests the following new notion of a homomorphism on $\mathcal{ALE}$ description trees.

Definition 9 *A* homomorphism *from an $\mathcal{ALE}$ description tree $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ into an $\mathcal{ALE}$ description tree $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ is a mapping $\varphi : V_H \longrightarrow V_G$ such that*

1. $\varphi(w_0) = v_0$,
2. $\ell_H(v) \subseteq \ell_G(\varphi(v))$ *or* $\ell_G(\varphi(v)) = \{\bot\}$ *for all* $v \in V_H$,
3. *for all* $vrw \in E_H$, *either* $\varphi(v)r\varphi(w) \in E_G$, *or* $\varphi(v) = \varphi(w)$ *and* $\ell_G(\varphi(v)) = \{\bot\}$, *and*
4. *for all* $v\forall rw \in E_H$, *either* $\varphi(v)\forall r\varphi(w) \in E_G$, *or* $\varphi(v) = \varphi(w)$ *and* $\ell_G(\varphi(v)) = \{\bot\}$.

In Figure 3, if we map w_0 onto v_0; w_1, w_2, and w_3 onto v_1; w_4 onto v_2; and w_5 onto v_3, then the above conditions are satisfied, i.e., this mapping yields a homomorphism from $\mathcal{G}_D$ into $\mathcal{G}_C$. With this new notion of a homomorphism between $\mathcal{ALE}$ description trees, we can characterize subsumption in $\mathcal{ALE}$ in a sound and complete way.

Theorem 10 *Let C, D be two $\mathcal{ALE}$ concept descriptions and $\mathcal{G}_C$, $\mathcal{G}_D$ the corresponding $\mathcal{ALE}$ description trees. Then $C \sqsubseteq D$ iff there exists a homomorphism from $\mathcal{G}_D$ into $\mathcal{G}_C$.*

It should be noted that there is a close relationship between the normalization rules introduced above and some of the rules employed by tableaux-based subsumption algorithms, as e.g. introduced in [50]. As shown in [50], the propagation of value restrictions on existential restrictions may lead to an exponential blow-up (see the concept descriptions C_n introduced below Theorem 15). Consequently, the size of the normal forms, and thus also of the description trees, may grow exponentially in the size of the original $\mathcal{ALE}$ concept descriptions. It is easy to see that this exponential blow-up cannot be avoided: On the one hand, as for $\mathcal{EL}$, the existence of a homomorphism between $\mathcal{ALE}$ description trees can still be tested in polynomial time. On the other hand, subsumption in $\mathcal{ALE}$ is an NP-complete problem [50].

4.3 Characterization of Subsumption for Other DLs

The characterization of subsumption for $\mathcal{ALE}$ has been extended to $\mathcal{ALEN}$ in [74]. There, description trees are not used explicitly. Subsumption is rather characterized directly for the normalized concept descriptions, by using induction on the role depth of the descriptions.

For the sublanguage $\mathcal{ALNS}$ of the DL employed by the CLASSIC system [36], which extends $\mathcal{ALN}$ by the so-called same-as operator, subsumption has been characterized in [72]. Due to the presence of the same-as operator in $\mathcal{ALNS}$, description graphs instead of description trees are used in this characterization.

For DLs with cyclic TBoxes, subsumption has been characterized for $\mathcal{FL}_0$ [4], $\mathcal{ALN}$ [70], and $\mathcal{EL}$ [8] w.r.t. the three types of semantics employed for cyclic TBoxes: descriptive semantics, and greatest and least fixed point semantics. For $\mathcal{FL}_0$ and $\mathcal{ALN}$, subsumption has been characterized using the language-based approach (see Section 3.3). For $\mathcal{EL}$, the characterization extends the approach for $\mathcal{EL}$ concept descriptions presented in Section 4.1. However, instead of homomorphisms between description trees, simulation relationships on description graphs are employed.

5 The Least Common Subsumer

In this section, we study the existence of the lcs and how it can be computed (if it exists). Our exposition again concentrates on $\mathcal{ALE}$. It is based on the results shown in [22]. In addition, we briefly mention results for other DLs.

As we will see, once the structural characterization of subsumption is in place, it is rather easy to derive algorithms for computing the lcs. As a warming up exercise, in the following subsection, we present an lcs algorithm for $\mathcal{EL}$. Its extension to $\mathcal{ALE}$ is described in Section 5.2. An overview of results for other DLs is provided in Section 5.3.

5.1 The LCS for $\mathcal{EL}$

The characterization of subsumption by homomorphisms allows us to characterize the lcs by the product of $\mathcal{EL}$ description trees.

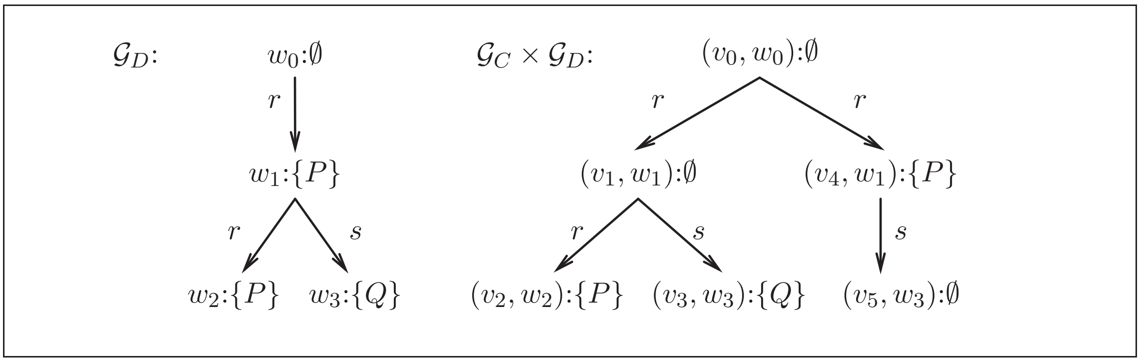

Figure 4: The product of $\mathcal{EL}$ description trees.

Definition 11 *The* product $\mathcal{G} \times \mathcal{H}$ *of two $\mathcal{EL}$ description trees $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ is defined inductively on the depth of the trees. Let $\mathcal{G}(v)$ denote the subtree of $\mathcal{G}$ rooted at v. We define (v_0, w_0) to be the root of $\mathcal{G} \times \mathcal{H}$, labeled with $\ell_G(v_0) \cap \ell_H(w_0)$. For each r-successor v of v_0 in $\mathcal{G}$ and w of w_0 in $\mathcal{H}$, we obtain an r-successor (v, w) of (v_0, w_0) in $\mathcal{G} \times \mathcal{H}$ that is the root of the product of $\mathcal{G}(v)$ and $\mathcal{H}(w)$.*

For example, consider the $\mathcal{EL}$ description tree $\mathcal{G}_C$ (Figure 1) and the $\mathcal{EL}$ description tree $\mathcal{G}_D$ (Figure 4), where $\mathcal{G}_D$ corresponds to the $\mathcal{EL}$ concept description $D := \exists r.(P \sqcap \exists r.P \sqcap \exists s.Q)$. The product $\mathcal{G}_C \times \mathcal{G}_D$ is depicted on the right-hand side of Figure 4.

Theorem 12 *Let C, D be two $\mathcal{EL}$ concept descriptions and $\mathcal{G}_C$, $\mathcal{G}_D$ the corresponding $\mathcal{EL}$ description trees. Then $C_{\mathcal{G}_C \times \mathcal{G}_D}$ is the lcs of C and D. In particular, the lcs of $\mathcal{EL}$ concept descriptions always exists.*

In our example, we thus obtain that the lcs of $C = P \sqcap \exists r.(\exists r.(P \sqcap Q) \sqcap \exists s.Q) \sqcap \exists r.(P \sqcap \exists s.P)$ and $D = \exists r.(P \sqcap \exists r.P \sqcap \exists s.Q)$ is

$$lcs_{\mathcal{EL}}(C, D) = \exists r.(\exists r.P \sqcap \exists s.Q) \sqcap \exists r.(P \sqcap \exists s.\top).$$

The size of the lcs of two $\mathcal{EL}$ concept descriptions C, D can be bounded by the size of $\mathcal{G}_C \times \mathcal{G}_D$, which is polynomial in the size of $\mathcal{G}_C$ and $\mathcal{G}_D$. Since the size of the description tree corresponding to a given description is linear in the size of the description, we obtain:

Proposition 13 *The size of the lcs of* two *$\mathcal{EL}$ concept descriptions C, D is polynomial in the size of C and D, and the lcs can be computed in polynomial time.*

In many applications, however, one is interested in the lcs of $n > 2$ concept descriptions $C_1, \ldots, C_n$. This lcs can be obtained from the n-ary product $\mathcal{G}_{C_1} \times \cdots \times \mathcal{G}_{C_n}$ of their corresponding $\mathcal{EL}$ description trees. Therefore, the size of the lcs can be bounded by the size of this product. It is not hard to show that in general this size cannot be polynomially bounded [22, 31].

Proposition 14 *The size of the lcs of n $\mathcal{EL}$ concept descriptions $C_1, \ldots, C_n$ of size linear in n may grow exponentially in n.*

5.2 The LCS for $\mathcal{ALE}$

Just as for $\mathcal{EL}$, the lcs of $\mathcal{ALE}$ concept descriptions can be obtain as the product of the corresponding $\mathcal{ALE}$ description trees. However, the definition of the product must be adapted to the modified notion of a homomorphism. In particular, this definition must treat leaves with label $\{\bot\}$ in a special manner. Such a leaf corresponds to the bottom-concept, and since $\bot \sqsubseteq C$ for all $\mathcal{ALE}$ concept descriptions C, we have $lcs(\bot, C) \equiv C$. Thus, our product operation should be defined such that $C_{\mathcal{G}_\bot \times \mathcal{G}_C} \equiv C$.

More precisely, the *product $\mathcal{G} \times \mathcal{H}$ of two $\mathcal{ALE}$ description trees* $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ is defined as follows. If $\ell_G(v_0) = \{\bot\}$ ($\ell_H(w_0) = \{\bot\}$), then we define $\mathcal{G} \times \mathcal{H}$ by replacing each node w in $\mathcal{H}$ (v in $\mathcal{G}$) by (v_0, w) ((v, w_0)). Otherwise, we define $\mathcal{G} \times \mathcal{H}$ by induction on the depth of the trees analogously to the definition of the product of $\leq$ description trees.

For the $\mathcal{ALE}$ description trees depicted in Figure 3, $\mathcal{G}_C \times \mathcal{G}_D$ is obtained from $\mathcal{G}_D$ by replacing w_0 by (v_0, w_0), w_i by (v_1, w_i) for $i = 1, 2, 3$, w_4 by (v_2, w_4), and w_5 by (v_3, w_5).[2]

Theorem 15 *Let C, D be two $\mathcal{ALE}$ concept descriptions and $\mathcal{G}_C$, $\mathcal{G}_D$ their corresponding $\mathcal{ALE}$ description trees. Then $C_{\mathcal{G}_C \times \mathcal{G}_D}$ is the lcs of C and D. In particular, the lcs of $\mathcal{ALE}$ concept descriptions always exists.*

Unlike $\mathcal{EL}$, the size of the lcs of two $\mathcal{ALE}$ concept descriptions may already grow exponentially. To see this, consider the following example. Let C_n, $n \geq 1$, be defined inductively as

$$C_1 := \exists r.P \sqcap \exists r.Q \text{ and } C_n := \exists r.P \sqcap \exists r.Q \sqcap \forall r.C_{n-1}$$

and let D_n, $n \geq 1$, be defined as

$$D_1 := \exists r.(P \sqcap Q) \text{ and } D_n := \exists r.(P \sqcap Q \sqcap D_{n-1}).$$

Note that the size of the normal form of C_n grows exponentially in n. It is easy to verify that the lcs of C_n and D_n is equivalent to the concept description E_n where

$$E_1 := \exists r.P \sqcap \exists r.Q \text{ and } E_n := \exists r.(P \sqcap E_{n-1}) \sqcap \exists r.(Q \sqcap E_{n-1}).$$

The size of E_n grows exponentially in n. It is not hard to check that there does not exist a smaller concept description equivalent to the lcs of C_n and D_n. Hence, we obtain:

Proposition 16 *The size of the lcs of* two *$\mathcal{ALE}$ concept descriptions C, D may be exponential in the size of C, D.*

[2] Note that this is a somewhat atypical example since in this case C is subsumed by D, and thus the lcs is equivalent to D.

The above example suggests that, by employing structure sharing, the size of the lcs can be reduced. However, in general this is not the case. More specifically, it was shown in [31] that even if equivalent sub-concept descriptions of the lcs can be represented as defined concepts in an acyclic TBox, the representation of the lcs may still grow exponentially.

5.3 The LCS for other DLs

Based on the structural characterization of subsumption for the DLs mentioned in Section 4.3, algorithms for computing the lcs have been employed in a similar manner as illustrated above [74, 72, 17, 7, 5]. Interestingly, for the DL $\mathcal{ALNS}$ it has turned out that the existence of the lcs depends on whether features, i.e., roles that are restricted to be functional, are interpreted as partial or total functions. While in the former case, the lcs always exists, this is not true in the latter case [72]. As mentioned above, for the DL $\mathcal{EL}$ with cyclic TBoxes interpreted with descriptive semantics the lcs also does not need to exist [5].

6 The Most Specific Concept

As illustrated in Section 3.2, most specific concepts need not exist for DLs with number restrictions or existential restrictions. There are two ways to overcome this problem. First, the languages can be extended to allow for cyclic TBoxes interpreted with the greatest fixed point semantics. Second, one can resort to approximating the most specific concept. In the following subsection, we consider the latter approach in more detail, mainly concentrating on the simple DL $\mathcal{EL}$. Besides introducing methods for computing approximations, we will also characterize the existence of the msc. In Section 6.2, we will summarize results obtained following the first approach.

6.1 Existence and Approximation of the MSC

The example presented in Section 3.2 illustrates that describing an msc may require a concept description with infinite role depth. Such a concept description can be approximated by restricting the role depth to a fixed constant k. This leads to the notion of a k-approximation. In Section 3.3 we have pointed out that the basis for solving non-standard inferences is an appropriate characterization of the underlying standard inference. For the lcs, this standard inference is the subsumption problem. In order characterize the msc and to design algorithms for computing (approximations of) it, an appropriate characterization of the instance problem is needed.

After defining k-approximations in Section 6.1.1, we first present a characterization of the instance problem in Section 6.1.2 and then, in Section 6.1.3, apply this characterization to compute k-approximations. All this is done for the simple case that the DL is $\mathcal{EL}$.

Extensions to more expressive DLs are discussed in Section 6.1.4. The results presented in this section are mainly based on [73].

6.1.1 Defining k-Approximations

To give a formal definition of k-approximations of the msc, we first need to define the role depth of concept descriptions. The role depth $depth(C)$ of an $\mathcal{EL}$ concept description C is defined as the maximum number of nested quantifiers in C:

$$\begin{array}{rcl} depth(\top) & = & depth(P) \ = \ 0, \\ depth(C \sqcap D) & = & \max(depth(C), depth(D)), \\ depth(\exists r.C) & = & depth(C) + 1. \end{array}$$

Definition 17 *Let $\mathcal{A}$ be an $\mathcal{EL}$-ABox, b an individual in $\mathcal{A}$, C an $\mathcal{EL}$ concept descriptions, and $k \in \mathbb{N}$. Then, C is a* k-approximation *of b w.r.t. $\mathcal{A}$ ($msc^k_{\mathcal{EL}}(b)$) iff*

1. *$\mathcal{A} \models C(b)$,*
2. *$depth(C) \leq k$, and*
3. *$C \sqsubseteq C'$ for all $\mathcal{EL}$ concept descriptions C' with $\mathcal{A} \models C'(b)$ and $depth(C') \leq k$.*

It is an easy consequence of this definition that k-approximations are unique up to equivalence (if they exist). Thus, we can talk about *the* k-approximation of a given individual.

The k-approximation of the individual Narcis in the example presented in Section 3.2 is the $\mathcal{EL}$ concept description

$$\underbrace{\exists \mathsf{loves}. \ldots . \exists \mathsf{loves}}_{k \text{ times}}.\mathsf{Vain}.$$

6.1.2 Characterizing the Instance Problem in $\mathcal{EL}$

In order to characterize instance relationships, we need to introduce description graphs (representing ABoxes) as a generalization of description trees (representing concept descriptions). An $\mathcal{EL}$ *description graph* is a labeled graph of the form $\mathcal{G} = (V, E, \ell)$ whose edges $vrw \in E$ are labeled with role names $r \in N_R$ and whose nodes $v \in V$ are labeled with sets $\ell(v)$ of concept names from N_C. The empty label corresponds to the top-concept.

Similarly to the translation of $\mathcal{EL}$ concept descriptions into $\mathcal{EL}$ description trees, an $\mathcal{EL}$-ABox $\mathcal{A}$ is translated into an $\mathcal{EL}$ description graph $\mathcal{G}(\mathcal{A})$ in the following way. Let $Ind(\mathcal{A})$ denote the set of all individuals occurring in $\mathcal{A}$. For each $a \in Ind(\mathcal{A})$, let

$$C_a = \begin{cases} \displaystyle\bigsqcap_{D(a) \in \mathcal{A}} D & \text{if there exists a concept assertion of the form } D(a) \in \mathcal{A}, \\ \top & \text{otherwise.} \end{cases}$$

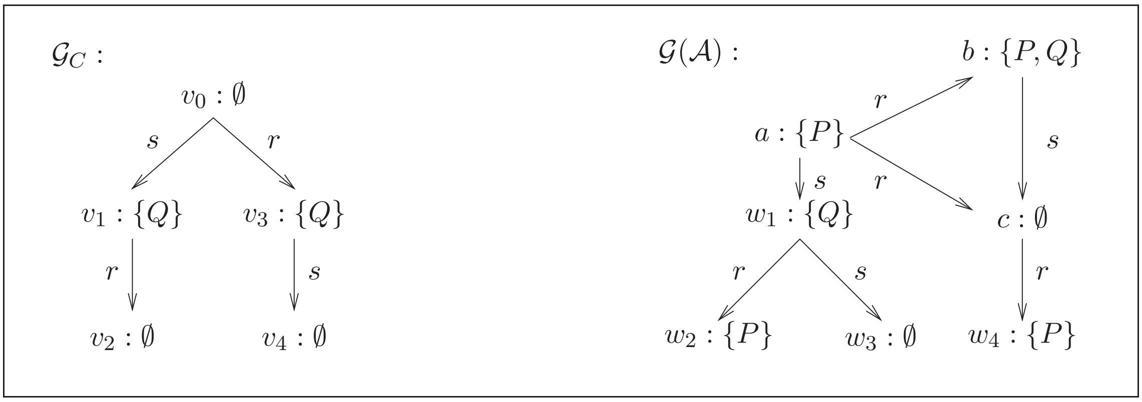

Figure 5: The $\mathcal{EL}$ description tree of C and the $\mathcal{EL}$ description graph of $\mathcal{A}$.

Let $\mathcal{G}_{C_a} = (V_a, E_a, a, \ell_a)$ denote the $\mathcal{EL}$ description tree obtained from C_a.[3] Without loss of generality we assume that the sets V_a for $a \in \mathit{Ind}(\mathcal{A})$ are pairwise disjoint. Then, $\mathcal{G}(\mathcal{A}) = (V, E, \ell)$ is defined as

- $V = \bigcup_{a \in \mathit{Ind}(\mathcal{A})} V_a$,
- $E = \{arb \mid r(a,b) \in \mathcal{A}\} \cup \bigcup_{a \in \mathit{Ind}(\mathcal{A})} E_a$, and
- $\ell(v) = \ell_a(v)$ for all $v \in V_a$.

As an example, consider the $\mathcal{EL}$ ABox

$$\begin{array}{rcl}\mathcal{A} & = & \{(P \sqcap \exists s.(Q \sqcap \exists r.P \sqcap \exists s.\top))(a), \quad (P \sqcap Q)(b), \quad (\exists r.P)(c), \\ & & r(a,b), \quad r(a,c), \quad s(b,c)\}.\end{array}$$

The corresponding $\mathcal{EL}$ description graph $\mathcal{G}(\mathcal{A})$ is depicted on the right-hand side of Figure 5. Later on we will also consider the $\mathcal{EL}$ description tree of

$$C = \exists s.(Q \sqcap \exists r.\top) \sqcap \exists r.(Q \sqcap \exists s.\top),$$

which is depicted on the left-hand side of this figure.

Now, an instance relationship $\mathcal{A} \models C(a)$ in $\mathcal{EL}$ can be characterized via the existence of a homomorphism from the description tree of C into the description graph of $\mathcal{A}$, where such *homomorphisms* are defined analogously to the case of homomorphisms between $\mathcal{EL}$ description trees. Given an individual a, we must require that homomorphism maps the root of the description tree to the node a in $\mathcal{G}(\mathcal{A})$.

Theorem 18 *Let $\mathcal{A}$ be an $\mathcal{EL}$-ABox, $a \in \mathit{Ind}(\mathcal{A})$ be an individual in $\mathcal{A}$, and C be an $\mathcal{EL}$ concept description. Then, $\mathcal{A} \models C(a)$ iff there exists a homomorphism φ from $\mathcal{G}_C$ into $\mathcal{G}(\mathcal{A})$ such that $\varphi(v_0) = a$, where v_0 is the root of $\mathcal{G}_C$.*

[3]Note that the individual a is defined to be the root of $\mathcal{G}(C_a)$; in particular, this means that $a \in V_a$.

In our example (Figure 5), a is an instance of C, since mapping v_0 on a, v_i on w_i, $i = 1, 2$, and v_3 on b and v_4 on c yields a homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}(\mathcal{A})$.

As mentioned in Section 4, existence of a homomorphism between graphs is an NP-complete problem. In the restricted case of testing for the existence of homomorphisms mapping trees into graphs, the problem is polynomial [52]. Thus, as a corollary of Theorem 18, we obtain the following complexity result.

Corollary 19 *The instance problem for $\mathcal{EL}$ can be decided in polynomial time.*

6.1.3 Computing k-Approximations

Our algorithm for computing $msc^k_{\mathcal{EL}}(a)$ is based on the following idea. Let $\mathcal{T}(a, \mathcal{G}(\mathcal{A}))$ denote the tree with root a obtained from the graph $\mathcal{G}(\mathcal{A})$ by unraveling. This tree has a finite branching factor, but possibly infinitely long paths. Pruning all paths to length k yields an $\mathcal{EL}$ description tree $\mathcal{T}_k(a, \mathcal{G}(\mathcal{A}))$ of depth $\leq k$. Using Theorem 18 and Theorem 8, it is easy to show that the $\mathcal{EL}$ concept description $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ is equivalent to $msc^k_{\mathcal{EL}}(a)$. In addition, we obtain a characterization of the existence of the msc. The following theorem summarizes the results.

Theorem 20 *Let $\mathcal{A}$ be an $\mathcal{EL}$-ABox, $a \in Ind(\mathcal{A})$, and $k \in \mathbb{N}$. Then, $C_{\mathcal{T}_k(a,\mathcal{G}(\mathcal{A}))}$ is the k-approximation of a w.r.t. $\mathcal{A}$. If, starting from a, no cyclic can be reached in $\mathcal{A}$ (i.e., $\mathcal{T}(a, \mathcal{G}(\mathcal{A}))$ is finite), then $C_{\mathcal{T}(a,\mathcal{G}(\mathcal{A}))}$ is the msc of a w.r.t. $\mathcal{A}$; otherwise no msc exists.*

As a corollary of this theorem we obtain:

Corollary 21 *For an $\mathcal{EL}$-ABox $\mathcal{A}$, an individual $a \in Ind(\mathcal{A})$, and $k \in \mathbb{N}$, the k-approximation of a w.r.t. $\mathcal{A}$ always exists and it can be computed in time polynomial in the size of $\mathcal{A}$ if k is assumed to be constant, and in exponential time otherwise. The existence of the msc can be decided in polynomial time. If the msc exists, then it can be computed in time exponential in the size of $\mathcal{A}$.*

Taking the ABox $\mathcal{A} = \{r(a, a),\ s(a, a)\}$ as an example, it is easy to see that the size of the k-approximation of $\mathcal{A}$ may grow exponentially in k if no structure sharing is employed. However, this exponential blow-up can be avoided when the k-approximations are defined by acyclic TBoxes. The same is true for the msc in case it exists: Consider, for instance, the ABox which consists of a sequence $a_1, \ldots, a_n$ of n individuals where there is an r and an s edge from a_i to a_{i+1} for every i.

6.1.4 Extensions to more Expressive DLs

So far, not much is known about computing k-approximations of the msc for DLs more expressible than $\mathcal{EL}$. In [73], the approach presented above is extended to the DL $\mathcal{EL}_\neg$, which extends $\mathcal{EL}$ by the bottom concept $\bot$ and primitive negation $\neg P$. In the following,

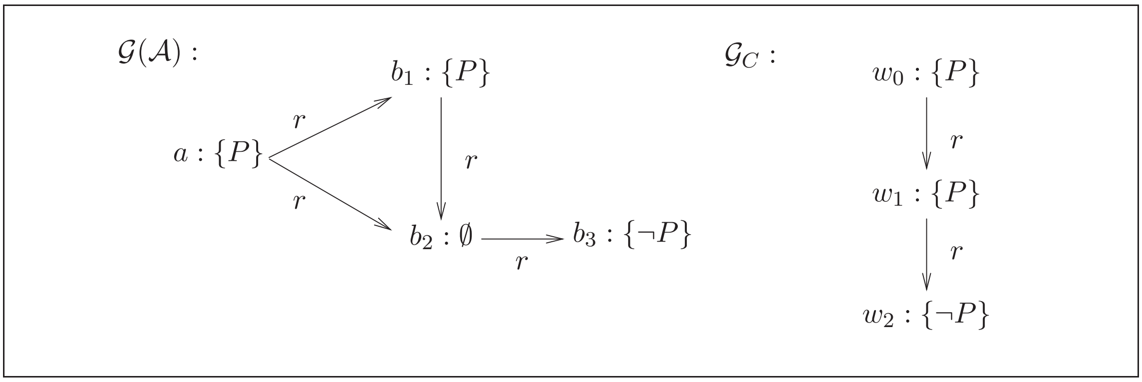

Figure 6: The $\mathcal{EL}_\neg$ description graph and the $\mathcal{EL}_\neg$ description tree of our example.

we briefly present the ideas behind computing k-approximations of the msc in $\mathcal{EL}_\neg$, and discuss the problems that arise when considering more expressive DLs.

The following example illustrates that a naïve extension of the approach for $\mathcal{EL}$ does not work for $\mathcal{EL}_\neg$. Consider, for instance, the following $\mathcal{EL}_\neg$ concept description C and $\mathcal{EL}_\neg$ ABox $\mathcal{A}$:

$$\begin{array}{rcl} C & = & P \sqcap \exists r.(P \sqcap \exists r.\neg P) \\ \mathcal{A} & = & \{P(a),\ P(b_1),\ \neg P(b_3),\ r(a,b_1),\ r(a,b_2),\ r(b_1,b_2),\ r(b_2,b_3)\}. \end{array}$$

The corresponding description tree and graph are depicted in Figure 6. Obviously, there does not exist a homomorphism φ from $\mathcal{G}_C$ into $\mathcal{G}(\mathcal{A})$ with $\varphi(w_0) = a$, because neither $P \in \ell(b_2)$ nor $\neg P \in \ell(b_2)$. For each model $\mathcal{I}$ of $\mathcal{A}$, however, either $b_2^{\mathcal{I}} \in P^{\mathcal{I}}$ or $b_2^{\mathcal{I}} \in (\neg P)^{\mathcal{I}}$, and thus $a^{\mathcal{I}} \in C^{\mathcal{I}}$. Hence, a is an instance of C w.r.t. $\mathcal{A}$ even though there does not exist a homomorphism φ from $\mathcal{G}_C$ into $\mathcal{G}(\mathcal{A})$ with $\varphi(w_0) = a$.

The reason for the problem illustrated by the example is that for the individuals in the ABox it is not always fixed whether they are instances of a given atomic concept or of its negation. In order to obtain a sound and complete characterization analogous to Theorem 18, we therefore consider all so-called atomic completions of $\mathcal{G}(\mathcal{A})$. An atomic completion of $\mathcal{G}(\mathcal{A})$ is obtained from $\mathcal{G}(\mathcal{A})$ by adding, for all concept names P and all nodes whose label contains neither P nor $\neg P$, either P or $\neg P$ to the label of this node.

In [73], it is shown that an individual a of the consistent $\mathcal{EL}_\neg$-ABox $\mathcal{A}$ is an instance of the $\mathcal{EL}_\neg$ concept description C iff *for every atomic completion* $\mathcal{G}'$ *of* $\mathcal{G}(\mathcal{A})$ there exists a homomorphism from $\mathcal{G}_C$ into $\mathcal{G}'$ that maps the root of $\mathcal{G}_C$ onto a.

Using this characterization of the instance problem, it is possible to show that the instance problem for $\mathcal{EL}_\neg$ is coNP-complete. Also, k-approximations can be obtained by unraveling the completions up to depth k and then taking the lcs of these completions.

Unfortunately, for $\mathcal{ALE}$ (or even more expressive DLs), analogous characterizations of the instance problem are not known. However, given finite sets of concept and role names, the set of all $\mathcal{ALE}$ concept descriptions of depth $\leq k$ is finite (up to equivalence) and can be computed effectively. The fact that $\mathcal{ALE}$ allows for conjunction implies that a k-approximation always exists: it can be obtained as the conjunction of all concepts (up to equivalence) of depth $\leq k$ that have the individual a as an instance. Obviously, this generic argument also carries over to more expressive DLs, including $\mathcal{ALCN}$ and beyond. However, such an enumeration algorithm is clearly to complex, and thus of no practical use.

6.2 The Most Specific Concept in the Presence of Cyclic TBoxes

It has first been shown for cyclic $\mathcal{ALN}$ TBoxes [17] and more recently for cyclic $\mathcal{EL}$ TBoxes [7] that the msc always exists if the TBoxes are interpreted with the greatest fixed point semantics. In addition, this msc can effectively be computed. In contrast, the msc need not exist if the TBoxes are interpreted with the least fixed-point semantics or descriptive semantics. The problem of computing the msc w.r.t. $\mathcal{EL}$ TBoxes interpreted with descriptive semantics is investigated in [6, 9].

7 Rewriting

In this section, we review results obtained for computing minimal rewritings, as defined in Section 3.2. As before, in our exposition we concentrate on $\mathcal{ALE}$ and sublanguages thereof, and comment on results for other DLs only briefly. As introduced in Section 3.2, the minimal rewriting problem is one instance of a more general rewriting framework. Another instance is approximation, which is also briefly discussed here.

In the following subsection, we consider the minimal rewriting decision problem. This will provide us with complexity lower bounds for the problem of computing minimal rewritings. The minimal rewriting computation problem itself is covered in Section 7.2. Approximation is discussed in Section 7.3. The results for minimal rewriting presented in this section are based mainly on the results of [23].

7.1 The Minimal Rewriting Decision Problem

Formulated for $\mathcal{ALE}$, the *minimal rewriting decision problem* is concerned with the following question: given an $\mathcal{ALE}$ concept description C, an $\mathcal{ALE}$ TBox $\mathcal{T}$, and a non-negative integer κ, does there exist an $\mathcal{ALE}$-rewriting E of C using $\mathcal{T}$ such that $|E| \leq \kappa$.

Clearly, this problem is decidable in nondeterministic polynomial time using an oracle for deciding equivalence modulo TBoxes by the following algorithm. First, guess an $\mathcal{ALE}$ concept description E of size $\leq \kappa$. Then check whether E is equivalent to C modulo $\mathcal{T}$.

This simple algorithm yields the following complexity upper bounds for the minimal rewriting decision problem in $\mathcal{ALE}$. If $\mathcal{T}$ is unfolded, i.e., the right-hand sides of the concept definitions do not contain defined concepts, we know that equivalence in $\mathcal{ALE}$ is in NP (see Section 2). Otherwise, if we do not assume $\mathcal{T}$ to be unfolded, equivalence is in PSPACE[4] since this is even the case for the larger DL $\mathcal{ALC}$ (see Section 2). Hence, for unfolded TBoxes the minimal rewriting decision problem for $\mathcal{ALE}$ is in NP, and otherwise it is in PSPACE.

Conversely, it is easy to see that the minimal rewriting decision problem is at least as hard as deciding subsumption. Let C and D be $\mathcal{ALE}$ concept descriptions, and A, P_1, P_2 be three different concept names not occurring in C, D. It is easy to see that $C \sqsubseteq D$ iff there exists a minimal rewriting of size ≤ 1 of the $\mathcal{ALE}$ concept description $P_1 \sqcap P_1 \sqcap C$ using the TBox $\mathcal{T} = \{A \doteq P_1 \sqcap P_2 \sqcap C \sqcap D\}$. Since subsumption in $\mathcal{ALE}$ w.r.t. an (unfolded or non-unfolded) $\mathcal{ALE}$ TBox is NP-hard, it follows that the minimal rewriting decision problem is NP-hard for $\mathcal{ALE}$.

Theorem 22 *In $\mathcal{ALE}$, the minimal rewriting decision problem is NP-complete for unfolded $\mathcal{ALE}$ TBoxes. With respect to arbitrary acyclic TBoxes, this problem is NP-hard and in PSPACE.*

Clearly, the above arguments also apply to other DLs. For example, we can use these arguments and the known complexity results for subsumption and equivalence in $\mathcal{ALC}$ to show that the minimal rewriting decision problem is PSPACE-complete for $\mathcal{ALC}$ (independently of whether the $\mathcal{ALC}$ TBox is unfolded or not). It should be noted, however, that the complexity of the subsumption problem is not the only source of complexity for the minimal rewriting decision problem. As an optimization problem, the minimal rewriting decision problem may also be intractable even if the subsumption problem is tractable. For example, subsumption w.r.t. unfolded TBoxes in $\mathcal{FL}_0$ and $\mathcal{ALN}$ is in P, but the NP-hardness of the minimal rewriting decision problem can nevertheless be shown by a reduction from SETCOVER (see [23] for details).

7.2 The Minimal Rewriting Computation Problem

Whereas the previous subsection was concerned with deciding whether there exists a rewriting within a given size bound, this subsection considers the problem of actually computing minimal rewritings. This is called the *minimal rewriting computation problem*. Since the minimal rewriting decision problem can obviously be reduced in polynomial time to the minimal rewriting computation problem, the lower bounds shown above immediately carry over to the computation problem.

To be more precise, there are actually two different variants of the computation problem. For a given instance $(C, \mathcal{T})$ of the minimal rewriting computation problem, one can be interested in computing either (1) *one* minimal rewriting of C using $\mathcal{T}$, or (2) *all* minimal rewritings of C using $\mathcal{T}$.

[4]This is only an upper bound. The exact complexity of the equivalence problem in $\mathcal{ALE}$ with acyclic TBoxes is not known.

The hardness results of the previous subsection imply that even computing one minimal rewriting is in general a hard problem. In addition, it is easy to see that the number of minimal rewritings of a concept description C w.r.t. a TBox $\mathcal{T}$ can be exponential in the size of C and $\mathcal{T}$. Consider, for instance, the concept description $C_n = P_1 \sqcap \ldots \sqcap P_n$ and the TBox $\mathcal{T}_n = \{A_i \doteq P_i \mid 1 \leq i \leq n\}$. The minimal rewritings are of the form $E = P_{i_1} \sqcap \cdots P_{i_k} \sqcap A_{j_1} \sqcap \cdots A_{j_l}$ where $l + k = n$ and $\{1, \ldots, n\} = \{i_1, \ldots, i_k, j_1, \ldots, j_l\}$. Obviously, there are exponentially many such rewritings.

It is very easy to come up with an algorithm for computing one or all minimal rewritings of a concept description C w.r.t. the TBox $\mathcal{T}$. Since the size of the minimal rewritings is bounded by the size of C, one can simply enumerate all concept descriptions of size less than or equal to the size of C, and check which of them are equivalent to C w.r.t. $\mathcal{T}$. Those of minimal size are the minimal rewritings. Clearly, this algorithm works for all DLs where equivalence w.r.t. a TBox is decidable. However, such a brute-force enumeration algorithm is clearly too inefficient to be of any practical interest.

In what follows, we present a more source-driven algorithm for $\mathcal{ALE}$ which uses the form of C (rather than only the size of C) to prune the search space.[5] The algorithm assumes the concept description C to be in $\forall$-*normal form.* This normal form is obtained from C (in polynomial time) by exhaustively applying the rule $\forall r.E \sqcap \forall r.F \longrightarrow \forall r.(E \sqcap F)$ to C. As a result, every conjunction in C contains at most one value restriction $\forall r.D$ for a given role $r \in N_R$.

Given an $\mathcal{ALE}$ concept description C in $\forall$-normal form and an $\mathcal{ALE}$ TBox $\mathcal{T}$, the algorithm for computing minimal rewritings works as follows:

1. Compute an extension C^* of C w.r.t. $\mathcal{T}$, which adds some defined concepts to C without changing its meaning.

2. Compute a reduction $\widehat{C}$ of C^* w.r.t. $\mathcal{T}$, which removes parts of C^* without changing its meaning..

3. Return $\widehat{C}$.

It remains to give formal definitions of the notions "extension" and "reduction."

Definition 23 *Let C be an $\mathcal{ALE}$ concept description and $\mathcal{T}$ be an $\mathcal{ALE}$ TBox. An* extension *C^* of C w.r.t. $\mathcal{T}$ is an $\mathcal{ALE}$ concept description obtained from C by conjoining defined names at some positions in C such that C^* is equivalent to C modulo $\mathcal{T}$.*

Obviously, there may exist exponentially many different extensions of C^*, which shows that this step may take exponential time. Alternatively, we could considered this to be a non-deterministic step, in which an appropriate extension is guessed.

[5] A similar approach works also for the DL $\mathcal{ALN}$ [23].

Informally speaking, a reduction $\widehat{C}$ of C^* w.r.t. $\mathcal{T}$ is an $\mathcal{ALE}$ concept description obtained from C^* by "eliminating all redundancies in C^*" such that the resulting concept description is still equivalent to C^* modulo $\mathcal{T}$. A concept description may have exponentially many different reductions, and hence computing reductions may also be considered to be a non-deterministic step.

Before defining the notion of a "reduction" formally, we illustrate how our algorithm works by a simple example. Consider the $\mathcal{ALE}$ concept description

$$C = P \sqcap Q \sqcap \forall r.P \sqcap \exists r.(P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q)),$$

and the $\mathcal{ALE}$ TBox $\mathcal{T} = \{\ A_1 \doteq \exists r.Q,\ A_2 \doteq P \sqcap \forall r.P,\ A_3 \doteq \forall r.P\ \}$.

The concept description

$$\begin{aligned} C^* \quad = \quad & A_2 \sqcap P \sqcap Q \sqcap \forall r.P \sqcap \\ & \exists r.(A_1 \sqcap P \sqcap \exists r.Q) \sqcap \exists r.(P \sqcap \forall r.(Q \sqcap \neg Q)) \end{aligned}$$

is an extension of C. A reduction of C^* can be obtained by eliminating

- P and $\forall r.P$ on the top-level of C^*, because they are redundant w.r.t. A_2;
- P in both of the existential restrictions on the top-level of C^*, because it is redundant due to the value restriction $\forall r.P$ on the top-level of C;
- the existential restriction $\exists r.Q$, because it is redundant w.r.t. A_1; and
- replacing $Q \sqcap \neg Q$ by $\bot$, since $\bot$ is the minimal inconsistent concept description.

The resulting concept description $\widehat{C} = A_2 \sqcap Q \sqcap \exists r.A_1 \sqcap \exists r.\forall r.\bot$ is equivalent to C modulo $\mathcal{T}$, i.e., $\widehat{C}$ is a rewriting of C using $\mathcal{T}$. Furthermore, it is easy to see that $\widehat{C}$ is in fact a *minimal* rewriting of C using $\mathcal{T}$.

Before we can define the notion of a "reduction" formally, we must formalize the notion of a "subdescription."

Definition 24 *The $\mathcal{ALE}$ concept description $\widehat{C}$ is a* subdescription *of the $\mathcal{ALE}$ concept description C iff it is equivalent to*

1. *$\widehat{C} = C$; or*
2. *$\widehat{C} = \bot$; or*
3. *$\widehat{C}$ is obtained from C by*
   - *removing some (negated) primitive concept names, value restrictions, or existential restrictions on the top-level of C, and*

- *for all remaining value/existential restrictions $\forall r.D/\exists r.D$ replacing D by a subdescription $\widehat{D}$ of D.*

The subdescription $\widehat{C}$ of C is a proper *subdescription of C iff it is different from C.*

Now, reductions can be defined as follows:

Definition 25 *Let C^* be an $\mathcal{ALE}$ concept description and $\mathcal{T}$ be an $\mathcal{ALE}$ TBox. The $\mathcal{ALE}$ concept description $\widehat{C}$ is called a* reduction of C^* w.r.t. $\mathcal{T}$ *iff $\widehat{C}$ is equivalent to C^* w.r.t. $\mathcal{T}$ and minimal in the following sense: there does not exist a proper subdescription of $\widehat{C}$ that is also equivalent to C^* w.r.t. $\mathcal{T}$.*

Note that, in the definition of a reduction, we do not allow removal of defined concepts unless they occur within value or existential restrictions that are removed as a whole. This makes sense since such defined concepts could have been omitted in the first place when computing the extension C^* of C.

From the definition of a reduction, it is not immediately clear how to actually compute one. Intuitively, a reduction $\widehat{C}$ of an $\mathcal{ALE}$ concept C^* in $\forall$-normal form is computed in a top-down manner. If $C \equiv_{\mathcal{T}} \bot$, then $\widehat{C} := \bot$. Otherwise, let $\forall r.C'$ be the (unique!) value restriction on the role r and $A_1 \sqcap \ldots \sqcap A_n$ the conjunction of the names of defined concepts on the top-level of C. Basically, $\widehat{C}$ is then obtained from C^* as follows:

1. Remove any (negated) primitive concept Q occurring on the top-level of C^*, if $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq_{\mathcal{T}} Q$.
2. Remove any existential restriction $\exists r.C_1$ occurring on the top-level of C^*, if
   (a) $A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.C' \sqsubseteq_{\mathcal{T}} \exists r.C_1$, or
   (b) there is another existential restriction $\exists r.C_2$ on the top-level of C^* such that $A_1 \sqcap \ldots \sqcap A_n \sqcap \forall r.C' \sqcap \exists r.C_2 \sqsubseteq_{\mathcal{T}} \exists r.C_1$.
3. Remove the value restriction $\forall r.C'$ if $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq_{\mathcal{T}} \forall r.C'$.
4. Finally, all concept descriptions D occurring in the remaining value and existential restrictions are reduced recursively.

The formal specification of the reduction algorithm given in [23] is more complex than the informal description given above mainly for two reasons. First, in (2b) it could be the case that the subsumption relation also holds if the rôles of $\exists r.C_1$ and $\exists r.C_2$ are exchanged. In this case, one has a choice of which existential restriction to remove. If the (recursive) reduction of C_1 and C_2 yields descriptions of different size, then we must remove the existential restriction for the concept with the larger reduction. If, however, the reductions are of equal size, then we must make a (don't know) nondeterministic choice between removing the one or the other.

Second, in (4) we cannot reduce the descriptions D without considering the context in which they occur. The reduction of these concepts must take into account the concept C' of the top-level value restriction of C as well as all concepts D' occurring in value restrictions of the form $\forall r.D'$ on the top-level of the defining concepts for $A_1, \ldots, A_n$. For instance, in our example the removal of P within the existential restrictions on the top-level of C^* was justified by the presence of $\forall r.P$ on the top-level of C^*. For this purpose, the algorithm described in [23] employs a third input parameter that takes care of such contexts.

Theorem 26 *The rewriting algorithm for $\mathcal{ALE}$ defined in [23] has the following properties:*

1. *Every possible output of the algorithm is a rewriting of the input concept description C using the input TBox $\mathcal{T}$, though it need not always be minimal.*

2. *The set of all computed rewritings contains all minimal rewritings of C using $\mathcal{T}$ (modulo associativity, commutativity and idempotence of conjunction, and the equivalence $C \sqcap \top \equiv C$).*

3. *One minimal rewriting of C w.r.t. $\mathcal{T}$ can be computed using polynomial space.*

4. *The set of all minimal rewritings of C w.r.t. $\mathcal{T}$ can be computed in exponential time.*

In practice, it often suffices to compute one (not necessarily minimal, but "small") rewriting. The sketch of the rewriting algorithm presented above suggests the following greedy algorithm for computing such a small rewriting. First, compute the extension C^* of C in which at all positions of C all possible defined concepts are conjoined. Then compute just one reduction $\widehat{C}$ of C^*. This yields a polynomial-time algorithm—given an oracle for equivalence testing—which does not always return a minimal rewriting, but nevertheless behaves well in practice, both in terms of the quality of the returned rewritings and in terms of runtime (see [23] for more details).

7.3 Approximation

Given two DLs $\mathcal{L}_s$ and $\mathcal{L}_d$, an $\mathcal{L}_d$ *approximation* of an $\mathcal{L}_s$ concept description C is an $\mathcal{L}_d$ concept description D such that $C \sqsubseteq D$ and D is minimal (w.r.t. subsumption) in $\mathcal{L}_d$ with this property.

In [42] the case where $\mathcal{L}_s$ is $\mathcal{ALC}$ and $\mathcal{L}_d$ is $\mathcal{ALE}$ was investigated in detail. It was shown that for every $\mathcal{ALC}$ concept description there exists a unique (up to equivalence) approximation in $\mathcal{ALE}$. The size of the $\mathcal{ALE}$ approximation may grow exponentially in the size of the given $\mathcal{ALC}$ concept description, and it can be computed in double exponential time.

	$\mathcal{EL}$	$\mathcal{ALE}$
subsumption	P	NP-complete
equivalence	NP-complete	NP-complete

Table 3: Deciding the solvability of matching problems

To measure the information that is lost by using an approximation rather than the original concept, in [42] the notion of the difference between concepts has been refined from an early definition by Teege [91]. Intuitively, the difference between a concept description C and its approximation is the concept description that needs to be conjoined to the approximation to obtain a concept description equivalent to C.

8 Matching

The matching problem has been introduced in Section 3.2. In this section, we sketch how it can be solved. As usual, our exposition concentrates on the DL $\mathcal{ALE}$. However, we will also comment on other DLs and on extensions of the basic matching problem. Most results presented here are based on [18, 71].

In what follows, we first consider the complexity of deciding whether a given matching problem has a solution (Subsection 8.1). In case a matching problem has a solution, we are also interested in computing a solution. In general, a solvable matching problem may have several (even infinitely many) solutions. Thus, the question arises what solutions are actually interesting ones. We try to answer this question in Subsection 8.2, where we define a precedence orderings on matchers. This ordering tells us which matchers are more interesting than others. Algorithms for computing such matchers in $\mathcal{ALE}$ are presented in Subsection 8.3. A summary of results for matching in other DLs as well as extensions of the basic matching problem is provided in Subsection 8.4.

8.1 Deciding Matching Problems

We study the question of how to decide whether a given matching problem has a matcher or not, and investigate the complexity of this problem. For the DLs $\mathcal{EL}$ and $\mathcal{ALE}$ we obtain the complexity results summarized in Table 3. The first and the second row of the table refer to matching modulo subsumption and matching modulo equivalence, respectively.

These results can be obtained as follows: First, note that patterns are not required to contain variables. Consequently, matching modulo subsumption (equivalence) is at least as hard as subsumption (equivalence). Thus, NP-completeness of subsumption in $\mathcal{ALE}$ [50] yields hardness in the second column of Table 3. Second, for the languages $\mathcal{ALE}$ and $\mathcal{EL}$, as already mentioned in Section 3.2, matching modulo subsumption can be reduced to subsumption: $C \sqsubseteq^? D$ has a matcher iff the substitution $\sigma_\top$, which replaces every variable by $\top$, is a matcher of $C \sqsubseteq^? D$. Thus, the known complexity results for subsumption in $\mathcal{ALE}$ and $\mathcal{EL}$ [50, 22] complete the first row of Table 3. Third, NP-hardness of matching modulo equivalence for $\mathcal{EL}$ can be shown by a reduction from SAT. It remains to show that matching modulo equivalence in $\mathcal{EL}$ and $\mathcal{ALE}$ can in fact be decided in non-deterministic polynomial time. This is an easy consequence of the following (non-trivial) lemma [71].

Lemma 27 *If an $\mathcal{EL}$ or $\mathcal{ALE}$ matching problem modulo equivalence has a matcher, then it has one of size polynomially bounded in the size of the problem. Furthermore, this matcher uses only concept and role names already contained in the matching problem.*

The lemma (together with the known complexity results for subsumption) shows that the following can be realized in NP: "guess" a substitution satisfying the given size bound, and then test whether it is a matcher.

8.2 Solutions of Matching Problems

As mentioned above, solvable matching problems may have infinitely many solutions. Hence, it is necessary to define a class of "interesting" matchers to be presented to the user. Such a definition certainly depends on the specific application in mind. Our definition is motivated by the application in chemical process engineering mentioned before. However, it is general enough to apply also to other applications.

We use the $\mathcal{EL}$ concept description C^1_{ex} and the pattern D^1_{ex} shown in Figure 7 to illustrate and motivate our definitions. Along with the concept descriptions, Figure 7 also depicts the description trees corresponding to C^1_{ex} and D^1_{ex} as defined in Section 4.1, where concept variables are simply dealt with like concept names.

It is easy to see that the substitution $\sigma_\top$ is a matcher of $C^1_{\mathrm{ex}} \sqsubseteq^? D^1_{\mathrm{ex}}$, and thus this matching problem modulo subsumption is indeed solvable. However, the matcher $\sigma_\top$ is obviously not an interesting one. We are interested in matchers that bring us as close as possible to the description C^1_{ex}. In this sense, the matcher

$$\sigma_1 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.\mathsf{W},\ Y \mapsto \mathsf{W}\}$$

is better than $\sigma_\top$, but still not optimal. In fact,

$$\sigma_2 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.\mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}),\ Y \mapsto \mathsf{W} \sqcap \mathsf{D}\}$$

is better than σ_1 since it satisfies $C^1_{\mathrm{ex}} \equiv \sigma_2(D^1_{\mathrm{ex}}) \sqsubset \sigma_1(D^1_{\mathrm{ex}})$.

We formalize this intuition with the help of the following precedence ordering on matchers. For a given matching problem $C \sqsubseteq^? D$ and two matchers σ, τ we define

$$\sigma \sqsubseteq_i \tau \ \text{ iff } \ \sigma(D) \sqsubseteq \tau(D).$$

Here "i" stands for "instance". Two matchers σ, τ are *i-equivalent* ($\sigma \equiv_i \tau$) iff $\sigma \sqsubseteq_i \tau$ and $\tau \sqsubseteq_i \sigma$. A matcher σ is called *i-minimal* iff $\tau \sqsubseteq_i \sigma$ implies $\tau \equiv_i \sigma$ for every matcher τ.

$C^1_{\text{ex}} := \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{D}) \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P})) \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{D} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}))$
$D^1_{\text{ex}} := \mathsf{W} \sqcap \exists \mathsf{hc}.(X \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap Y)) \sqcap \exists \mathsf{hc}.(X \sqcap Y)$

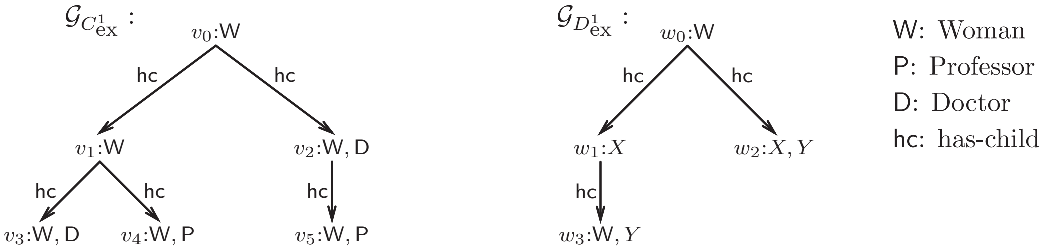

Figure 7: $\mathcal{EL}$ concept description and pattern, and their $\mathcal{EL}$ description trees.

We are interested in *computing i-minimal matchers*. More precisely, we want to obtain at least one i-minimal matcher for each of the minimal i-equivalence classes (i.e., i-equivalence classes of i-minimal matchers). Note that, given an i-minimal matcher σ of a matching problem $C \sqsubseteq^? D$, its equivalence class, i.e., the set of all matchers that are i-equivalent to σ, consists of the matchers of the problem $\sigma(D) \equiv^? D$.

The matching problem

$$\exists r.A \sqcap \exists r.B \sqsubseteq^? \exists r.X$$

illustrates that there may in fact be different minimal i-equivalence classes: mapping X to A and mapping X to B, respectively, yields two i-minimal matchers which, however, do not belong to the same i-equivalence class.

Since an i-equivalence class usually contains more than one matcher, the question is which ones to prefer within this class. In our running example, σ_2 is a least and therefore i-minimal matcher. Nevertheless, it is not the one we really want to compute since it contains redundancies, i.e., expressions that are not really necessary for obtaining the instance $\sigma_2(D^1_{\text{ex}})$ (modulo equivalence). In fact, σ_2 contains two different kinds of redundancies. First, the existential restriction $\exists \mathsf{hc}.\mathsf{W}$ in $\sigma_2(X)$ is redundant since removing it still yields a concept description equivalent to $\sigma_2(X)$. Second, W in $\sigma_2(Y)$ is redundant in that the substitution obtained by deleting W from $\sigma_2(Y)$ still yields the same instance of D^1_{ex} (although the resulting concept description is no longer equivalent to $\sigma_2(Y)$). In our example, the only i-minimal matcher (modulo associativity and commutativity of concept conjunction) that is free of redundancies in this sense is

$$\sigma_3 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}),\ Y \mapsto \mathsf{D}\}.$$

Summing up, we want to compute *all i-minimal matchers that are reduced*, i.e., free of redundancies. We use the notion of subdescriptions introduced above (Definition 24) to capture the notion "reduced" in a formal way. Given two matchers σ, τ of $C \sqsubseteq^? D$, we say that τ is a *submatcher* of σ iff $\tau(Y)$ is a (not necessarily strict) subdescription of $\sigma(Y)$ for all variables Y. If τ is a *submatcher* of σ and there is at least one variable X for which $\tau(X)$ is a strict subdescription of $\sigma(X)$, then we say that τ is a *strict submatcher* of σ.

Definition 28 *The matcher σ of $C \sqsubseteq^? D$ is* i-minimal and reduced *iff*

1. *σ is i-minimal,*

2. *σ is in $\forall$-normal form, i.e., $\sigma(X)$ is in $\forall$-normal form for all variables X (see Section 7.2 for the definition of $\forall$-normal form), and*

3. *there does not exist a matcher τ of $C \sqsubseteq^? D$ that is both i-equivalent to σ and a strict submatcher of σ.*

8.3 Computing Matchers

In the previous section, we have identified the set of all i-minimal and reduced matchers (in $\forall$-normal form) as the set of "interesting" matchers. We now show how these matchers can be computed. Given a matching problem $C \sqsubseteq^? D$, our algorithm for computing i-minimal and reduced matchers in principle proceeds as follows:

1. Compute the set of all i-minimal matchers of $C \sqsubseteq^? D$ up to i-equivalence (i.e., one matcher for each i-equivalence class).

2. For each i-minimal matcher σ computed in the first step, compute the set of all reduced matchers in $\forall$-normal form up to commutativity and associativity of conjunction for the problem $\sigma(D) \equiv^? D$.

If we are interested in matching modulo equivalence instead of subsumption, we just apply the second step to $C \equiv^? D$.

In the following two subsections, we illustrate the first step of the algorithm—computing i-minimal matchers—for $\mathcal{EL}$ and $\mathcal{ALE}$. For the second step, we refer the reader to [18, 71]. In particular, this step involves to show that every solvable $\mathcal{ALE}$ matching problem has a matcher of size polynomially bounded in the size of the matching problem.

The main results on computing matchers shown in [18, 71] are summarized in the following theorem. We call a set containing all i-minimal matchers up to i-equivalence *i-complete*. Such a set is called *minimal i-complete* if it contains only i-minimal matchers. Similarly, a set containing all reduced matchers in $\forall$-normal form (up to commutativity and associativity of conjunction) is called *complete w.r.t. reduction*, and it is called *minimal* if it contains only reduced matchers.

Theorem 29 *1. For a solvable $\mathcal{ALE}$ or $\mathcal{EL}$ matching problem modulo subsumption, the cardinality of a (minimal) i-complete set can be bounded exponentially in the size of the matching problem. This upper bounds is tight. Furthermore, minimal i-complete* *sets can be computed in exponential time in case of $\mathcal{EL}$ and in exponential space in case of $\mathcal{ALE}$. If minimality is not required, such a set can be computed in exponential time also for $\mathcal{ALE}$.*

2. *For a solvable $\mathcal{ALE}$ or $\mathcal{EL}$ matching problem modulo equivalence, the cardinality of a (minimal) complete set w.r.t. reduction may grow exponentially in the size of the matching problem. However, the size of the matchers in this set can polynomially be bounded. This immediately implies that there exists an exponential time algorithm for computing minimal complete sets w.r.t. reduction (both for $\mathcal{ALE}$ and $\mathcal{EL}$).*

8.3.1 Computing i-minimal Matchers in $\mathcal{EL}$

The algorithm for computing i-minimal matchers in $\mathcal{EL}$ is based on the characterization of subsumption via homomorphisms between description trees presented in Section 4.1.

Given a matching problem of the form $C \sqsubseteq^? D$, our algorithm computes homomorphisms from the description tree $\mathcal{G}_D$ corresponding to D into the description tree $\mathcal{G}_C$ corresponding to C. Concept patters are turned into description trees in the obvious way, i.e., concept variables are dealt with as concept names (see, e.g., Figure 7). When computing the homomorphisms from $\mathcal{G}_D$ into $\mathcal{G}_C$, the variables in $\mathcal{G}_D$ are ignored. For instance, in our example, there are six homomorphisms from $\mathcal{G}_{D^1_{\mathsf{ex}}}$ into $\mathcal{G}_{C^1_{\mathsf{ex}}}$. We will later consider the ones mapping w_i onto v_i for $i = 0, 1, 2$, and w_3 onto v_3 or w_3 onto v_4, which we denote by φ_1 and φ_2, respectively.

The complete algorithm is depicted in Figure 8. With $C_{\varphi(v)}$ we denote the $\mathcal{EL}$ concept description that corresponds to the $\mathcal{EL}$ description tree rooted at the node $\varphi(v)$ in $\mathcal{G}_C$. The algorithm constructs substitutions τ such that $C \sqsubseteq \tau(D)$, i.e., there is a homomorphism from $\mathcal{G}_{\tau(D)}$ into $\mathcal{G}_C$. This is achieved by first computing all homomorphisms from $\mathcal{G}_D$ into $\mathcal{G}_C$. Assume that the node v in $\mathcal{G}_D$, whose label contains X, is mapped onto the note $w = \varphi(v)$ of $\mathcal{G}_C$. The idea is then to substitute X with the concept description corresponding to the subtree of $\mathcal{G}_C$ starting with the node $w = \varphi(v)$, i.e., with $C_{\varphi(v)}$. The remaining problem is that a variable X may occur more than once in D. Thus, we cannot simply define $\tau(X)$ as $C_{\varphi(v)}$ where v is such that X occurs in the label of v. Since there may exist several nodes v with this property, we take the least common subsumer of the corresponding parts of C. The reason for taking the *least* common subsumer is that we want to compute substitutions that are as specific as possible.

In our example, the homomorphism φ_1 yields the substitution τ_1:

$$\begin{aligned}
\tau_1(X) &:= lcs\{C^1_{\mathsf{ex},v_1},\ C^1_{\mathsf{ex},v_2}\} \equiv \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}),\\
\tau_1(Y) &:= lcs\{C^1_{\mathsf{ex},v_2},\ C^1_{\mathsf{ex},v_3}\} \equiv \mathsf{W} \sqcap \mathsf{D},
\end{aligned}$$

whereas φ_2 yields the substitution τ_2:

$$\begin{aligned}
\tau_2(X) &:= lcs\{C^1_{\mathsf{ex},v_1},\ C^1_{\mathsf{ex},v_2}\} \equiv \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}),\\
\tau_2(Y) &:= lcs\{C^1_{\mathsf{ex},v_2},\ C^1_{\mathsf{ex},v_4}\} \equiv \mathsf{W}.
\end{aligned}$$

Input: $\mathcal{EL}$ matching problem $C \sqsubseteq^? D$.
Output: i-complete set $\mathcal{C}$ for $C \sqsubseteq^? D$.

$\mathcal{C} := \emptyset$;
For all homomorphisms φ from
  $\mathcal{G}_D = (V, E, v_0, \ell)$ into $\mathcal{G}_C$ do
    Define τ by $\tau(X) := lcs\{C_{\varphi(v)} \mid X \in \ell(v)\}$
      for all variables X in D;
    $\mathcal{C} := \mathcal{C} \cup \{\tau\}$;

Figure 8: The $\mathcal{EL}$ matching algorithm

Unlike τ_1, the substitution τ_2 is not i-minimal. Therefore, τ_2 will be removed in a post-processing step, which extracts a *minimal* i-complete set from the i-complete one. By applying Theorem 8, the following theorem is easy to show:

Theorem 30 *The algorithm described in Figure 8 always computes an i-complete set of matchers for a given $\mathcal{EL}$ matching problem modulo subsumption.*

8.3.2 Computing i-minimal Matchers in $\mathcal{ALE}$

The idea underlying the algorithm for computing i-minimal matchers in $\mathcal{ALE}$ is similar to the one for $\mathcal{EL}$. Again, we apply the characterization of subsumption by homomorphisms (Theorem 10). One problem is that this characterization requires the subsuming description to be normalized.[6] However, the pattern D contains variables, and hence the normalization of $\sigma(D)$ depends on what is substituted for these variables by the matcher σ. However, this matcher is exactly what we want to compute in the first place.

Fortunately, Theorem 10 can be relaxed as follows. To characterize the subsumption relation $C \sqsubseteq D$, it is not necessary to normalize D completely. Instead of $\mathcal{G}_D$, which is based on the normal form of D, it suffices to employ the tree $\mathcal{G}_D^\top$ that is obtained from the so-called $\top$-*normal form* of D. This normal form is obtained from D by exhaustively applying the rule $\forall r.\top \longrightarrow \top$. As an easy consequence of the proof of Theorem 10, we obtain the following corollary:

Corollary 31 *Let C, D be $\mathcal{ALE}$ concept descriptions. Then, $C \sqsubseteq D$ iff there exists a homomorphism from $\mathcal{G}_D^\top$ to $\mathcal{G}_C$.*

Given the $\mathcal{ALE}$ matching problem $C \sqsubseteq^? D$, the following example illustrates that it does not suffice to consider just all homomorphisms from $\mathcal{G}_D^\top$ to $\mathcal{G}_C$ in order to compute an i-complete set.

[6] Recall that, in the case of $\mathcal{ALE}$, the description tree $\mathcal{G}_C$ of a concept description C is obtained from the normal form of C.

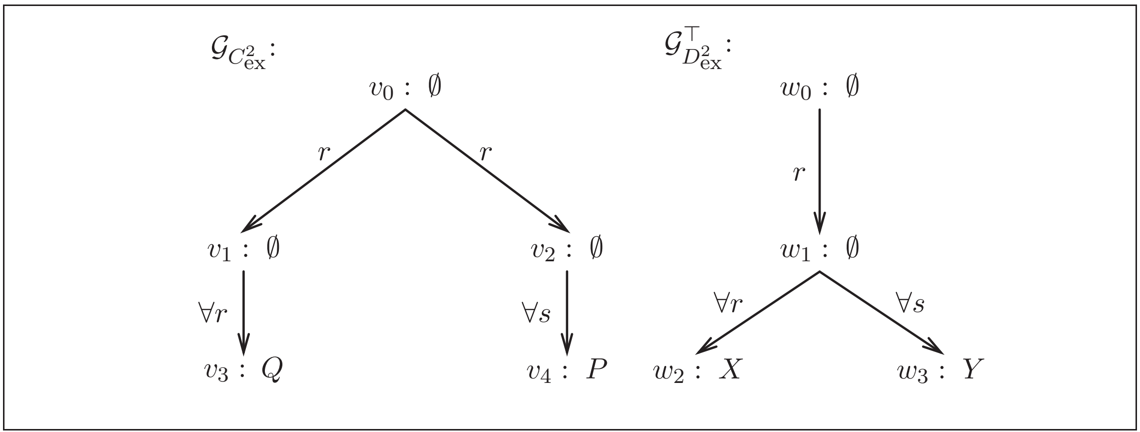

Figure 9: The description trees for C^2_{ex} and D^2_{ex}.

Example 32 *Consider the $\mathcal{ALE}$ matching problem $C^2_{ex} \sqsubseteq^? D^2_{ex}$, where*

$$\begin{aligned} C^2_{ex} &:= (\exists r.\forall r.Q) \sqcap (\exists r.\forall s.P) \\ D^2_{ex} &:= \exists r.(\forall r.X \sqcap \forall s.Y). \end{aligned}$$

The description trees corresponding to C^2_{ex} and D^2_{ex} are depicted in Figure 9. Obviously, $\sigma := \{X \mapsto Q, Y \mapsto \top\}$ and $\tau := \{X \mapsto \top, Y \mapsto P\}$ are solutions of the matching problem. However, there is no homomorphism from $\mathcal{G}^{\top}_{D^2_{ex}}$ into $\mathcal{G}_{C^2_{ex}}$. Indeed, the node w_1 can be mapped either to v_1 or v_2. In the former case, w_2 can be mapped to v_3, but then there is no way to map w_3. In the latter case, w_3 must be mapped to v_4, but then there is no node w_2 can be mapped to.

The problem is that Corollary 31 requires the subsumer to be in $\top$-normal form. However, the $\top$-normal form of the instantiated concept pattern depends on the matcher, and thus cannot be computed in advance. Fortunately, only matchers that substitute variables by $\top$ cause problems. Thus, the problem can be fixed by first guessing which variables are replaced by $\top$. Replacing these variables in D by $\top$ yields a so-called *$\top$-pattern E*. Now, instead of computing all homomorphisms from $\mathcal{G}^{\top}_D$ into $\mathcal{G}_C$, our matching algorithm computes for *all* $\top$-patterns E of D all homomorphism from $\mathcal{G}^{\top}_E$ into $\mathcal{G}_C$. With this modification, we obtain:

Theorem 33 *There is an algorithm that computes an i-complete set of matchers for a given $\mathcal{ALE}$ matching problem modulo subsumption.*

8.4 Matching in other DLs and Extensions of Matching

We give only a very brief overview on results for other DLs and on extensions of matching (see also [71] for a more detailed overview).

Matching has also been considered for the DLs $\mathcal{ALN}$ [21], $\mathcal{ALNS}$ [71], and $\mathcal{ALN}$ with cyclic TBoxes [71], based on the characterization of subsumption proved for these DLs.

The basic matching problem, as introduced in Section 3.2, has been extended in the following two directions. First, matching where variables are further constrained by side conditions of the form $X \sqsubseteq E$ or $X \sqsubset E$ (where E is a concept pattern and X is a concept variable) was first introduced in [37], and further studied in [21, 10] for the DL $\mathcal{ALN}$.

Second, unification, which extends matching modulo equivalence in that both sides of the equation may contain variables, has first been introduced in the context of DLs in [25], and studied there for the DL $\mathcal{FL}_0$. It is shown there that unification is considerably more complex than matching: even for the small DL $\mathcal{FL}_0$, deciding whether a given unification problem has a solution or not is EXPTIME-complete. Later on, these results were extended to unification in $\mathcal{FL}_{trans}$, the extension of $\mathcal{FL}_0$ by transitive closure of roles [19], and to the extension of this DL by atomic negation [20].

9 Conclusion and Future Perspectives

Compared to the large body of results for standard inferences in DLs, the investigation of non-standard inferences is only at its beginning. Nevertheless, for the DLs $\mathcal{ALE}$ and $\mathcal{ALN}$ and their sublanguages, we now have a relatively good understanding of how to solve non-standard inferences like computing the least common subsumer, matching, and rewriting. For these results to be useful in practice, two more problems must be addressed, though.

First, there is a need for good implementations of the algorithms developed for non-standard inferences, which must be able to interact with existing systems implementing standard inferences. The system Sonic [92, 93] is a first step in this direction. It extends the ontology editor OilEd [32] by implementations of the non-standard inferences lcs and approximation, and uses the system Racer [54] as standard reasoner. There also exist first implementations of matching algorithms for $\mathcal{ALE}$ [41] and $\mathcal{ALN}$ [43].

Second, modern DL systems like FaCT [61] and Racer [54] are based on very expressive DLs, and there exist large knowledge bases that use this expressive power and can be processed by these systems [85, 90, 53]. In contrast, results for non-standard inferences are currently restricted to rather inexpressive DLs, and some of these inferences do not even make sense for more expressive DLs.[7] In order to allow the user to re-use concepts defined in such existing expressive knowledge bases and still support the user with non-standard inferences, one can either use approximation or consider non-standard inferences w.r.t. a background terminology.

To explain these two options in more detail, assume that $\mathcal{L}_2$ is an expressive DL, and that $\mathcal{L}_1$ is a sublanguage of $\mathcal{L}_2$ for which we know how to compute non-standard inferences. In the first case, one first computes the $\mathcal{L}_1$ approximation of the concepts

[7]For example, as pointed out before, using the lcs does not make sense in DLs allowing for disjunction.

expressed in $\mathcal{L}_2$, and then applies the non-standard inferences in $\mathcal{L}_1$. As mentioned, first results for approximation have been obtained in [42]. In the second case, one considers a *background terminology* $\mathcal{T}$ defined in $\mathcal{L}_2$. When defining new concepts, the user employs only the sublanguage $\mathcal{L}_1$ of $\mathcal{L}_2$. However, in addition to primitive concepts and roles, the concept descriptions written in the DL $\mathcal{L}_1$ may also contain names of concepts defined in $\mathcal{T}$. The non-standard inferences are then defined modulo the TBox $\mathcal{T}$, i.e., instead of using subsumption between $\mathcal{L}_1$ concept descriptions, one uses subsumption w.r.t. the TBox $\mathcal{T}$. First results for the lcs modulo background terminologies have been obtained in [30].

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[2] Franz Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, 1991.

[3] Franz Baader. A formal definition for the expressive power of terminological knowledge representation languages. *J. of Logic and Computation*, 6:33–54, 1996.

[4] Franz Baader. Using automata theory for characterizing the semantics of terminological cycles. *Ann. of Mathematics and Artificial Intelligence*, 18:175–219, 1996.

[5] Franz Baader. Computing the least common subsumer in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In *Proceedings of the 11th International Conference on Conceptual Structures, ICCS 2003*, volume 2746 of *Lecture Notes in Artificial Intelligence*, pages 117–130. Springer-Verlag, 2003.

[6] Franz Baader. The instance problem and the most specific concept in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In *Proceedings of the 26th Annual German Conference on Artificial Intelligence, KI 2003*, volume 2821 of *Lecture Notes in Artificial Intelligence*, pages 64–78, Hamburg, Germany, 2003. Springer-Verlag.

[7] Franz Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 319–324. Morgan Kaufmann, 2003.

[8] Franz Baader. Terminological cycles in a description logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 325–330. Morgan Kaufmann, 2003.

[9] Franz Baader. A graph-theoretic generalization of the least common subsumer and the most specific concept in the description logic $\mathcal{EL}$. In Jurai Hromkovic and Manfred Nagl, editors, *Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2004)*, Lecture Notes in Computer Science, Bad Honnef, Germany, 2004. Springer-Verlag.

[10] Franz Baader, Sebastian Brandt, and Ralf Küsters. Matching under side conditions in description logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 213–218, 2001.

[11] Franz Baader, Martin Buchheit, and Bernhard Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[12] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[13] Franz Baader and Philipp Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 452–457, 1991.

[14] Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge (PDK'91)*, volume 567 of *Lecture Notes in Artificial Intelligence*, pages 67–86. Springer-Verlag, 1991.

[15] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics for the semantic web. *KI – Künstliche Intelligenz*, 4, 2002.

[16] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks in Information Systems, pages 3–28. Springer–Verlag, Berlin, Germany, 2003.

[17] Franz Baader and Ralf Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of the 22nd German Annual Conf. on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1998.

[18] Franz Baader and Ralf Küsters. Matching in description logics with existential restrictions. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 261–272, 2000.

[19] Franz Baader and Ralf Küsters. Unification in a description logic with transitive closure of roles. In Robert Nieuwenhuis and Andrei Voronkov, editors, *Proceedings*

of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, (LPAR 2001), Lecture Notes in Artificial Intelligence, Havana, Cuba, 2001. Springer-Verlag.

[20] Franz Baader and Ralf Küsters. Unification in a Description Logic with Inconsistency and Transitive Closure of Roles. In *Proceedings of the 2002 International Workshop on Description Logics (DL 2002)*, 2002. Available from http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/.

[21] Franz Baader, Ralf Küsters, Alex Borgida, and Deborah L. McGuinness. Matching in description logics. *J. of Logic and Computation*, 9(3):411–447, 1999.

[22] Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.

[23] Franz Baader, Ralf Küsters, and Ralf Molitor. Rewriting concepts using terminologies. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 297–308, 2000.

[24] Franz Baader, Ralf Küsters, and Frank Wolter. Extensions to description logics. In *[12]*, pages 219–261. 2003.

[25] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. In H. Prade, editor, *Proc. of the 13th Eur. Conf. on Artificial Intelligence (ECAI'98)*, pages 331–335. John Wiley & Sons, 1998.

[26] Franz Baader and Paliath Narendran. Unification of concepts terms in description logics. *J. of Symbolic Computation*, 31(3):277–305, 2001.

[27] Franz Baader and Werner Nutt. Basic description logics. In *[12]*, pages 43–95. 2003.

[28] Franz Baader and Ulrike Sattler. Expressive number restrictions in description logics. *J. of Logic and Computation*, 9(3):319–350, 1999.

[29] Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

[30] Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the least common subsumer w.r.t. a background terminology. In José Júlio Alferes and João Alexandre Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004)*, volume 3229 of *Lecture Notes in Artificial Intelligence*, pages 400–412, Lisbon, Portugal, 2004. Springer-Verlag.

[31] Franz Baader and Anni-Yasmin Turhan. On the problem of computing small representations of least common subsumers. In *Proceedings of the 25th German Conference on Artificial Intelligence (KI 2002)*, volume 2479 of *Lecture Notes in Artificial Intelligence*, pages 99–113, Aachen, Germany, 2002. Springer-Verlag.

[32] Sean Bechhofer, Ian Horrocks, Carol Goble, and Robert Stevens. OilEd: A reasonable ontology editor for the semantic web. In Franz Baader, Gerhard Brewka, and Thomas Eiter, editors, *KI-2001: Advances in Artificial Intelligence*, volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 396–408. Springer-Verlag, 2001.

[33] Tim Berners-Lee, James A. Hendler, and Ora Lassila. The semantic Web. *Scientific American*, 284(5):34–43, 2001.

[34] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[35] Ralf Bogusch, Bernd Lohmann, and Wolfgang Marquardt. Computer-aided process modeling with MODKIT. In *Proceedings of Chemputers Europe III*, Frankfurt, 1996.

[36] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.

[37] Alexander Borgida and Deborah L. McGuinness. Asking queries about frames. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 340–349, 1996.

[38] Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.

[39] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos, 1985.

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

[41] Sebastian Brandt. Implementing matching in $\mathcal{ALE}$—first results. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*. CEUR Electronic Workshop Proceedings, http://CEUR-WS.org/Vol-81/, 2003.

[42] Sebastian Brandt, Ralf Küsters, and Anni-Yasmin Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Principles of*

Knowledge Representation and Reasoning (KR2002), pages 203–214, San Francisco, CA, 2002. Morgan Kaufman.

[43] 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 Electronic Workshop Proceedings, http://CEUR-WS.org/Vol-115/, 2004.

[44] Sebastian Brandt and Anni-Yasmin 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 (ADL'01)*. CEUR Electronic Workshop Proceedings, http://CEUR-WS.org/Vol-44/, 2001.

[45] Paolo Bresciani, Enrico Franconi, and Sergio Tessaris. Implementing and testing expressive description logics: Preliminary report. In *Proc. of the 1995 Description Logic Workshop (DL'95)*, pages 131–139, 1995.

[46] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

[47] Michel Chein and Marie-Laure Mugnier. Conceptual graphs: Fundamental notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[48] William W. Cohen and Haym Hirsh. Learning the CLASSIC description logics: Theoretical and experimental results. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 121–133, 1994.

[49] Francesco Donini. Complexity of reasoning. In *[12]*, pages 96–136. 2003.

[50] Francesco M. Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.

[51] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.

[52] Michael R. Garey and David S. Johnson. *Computers and Intractability — A guide to NP-completeness*. W. H. Freeman and Company, San Francisco (CA, USA), 1979.

[53] Volker Haarslev and Ralf Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 2001.

[54] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

[55] Philipp Hanschke. Specifying role interaction in concept languages. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 318–329. Morgan Kaufmann, Los Altos, 1992.

[56] Bernhard Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of the German Workshop on Artificial Intelligence*, pages 38–47. Springer-Verlag, 1990.

[57] Bernhard Hollunder. Consistency checking reduced to satisfiability of concepts in terminological systems. *Ann. of Mathematics and Artificial Intelligence*, 18(2–4):133–157, 1996.

[58] Bernhard Hollunder and Franz Baader. Qualifying number restrictions in concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 335–346, 1991.

[59] Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of the 9th Eur. Conf. on Artificial Intelligence (ECAI'90)*, pages 348–353, London (United Kingdom), 1990. Pitman.

[60] Ian Horrocks. The FaCT system. In Harrie de Swart, editor, *Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 307–312. Springer-Verlag, 1998.

[61] Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[62] Ian Horrocks. Implementation and optimization techniques. In *[12]*, pages 306–346. 2003.

[63] Ian Horrocks and Peter F. Patel-Schneider. DL systems comparison. In *Proc. of the 1998 Description Logic Workshop (DL'98)*, pages 55–57. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-11/, 1998.

[64] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[65] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

[66] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*. Morgan Kaufmann, Los Altos, 2001.

[67] Ian Horrocks, Ulrike Sattler, and Stefan Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.

[68] Yevgeny Kazakov and Hans de Nivelle. Subsumption of concepts in $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*. CEUR Electronic Workshop Proceedings, http://CEUR-WS.org/Vol-81/, 2003.

[69] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protégé OWL plugin: An open development environment for semantic web applications. In *Proceedings of the Third International Semantic Web Conference*, Hiroshima, Japan, 2004.

[70] Ralf Küsters. Characterizing the semantics of terminological cycles in $\mathcal{ALN}$ using finite automata. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 499–510, 1998.

[71] Ralf Küsters. *Non-standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[72] Ralf Küsters and Alex Borgida. What's in an attribute? Consequences for the least common subsumer. *J. of Artificial Intelligence Research*, 14:167–203, 2001.

[73] Ralf Küsters and Ralf Molitor. Approximating most specific concepts in description logics with existential restrictions. In Franz Baader, Gerd Brewka, and Thomas Eiter, editors, *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 33–47, Vienna, Austria, 2001. Springer-Verlag.

[74] Ralf Küsters and Ralf Molitor. Computing least common subsumers in $\mathcal{ALEN}$. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 219–224, 2001.

[75] Carsten Lutz. Complexity of terminological reasoning revisited. In *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, pages 181–200. Springer-Verlag, 1999.

[76] Robert MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, Los Altos, 1991.

[77] Wolfgang Marquardt, Lars von Wedel, and Birgit Bayer. Perspectives on Lifecycle Process Modeling. In *Proceedings of fifth International Conference on Foundations of Computer-Aided (FOCAPD'99)*, Breckenridge, Colorado, USA, 1999. Process Design.

[78] Eric Mays, Robert Dionne, and Robert Weida. K-REP system overview. *SIGART Bull.*, 2(3), 1991.

[79] Marvin Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*. The MIT Press, 1981. A longer version appeared in *The Psychology of Computer Vision* (1975). Republished in [39].

[80] Ralf Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken (Supporting the Modelling of of Chemical Processes by Using Non-standard Inferences in Description Logics)*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2000. In German.

[81] Ralf Möller and Volker Haarslev. Description logic systems. In *[12]*, pages 282–305. 2003.

[82] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[83] Christof Peltason. The BACK system — an overview. *SIGART Bull.*, 2(3):114–119, 1991.

[84] M. Ross Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967. Republished in [39].

[85] Alan Rector and Ian Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, Stanford, CA, 1997. AAAI Press.

[86] Steven W. Reyner. An analysis of a good algorithm for the subtree problem. *SIAM Journal of Computing*, 6(4):730–732, 1977.

[87] Ulrike Sattler. *Terminological Knowledge Representation Systems in a Process Engineering Application*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1998.

[88] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

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

[90] Stefan Schultz and Udo Hahn. Knowledge engineering by large-scale knowledge reuse—experience from the medical domain. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 601–610. Morgan Kaufmann, 2000.

[91] Gunnar Teege. Making the Difference: A Subtraction Operation for Description Logics. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 540–550, Bonn, Germany, 1994. Morgan Kaufmann, Los Altos.

[92] Anni-Yasmin Turhan and Christian Kissig. SONIC—non-standard inferences go OILED. In D. Basin and M. Rusinowitch, editors, *Proceedings of the 2nd International Joint Conference on Automated Reasoning (IJCAR'04)*, volume 3097 of *Lecture Notes in Artificial Intelligence*, pages 321–325. Springer-Verlag, 2004.

[93] Anni-Yasmin Turhan and Christian Kissig. SONIC—system description. In *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*. CEUR Electronic Workshop Proceedings, http://CEUR-WS.org/Vol-104/, 2004.

[94] Lars von Wedel and Wolfgang Marquardt. ROME: A Repository to Support the Integration of Models over the Lifecycle of Model-based Engineering Processes. In *Proceedings of the 10th European Symposium on Computer Aided Process Engineering (ESCAPE-10)*, 2000.

[95] William A. Woods and James G. Schmolze. The KL-ONE family. In F. W. Lehmann, editor, *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992. Published as a special issue of *Computers & Mathematics with Applications*, Volume 23, Number 2–9.