Reasoning Services for the Maintenance and Flexible Access to Description Logic Ontologies

an Stelle einer Habilitationsschrift

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dr.-Ing. Anni-Yasmin Turhan

Betreuender Hochschullehrer: Prof. Dr.-Ing. Franz Baader

Dresden, November 2013

Contents

1 Introduction

In Computer Science logics play an important role in many fields. Applied Computer Science employs logical formalisms and their reasoning services in data base systems or to verify properties of complex software and hardware systems. In theoretical computer science logics are investigated in regard to their close link to automata. Their expressivity is subject in the field of descriptive complexity and the reasoning algorithms for logics are assessed by means of computational complexity. In the field of Artificial Intelligence logics and their reasoning problems are investigated since the early beginnings—in particular in the field of knowledge representation. In logic-based knowledge representation one of the main research goals is to devise formalisms for which reasoning services that are vital to applications can be solved by sound, complete and terminating algorithms—and ideally have low computational complexity. By reasoning services in general we refer to tasks that infer from a given representation of facts new facts. This thesis studies some of these reasoning services in the context of formal ontology languages.

Description Logics (DLs) [7] are a family of knowledge representation formalisms with model-based semantics that allow to represent concepts and (binary) relations from an application domain. A DL knowledge base (KB) consists of a TBox, which contains intensional knowledge such as concept definitions, and an ABox, which contains extensional knowledge and is used to describe individual facts. Essentially, a TBox is a first order logic (FOL) theory and the ABox is a set of ground facts.

DLs are closely related to Modal Logics or Hybrid Logics [74, 9] and most DLs are decidable fragments of First Order Logic. DLs are investigated with respect to different reasoning services. A main research goal is to devise DLs that provide sufficient expressivity while allowing for reasoning algorithms with low computational complexity. Reasoning algorithms for DLs that offer a good trade-off for these criteria are implemented in highly optimized reasoning systems. This motivated the choice of DLs as the basis for the ontology web language OWL standardized by the W3C.

From the mid-nineties on, the main research effort in the field of DLs was to investigate highly expressive DLs for which reasoning is still decidable [41, 81, 13, 42, 40, 43]. One of these highly expressive DL is $\mathcal{SROIQ}$ that allows for reasoning in 2NExpTime [40, 45] and that is the DL underlying OWL 2 [64]. In the last decade so-called lightweight DLs, which have rather limited expressivity, but allow for tractable reasoning procedures were investigated as well. Lightweight DLs are tailored to yield efficient algorithms for a particular reasoning problem. The DLs from the $\mathcal{EL}$-family of DLs mostly yield tractable reasoning algorithms for testing subsumption, i.e., for testing whether one concept is more general than another [5, 6]. The DL $\mathcal{EL}^{++}$ is a DL that still allows for subsumption tests in polynomial time and is a

maximal DL with this property. $\mathcal{EL}^{++}$ is the DL underlying the OWL 2 EL profile [64].

Flexible access to instance data in DL knowledge bases can be facilitated by answering of conjunctive queries. Here the task is to evaluate over a DL knowledge base (unions of) conjunctive queries, which correspond to the select-project-join fragment of SQL. While for highly expressive DL this task co-NP-hard [73, 34, 57], the DLs from the DL Lite family allow for conjunctive query entailment[1] in LogSpace [20, 21]—albeit at the cost of severe restrictions on the expressivity. For applications with huge instance data and only lightweight modelling of the background information this DL can still be very useful. In fact, DL Lite$_\mathcal{R}$ is the basis of the OWL 2 QL profile [64].

The reasoner systems for OWL 2 and its profiles support standard reasoning tasks for DL KBs, which are used in many application domains. Besides in the semantic web, geography or for context-aware systems DLs are employed, most prominently, in the bio-medical field, where large ontologies are in use for more than a decade. For example, the well-known ontologies SNOMED [78] and the GENE ONTOLOGY [25] use only the expressivity of the OWL 2 profile and they contain ten thousands of statements. Now, building or maintaining such ontologies is certainly a non-trivial task for users with little expertise in knowledge representation or logic and calls for automated support. To this end, a number of so-called non-standard inferences have been defined and investigated. This thesis presents results on such non-standard inferences that can be employed for the maintenance and flexible access to description logic ontologies.

1.1 Basic Notions of Description Logics

The main building blocks of a DL knowledge base are concepts. Each DL provides a set of *concept constructors*, which allows to build complex concepts. Starting from two disjoint sets: a set of concept names $\mathsf{N_C}$ and a set of role names $\mathsf{N_R}$, *concepts* can be built. In the DL $\mathcal{EL}$, concept names $A \in \mathsf{N_C}$ and also the top concept $\top$ are $\mathcal{EL}$-concepts. Complex concepts are defined inductively. Let C and D be $\mathcal{EL}$-concepts and r be a role name, then concept conjunctions $C \sqcap D$ and existential restrictions $\exists r.C$ are $\mathcal{EL}$-concepts.

The DL $\mathcal{ALC}$ extends $\mathcal{EL}$ by disjunction, value restrictions and negation, which makes it a propositionally complete DL. It also provides the *bottom concept* $\bot$. $\mathcal{ALC}$ is extended to the highly expressive DL $\mathcal{SROIQ}$ by qualified number restrictions and nominals. These concept constructors and their syntax are shown in Table 1.

Concepts typically provide an intensional characterization of a set of objects. Formally, the semantics of concepts are given by interpretations. An

[1] Query entailment is the decision problem corresponding to query answering

Name	Syntax	Semantics
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = (C^{\mathcal{I}} \cap D^{\mathcal{I}})$
disjunction	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = (C^{\mathcal{I}} \cup D^{\mathcal{I}})$
negation	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
value restriction	$\forall r.C$	$(\forall r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall e.(d,e) \in r^{\mathcal{I}} \longrightarrow e \in C^{\mathcal{I}}\}$
nominals	$\{a\}$	$\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$
qualified at-least restriction	$(\geq n\ r\ C)$	$(\geq n\ r\ C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e_1, \ldots, e_n.\ e_i \neq e_j, 1 \leq i < j \leq n \wedge (d, e_i) \in r^{\mathcal{I}} \wedge e_i \in C^{\mathcal{I}}, 1 \leq i \leq n\}$
qualified at-most restriction	$(\leq n\ r\ C)$	$(\leq n\ r\ C)^{\mathcal{I}} = \big(\neg(\geq n{+}1\ r\ C)\big)^{\mathcal{I}}$

Table 1: DL Concept Constructors.

interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a pair consisting of an *interpretation domain* $\Delta^{\mathcal{I}}$ and a mapping function $\cdot^{\mathcal{I}}$. Concept names ($A \in \mathsf{N_C}$) are interpreted as subsets of $\Delta^{\mathcal{I}}$: $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. Role names ($r \in \mathsf{N_R}$) are interpreted as binary relations over $\Delta^{\mathcal{I}}$: $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The concept $\top$ is always interpreted as the whole interpretation domain ($\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$) and the bottom concept as the empty set ($\bot^{\mathcal{I}} = \emptyset$). The interpretation function is extended to complex concepts according to the semantics of concept constructors, which is shown in the third column of Table 1.

In many DLs roles can be refined to model knowledge about binary relations from an application domain by *role axioms*. For instance, roles can be declared to be transitive. Furthermore, some DLs allow to use *simple role inclusion axioms* to state sub-role and super-role relationships, which together constitute the *role hierarchy*. Commonly used role axioms are shown in Table 2. The DL $\mathcal{SROIQ}$, for example, offers besides the mentioned concept constructors also transitive roles, inverse roles, and complex role

Name	Syntax	Semantics
transitive role	$\mathrm{trans}(r)$	$(d,e) \in r^{\mathcal{I}} \wedge (e,f) \in r^{\mathcal{I}} \longrightarrow (d,f) \in r^{\mathcal{I}}$
simple role inclusion	$r \sqsubseteq s$	$(d,e) \in r^{\mathcal{I}} \longrightarrow (d,e) \in s^{\mathcal{I}}$
complex role inclusion	$r \circ s \sqsubseteq t$	$(d,e) \in r^{\mathcal{I}} \wedge (e,f) \in s^{\mathcal{I}} \longrightarrow (d,f) \in t^{\mathcal{I}}$

Table 2: DL Role Axioms.

inclusion axioms. $\mathcal{SROIQ}$ is the DL corresponding to OWL 2.

The semantics of the role axioms are given in Table 2. A role axiom is *satisfied* by an interpretation $\mathcal{I}$, if $\mathcal{I}$ fulfills the conditions from the third column of Table 2. Note that complex role inclusions can be used to express transitive roles ($t \circ t \sqsubseteq t$) and simple role inclusion axioms, if the identity relation is captured in a role id ($id \circ r \sqsubseteq s$).

The $\mathcal{EL}$-family consists of $\mathcal{EL}$ and of DLs that extend $\mathcal{EL}$, but that neither offer value restrictions nor full negation. The DL that extends $\mathcal{EL}$ with nominals is called $\mathcal{ELO}$. Nominals allow to refer to individuals in complex concepts. DLs with nominals correspond to Hybrid Logics. The extension of $\mathcal{EL}$ with simple role inclusion axioms is called $\mathcal{ELH}$ and the one by complex role inclusion axioms is called $\mathcal{ELR}$. Some DLs allow to declare *inverse roles*, where $inv(r)^{\mathcal{I}} = \{(e,d) \mid (d,e) \in r^{\mathcal{I}}\}$. The extension of $\mathcal{EL}$ by inverse roles is called $\mathcal{ELI}$. The DL $\mathcal{ELOR}$ extends $\mathcal{EL}$ by nominals and complex role inclusions. $\mathcal{ELOR}$ extended by data types in turn yields $\mathcal{EL}^{++}$, which is the DL underlying the OWL 2 EL profile of OWL 2.

A concept name can be assigned to a (complex) concept or two concepts can be related to each other by general concept axioms.

Definition 1 (General concept axioms)**.** *Let C and D be concepts and A a concept name. Then a statement of the form $C \sqsubseteq D$ is called a* general concept inclusion axiom *(GCI). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$* satisfies *a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$*

We use *concept equivalence axioms*, statements of the form $C \equiv D$, as an abbreviation for the two GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$. If, in a concept equivalence axiom the left-hand side is a concept name ($A \equiv C$, $A \in NC$), then this kind of axiom is called a *concept definition*. Note, that the disjointness of concepts, say C and D can be expressed by the GCI $C \sqcap D \sqsubseteq \bot$. Role axioms and concept axioms are collected in the TBox.

Definition 2 (TBox, unfoldable TBox)**.** *A* TBox $\mathcal{T}$ *is a finite set of concept axioms and role axioms. An* unfoldable TBox *is a TBox, where*

- *all concept axioms are concept definitions,*
- *each concept name appears at most once on the left-hand sides of a concept axiom, and*
- *each concept definition is* acyclic*, i.e., the concept name on the left-hand side is not referred to directly or indirectly in the right-hand side.*

To distinguish unfoldable TBoxes from those that do not fulfill the conditions for unfoldable TBoxes, these TBoxes are sometimes referred to as *general TBoxes*. The concept names appearing on the left-hand sides in unfoldable TBoxes are called *defined concepts*, whereas the remaining concept

names are called *primitive concepts*. TBoxes have model-based semantics. An interpretation is a *model* of a TBox $\mathcal{T}$, if each GCI and role axiom in $\mathcal{T}$ is satisfied.

When using unfoldable TBoxes, the information on concepts from the TBox can be treated by a pre-processing step. Concept definitions can be *unfolded*, i.e., the defined concepts appearing in the left-hand side of the concept definitions are replaced by the right-hand sides of their concept definitions. This kind of replacement is done exhaustively when unfolding a TBox. Due to the conditions for unfoldable TBoxes, this process always terminates, but it may generate concepts of size exponential in the worst case [66].

Assertions are statements that capture information on individual facts—such as concept memberships or relationships to other individuals. $\mathsf{N_I}$ is the set of individual names and pairwise disjoint to $\mathsf{N_C}$ and $\mathsf{N_R}$. Let a and b be individual names ($\{a, b\} \subseteq \mathsf{N_I}$), C be a concept and r a role. *Concept assertions* are statements of the form $C(a)$. *Role assertions* are statements of the form $r(a, b)$. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation. For individuals interpretations are extended, so that they map every $a \in \mathsf{N_I}$ to an $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. A concept assertion $C(a)$ is satisfied in an interpretation $\mathcal{I}$, iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$. A role assertion $r(a, b)$ is satisfied in an interpretation $\mathcal{I}$, iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$.

Definition 3 (ABox, knowledge base)**.** *An ABox $\mathcal{A}$ is a finite set of concept assertions and role assertions. An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$, if it satisfies all assertions contained in $\mathcal{A}$.*

A knowledge base $\mathcal{K}$ *is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, if it is a model of $\mathcal{T}$ and of $\mathcal{A}$.*

Sometimes we call a knowledge base an *ontology*. The model-based semantics of knowledge bases are the basis for the reasoning services defined for DLs.

1.2 Reasoning in Description Logics

Reasoning services allow to derive implicitly captured information from the explicitly given. The classical reasoning problems underlying common such services can be defined as follows.

Definition 4. *Let C and D be concepts, a an individual, $\mathcal{T}$ a general TBox, $\mathcal{A}$ an ABox and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.*

Concept satisfiability: *A concept C is* satisfiable w.r.t. $\mathcal{T}$ *iff $C^{\mathcal{I}} \neq \emptyset$ for some model $\mathcal{I}$ of $\mathcal{T}$.*

Subsumption: *A concept C is* subsumed by a concept D w.r.t. $\mathcal{T}$ *(denoted $C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$.*

Concept equivalence: *a concept C is* equivalent to a concept D w.r.t. $\mathcal{T}$ *(denoted $C \equiv_{\mathcal{T}} D$) iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$.*

Knowledge base consistency: *$\mathcal{K}$ is* consistent *iff it has a model.*

Instance checking: *a is an* instance of C w.r.t. $\mathcal{K}$ *(denoted $\mathcal{K} \models C(a)$) iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$.*

Obviously, concept equivalences can be tested by two subsumption tests in the following way: $C \equiv_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. In DL systems complex reasoning services are offered, which are defined based on the above reasoning problems as follows.

Classification: Given a TBox $\mathcal{T}$. Compute the subsumption relationships between all *concept names* in $\mathcal{T}$.

Instance retrieval: Given a concept C and a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, compute all individuals a in $\mathcal{A}$ s.t. $\mathcal{K} \models C(a)$ holds.[2]

ABox realization: Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Compute for each individual a in $\mathcal{A}$ the set of those *concept names* in $\mathcal{K}$ to which a is an instance of.

The reasoning problems from Definition 4 can be reduced to each other, if negation is provided in the DL under consideration (see [7, 83]). In fact, most of these reductions can be done in polynomial time. Due to the reductions, it suffices to investigate reasoning algorithms for only one of these reasoning problems. For DLs that are extensions of $\mathcal{ALC}$ it is a common approach to reduce the other reasoning problems to knowledge base consistency, which is then tested by the *tableaux method*. Essentially, this method constructs a model (or a model-like structure) that gives evidence to the consistency of the KB.

For $\mathcal{ALC}$ the computational complexity for testing consistency of a KB depends on whether general or unfoldable TBoxes are part of the KB. In case of unfoldable TBoxes KB consistency is PSpace-complete while for general TBoxes the complexity becomes ExpTime-complete. In case of $\mathcal{SROIQ}$ the complexity rises to 2NExpTime [45].

Another reasoning service that has been intensively investigated in recent years is answering (unions of) conjunctive queries. Intuitively, a conjunctive query is a conjunction of assertions that may also contain variables, of which some can be existentially quantified. Some of the variables in the conjunctive query are so-called *answer variables*. If all occurrences of these variables are replaced in the query by a tuple of individuals from the ABox such that the obtained query is entailed by the KB, then this tuple of individuals is a *matcher* for the query. *Conjunctive query answering* is to compute all such

[2] Instance retrieval is sometimes also called instance query.

matchers. Answering conjunctive queries thus offers a much more powerful way to query the KB, than instance retrieval, where the queries are limited by the expressivity of the DL in use. For example, $\mathcal{ALC}$ only allows to query for tree-shaped structures by the use of concepts in the query. In contrast conjunctive queries allow to query about finite graph structures. *Query entailment* tests whether a given query has a matcher in a given KB. Query entailment is the decision problem corresponding to conjunctive query answering.

The complexity of query entailment for $\mathcal{ALC}$ is ExpTime, if measured w.r.t. the whole KB, whereas its data complexity (i.e. measured w.r.t. the ABox alone) is co-NP-complete [57]. Conjunctive query answering in expressive DLs such as the DL corresponding to OWL 2 is 2ExpTime-hard regarding combined complexity [57, 45], and coNP-hard for the data complexity [73, 67]. For the lightweight profiles the data complexity for conjunctive query answering is polynomial in case of OWL 2 EL [72, 51, 50] and even in AC^0 for OWL 2 QL [21].

1.3 Generalization Inferences for DLs

Generalization inferences are reasoning services that generalize either a group of concepts or an individual into one concept. These inferences are particularly useful when building or structuring TBoxes. They allow to derive a complex concept from a set of individuals that has all individuals from the set as instances and thus allow to derive a new complex concept in an example-driven way. This can be done by the *bottom-up approach* (see [8, 19]) in two steps: first generalize each individual into a concept and then generalizing all of these into a single concept. The first step in the bottom-up approach is realized by the computation of the most specific concept, initially defined in [24]. The second step is realized by the computation of the least common subsumer, an inference initially defined in [23]. Both inferences compute possibly complex concepts written in a *target DL*.

Definition 5 (least common subsumer, most specific concept)**.** *Let $\mathcal{L}$ be a DL, C and D be $\mathcal{L}$-concepts, $\mathcal{T}$ an $\mathcal{L}$-TBox, $\mathcal{A}$ an $\mathcal{L}$-ABox, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a an individual from $\mathcal{A}$. A* most specific concept *(MSC) of a w.r.t. $\mathcal{K}$ (denoted $msc_{\mathcal{K}}(a)$) is a concept E such that*

- $\mathcal{K} \models E(a)$ *and*
- *for each $\mathcal{L}$-concept F holds: if $\mathcal{K} \models F(a)$, then $E \sqsubseteq_{\mathcal{T}} F$.*

A least common subsumer *(LCS) of C and D w.r.t. $\mathcal{T}$ (denoted $lcs_{\mathcal{T}}(C, D)$) is an $\mathcal{L}$-concept E such that*

1. *$C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$ and*
2. *for each $\mathcal{L}$-concept F holds: if $C \sqsubseteq_{\mathcal{T}} F$ and $D \sqsubseteq_{\mathcal{T}} F$, then $E \sqsubseteq_{\mathcal{T}} F$.*

Since in DLs with conjunction the MSC and the LCS are unique modulo equivalence, we speak about *the* LCS and *the* MSC, respectively. For DLs that offer disjunction the LCS of a group of concepts is simply is their disjunction. Thus mostly sub-Boolean DLs are considered for the computation of the LCS.

Computation algorithms for the LCS of concept (w.r.t. an empty TBox) have been defined for $\mathcal{EL}$ and extensions of it in [8, 52]. These algorithms represent the input concepts as labelled trees and use a characterization of subsumption by homomorphisms between such trees.

1.4 Reasoning in $\mathcal{EL}$ by Completion

A reasoning procedure that computes classification for $\mathcal{EL}$-TBoxes in polynomial time is the *completion algorithm* [17, 5]. This algorithm proceeds in three steps:

1. Transform the TBox into normal form.
   An $\mathcal{EL}$-TBox is in *normal form*, if it only contains GCIs of the form:

   $$A_1 \sqsubseteq B, \quad A_1 \sqcap A_2 \sqsubseteq B, \quad A_1 \sqsubseteq \exists r.A_2 \quad \text{or} \quad \exists r.A_1 \sqsubseteq B.$$

   where $A_1, A_2 \in \mathsf{N_C}$ and $D \in \mathsf{N_C} \cup \{\bot\}$ appear in $\mathcal{T}$.

2. Apply completion rules to completion sets. Two kinds of completion sets are used: one for each named concept A in $\mathcal{T}$ ($S(A)$) and one for pairs of named concepts and roles in $\mathcal{T}$ ($S(A, r)$). Initially, the algorithm sets each $S(A) = \{A, \top\}$ and each $(S(A, r) = \emptyset)$. The following invariants hold during the application of the completion rules: (I1) if $B \in S(A)$ then $A \sqsubseteq_{\mathcal{T}} B$ and (I2) if $B \in S(A, r)$ then $A \sqsubseteq_{\mathcal{T}} \exists r.B$.

3. Read-off subsumption relationships from saturated completion sets. Once the completion rules have been applied exhaustively, the following holds:

   - $B \in S(A)$ iff $A \sqsubseteq_{\mathcal{T}} B$ and
   - $B \in S(A, r)$ iff $A \sqsubseteq_{\mathcal{T}} \exists r.B$.

A more thorough introduction on classification by the completion method is given in [83]. The completion method or variants thereof are implemented in a number of DL reasoners—such as CEL [10], SNOROCKET [54], JCEL [62, 61], and ELK[3] [46]. The completion algorithm computes the canonical model of the TBox.

Definition 6 (Canonical model [60])**.** *Let C be a concept and $\mathcal{T}$ an $\mathcal{EL}$-TBox. The* canonical model *$\mathcal{I}_{C,\mathcal{T}}$ of C and $\mathcal{T}$ is defined as follows:*

[3] ELK uses an optimized variant of completion that performs normalization and introduction of completion sets 'on demand'.

- $\Delta^{\mathcal{I}_{C,\mathcal{T}}} := \{d_C\} \cup \{d_A \mid \exists A \in \mathsf{N_C} \cap (\mathit{sub}(C) \cup \mathit{sub}(\mathcal{T}))\} \cup \{d_{C'} \mid \exists r.C' \in \mathit{sub}(C) \cup \mathit{sub}(\mathcal{T})\}$;
- $A^{\mathcal{I}_{C,\mathcal{T}}} := \{d_D \mid D \sqsubseteq_{\mathcal{T}} A\}$, *for all* $A \in \mathsf{N_C}$ *appearing in* $\mathcal{T}$ *or* C;
- $r^{\mathcal{I}_{C,\mathcal{T}}} := \{(d_D, d_{D'}) \mid D \sqsubseteq_{\mathcal{T}} \exists r.D'$ *for* $D' \in \mathit{sub}(\mathcal{T})$ *or* $D \sqsubseteq_R \exists r.D'$ *for* $D' \in \mathit{sub}(C)\}$ *for all* $r \in \mathsf{N_R}$ *appearing in* $\mathcal{T}$ *or* C;

Starting from the canonical model, the subsumption relation between arbitrary $\mathcal{EL}$-concepts (formulated over the signature of the TBox) can be tested by simulations [60].

Definition 7 (Simulation)**.** *Let* $\mathcal{I}_1, \mathcal{I}_2$ *be interpretations and* $\lesssim$ *be a relation on* $\Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$*. The relation* $\lesssim$ *is a* simulation between $\mathcal{I}_1$ and $\mathcal{I}_2$ *if the following conditions are satisfied:*

- *If* $e_1 \lesssim e_2$ *and* $e_1 \in A^{\mathcal{I}_1}$*, then* $e_2 \in A^{\mathcal{I}_2}$ *(*$A \in \mathsf{N_C}$*).*
- *If* $e_1 \lesssim e_2$ *and* $(e_1, e_1') \in r^{\mathcal{I}_1}$*, then there exists* $e_2' \in \Delta^{\mathcal{I}_2}$ *s.t.* $(e_2, e_2') \in r^{\mathcal{I}_2}$ *and* $e_1' \lesssim e_2'$*.*

The completion method can be extended to an algorithm for ABox realization [17, 5]. Even for this task the complexity stays polynomial [5]. The completion procedures for $\mathcal{EL}$ have been lifted to several extensions of $\mathcal{EL}$ [5, 6, 47].

The completion method or the computation of canonical models, respectively, can be employed for computing other inferences than TBox classification or ABox realization as well—such as conjunctive query answering [72, 51, 50], ontology modularization [60] or computing explanations for unintuitive entailments [12].

1.5 Overview of the Thesis

Besides the two introductory articles [83, 11], this thesis presents the content of eight more publications, which essentially are grouped into three main topics.

1. **Application of DL reasoning for context-aware systems.** In Section 2 we address the task of situation recognition by standard DL reasoning services. The studies in [79] and [27] employ different versions of the web ontology language OWL and demonstrate the feasibility of this approach.

2. **Computing generalizations in the $\mathcal{EL}$ family.** Building DL knowledge bases is often a costly and difficult task. This task can be facilitated by the computation of the LCS and the MSC. We investigate the

computation of generalization inferences in the $\mathcal{EL}$-family. If general TBoxes or ABoxes with cyclic role assertions are used, then the LCS or MSC, respectively, need not exist. Section 3 shows how approximative solutions for the LCS and the MSC for members of the $\mathcal{EL}$-family [70, 29] can be computed based on the completion method for $\mathcal{EL}$.

This approach is extended to a variant of $\mathcal{EL}$ that can express probabilities in Section 4. We study a completion algorithm for this logic and give computation methods for the MSC and explanations [68].

In some cases the LCS and the MSC do exist even in the presence of cyclic structures in the knowledge base. Conditions characterizing these cases were described in [86] and are the focus of Section 5.

3. **Concept similarity measures.** Section 6 presents a flexible framework for generating $\mathcal{EL}$-concept similarity measures [55]. For the measures generated by our framework we can guarantee several desirable properties. Furthermore, first results on answering relaxed instance queries are presented. We investigated instance retrieval for concepts relaxed by concept similarity measures in [31].

2 DL Standard Reasoning for Situation Recognition in Context-aware Systems

Context-aware systems act proactively and autonomously to changes in their environment. These systems are equipped with sensing capabilities in the wider sense. Depending on the information gathered about the context of the system, adaptation decisions are invoked. Here, the central task is to monitor the context data from the environment and recognize possibly critical situations to which the system may have to adapt.

Typically, context information comes from different sources, such as sensor data, data from other applications or direct input from the user of the system. This heterogeneous information describes the environment on different levels of detail, which need to be integrated in order to derive the high-level context of the system. Furthermore, the context information can be incomplete due to unavailability of information sources or it can even be incoherent, if context sources are faulty or corrupted.

To detect inconsistent information and to deal gracefully with incomplete information, logical reasoning and DL reasoning in particular [4, 84, 80] have been proposed for situation recognition. DLs seem a natural candidate for representing context information, since they allow for representing facts on different levels of granularity. DL reasoning systems (typically) adopt the open world assumption and thus can deal with incomplete information in a graceful way.

In [79] we describe a general framework for employing standard DL reasoning services for situation recognition. The idea is to encode the situations to be recognized as queries over a DL KB. The TBox of this KB stores the background information on the system and its environment in terms of concepts. Such a TBox is built manually at design-time. The gathered information about the status of the environment is preprocessed and then collected in ABoxes, which are automatically generated at run-time. To recognize critical situations, a query is executed over the KB. If individuals matching the query are found, then the corresponding situation is recognized. The reasoning tasks instance retrieval and conjunctive query answering can be employed for situation recognition.

In [79] we employ concepts as queries to recognize situations. In order to detect whether any individual in the ABox is an instance of any of the (possibly many) query concepts, instance checks can be performed for those individuals representing the current situation.

We demonstrated the feasibility of our approach by a study in the intelligent home domain using OWL 1 reasoning systems that were available at that time (in 2008). In the application scenario an intelligent door lock is modelled that acts, when the door bell is rung. Here, essentially two kinds of situations were relevant. The first kind are situations, where it is clear

whether or not to open the door. The second kind are situations regarding the where-abouts of the inhabitant of the house, in order to delegate the decision on whether to open. We modelled the TBox and the query concepts representing situations in OWL. The resulting TBox used the expressivity of the DL $\mathcal{ALCHIF}$, which is a proper sub-logic of OWL Lite (corresponding to the DL $\mathcal{SHIF}$).

For the two complex situations to be recognized in this application we modelled the situations as concepts. In order to be able to recognize at least a generalization of these situations, we modelled relaxed variants of these situations—each of which captured only a single aspect of the complex situation. For instance, one complex situation was generalized in one variant modeling only information on the person who is ringing and in another variant by modeling only the current location of the inhabitant of the house. For these variants we also added generalized concepts to the TBox. This hierarchy of concepts describing more and more general, but simpler situations allows to recognize at least more general kinds of situations and thus allows the context-aware system to act.

The complexity of reasoning in OWL Lite is ExpTime-complete. To achieve acceptable running times, we built two variants of the TBox that avoid expressive means known to increase running times: GCIs and inverse roles together with role hierarchies. For our test suite of a TBox with 135 concepts, 27 roles and an ABox with 98 individuals, the process of ABox realization (after the TBox was classified) took about 2 seconds for most tested systems.[4] The two simplified variants of the TBox did decrease running times only moderately (if at all), while missing several inferences.

In [27] we tested the capabilities of OWL 2 reasoners for the different profiles regarding situation recognition for service management systems. Service management systems orchestrate the execution of complex services in distributed computing environments. The goal in such settings is to ensure that computing resources are efficiently utilized while functional and nonfunctional requirements of individual services are respected. This can be achieved by *service migration* [22], where the main memory content of a service is transferred from one physical machine to another at runtime or by *service rebinding* [28], where for a service instance running on an underutilized server, a new instance is started on a second server and all future service requests are redirected to this second one, so that the old service instance can be gradually terminated and the underutilized server can be switched off.

We modelled the situations to be recognized as concepts and also as unions of conjunctive queries and used the DL reasoning services instance query answering and answering of (unions of) conjunctive queries for rec-

[4]The difference of running times between FaCT++ and Pellet was almost 3 orders of magnitude.

Reasoner	Version	Query type		Profile		
		Inst.	Conj.	OWL	EL	QL
FaCT++ [82]	v1.6.1	✓		✓	✓	✓
HermiT [65]	v1.3.6	✓		$\mathcal{SHOIQ}$	✓	✓
Pellet [77]	v2.3.0	✓	✓	$\mathcal{SHOIN}(\mathcal{D})$	✓	✓
RacerPro [37]	v2.0	✓	✓	$\mathcal{SHIQ}(\mathcal{D})$	✓	✓
ELK [46]	v0.3.1	✓			$\mathcal{EL}^+$	
jCel [61]	v0.18.0	✓			$\mathcal{EL}^+$	
Quest [71]	v1.7-alpha		✓			✓

Table 3: Reasoners and their supported queries and profiles.

ognizing situations. We were interested in the performance of the OWL 2 reasoners w.r.t. the different DL-based OWL 2 profiles and the two reasoning services.

We manually built a TBox modelling a video player platform containing 113 concept definitions and 66 properties. This TBox uses $\mathcal{ALCIQ}$ a sublogic of OWL 2, for which reasoning is also ExpTime-complete. For both lightweight profiles, OWL 2 EL and OWL 2 QL, we built variations of the OWL 2 TBox manually—keeping much of the original information. In the case of OWL 2 QL, which does not offer much expressivity for building complex concepts, we encoded the necessary information in the queries. We modelled 13 situations as OWL 2 concepts (of size of approximately 10). Since OWL 2 EL does not offer universal quantification, only 11 of them are modeled as OWL 2 EL classes. For these 11 situations we formulated the corresponding union of conjunctive queries, which contain on average 15 disjuncts of conjunctions with 8 conjuncts each. We used two simple ABoxes, i.e., ABoxes which have no complex concepts, but only concept names in the concept assertions, and which contained 380 individuals, more than 770 class assertions, and more than 545 property assertions.

Our evaluation used seven DL reasoners, which differ w.r.t. the supported DL and the reasoning services provided. Table 3 depicts the tested reasoners, the used version, and the closest DL of the respective profile they implement ('✓' stands for full coverage). At the time of writing no reasoner for answering conjunctive queries in OWL 2 EL under the standard semantics was available.

The experiments showed that in case of the lightweight profiles not all situations could be recognized, due to the limited expressiveness.

For the use of query concepts the time for loading the ontology and instance retrieval for all situation concepts in OWL 2 took about 2 seconds for most systems. When using the OWL 2 EL profiles these times dropped to $0.5s$ for most systems. The time for answering conjunctive queries that

modelled the 11 situations was higher. For OWL 2, RacerPro[5] used 41*s*. For the two lightweight profiles the running times were similar: Pellet needed almost 3*s*, while it took 7.3*s* for RacerPro.

We carried out a similar study in [35] that addressed the task of situation recognition for the energy efficient use of software variants in a very similar test setting. Here, we found that the part of the video player ontology that modelled the hardware could easily be reused. The running times measured for a similar setup as above were all comparable.

Unsurprisingly, the use of query concepts is sometimes a limitation. In particular, query concepts only allow to query for tree-like structures for most DLs. In contrast, conjunctive queries allow to query for arbitrary structures by the use of variables. For instance, one would like to express that 'the provider of a network available at the current location of the user and the provider contracted by the user are the same', which are multiple relationships to the same (set of) objects via different roles. This kind of query can easily be captured by conjunctive queries. To express this in a concept, would then require the use of nominals.

[5] under nrql semantics

3 Computing Role-depth Bounded Generalizations

The least common subsumer (Def. 5) and the most specific concept (Def. 5) are the two generalization inferences we have investigated for some members of the $\mathcal{EL}$-family of Description Logics. If computed w.r.t. cyclic ABoxes [53] the MSC, and if computed w.r.t. general TBoxes the LCS do not need to exist [2], since a finite $\mathcal{EL}$-concept cannot capture the cyclic information.

There have been several approaches to address this problem. If computed w.r.t. greatest fixed points semantics the LCS always exists in the presence of general TBoxes [3, 18]. Similarly, in [58] an extension of $\mathcal{EL}$ with greatest fix points in the concept language has been proposed. In this extension the LCS and MSC always exist.

To compute generalizations in OWL 2 EL, extending the syntax or changing the semantics is not a feasible approach. Instead, we pursue a pragmatic approach in [70] by computing the generalization in $\mathcal{EL}$ w.r.t. general TBoxes up to a maximal role-depth (specified in the input) and thus obtaining only an approximation, if applied to cyclic concepts or individuals with cyclic role relationships, respectively. To approximate the MSC computed w.r.t. cyclic ABoxes (and unfoldable TBoxes) by limiting the role-depth had already been suggested in [53].

In case the LCS is computed w.r.t. an acyclic TBox, the obtained concept is the exact solution, provided that a sufficiently large role-depth bound has been chosen, which is the role-depth of the input concepts unfolded w.r.t. the TBox. Similarly, for KBs consisting of an acyclic TBox and an acyclic ABox, the MSC of an individual a can be obtained by taking the sum of the longest path from a in the ABox and the maximal role-depth of unfolded concept assertions of individuals reachable from a as the bound.

3.1 The Role-depth Bounded LCS

In [70] we introduced the notion of a role-depth bounded LCS limited by a bound k. The so-called k-*LCS* is a common subsumer of the input concepts, but it has a maximal role-depth of k and is the least such concept w.r.t. subsumption among all concepts of role-depth at most k. The idea underlying our computation algorithm for the k-LCS is to use the data structure that is computed already during completion for the classification of the TBox, i.e. to use the canonical model of the TBox.

Computing the Role-depth Bounded LCS in $\mathcal{EL}$

The computation algorithm for the k-LCS proceeds in the following steps:

1. it introduces concept names for the possibly complex input concepts by adding concept definitions to the TBox,
2. performs classification of the enriched TBox

3. starting from the completion sets of the newly introduced concepts, it computes the cross-product of their existential restrictions taken from the completion sets for subsuming existential restrictions ($S(A, r)$) and applies the k-LCS recursively with decreased role-depth.

Through the traversal of the completion sets, a tree unraveling of the canonical model is produced. The cross-product construction of these tree unravelings yields the k-LCS. (The concept names introduced by completion can safely be omitted.) It was already shown in [8] that the computation of the LCS of $\mathcal{EL}$-concepts (without a TBox) can be done by a cross-product construction of the syntax trees of the input concepts.

Unfortunately, the k-LCS for $\mathcal{EL}$ can grow exponentially in the size of the TBox.

Computing the Role-depth Bounded LCS in $\mathcal{ELOR}$

$\mathcal{EL}$-concepts can describe (classes of) tree structures. In particular they do not allow to express multiple relationships to the same (set of) objects via different roles. Nominals can be used to describe such structures. This and a better coverage of the OWL 2 EL profile by generalization inferences motivated our extension of the above approach to nominals and complex role hierarchies.

Nominals can lead to conditional subsumption relationships, depending on whether a concept has an empty interpretation or not. To handle these conditional subsumption relationships the non-emptiness of concepts needs to be propagated along the reachability relation between concepts. This means that in the presence of nominals the completion sets are computed w.r.t. an additional parameter—a concept name that is assumed to have a non-empty interpretation. Such an extension has been proposed in [5][6] and [47][7] for $\mathcal{ELO}$ and can be classified in polynomial time. We gave a completion-based version of this algorithm in [29] as a basis for a computation algorithm for the k-LCS in $\mathcal{ELOR}$. The k-LCS algorithm performs the same three steps as the one for $\mathcal{EL}$. However, it only needs to consider those concepts that are related to the input concepts by the reachability relation, when re-classifying the TBox augmented by the k-LCS input concepts. Again, the resulting concept can grow exponentially in the size of the input.

Additional Results on the Role-depth Bounded LCS. On the one hand we investigated the completion-based method to compute the k-LCS

[6] The algorithm turned out to be incomplete.

[7] This optimized method performs normalization and the computation of parts of the canonical model only if necessary.

also for $\mathcal{ELI}$ in [33]. On the other hand it turns out that the completion-based method yields concepts for the MSC and the LCS that have redundant parts, since *all* subsumers are stored in the completion sets. To obtain smaller representations of the k-LCS, we investigated rewriting methods together with optimization methods for the k-LCS in [32], which we implemented in our system GEL. An evaluation for the $\mathcal{EL}$ case on a version of the medical ontology GALEN showed that even for a role-depth bound of $k = 4$ computation times for the un-optimized k-LCS can take an hour, while the optimized version takes only 1.9 seconds.

3.2 The Role-depth Bounded MSC

The k-MSC is a concept that has the input individual as an instance, has a maximal role-depth of at most k, and is minimal w.r.t. subsumption among all concepts of role-depth k. The k-MSC can be computed from the canonical model of the ABox, which is computed by the completion method for ABox realization.

Role-depth Bounded MSC in $\mathcal{EL}$

Our computation algorithm for the k-MSC in $\mathcal{EL}$ uses the completion sets obtained during ABox realization [70]. The algorithm unravels the canonical model starting from the element representing the input individual. It starts from the completion set of the input individual and invoking the k-MSC (with decreased role-depth bound) for each directly related individual recursively. Similarly to the completion-based k-LCS, this method yields redundant (sub-)concepts, for which the same rewriting techniques can be applied to obtain more succinct, but equivalent concepts. In general the exponential growth of the resulting concept cannot be avoided. One can show that the k-MSC in $\mathcal{EL}$ can grow exponentially in k, but only polynomially in the size of the knowledge base.

Role-depth Bounded MSC w.r.t. $\mathcal{ELOR}$ KBs

The MSC expressed in $\mathcal{ELO}$ is trivial, since $msc_{\mathcal{K}}(a) = \{a\}$. This kind of msc does not provide much insight regarding the concept containing a as an instance. Thus we investigated the MSC expressed in $\mathcal{EL}$, but computed w.r.t. an $\mathcal{ELOR}$ KB. In DLs with nominals and existential restrictions, the ABox can be 'absorbed' into the TBox by encoding concept assertions $C(a)$ as $\{a\} \sqsubseteq C$ and role assertions $r(a, b)$ as $\{a\} \sqsubseteq \exists r.\{b\}$. Once the completion rules for the enriched TBox have been applied exhaustively, the k-MSC in $\mathcal{EL}$ can be obtained by starting from the completion set that represents the input individual.

For $\mathcal{ELOR}$ highly redundant concepts are obtained for the MSC when using the completion-based algorithm naively, since (1) *all* named subsumers

are considered (not only the direct ones) and (2) existential restrictions are duplicated for all super-roles of a role. Similar rewriting techniques that result in a more succinct concept as in the case of the LCS can be employed. Similarly to the case of the $\mathcal{EL}$-MSC w.r.t. an $\mathcal{EL}$-KB, the $\mathcal{EL}$-MSC w.r.t. an $\mathcal{ELOR}$-KB can grow exponentially in k, but only polynomially in the size of the knowledge base.

By the methods for computing the role-depth bounded generalizations for $\mathcal{ELOR}$-KBs, we can now cover most of the expressivity of the OWL 2 EL profile by our algorithms—data-types (and disjointness axioms) are missing.

4 Non-standard Inferences for DLs with Subjective Probabilities

In applications, information is not always crisp, but holds only with a degree of certainty. Crisp DLs cannot represent this kind of information faithfully. In recent years several ways of extending DLs by expressive means for uncertainty or imprecision have been investigated. For instance, fuzzy DLs [15, 14, 16], rough DLs [76, 26, 48], or probabilistic DLs [49, 56] were defined and reasoning in these DLs was investigated.

A DL that can express subjective probabilities in the concepts is the DL Prob-$\mathcal{EL}_c$ devised in [59]. Prob-$\mathcal{EL}_c$ extends $\mathcal{EL}$ and allows for concepts following the syntax rule:

$$C, D \quad ::= \quad \top \quad | \quad A \quad | \quad C \sqcap D \quad | \quad \exists r.C \quad | \quad P_{\triangleright q} C \quad | \quad \exists P_{\triangleright q} r.C,$$

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, $\triangleright \in \{>, <, \geq, \leq, =\}$, and $q \in [0, 1]$. Intuitively, a concept of the form $P_{\triangleright q} C$ denotes the class of all objects that belong to C with a probability $\triangleright q$.

The semantics of Prob-$\mathcal{EL}_c$ generalizes the semantics of classical $\mathcal{EL}$ by considering a set of possible worlds, corresponding to an extension of FOL by subjective (or Type 2) probabilities [38]. More precisely, the semantics of Prob-$\mathcal{EL}_c$ is based on probabilistic interpretations. A *probabilistic interpretation* is a tuple of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, W, (\mathcal{I}_w)_{w \in W}, \mu)$, where $\Delta^{\mathcal{I}}$ is a (non-empty) domain, W is a non-empty set of (possible) *worlds*, μ is a discrete probability distribution over the set of worlds W, and for every $w \in W$, $\mathcal{I}_w$ is a classical $\mathcal{EL}$ interpretation with domain $\Delta^{\mathcal{I}}$.

While (crisp) concept names are interpreted as subsets of each world in a probabilistic interpretation, probabilistic named concepts $P_{\triangleright q} A$ are interpreted by:

$$\begin{aligned} P_{\triangleright q} A^{\mathcal{I}_w} &:= \{d \in \Delta \mid p_d^{\mathcal{I}}(A) \triangleright q\} \\ p_d^{\mathcal{I}}(A) &:= \mu(\{w \in W \mid d \in A^{\mathcal{I}_w}\}), \\ p_{d,e}^{\mathcal{I}}(r) &:= \mu(\{w \in W \mid (d, e) \in r^{\mathcal{I}_w}\}). \end{aligned}$$

The functions $\mathcal{I}_w$ and $p_d^{\mathcal{I}}$ are extended to complex concept descriptions through a mutual recursion, see [59] for details. A probabilistic interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, W, (\mathcal{I}_w)_{w \in W}, \mu)$ *satisfies*

- the GCI $C \sqsubseteq D$, if for every $w \in W$ it holds that $C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w}$.
- the assertion $C(a)$, if for every world $w \in W$ it holds that $a_w^{\mathcal{I}} \in C_w^{\mathcal{I}}$,
- the assertion $r(a, b)$, if $(a_w^{\mathcal{I}}, b_w^{\mathcal{I}}) \in r_w^{\mathcal{I}}$ for all $w \in W$, and
- the assertion $P_{\triangleright q} r(a, b)$, if $p_{a^{\mathcal{I}}, b^{\mathcal{I}}}^{\mathcal{I}}(r) \triangleright q$.

Based on this definition, the notions of model of a TBox, model of an ABox, concept subsumption, and instance checking are defined as before.

Unfortunately, the probabilistic constructors increase the complexity of reasoning, and deciding subsumption becomes intractable in general (more precisely ExpTime-complete) [36]. However, reasoning stays tractable if only the probabilistic concepts $P_{>0}C$ and $P_{=1}C$ are allowed. This DL, called Prob-$\mathcal{EL}_c^{01}$, allows only to express that a concept C may hold almost surely ($P_{=1}C$), or that C may be improbable, but not impossible ($P_{>0}C$). A completion algorithm for Prob-$\mathcal{EL}_c^{01}$ was devised in [59][8], which we used as a foundation to investigate generalization inferences in Prob-$\mathcal{EL}_c^{01}$. Prob-$\mathcal{EL}_c^{01}$ has the *uniform model property*, which states that, if a Prob-$\mathcal{EL}_c^{01}$-KB has a model, then it also has a model where the probability distribution assigns the same probability to each world that has a probability greater than 0. This allows to compute canonical models with uniform probability.

In the completion algorithm completion sets are introduced for several worlds. Intuitively, there are three kinds of worlds to be distinguished. One world represents 'almost certain' facts that hold with probability 1. Another one represents the world that has probability 0, which is used to compute the subsumption relationships between classical (non-probabilistic) concepts. The third kind are worlds for each concept name A that appears in a concept of the form $P_{>0}A$. These worlds 'give evidence' in the model that A is not impossible.

4.1 Generalizations in Prob-$\mathcal{EL}_c^{01}$

As in $\mathcal{EL}$, neither the LCS nor the MSC need to exist in Prob-$\mathcal{EL}_c^{01}$, if considered w.r.t. cyclic TBoxes or acyclic ABoxes, respectively. A computation algorithm for the k-LCS in Prob-$\mathcal{EL}_c^{01}$ was described in [69]. It turns out that the cross-product construction needs to be carried out separately for the three kinds of worlds. To obtain crisp existential restrictions, existential restrictions in the world representing probability 0 need to be combined. While for concepts of the form $P_{=1}\exists r.C$ existential restrictions from the 'almost certain world', i.e. the world with probability 1 need to be combined. To obtain concepts of the form $P_{>0}\exists r.C$, the cross-product construction is carried out over the remaining worlds, i.e. those with probability greater 0.

The completion algorithm for ABox realization in Prob-$\mathcal{EL}_c^{01}$ from [59] is the basis for the computation algorithm for the k-MSC presented in [68]. Here, we extended the method from Section 3.2 to probabilistic concepts. Essentially, the MSC in Prob-$\mathcal{EL}_c^{01}$ can be constructed from the completion sets similarly as the k-LCS, by distinguishing the three kinds of worlds when constructing existential restrictions.

[8] The completion algorithm turned out to be incomplete recently, the missing completion rule can be found in [30].

Similarly as for subsumption or instance checking, the computational complexity for the computation role-depth bounded LCS or MSC does not change when going from $\mathcal{EL}$ to Prob-$\mathcal{EL}_c^{01}$. The resulting LCS-concept can be exponential in the size of the TBox [68]. Note, that our approximative methods yield the exact generalizations, if the TBox (/the KB $\mathcal{K}$) is unfoldable and the k is sufficiently large.

The k-LCS has been extended to the use of nominals in the probabilistic concepts in the DL Prob-$\mathcal{ELO}_c^{01}$ [30], which required to extend the completion algorithm for Prob-$\mathcal{EL}_c^{01}$ by the propagation of concept non-emptiness. Again, the complexities for the standard inferences as well as for the generalization inferences do not change in the presence of nominals.

4.2 Computing Explanations in Prob-$\mathcal{EL}_c^{01}$

Since the LCS and MSC inference may return results that are unintuitive to users—which might easily be the case when obtained for DLs with probabilities—computing explanations automatically is a useful service. In case an unintuitive concept is returned by the MSC, the set of axioms and assertions causing the instance relationship between the input individual and this concept to hold can be returned as an explanation. In [68] we extended the method for computing explanations for subsumption relationships in $\mathcal{EL}$ from [12] to explaining subsumption and instance relationships in Prob-$\mathcal{EL}_c^{01}$. The approach in [12] computes all the *minimal axiom sets* (MinAs) causing such a relationship to hold. It labels each axiom with a propositional variable and each element of the completion sets gets labelled with a monotone Boolean formula. The idea is that these monotone Boolean formulas get combined during completion. If completion set elements together with the axiom cause a completion rule to 'fire', then their corresponding formulas form a conjunction together with the propositional variable for the axiom. If the consequence was new, it gets the conjunction as a label. In case the consequence was already derived, the new conjunction forms a disjunction with the old label of the consequence, unless the new conjunction is redundant to the old one. The resulting formula in the label of the consequence that is to be explained, transformed into DNF collects in each disjunct which axioms need to be combined to derive the consequence.

This approach was transferred to the completion algorithm for realization of Prob-$\mathcal{EL}_c^{01}$-ABoxes in [68] by labelling the assertions also. There can be exponentially many models of a monotone Boolean formula, i.e. exponentially many different conjunctions in the DNF of the formula. Thus there can be exponentially many MinAs for one consequence in Prob-$\mathcal{EL}_c^{01}$ and the modified completion algorithm can run in ExpTime—as it is also the case for $\mathcal{EL}$.

5 Computing Exact Generalizations in $\mathcal{EL}$

In [86] we investigated conditions under which the LCS or the MSC exist in $\mathcal{EL}$, if computed w.r.t. cyclic TBoxes or KBs. In both cases we can follow the same approach:

1. Identify a set of candidate concepts of increasing role-depth for the exact generalization,
2. find a characterization for the concept obtained by the exact generalization, and
3. find an upper bound for the role-depth of the exact generalization.

Testing whether the set of candidates with role-depth up to the upper bound contains a concept that fulfills the characterization, decides whether the exact generalization exists. If such an element is in the set, then this element is the exact generalization.

5.1 Conditions for the Existence of the LCS w.r.t. General $\mathcal{EL}$-TBoxes

In case the LCS exists for a given $\mathcal{EL}$-TBox and two given $\mathcal{EL}$-concepts C and D, then there must be an $\ell \in \mathbb{N}$ s.t. the k-$lcs_{\mathcal{T}}(C, D)$ up to role-depth ℓ is the $lcs_{\mathcal{T}}(C, D)$. Thus the sequence of k-LCSs with increasing role-depth is a set of *LCS candidates*. This sequence can be computed from pointed models, which are pairs of an element and a canonical interpretation (Def. 6), by building the characteristic concept from them. The *characteristic concept* of the pointed (canonical) model $(\mathcal{I}, d)$ is the concept that is a conjunction of (1) all concept names that hold for d in $\mathcal{I}$ and (2) all existential restrictions built from each role for which d has a role-successor in $\mathcal{I}$ and the characteristic concept of that role-successor in $\mathcal{I}$. In detail the sequence of LCSs can be computed from the pointed canonical models $(\mathcal{I}_{C,\mathcal{T}}, d_C)$ and $(\mathcal{I}_{D,\mathcal{T}}, d_D)$ using simulations (Def. 7) by:

1. taking their cross-product $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$,
2. building its *tree unraveling* starting from (d_C, d_D) up to path-length k, yielding a new interpretation, and
3. building from this interpretation the k-characteristic concept of (d_C, d_D).

The characteristic concept of the tree unraveling up to paths of length k of the cross-product of the (pointed) canonical models $X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ yields the k-LCS of C and D w.r.t. $\mathcal{T}$ and thus a common subsumer. We showed the following characterization of the LCS in [86]:

$$E \equiv_{\mathcal{T}} lcs_{\mathcal{T}}(C, D) \text{ iff } (\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \lesssim (\mathcal{I}_{E,\mathcal{T}}, d_E) \text{ and } (\mathcal{I}_{E,\mathcal{T}}, d_E) \lesssim (\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)). \quad (1)$$

The existence of (maximal) simulations between finite trees can be decided in polynomial time [39]. This test allows us to check for an obtained k-LCS whether it is the LCS.

Furthermore, we have shown an upper bound on the role-depth of the LCS of two $\mathcal{EL}$-concepts w.r.t. to a general $\mathcal{EL}$-TBox, if it exists. In that case the role-depth of the LCS is quadratic in the size of the product model.

Together with the characterization of the LCS (1), this bound allows to test for the existence of the LCS for a given TBox and a pair of concepts by the following steps: compute the size of the product of the pointed canonical models, obtain the bound and test for k-LCSs up to this bound whether the simulation required by (1) exists.[9]

Using the above techniques, we can show that it is decidable in polynomial time whether the LCS of a given pair of concepts w.r.t. a given TBox exists.

5.2 Conditions for the Existence of the MSC w.r.t. Cyclic $\mathcal{EL}$-ABoxes

Following the same approach as for the LCS, we showed in [86] conditions under which the MSC does exist—even if computed w.r.t. cyclic ABoxes. Assume we want to compute the $msc_{\mathcal{K}}(a)$ of individual a w.r.t. the KB $\mathcal{K}$, then the sequence of k-MSCs for an increasing role-depth bound k is a set of candidates for this MSC. The k-MSC can be computed from the pointed canonical model of the KB $(\mathcal{I}_{\mathcal{K}}, d_a)$ by computing the tree unraveling of path length up to k starting from d_a and then building the k-characteristic concept $X^k(\mathcal{I}_{\mathcal{K}}, d_a)$ of it. We showed the following characterization of the MSC:

$$C \equiv_{\mathcal{T}} msc_{\mathcal{K}}(a) \text{ iff } (\mathcal{I}_{\mathcal{K}}, d_a) \lesssim (\mathcal{I}_{C,\mathcal{T}}, d_C) \text{ and } (\mathcal{I}_{C,\mathcal{T}}, d_C) \lesssim (\mathcal{I}_{\mathcal{K}}, d_a). \tag{2}$$

Furthermore, we have shown an upper bound on the role-depth of the MSC of an individual w.r.t. an $\mathcal{EL}$-KB with a cyclic $\mathcal{EL}$-ABox, if it exists. In that case the role-depth of the MSC is quadratic in the size of the canonical model. Together with (2) this yields that it is decidable in polynomial time whether the MSC of a given individual w.r.t. a given KB exists.

The same upper bounds for the role-depth of the LCS and of the MSC as obtained for $\mathcal{EL}$ hold, if the TBox and the KB, respectively, use role hierarchies [85].

[9] The result in [86] on the size of the LCS (and the MSC) needs to be corrected: in the worst case both of them can grow exponentially in the size of the TBox/KB.

6 Concept Similarity and Relaxed Instance Queries

The notion of concept similarity is widely used in different application domains for ontologies. For example, for the GENE ONTOLOGY the similarity of concepts describing genes is used to assess whether these genes realize similar functionalities [75, 63]. The similarity of concepts is assessed by similarity measures. A *concept similarity measure* (CSM) is a function that returns a value between 0 and 1 for a pair of (complex) concepts. These measures play a central role in ontology alignment, where the task is to find the corresponding concepts from different ontologies.

6.1 A Framework for $\mathcal{EL}$-concept Similarity Measures

Despite its popular use, the notion of concept similarity is not formally defined and may even depend on the context or application. Many concept similarity measures have been devised in the literature [44, 63, 1] for different purposes. However, there are some properties that should be required to hold for a well-founded similarity measure for concepts. In [55] we list properties for CSMs that are well-established from the literature or that match the intuition for similarity measures.

Among other properties, we require that a CSM regards the semantics rather than the syntactic form of the concepts to assess. A CSM $\sim$ is *equivalence invariant* iff for two concepts C_1 and C_2, $C_1 \equiv C_2$ implies that $C_1 \sim D = C_2 \sim D$ for any concept D. This property can be achieved by transforming the input concepts into a unique normal form. Other properties of a CSM are those of a metric, namely, symmetry, triangle inequality (for the dual notion, i.e. for the distance of concepts) and identity of the indiscernible. The (partial) order induced by the subsumption relationships is reflected by the property *subsumption preserving*, which requires that if $C \sqsubseteq D \sqsubseteq E$, then $C \sim D \geq C \sim E$ holds. Our framework *simi* for concept similarity measures for $\mathcal{EL}$-concepts and role hierarchies allows to construct measures, which have most of the properties discussed in [55].[10]

An individual CSM can be obtained by parameterizing *simi* by

- a *primitive measure* which allows to set the similarity of pairs of concept names or pairs of roles, and
- several operators,
- a *weighting function* g that sets a weight for concept names and existential restrictions and thus allows to highlight thematic sub-domains of the domain of discourse.

[10] The triangular inequality could not be achieved in *simi*.

The measures obtained by the framework can be computed in polynomial time in the size of the input concepts, if the functions employed in the measure can be computed in polynomial time.

Although defined only for concepts, the similarity measure framework allows to construct measures that can be applied to concepts defined in unfoldable TBoxes. In such a case the concepts need to be expanded w.r.t. the TBox before applying the measure. Similarity measures constructed by this framework can be utilized to realize new reasoning services.

6.2 Towards Instance Queries for Concepts Relaxed by Similarity Measures

Instance queries allow to query ABoxes by using concepts as a query. In case such a query does not return any individual from the ABox, feasible alternatives can be retrieved by relaxing the query concept C_q, i.e., retrieve individuals that 'come close' to those that fulfill the specification C_q.

The appropriate notion of similarity to be used may vary for different applications or even for different queries within one application. Some aspects of the query concept need to be kept in a proposed alternative, while for others a variation is acceptable.

In [31] we propose to employ CSMs to relax query concepts. The idea is that for a query concept C_q all those individuals are retrieved that are instances of 'sufficiently similar' concepts to C_q. More precisely, we define the following new reasoning service:

Definition 8 (Relaxed instance, relaxed instance retrieval)**.** *Let $\mathcal{L}$ be some DL, C_q be an $\mathcal{L}$-concept, $\sim$ a CSM over $\mathcal{L}$-concepts, and $t \in (0, 1]$ a degree. The individual $a \in \mathsf{N_I}$ is a* relaxed instance *of C_q w.r.t. the $\mathcal{L}$-KB $\mathcal{K}$, $\sim$ and the threshold t, denoted $a \in_t^{\sim} C_q$, iff there exists an $\mathcal{L}$-concept X s.t. $C_q \sim X \geq t$ and $\mathcal{K} \models X(a)$.*

Relaxed instance retrieval *is to compute the set of all individuals that are relaxed instances of C_q w.r.t. the $\mathcal{L}$-KB $\mathcal{K}$, $\sim$ and the threshold t.*

This approach allows flexibility w.r.t. the degree of similarity and w.r.t. the notion of similarity by choosing an appropriate degree t and a CSM $\sim$, respectively. An illustration for the different relaxed instances obtained for different CSMs w.r.t. the same C_q is given in Figure 1.

We investigated this inference for $\mathcal{EL}$-KBs with unfoldable TBoxes in [31] and identified properties of CSMs that allow to compute relaxed instances in our approach. Now, to compute the relaxed instances of an $\mathcal{EL}$-concept (w.r.t. an $\mathcal{EL}$-KB) it is not feasible to compute all sufficiently similar concepts and then perform instance checking for those, since (1) the number of those concepts can be infinite leading to an infinite number of queries and (2) a similarity measure does not provide a method how to obtain a 'sufficiently similar' concept.

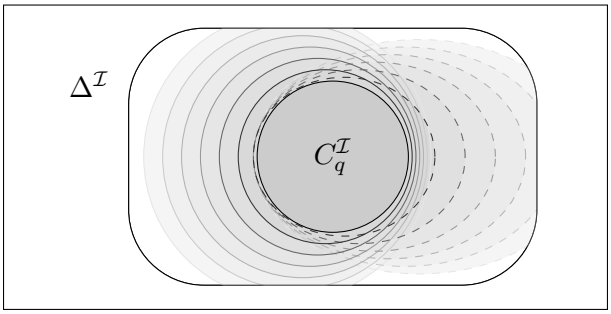

Figure 1: Relaxed instances w.r.t. two different similarity measures. Darker colors represent the relaxed instances of C_q w.r.t. higher degrees t.

Instead we proceed by computing, for each individual a in the ABox, a concept that has the individual a as an instance and resembles C_q most w.r.t. the CSM $\sim$. We call this the *mimic* of C_q w.r.t. a and $\sim$ (denoted $\mathfrak{M}(C_q, a)$). Now, if $\mathfrak{M}(C_q, a) \sim C_q \geq t$ holds, then a is a relaxed instance of C_q; otherwise, it cannot be a relaxed instance, as no concept can have a greater similarity degree with C_q while still containing a. In [31] we give an algorithm for computing mimics in $\mathcal{EL}$. The idea is to compute the k-MSC of a with the role-depth of C_q as the role-depth bound, and then remove sub-concepts from the resulting concept to make it more similar to C_q. This approach requires that the CSM $\sim$ is symmetric, equivalence invariant, structural, i.e., it computes the similarity by induction on the structure of concepts, and is monotone in the sense that, for $N \subseteq \mathsf{N_C}$:

$$\big(X \ \sim \bigsqcap_{A \in N} A\big) \ \geq \ \big(X \sqcap \exists r.B \ \sim \bigsqcap_{A \in N} A\big).$$

Deciding whether an individual is a relaxed instance of a concept (w.r.t. a similarity measure and a degree t) can be done in NExpTime in $k = \mathit{role\text{-}depth}(C_q)$ (provided that $\sim$ can be computed in NExpTime). However, the algorithm runs in NP-time in the size of $\mathcal{K}$, if $\sim$ can be computed in NP. Tight lower bounds for the decision problem whether an individual is a relaxed instance remain future work.

7 Conclusions and Outlook

In this thesis we have presented results on how to realize situation recognition for context-aware systems by means of standard DL reasoning. We have supplied studies on realizing this approach by reasoning systems for OWL 1 and OWL 2, which demonstrated the feasibility of our approach. However, the ontology languages from the OWL standard do lack expressive means needed in this kind of application, such as for uncertain or temporal information, for example.

The computation of generalizations can play an important role for building and maintenance of DL TBoxes. We have investigated an approximative method to compute the LCS and the MSC for members of the $\mathcal{EL}$-family in the presence of cycles in the TBox and the ABox, respectively. Our algorithms for computing generalizations are based on the completion method, which is the reasoning method for standard reasoning tasks and implemented in DL reasoners. For generalizations, nominals may tend to be too restrictive since the LCS only returns an $\mathcal{ELO}$-concept, if both concepts use the same individual. Here, the use of variables, similarly as in nominal schemas, instead of individuals could be useful.

We also devised algorithms for computing generalizations and explanations for $\mathcal{EL}$ augmented with (restricted) subjective probabilities. We conjecture that our methods can be extended to Prob-$\mathcal{EL}_c$ (without restricting the used probabilities).

For the LCS and the MSC in $\mathcal{EL}$ computed w.r.t. cyclic KBs we showed exact conditions for their existence. It requires an empirical evaluation to see whether there are $\mathcal{EL}$ KBs from applications that are cyclic and where the LCS exists for some concepts or where the MSC exists for some individuals. On the theoretical side, we would like to devise such existence conditions for more expressive variants of $\mathcal{EL}$. Similar existence conditions based on canonical models for Horn-DLs can be investigated.

The framework for concept similarity measures allows to construct such measures with certain properties. An interesting extension is to devise such a framework for more expressive DLs or for $\mathcal{EL}$-concepts defined w.r.t. general TBoxes. Here, the use of canonical models appear to be a promising idea.

The new inference services relaxed instance retrieval and the computation of mimics open the road for several follow-up questions. A natural task would be to find the exact complexity bounds for the relaxed instance retrieval in $\mathcal{EL}$. One could also investigate relaxed versions of other standard inferences, such as relaxed subsumption or query answering. As for the concept similarity measures, the extension of the two inferences to general TBoxes is interesting future work.

References

[1] T. Alsubait, B. Parsia, and U. Sattler. A similarity-based theory of controlling mcq difficulty. In J. Stando and Y. Imai, editors, *Proceedings of Second International Conference on e-Learning and e-Technologies in Education (ICEEE)*, pages 283–288, 2013.

[2] F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 325–330. Morgan Kaufmann, 2003.

[3] F. Baader. Terminological cycles in a description logic with existential restrictions. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 319–324. Morgan Kaufmann, 2003.

[4] F. Baader, A. Bauer, P. Baumgartner, A. Cregan, A. Gabaldon, K. Ji, K. Lee, D. Rajaratnam, and R. Schwitter. A novel architecture for situation awareness systems. In M. Giese and A. Waaler, editors, *Proceedings of the 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux 2009)*, volume 5607 of *Lecture Notes in Computer Science*, pages 77–92. Springer-Verlag, 2009.

[5] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.

[6] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In K. Clark and P. F. Patel-Schneider, editors, *In Proc. of the OWLED Workshop*, 2008.

[7] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[8] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann, Los Altos.

[9] F. Baader and C. Lutz. Description logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *The Handbook of Modal Logic*, pages 757–820. Elsevier, 2006.

[10] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, volume 4130 of *Lecture Notes In Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006. CEL download page: `http://lat.inf.tu-dresden.de/systems/cel/`.

[11] F. Baader, C. Lutz, and A.-Y. Turhan. Small is again beautiful in description logics. *KI*, 24(1):25–33, 2010.

[12] F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of the 30th German Annual Conf. on Artificial Intelligence (KI'07)*, volume 4667 of *Lecture Notes In Artificial Intelligence*, pages 52–67, Osnabrück, Germany, 2007. Springer.

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

[14] F. Bobillo, M. Delgado, J. Gómez-Romero, and U. Straccia. Fuzzy description logics under Gödel semantics. *International Journal of Approximate Reasoning*, 50(3):494–514, 2009.

[15] F. Bobillo and U. Straccia. Fuzzy description logics with general t-norms and datatypes. *Fuzzy Sets and Systems*, 160(23):3382–3402, 2009.

[16] S. Borgwardt and R. Peñaloza. Undecidability of fuzzy description logics. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-12)*, pages 232–242. AAAI Press, 2012.

[17] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. L. de Mantáras and L. Saitta, editors, *Proc. of the 16th European Conf. on Artificial Intelligence (ECAI-04)*, pages 298–302. IOS Press, 2004.

[18] S. Brandt. *Standard and Non-standard Reasoning in Description Logics*. PhD thesis, Institute for Theoretical Computer Science, TU Dresden, January 2006.

[19] S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In G. Görz, V. Haarslev, C. Lutz, and R. Möller, editors, *Proc. of the 2001 Applications of Description Logic Workshop (ADL 2001)*, number 44 in CEUR Workshop, Vienna, Austria, September 2001. RWTH Aachen. See `http://CEUR-WS.org/Vol-44/`.

[20] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In M. M. Veloso and S. Kambhampati, editors, *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05)*, pages 602–607. AAAI Press/The MIT Press, 2005.

[21] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[22] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Vol. 2*, pages 273–286. USENIX Association, 2005.

[23] W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, editor, *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI-92)*, pages 754–760, San Jose, CA, 1992. AAAI Press/The MIT Press.

[24] W. W. Cohen and H. 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, Bonn, 1994. Morgan Kaufmann, Los Altos.

[25] T. G. O. Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[26] C. d'Amato, N. Fanizzi, F. Esposito, and T. Lukasiewicz. Representing uncertain concepts in rough description logics via contextual indiscernibility relations. In F. Bobillo, P. C. Costa, C. d'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool, editors, *Uncertainty Reasoning for the Semantic Web II, International Workshops URSW 2008-2010 and UniDL 2010, Revised Selected Papers*, volume 7123 of *LNCS*, pages 300–314. Springer, 2013.

[27] W. Dargie, Eldora, J. Mendez, C. Möbius, K. Rybina, V. Thost, and A.-Y. Turhan. Situation recognition for service management systems using OWL 2 reasoners. In D. Nicklas, D. Riboni, and M. Wieland, editors, *In Proceedings of the Context Modeling and Reasoning Workshop (CoMoRea'13) collocated with IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 31–36, 2013.

[28] W. Dargie, A. Strunk, and A. Schill. Energy-aware service execution. In C. T. Chou, T. Pfeifer, and A. P. Jayasumana, editors, *IEEE 36th*

Conference on Local Computer Networks, LCN 2011, Bonn, Germany, October 4-7, 2011, pages 1064–1071. IEEE, 2011.

[29] A. Ecke, R. Peñaloza, and A.-Y. Turhan. Computing role-depth bounded generalizations in the description logic $\mathcal{ELOR}$. In I. J. Timm and M. Thimm, editors, *Proceedings of the 36th German Conference on Artificial Intelligence (KI 2013)*, volume 8077 of *Lecture Notes in Artificial Intelligence*, pages 49–60, Koblenz, Germany, 2013. Springer-Verlag. extended version: `http://lat.inf.tu-dresden.de/research/papers/2013/EcPeTu-KI-13.long.pdf`.

[30] A. Ecke, R. Peñaloza, and A.-Y. Turhan. Role-depth bounded least common subsumer in Prob-$\mathcal{EL}$ with nominals. In T. Eiter, B. Glimm, Y. Kazakov, and M. Krötzsch, editors, *Informal Proceedings of the 26th International Workshop on Description Logics (DL-2013)*, volume 1014 of *CEUR-WS*, pages 670–688, Ulm, Germany, 2013.

[31] A. Ecke, R. Peñaloza, and A.-Y. Turhan. Towards instance query answering for concepts relaxed by similarity measures. In L. Godo, H. Prade, and G. Qi, editors, *Workshop on Weighted Logics for AI (in conjunction with IJCAI'13)*, Beijing, China, 2013. An extended version is invited to a special issue on the Workshop Weighted Logics for AI of the Journal of Applied Logic.

[32] A. Ecke and A.-Y. Turhan. Optimizations for the role-depth bounded least common subsumer in $\mathcal{EL}+$. In P. Klinov and M. Horridge, editors, *Proceedings of the OWL Experiences and Directions Workshop (OWLed'12)*, volume 849 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[33] A. Ecke and A.-Y. Turhan. Role-depth bounded least common subsumers for $\mathcal{EL}+$ and $\mathcal{ELI}$. In Y. Kazakov, D. Lembo, and F. Wolter, editors, *Proceedings of the 2012 International Workshop on Description Logics*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[34] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In M. M. Veloso, editor, *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, pages 399–404, Hyderabad, India, 2007.

[35] S. Götz, J. Mendez, V. Thost, and A.-Y. Turhan. OWL 2 Reasoning To Detect Energy-Efficient Software Variants From Context. In *Proceedings of the OWL Experiences and Directions Workshop (OWLed'13)*, 2013.

[36] V. Gutiérrez-Basulto, J. C. Jung, C. Lutz, and L. Schröder. A closer look at the probabilistic description logic Prob-$\mathcal{EL}$. In *Proceedings of Twenty-Fifth Conference on Artificial Intelligence (AAAI-11)*, 2011.

[37] V. Haarslev, K. Hidde, R. Möller, and M. Wessel. The racerpro knowledge representation and reasoning system. *Semantic Web Journal*, 3(3):267–277, 2012.

[38] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

[39] M. Henzinger, T. Henzinger, and P. Kopke. Computing simulations on finite and infinite graphs. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:453, 1995.

[40] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-06)*, pages 57–67. AAAI Press, 2006.

[41] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.

[42] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*. Morgan Kaufmann, Jan. 2005.

[43] I. Horrocks and U. Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *J. of Automated Reasoning*, 39(3):249–276, 2007.

[44] K. Janowicz. Sim-dl: Towards a semantic similarity measurement theory for the description logic $\mathcal{ALCNR}$ in geographic information retrieval. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2006 (OTM 2006)*, volume 4278 of *Lecture Notes in Computer Science*, pages 1681–1692. Springer, 2006.

[45] Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than shoiq. In G. Brewka and J. Lang, editors, *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-08)*, pages 274–284. AAAI Press, 2008.

[46] Y. Kazakov, M. Krötzsch, and F. Simančík. ELK reasoner: Architecture and evaluation. In I. Horrocks, M. Yatskevich, and E. Jimenez-Ruiz, editors, *Proceedings of the OWL Reasoner Evaluation Workshop (ORE'12)*, volume 858 of *CEUR Workshop*. CEUR-WS.org, 2012.

[47] Y. Kazakov, M. Krötzsch, and F. Simančík. Practical reasoning with nominals in the $\mathcal{EL}$ family of description logics. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-12)*, pages 264–274. AAAI Press, 2012.

[48] C. M. Keet. Rough subsumption reasoning with rowl. In I. Brown, K. Sewchurran, and H. Suleman, editors, *Proc. of the 2011 Annual Conf. of the South African Inst. of Comp. Scientists and Inform. Tech. (SAICSIT 2011)*, pages 133–140. ACM, 2011.

[49] P. Klinov and B. Parsia. Pronto: A practical probabilistic description logic reasoner. In F. Bobillo, P. C. da Costa, C. d'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool, editors, *Uncertainty Reasoning for the Semantic Web II, International Workshops URSW 2008-2010 and UniDL 2010, Revised Selected Papers*, volume 7123 of *Lecture Notes in Computer Science*, pages 59–79. Springer, 2013.

[50] A. Krisnadhi and C. Lutz. Data complexity in the el family of dls. In D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A.-Y. Turhan, and S. Tessaris, editors, *Proc. of the 2007 Description Logic Workshop (DL 2007)*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[51] M. Krötzsch and S. Rudolph. Conjunctive queries for el with composition of roles. In D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A.-Y. Turhan, and S. Tessaris, editors, *Proc. of the 2007 Description Logic Workshop (DL 2007)*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[52] R. Küsters and R. Molitor. Computing Least Common Subsumers in $\mathcal{ALEN}$. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI-01)*, pages 219–224. Morgan Kaufman, 2001.

[53] R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. *AI Communications*, 15(1):47–59, 2002.

[54] M. Lawley and C. Bousquet. Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In T. Meyer, M. Orgun, and K. Taylor, editors, *Australasian Ontology Workshop 2010 (AOW 2010): Advances in Ontologies*, volume 122 of *CRPIT*, pages 45–50, Adelaide, Australia, 2010. ACS.

[55] K. Lehmann and A.-Y. Turhan. A framework for semantic-based similarity measures for $\mathcal{EL}\mathcal{H}$-concepts. In L. F. del Cerro, A. Herzig, and J. Mengin, editors, *Proceedings of the 13th European Conference on Logics in Artificial Intelligence, (JELIA'12)*, volume 7519 of *Lecture Notes in Computer Science*, pages 307–319. Springer, 2012.

[56] T. Lukasiewicz. Uncertainty reasoning for the semantic web. In A. Polleres and T. Swift, editors, *Web Reasoning and Rule Systems, Third International Conference, RR 2009, Chantilly, VA, USA, October 25-26, 2009, Proceedings*, volume 5837 of *Lecture Notes in Computer Science*, pages 26–39. Springer, 2009.

[57] C. Lutz. The complexity of conjunctive query answering in expressive description logics. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR2008)*, number 5195 in LNAI, pages 179–193. Springer, 2008.

[58] C. Lutz, R. Piro, and F. Wolter. $\mathcal{EL}$-concepts go second-order: Greatest fixpoints and simulation quantifiers. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proc. of the 19th European Conf. on Artificial Intelligence (ECAI-10)*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 41–46. IOS Press, 2010.

[59] C. Lutz and L. Schröder. Probabilistic description logics for subjective uncertainty. In F. Lin, U. Sattler, and M. Truszczynski, editors, *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-10)*. AAAI Press, 2010.

[60] C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation*, 45(2):194 – 228, 2010.

[61] J. Mendez. jCel: A modular rule-based reasoner. In *In Proc. of the 1st Int. Workshop on OWL Reasoner Evaluation (ORE'12)*, volume 858 of *CEUR*, 2012.

[62] J. Mendez, A. Ecke, and A.-Y. Turhan. Implementing completion-based inferences for the $\mathcal{EL}$-family. In R. Rosati, S. Rudolph, and M. Zakharyaschev, editors, *Proc. of the 2011 Description Logic Workshop (DL 2011)*, volume 745. CEUR Workshop, 2011.

[63] M. Mistry and P. Pavlidis. Gene ontology term overlap as a measure of gene functional similarity. *BMC Bioinformatics*, 9, 2008.

[64] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 web ontology language profiles. W3C Rec-

ommendation, 27 October 2009. `http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/`.

[65] B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In *Proc. of the 23th Conf. on Automated Deduction (CADE-23)*, LNAI, pages 67–83, Bremen, Germany, July 17–20 2007. Springer.

[66] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence Journal*, 43:235–249, 1990.

[67] M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008.

[68] R. Peñaloza and A.-Y. Turhan. Instance-based non-standard inferences in $\mathcal{EL}$ with subjective probabilities. In F. Bobillo, P. C. Costa, C. d'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool, editors, *Uncertainty Reasoning for the Semantic Web II, International Workshops URSW 2008-2010 and UniDL 2010, Revised Selected Papers*, number 7123 in Lecture Notes in Computer Science, pages 80–98. Springer-Verlag, 2013.

[69] R. Peñaloza and A.-Y. Turhan. Role-depth bounded least common subsumers by completion for $\mathcal{EL}$- and Prob-$\mathcal{EL}$-TBoxes. In V. Haarslev, D. Toman, and G. E. Weddell, editors, *Proceedings of the 23rd International Workshop on Description Logics (DL'10)*, volume 573 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[70] R. Peñaloza and A.-Y. Turhan. A practical approach for computing generalization inferences in $\mathcal{EL}$. In G. Antoniou, M. Grobelnik, E. P. B. Simperl, B. Parsia, D. Plexousakis, P. D. Leenheer, and J. Z. Pan, editors, *Proceedings of 8th Extended Semantic Web Conference (ESWC'11)*, volume 6643 of *Lecture Notes in Computer Science*, pages 410–423. Springer, 2011.

[71] M. Rodriguez-Muro and D. Calvanese. Quest, an OWL 2 QL reasoner for ontology-based data access. In *Proc. of the 9th Int. Workshop on OWL: Experiences and Directions (OWLED 2012)*, volume 849 of *CEUR*, 2012.

[72] R. Rosati. On conjunctive query answering in el. In D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A.-Y. Turhan, and S. Tessaris, editors, *Proc. of the 2007 Description Logic Workshop (DL 2007)*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[73] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.

[74] K. Schild. A correspondence theory for terminological logics: preliminary report. In J. Mylopoulos and R. Reiter, editors, *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, 1991.

[75] A. Schlicker, F. S. Domingues, J. Rahnenführer, and T. Lengauer. A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinformatics*, 7:302, 2006.

[76] S. Schlobach, M. C. A. Klein, and L. Peelen. Description logics with approximate definitions - precise modeling of vague concepts. In M. M. Veloso, editor, *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, pages 557–562, 2007.

[77] E. Sirin and B. Parsia. Pellet system description. In *Description Logics*, volume 189 of *CEUR*, 2006.

[78] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Assoc.*, 2000. Fall Symposium Special Issue.

[79] T. Springer and A.-Y. Turhan. Employing description logics in ambient intelligence for modeling and reasoning about complex situations. *Journal of Ambient Intelligence and Smart Environments*, 1(3):235–259, 2009.

[80] K. Taylor and L. Leidinger. Ontology-driven complex event processing in heterogeneous sensor networks. In G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. D. Leenheer, and J. Z. Pan, editors, *Proceedings of 8th Extended Semantic Web Conference (ESWC 2011)*, volume 6644 of *Lecture Notes in Computer Science*, pages 285–299. Springer, 2011.

[81] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, May 2000.

[82] D. Tsarkov, I. Horrocks, and P. Patel-Schneider. Optimising terminological reasoning for expressive description logics. *Journal of Automated Reasoning*, 2007.

[83] A.-Y. Turhan. Reasoning and explanation in $\mathcal{EL}$ and in expressive description logics. In U. Aßmann, A. Bartho, and C. Wende, editors, *Reasoning Web. Semantic Technologies for Software Engineering, 6th International Summer School 2010, Dresden, Germany. Tutorial Lectures*, volume 6325 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2010.

[84] A.-Y. Turhan, T. Springer, and M. Berger. Pushing doors for modeling contexts with OWL DL – a case study. In J. Indulska and D. Nicklas, editors, *Proceedings of the Workshop on Context Modeling and Reasoning (CoMoRea'06)*. IEEE Computer Society, March 2006.

[85] A.-Y. Turhan and B. Zarrieß. Computing the lcs w.r.t. general $\mathcal{EL}^+$-TBoxes. In T. Eiter, B. Glimm, Y. Kazakov, and M. Krötzsch, editors, *Informal Proceedings of the 26th International Workshop on Description Logics*, volume 1014 of *CEUR Workshop Proceedings*, pages 477–488. CEUR-WS.org, 2013.

[86] B. Zarrieß and A.-Y. Turhan. Most Specific Generalizations w.r.t. General $\mathcal{EL}$-TBoxes. In F. Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, 2013.

Appendix

the submitted papers in chronological order

Künstl Intell (2010) 24: 25–33
DOI 10.1007/s13218-010-0004-8

FACHBEITRAG

Small is Again Beautiful in Description Logics

Franz Baader · Carsten Lutz · Anni-Yasmin Turhan

Received: 24 June 2009 / Accepted: 10 August 2009 / Published online: 9 February 2010

Abstract The Description Logic (DL) research of the last 20 years was mainly concerned with increasing the expressive power of the employed description language without losing the ability of implementing highly-optimized reasoning systems that behave well in practice, in spite of the ever increasing worst-case complexity of the underlying inference problems. OWL DL, the standard ontology language for the Semantic Web, is based on such an expressive DL for which reasoning is highly intractable. Its sublanguage OWL Lite was intended to provide a tractable version of OWL, but turned out to be only of a slightly lower worst-case complexity than OWL DL. This and other reasons have led to the development of two new families of light-weight DLs, $\mathcal{EL}$ and DL-Lite, which recently have been proposed as profiles of OWL 2, the new version of the OWL standard. In this paper, we give an introduction to these new logics, explaining the rationales behind their design.

Keywords Knowledge representation · Description logics · Automated reasoning

F. Baader (✉) · A.-Y. Turhan
Institut für Theoretische Informatik, TU Dresden, 01062 Dresden, Deutschland
e-mail: baader@inf.tu-dresden.de

A.-Y. Turhan (✉)
e-mail: turhan@inf.tu-dresden.de

C. Lutz
Universität Bremen, Fachbereich 03, Postfach 330440, 28334 Bremen, Deutschland
e-mail: clu@informatik.uni-bremen.de

1 Introduction

Description Logics [8] are a well-investigated family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration, and databases, but their most notable success so far is the adoption of the DL-based language OWL[1] as a standard ontology language for the Semantic Web [11, 33].

In DLs, concepts are formally described by *concept descriptions*, i.e., expressions that are built from concept names (unary predicates) and role names (binary predicates) using concept constructors. The expressivity of a particular DL is determined by which concept constructors are available in it. From a semantic point of view, concept names and concept descriptions represent sets of individuals, whereas roles represent binary relations between individuals. For example, using the concept name *Woman*, and the role name *child*, the concept of *women having a daughter* can be represented by the concept description

$$\mathit{Woman} \sqcap \exists \mathit{child}.\mathit{Woman},$$

and the concept of *women having only daughters* by

$$\mathit{Woman} \sqcap \forall \mathit{child}.\mathit{Woman}.$$

In its simplest form, a DL *terminology* (usually called *TBox*) can be used to introduce abbreviations for complex concept descriptions. For example, the *concept definitions*

$$\mathit{Woman} \equiv \mathit{Human} \sqcap \mathit{Female},$$
$$\mathit{Mother} \equiv \mathit{Woman} \sqcap \exists \mathit{child}.\top$$

[1] http://www.w3.org/TR/owl-features/.

define the concept of a woman as a human that is female, and the concept of a mother as a woman that has a child, where $\top$ stands for the top concept (which is interpreted as the universe of all individuals in the application domain). So-called *general concept inclusions (GCIs)* can be used to state additional constraints on the interpretation of concepts and roles. In our example, it makes sense to state domain and range restrictions for the role *child*. The GCIs

$$\exists child.Human \sqsubseteq Human$$
$$Human \sqsubseteq \forall child.Human$$

respectively say that only human beings can have human children, and that the child of a human being must be human.

In the *assertional part (ABox)* of a DL knowledge base, facts about a specific application situation can be stated, by introducing named individuals and relating them to concepts and roles. For example, the assertions

$$Woman(LINDA), \qquad child(LINDA, JAMES)$$

state that Linda is a woman, who has the child James.

Knowledge representation systems based on DLs provide their users with various inference services that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the *subsumption* service allows one to determine subconcept-superconcept relationships. For example, w.r.t. the concept definitions from above, the concept *Female* subsumes the concept *Mother* since all instances of the second concept are necessarily instances of the first concept, i.e., whenever the above concept definitions are satisfied, then *Mother* is interpreted as a subset of *Female*. With the help of the subsumption service, one can compute the hierarchy of all concepts defined in a TBox. This compound inference service is usually called *classification*. The *instance* service can be used to check whether an individual occurring in an ABox is necessarily an instance of a given concept. For example, w.r.t. the above assertions, concept definitions, and GCIs, the individual *JAMES* is an instance of the concept *Human*. With the help of the instance service, one can also compute answers to *instance queries*, i.e., all individuals occurring in the ABox that are instances of the query concept C. In order to state more general search criteria, one can use so-called *conjunctive queries*, i.e., conjunctions of assertions that may also contain variables, of which some can be existentially quantified. For example, the conjunctive query

$$\exists y, z.Woman(x) \wedge child(x, y) \wedge child(z, y) \wedge Beatle(z)$$

asks for all women that have a child with a parent that is a Beatle. With respect to the knowledge base we have introduced so far, this conjunctive query has no individual as an answer.

In order to ensure a reasonable and predictable behavior of a DL system, the underlying inference problems (like the subsumption and the instance problem) should at least be decidable for the DL employed by the system, 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 in DL research.

The general opinion on the (worst-case) complexity that is acceptable for a DL has changed dramatically over time. Historically, in the early times of DL research people concentrated on identifying formalisms for which reasoning is tractable, i.e., can be performed in polynomial time [47]. The precursor of all DL systems, KL-ONE [16], as well as its early successor systems, like KANDOR [47], K-REP [43], BACK [48], and LOOM [42], indeed employed polynomial-time subsumption algorithms. Later on, however, it turned out that subsumption in rather inexpressive DLs may be intractable [38], that subsumption in KL-ONE is even undecidable [49], and that even for systems like KANDOR and BACK, for which the expressiveness of the underlying DL had been carefully restricted with the goal of retaining tractability, the subsumption problem is in fact intractable [44]. The reason for the discrepancy between the complexity of the subsumption algorithms employed in the above mentioned early DL systems and the worst-case complexity of the subsumption problems these algorithms were supposed to solve was due to the fact that these systems employed sound, but incomplete subsumption algorithms, i.e., algorithms whose positive answers to subsumption queries are correct, but whose negative answers may be incorrect. The use of incomplete algorithms has since then largely been abandoned in the DL community, mainly because of the problem that the behavior of the systems is no longer determined by the semantics of the description language: an incomplete algorithm may claim that a subsumption relationship does not hold, although it should hold according to the semantics. All the intractability results mentioned above already hold for subsumption between concept descriptions without a TBox. An even worse blow to the quest for a practically useful DL with a sound, complete, and polynomial-time subsumption algorithm was Nebel's result [45] that subsumption w.r.t. an acyclic TBox (i.e., an unambiguous set of concept definitions without cyclic dependencies) in a DL with conjunction ($\sqcap$) and value restriction ($\forall r.C$) is already intractable.[2]

[2] All the systems mentioned above supported these two concept constructors, which were at that time viewed as being indispensable for a DL. The DL with exactly these two concept constructors is called $\mathcal{FL}_0$ [4].

At about the time when these (negative) complexity results were obtained, a new approach for solving inference problems in DLs, such as the subsumption and the instance problem, was introduced. This so-called *tableau-based approach* was first introduced in the context of DLs by Schmidt-Schauß and Smolka [50], though it had already been used for modal logics long before that [22]. It has turned out that this approach can be used to handle a great variety of different DLs [7, 10, 15, 26, 27, 30, 34, 35], and it yields sound and complete inference algorithms also for very expressive DLs. Although the worst-case complexity of these algorithms is quite high, the tableau-based approach nevertheless often yields practical procedures: optimized implementations of such procedures have turned out to behave quite well in applications [9, 23, 25, 28, 29, 31], even for expressive DLs with a high worst-case complexity (ExpTime and beyond). The advent of efficient tableau-based algorithms was the main reason why the DL community basically abandoned the search for DLs with tractable inference problems, and concentrated on the design of practical tableau-based algorithms for expressive DLs. The most prominent modern DL systems, FaCT++ [53], Racer [24], and Pellet [51] support very expressive DLs and employ highly-optimized tableau-based algorithms.

In addition to the fact that DLs are equipped with a well-defined formal semantics, the availability of mature systems that support sound and complete reasoning in very expressive description formalisms was an important argument in favor of using DLs as the foundation of OWL, the standard ontology language for the Semantic Web. In fact, OWL DL is based on the expressive DL $\mathcal{SHOIN}(\mathcal{D})$, for which reasoning is NExpTime-complete, and its sublanguage OWL Lite is based on $\mathcal{SHIF}(\mathcal{D})$, for which reasoning is still ExpTime-complete [32]. The OWL 2 standard is based on the even more expressive DL $\mathcal{SROIQ}(\mathcal{D})$, which is even 2NExpTime-complete [36].

Due to the ever increasing expressive power and worst-case complexity of expressive DLs, there is also an increasing number of ontologies emerging from practical applications that cannot be handled by tableau-based reasoning systems without manual tuning by the system developers, despite highly optimized implementations. Perhaps the most prominent example is the well-known medical ontology SNOMED CT,[3] which comprises 380,000 concepts and is used to generate a standardized health care terminology used as a standard for medical data exchange in a variety of countries such as the US, Canada, and Australia. In tests performed in 2005 with FaCT++ and Racer, neither of the two systems could classify SNOMED CT [13],[4] and Pellet still could not classify SNOMED CT in tests performed in 2008 [52]. From the DL point of view, SNOMED CT is an acyclic TBox that contains only the concept constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$). The DL with exactly these three concept constructors is called $\mathcal{EL}$ [12]. In contrast to its counterpart with value restrictions, $\mathcal{FL}_0$, the light-weight DL $\mathcal{EL}$ has much better algorithmic properties. Whereas subsumption without a TBox is polynomial in both $\mathcal{EL}$ [12] and $\mathcal{FL}_0$ [38], subsumption in $\mathcal{FL}_0$ w.r.t. an acyclic TBox is coNP-complete [45] and w.r.t. GCIs it is even ExpTime-complete [5]. In contrast, subsumption in $\mathcal{EL}$ stays tractable even w.r.t. GCIs [17], and this result is stable under the addition of several interesting means of expressivity [5, 6]. The DL $\mathcal{EL}$ and the mentioned tractability results will be introduced in more detail in the next section.

Another issue with expressive DLs and tableau-based algorithms is that they do not scale too well to knowledge bases with a very large ABox. In particular, query answering in expressive DLs such as the already mentioned $\mathcal{SHIF}$ and $\mathcal{SHOIN}$ is 2ExpTime-complete regarding combined complexity [39], i.e., the complexity w.r.t. the size of the TBox and the ABox. Thus query answering in these logics is even harder than subsumption while at the same time being much more time critical. Moreover, query answering in these DLs is coNP-complete [46] regarding data complexity (i.e., in the size of the ABox), which is viewed as 'unfeasible' in the database community. These results are dramatic since many DL applications, such as those that use ABoxes as kind of web repositories, involve ABoxes with hundred of thousands of individuals. It is a commonly held opinion that, in order to achieve truly scalable query answering in the short term, it is essential to make use of conventional relational database systems for query answering in DLs. Given this proviso, the question is what expressivity can a DL offer such that queries can be answered using relational database technology while at the same time meaningful concepts can be specified in the TBox. As an answer to this, the DL-Lite family has been introduced in [18, 19], designed to allow the implementation of conjunctive query answering 'on top of' a relational database system. In Sect. 3, we introduce DL-Lite$_{core}$ and two of its extensions DL-Lite$_{\mathcal{F}}$ and DL-Lite$_{\mathcal{R}}$. We also sketch the standard approach to query answering in these languages. Interestingly, also in $\mathcal{EL}$ it is possible to implement query answering using a database system, though with a different approach than in DL-Lite (see the end of Sect. 3).

2 The DL $\mathcal{EL}$ and Its Extension $\mathcal{EL}^{++}$

Starting with a set N_{con} of concept names and a set N_{role} of role names, *$\mathcal{EL}$-concept descriptions* are built using the concept constructors top concept ($\top$), conjunction ($\sqcap$), and

[3] http://www.ihtsdo.org/snomed-ct/.

[4] Note, however, that more recent versions of FaCT++ and Racer perform quite well on SNOMED CT [52], due to optimizations specifically tailored towards the classification of SNOMED CT.

Table 1 Syntax and semantics of $\mathcal{EL}$

Name	Syntax	Semantics
Concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Top concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
Conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
General concept inclusion (GCI)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$

existential restriction ($\exists r.C$). The semantics of $\mathcal{EL}$-concept descriptions is defined in the usual way, using the notion of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names and subsets of $\Delta^{\mathcal{I}}$ to concept descriptions, as shown in the semantics column of Table 1.

A *general concept inclusion (GCI)* is of the form $C \sqsubseteq D$ where C, D are $\mathcal{EL}$-concept descriptions, and a *concept definition* is of the form $A \equiv C$ where A is a concept name and C is an $\mathcal{EL}$-concept description. The interpretation $\mathcal{I}$ is a *model* of the GCI $C \sqsubseteq D$ or the concept definition $A \equiv C$ if it satisfies the condition stated in the semantics column of Table 1. Obviously, this semantics implies that the concept definition $A \equiv C$ is equivalent to the two GCIs $A \sqsubseteq C, C \sqsubseteq A$ in the sense that they have the same models. For this reason, in the following we will consider only GCIs. A finite set of GCIs is called a *TBox*.

Given a TBox $\mathcal{T}$ and two $\mathcal{EL}$-concept descriptions C, D, we say that C is *subsumed* by D w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$.[5]

When designing a subsumption algorithm for $\mathcal{EL}$ it is actually enough to consider the case where C, D are concept names occurring in the TBox. In fact, it is easy to see that $C \sqsubseteq_{\mathcal{T}} D$ iff $A \sqsubseteq_{\mathcal{T} \cup \{A \sqsubseteq C, D \sqsubseteq B\}} B$ where A, B are new concept names, i.e., concept names not occurring in C, D, and $\mathcal{T}$.

The polynomial-time subsumption algorithm for $\mathcal{EL}$ [5, 17] that will be sketched below actually classifies the given TBox $\mathcal{T}$, i.e., it simultaneously computes all subsumption relationships between the concept names occurring in $\mathcal{T}$. This algorithm proceeds in four steps:

1. Normalize the TBox.
2. Translate the normalized TBox into a graph.
3. Complete the graph using completion rules.
4. Read off the subsumption relationships from the normalized graph.

[5] In this section, we do not introduce ABoxes and the instance problem. It should be noted, however, that the tractability results sketched in this section extend to the instance problem.

(R1) If $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $A_1, A_2 \in S(A)$ then add B to $S(A)$

(R2) If $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$ and $A_1 \in S(A)$ then add r to $R(A, B)$

(R3) If $\exists r.B_1 \sqsubseteq A_1 \in \mathcal{T}$ and $B_1 \in S(B), r \in S(A, B)$ then add A_1 to $S(A)$

Fig. 1 The completion rules for subsumption in $\mathcal{EL}$

An $\mathcal{EL}$-TBox is *normalized* iff it only contains GCIs of the following form: $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$, $\exists r.A \sqsubseteq B$, where A, A_1, A_2, B are concept names or the top concept $\top$. Any $\mathcal{EL}$-TBox can be transformed in polynomial time into a normalized one by applying equivalence-preserving normalization rules [17].

In the next step, we build the *classification graph* $G_{\mathcal{T}} = (V, V \times V, S, R)$ where

- V is the set of concept names (including $\top$) occurring in the normalized TBox $\mathcal{T}$;
- S labels nodes with sets of concept names (again including $\top$);
- R labels edges with sets of role names.

The label sets are supposed to satisfy the following *invariants*:

- $B \in S(A)$ implies $A \sqsubseteq_{\mathcal{T}} B$, i.e., $S(A)$ contains only subsumers of A w.r.t. $\mathcal{T}$.
- $r \in R(A, B)$ implies $A \sqsubseteq_{\mathcal{T}} \exists r.B$, i.e., $R(A, B)$ contains only roles r such that $\exists r.B$ subsumes A w.r.t. $\mathcal{T}$.

Initially, we set $S(A) := \{A, \top\}$ for all nodes $A \in V$, and $R(A, B) := \emptyset$ for all edges $(A, B) \in V \times V$. Obviously, the above invariants are satisfied by these initial label sets.

The labels of nodes and edges are then extended by applying the rules of Fig. 1. Note that a rule is only applied if it really extends a label set. It is easy to see that these rules preserve the above invariants. For example, consider the (most complicated) rule (R3). Obviously, $\exists r.B_1 \sqsubseteq A_1 \in \mathcal{T}$

implies $\exists r.B_1 \sqsubseteq_{\mathcal{T}} A_1$, and the assumption that the invariants are satisfied before applying the rule yields $B \sqsubseteq_{\mathcal{T}} B_1$ and $A \sqsubseteq_{\mathcal{T}} \exists r.B$. The subsumption relationship $B \sqsubseteq_{\mathcal{T}} B_1$ obviously implies $\exists r.B \sqsubseteq_{\mathcal{T}} \exists r.B_1$. By applying transitivity of the subsumption relation $\sqsubseteq_{\mathcal{T}}$, we thus obtain $A \sqsubseteq_{\mathcal{T}} A_1$.

The fact that subsumption in $\mathcal{EL}$ w.r.t. TBoxes can be decided in polynomial time is an immediate consequence of the following two facts (see [5, 17] for proofs):

1. Rule application terminates after a polynomial number of steps.
2. If no more rules are applicable, then $A \sqsubseteq_{\mathcal{T}} B$ iff $B \in S(A)$.

Theorem 1 *Subsumption in $\mathcal{EL}$ w.r.t. TBoxes can be decided in polynomial time.*

This result is not only of theoretical interest. Experiments have shown that an optimized implementation [13] of the subsumption algorithm sketched above in the CEL system[6] [14] behaves very well on large life science ontologies [13, 52].

The tractability result for $\mathcal{EL}$ can be extended to $\mathcal{EL}^{++}$, which extends $\mathcal{EL}$ by the following means of expressiveness:

- The *bottom concept* $\bot$ is always interpreted as the empty set. It can, for example, be used to express disjointness of concepts, as in the GCI $Woman \sqcap Man \sqsubseteq \bot$.
- *Nominals* are basically names for individuals, but used as concept constructors with set brackets around the individual name. A nominal $\{n\}$ is always interpreted as a singleton set. For example, we can use the nominal $\{OBAMA\}$ to express the concept of all individuals that like Obama: $\exists likes.\{OBAMA\}$. Nominals can also be used to express ABox assertions through GCIs. For example, the role assertion $r(a, b)$ can be expressed as $\{a\} \sqsubseteq \exists r.\{b\}$.
- *Concrete domains* can be used to refer to data types like numbers or strings when defining concepts. For example, the concept description $Human \sqcap \geq_{18}(age)$ describes adult human beings. However, only very restricted forms of concrete domains are admissible in $\mathcal{EL}^{++}$ (see [5] for details).
- *Restricted role-value maps* are of the form $r_1 \circ \cdots \circ r_k \sqsubseteq r$. They are TBox axioms and not concept constructors. In a model of this role-value map, the composition of the roles $r_1, \ldots, r_k$ must be contained in the role r. Special cases of such role-value maps are *transitivity of a role* r, expressed as $r \circ r \sqsubseteq r$ and *right-identity rules* $r \circ s \sqsubseteq r$, which are both important for medical ontologies. For example, we may want to say that the *part_of* relation is transitive, which can be expressed as $part\_of \circ part\_of \sqsubseteq part\_of$, and that medical findings are inherited along *part_of*, expressed as $finding\_at \circ part\_of \sqsubseteq finding\_at$. Given the second role-value maps together with GCIs stating that a finger is part of the hand, an injury of the finger is an injury found at the finger, and an injury of the hand is an injury found at the hand, we can then deduce that an injury of the finger is an injury of the hand.
- A *reflexivity* axiom for the role r states that this role is reflexive, i.e., every individual is related to itself w.r.t. this role. For example, in a medical ontology one may want to state that the *part_of* relation is reflexive, i.e., every entity is part of itself.
- The range restriction $ran(r) \sqsubseteq C$ says that the second component of every tuple belonging to r must belong to C. For example, the range restriction $ran(finding\_at) \sqsubseteq Body\_structure$ says that finding sites must belong to the body structure, i.e., this role is used to specify where in the body something (e.g., an injury) is found. The range restriction $ran(r) \sqsubseteq C$ could of course be expressed using the GCI $\top \sqsubseteq \forall r.C$, but value restrictions $\forall r.C$ are not available in $\mathcal{EL}^{++}$. Thus, range restrictions can be seen as a restricted way of using value restrictions in $\mathcal{EL}^{++}$. Note, however, that the unrestricted use of value restrictions would destroy tractability.

Note that the original version of $\mathcal{EL}^{++}$ [5] did not have reflexive roles and range restrictions. They were added in the version introduced in [6], which is the version of $\mathcal{EL}^{++}$ that underlies the designated OWL EL profile of OWL 2. To keep tractability (even decidability), one must actually impose a syntactic restriction on $\mathcal{EL}^{++}$-TBoxes that prevents interactions between restricted role-value maps and range restrictions (see [6] for details). It should also be noted that basically all other additions of typical DL constructors to $\mathcal{EL}$ make subsumption w.r.t. TBoxes ExpTime-hard [5, 6].

3 The DL-Lite Family of DLs

DL-Lite$_{core}$ is the basic member of the DL-Lite family [20]. Concept descriptions of this DL are of the form

$$A, \quad \exists r.\top, \quad \exists r^-.\top,$$

where A is a concept name, r is a role name, and r^- denotes the inverse of the role name r, with the obvious semantics

$$(r^-)^{\mathcal{I}} = \{(y, x) \mid (x, y) \in r^{\mathcal{I}}\}.$$

A DL-Lite$_{core}$ knowledge base (KB) consists of a TBox and an ABox. The *TBox formalism* allows for GCIs and disjointness axioms between DL-Lite$_{core}$ concept descriptions C, D:

$$C \sqsubseteq D \quad \text{and} \quad \mathsf{disj}(C, D),$$

where an interpretation $\mathcal{I}$ is a model of $\mathsf{disj}(C, D)$ if it satisfies $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$. Although conjunction is not available

[6] http://cel.googlecode.com.

in DL-Lite$_{core}$, it can be simulated to a certain extent: a conjunction on the right-hand side of a GCI $C \sqsubseteq D_1 \sqcap D_2$ can be expressed by the two GCIs $C \sqsubseteq D_1$ and $C \sqsubseteq D_2$. Disjunction on the left-hand side of a GCI can be expressed in a similar way. The following is an example of a DL-Lite$_{core}$-TBox:

$$\mathcal{T}_{ex} = \{\exists child.\top \sqsubseteq Parent,\ Parent \sqsubseteq Human,$$
$$Human \sqsubseteq \exists child^{-}.\top,\ \mathsf{disj}(Human, Insect)\}.$$

A DL-Lite$_{core}$-ABox is a finite set of *concept and role assertions*: $A(a)$ and $r(a, b)$, where A is a concept name, r is a role name, and a, b are individual names. An interpretation $\mathcal{I}$ assigns an element $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to every individual name c such that the *unique name assumption* (UNA) is satisfied, i.e. $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for distinct individual names a, b.[7] It is a model of $A(a)$ if it satisfies $a^{\mathcal{I}} \in A^{\mathcal{I}}$ and of $r(a, b)$ if it satisfies $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. The restriction to concept names in concept assertions can be circumvented by introducing a GCI for a new concept name, say $A_{new} \sqsubseteq C$, in the TBox and then stating $A_{new}(a)$ in the ABox. The following is an example of a DL-Lite$_{core}$-ABox:

$$\mathcal{A}_{ex} = \{Woman(LINDA),\ child(LINDA, JAMES),$$
$$Beatle(PAUL),\ child(PAUL, JAMES)\}.$$

In [20], the following two extensions of DL-Lite$_{core}$ have also been considered:

- DL-Lite$_{\mathcal{F}}$, in which the TBox may additionally contain *functionality axioms* $\mathsf{func}(r)$ for role names and their inverses. Such an axiom can, e.g., be used to state that the role *father* is functional, i.e., every individual has at most one father.
- DL-Lite$_{\mathcal{R}}$, in which the TBox may additionally contain *role inclusion axioms* $r_1 \sqsubseteq r_2$ and *role disjointness axioms* $\mathsf{disj}(r_1, r_2)$ for role names and their inverses. Such axioms can, e.g., be used to state that the roles *father* and *mother* are disjoint subroles of $child^{-}$.

Other members of the DL-Lite family have, e.g., been defined in [2, 21, 37].

The DL-Lite family of DLs is tailored towards applications in which huge amounts of data (represented as an ABox) are queried w.r.t. fairly light-weight ontologies. In this setting, it is no longer sufficient that query answering is tractable. One needs to be able to store the ABox in a relational database system, and answer queries using a relational query engine. From a logical point of view, a relational database is a finite first-order interpretation $\mathcal{I}$, and the relational query engine can efficiently answer *first-order queries (FOL queries)*. Such a query is a first-order formula $\phi(\vec{x})$ over the vocabulary of the database and with free variables $\vec{x}$; an answer tuple $\vec{c}$ is a sequence of elements of the domain of $\mathcal{I}$ such that $\phi(\vec{c})$ evaluates to true in $\mathcal{I}$. Given an FOL query q, we denote the set of its answer tuples in the database $\mathcal{I}$ with $q^{\mathcal{I}}$.

In DL-Lite, one concentrates on answering a restricted form of FOL queries, so-called unions of conjunctive queries. A *conjunctive query* is a conjunction of atoms, built using concept and role names as predicate symbols, individual names as constant symbols, and variables, of which some may be existentially quantified. For example, the following is a conjunctive query:

$$q_{ex} = \exists y, z_1, z_2.\ Woman(x) \wedge child(x, y) \wedge child(z_1, y)$$
$$\wedge Human(z_1) \wedge child(z_2, z_1).$$

A *union of conjunctive queries* is a finite set of conjunctive queries, which is interpreted as the disjunction of its elements. Given a union of conjunctive queries or a conjunctive query q and a knowledge base $\mathcal{K}$, the *set of answers to q over $\mathcal{K}$* (denoted $ans(q, \mathcal{K})$) consists of all tuples $\vec{a}$ of individual names appearing in the knowledge base such that $\vec{a}^{\mathcal{I}} \in q^{\mathcal{I}}$ for every model $\mathcal{I}$ of the knowledge base. For the knowledge base $\mathcal{K}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$ of our example and the conjunctive query q_{ex}, it is easy to see that $ans(q_{ex}, \mathcal{K}_{ex}) = \{LINDA\}$.

The approach for query answering in DL-Lite using a relational database system proceeds as follows:

1. use the TBox $\mathcal{T}$ to reformulate the given union of conjunctive queries q into an FOL query $q_{\mathcal{T}}$ and then discard the TBox;
2. view the ABox $\mathcal{A}$ as a relational database $\mathcal{I}_{\mathcal{A}}$, which has as its domain all individuals names occurring in $\mathcal{A}$, interprets concept names A as $A^{\mathcal{I}_{\mathcal{A}}} = \{a \mid A(a) \in \mathcal{A}\}$, and role names r as $r^{\mathcal{I}_{\mathcal{A}}} = \{(a, b) \mid r(a, b) \in \mathcal{A}\}$;
3. evaluate $q_{\mathcal{T}}$ in the database $\mathcal{I}_{\mathcal{A}}$ using a relational query engine.

If this approach is correct for a given DL $\mathcal{L}$, i.e., there is a reformulation function $q \mapsto q_{\mathcal{T}}$ such that $q_{\mathcal{T}}{}^{\mathcal{I}_{\mathcal{A}}} = ans(q, (\mathcal{T}, \mathcal{A}))$ for all unions of conjunctive queries q, then one says that answering conjunctive queries in $\mathcal{L}$ is *FOL-reducible*. The following theorem is proved in [20].

Theorem 2 *Answering conjunctive queries in DL-Lite$_{core}$, DL-Lite$_{\mathcal{F}}$, and DL-Lite$_{\mathcal{R}}$ is FOL-reducible.*

Since the size of the reformulated query does not depend on the size of the ABox, the data complexity of evaluating the original query (i.e., the complexity in terms of the size of the ABox) is the same as evaluating the reformulated query. Because the data complexity of evaluating FOL queries in a relational database is complete for the complexity class AC^0, this implies that the data complexity of answering conjunctive queries in DL-Lite$_{core}$, DL-Lite$_{\mathcal{F}}$, and

[7] The impact of dropping the UNA on the complexity of reasoning in the DL-Lite family has been investigated in [3].

DL-Lite$_\mathcal{R}$ is in AC^0, which is a proper subclass of the class of all tractable problems P. This method for query answering in DL-Lite based on FOL-reducibility has been implemented in the QuOnto system [1].

The reformulation approach developed in [20] actually yields a union of conjunctive queries rather than an arbitrary FOL query. Instead of describing it in detail, we illustrate it with our example. The main idea is to use the GCIs in the TBox as rewrite rules from right to left. Each rewrite step replaces an atom in a conjunctive query q contained in the union of conjunctive queries. The rewritten conjunctive query q' is then added to the union of conjunctive queries (without removing the original query q). Consider the atom $child(z_2, z_1)$ in q_{ex}. Since z_2 is existentially quantified, this basically says that z_1 belongs to $\exists child^-.\top$, and thus the GCI $Human \sqsubseteq \exists child^-.\top$ can be used to replace this atom with $Human(z_1)$, which already occurs in the conjunctive query. Thus, the new conjunctive query $q^{(1)}$:

$$\exists y, z_1.Woman(x) \wedge child(x, y) \wedge child(z_1, y) \ \wedge Human(z_1)$$

is added. In $q^{(1)}$, the atom $Human(z_1)$ can be replaced by $Parent(z_1)$, which yields the additional conjunctive query $q^{(2)}$. Using the GCI $\exists child.\top \sqsubseteq Parent$, the atom $Parent(z_1)$ in $q^{(2)}$ can be replaced by $child(z_1, z_3)$, where z_3 is a new existentially quantified variable. This yields the new conjunctive query $q^{(3)}$:

$$\exists y, z_1, z_3.Woman(x) \wedge child(x, y) \wedge child(z_1, y) \wedge child(z_1, z_3).$$

It is easy to see that *LINDA* is an answer for the query $q^{(3)}$ in the database $\mathcal{I}_{\mathcal{A}_{ex}}$, and thus of the union of conjunctive queries generated by the reformulation process. In addition to rewriting atoms using GCIs, the general reformulation process also uses unification of atoms in a conjunctive query to generate new conjunctive queries (see [20] for details).

It should be noted that also for (a fragment of) $\mathcal{EL}^{++}$, an approach to conjunctive query answering using relational database systems has been developed [40, 41]. Since the data complexity of query answering in $\mathcal{EL}$ is PTime-complete, the approach follows a different route than the one for DL-Lite (since FOL-reducibility implies that the data complexity of query answering is in AC^0). In particular, the TBox is incorporated into the ABox and not into the query. However, some limited query reformulation (independent of both the TBox and the ABox) is still required. Interestingly, both the ABox rewriting and the query reformulation cause only a polynomial blow-up, in contrast to DL-Lite, where the blow-up of the query may be exponential in the size of the original query [20]. This alternative approach for query answering using a relational database system can also be applied to DL-Lite [37]. The approach introduced in [37] causes an exponential blow-up of the query, but we believe that this may be avoidable. Nevertheless, even with this blow-up the query execution times are typically smaller than those of the approach introduced in [20].

4 Conclusion

We have described the origins of two novel families of light-weight DLs: logics of the $\mathcal{EL}$ family were designed to admit subsumption and classification in polynomial time, while still providing sufficient expressive power for life-science ontologies; logics of the DL-Lite family have been designed to enable query answering using relational database systems, while still providing sufficient expressive power to capture conceptual modeling formalisms. The relevance of the small DLs discussed in this article is underlined by the fact that both of them are captured in the official W3C profiles[8] document for the recommendation of OWL 2. Each of the OWL 2 profiles are designed for specific application requirements. For applications that rely on reasoning services for ontologies with a large number of concepts, the profile OWL 2 EL has been introduced, which is based on $\mathcal{EL}^{++}$. For applications that deal with large sets of data and that mainly use the reasoning service of query answering, the profile OWL 2 QL has been defined. The DL underlying this profile is DL-Lite$_\mathcal{R}$. Both, the profile OWL 2 EL and OWL 2 QL pave the way to apply very efficient reasoning services in practical applications. The recent research and standardization efforts discussed in this paper suggest that small is indeed again beautiful in Description Logics.

References

1. Acciarri A, Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Palmieri M, Rosati R (2005) QUONTO QUerying ONTOlogies. In: Proc of the nat conf on AI (AAAI'05)
2. Artale A, Calvanese D, Kontchakov R, Zakharyaschev M (2007) DL-Lite in the light of first-order logic. In: Proc of the nat conf on AI (AAAI'07)
3. Artale A, Calvanese D, Kontchakov R, Zakharyaschev M (2009) DL-Lite without the unique name assumption. In: Proc of the description logic WS (DL'09), CEUR
4. Baader F (1990) Terminological cycles in KL-ONE-based knowledge representation languages. In: Proc of the nat conf on AI (AAAI'90)
5. Baader F, Brandt S, Lutz C (2005) Pushing the $\mathcal{EL}$ envelope. In: Proc of the int joint conf on AI (IJCAI'05)
6. Baader F, Brandt S, Lutz C (2008) Pushing the $\mathcal{EL}$ envelope further. In: Proc of the Int WS on OWL: experiences and directions (OWLED'08)
7. Baader F, Buchheit M, Hollunder B (1996) Cardinality restrictions on concepts. Artif Intell 88(1–2):195–213
8. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF (eds) (2003) The description logic handbook: theory, implementation, and applications. Cambridge University Press, Cambridge

[8] http://www.w3.org/TR/owl2-profiles/.

9. Baader F, Franconi E, Hollunder B, Nebel B, Profitlich H-J (1994) An empirical analysis of optimization techniques for terminological representation systems or: making KRIS get a move on. Appl AI Spec Iss on KB Management
10. Baader F, Hanschke P (1991) A schema for integrating concrete domains into concept languages. In: Proc of the int joint conf on AI (IJCAI'91)
11. Baader F, Horrocks I, Sattler U (2003) Description logics. In: Handbook on ontologies. Int handbooks in information systems. Springer, Berlin
12. Baader F, Küsters R, Molitor R (1999) Computing least common subsumers in description logics with existential restrictions. In: Proc of the int joint conf on AI (IJCAI'99)
13. Baader F, Lutz C, Suntisrivaraporn B (2005) Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In: Proc of the int WS on methods for modalities (M4M-05)
14. Baader F, Lutz C, Suntisrivaraporn B (2006) CEL—a polynomial-time reasoner for life science ontologies. In: Proc of the int joint conf on autom reasoning (IJCAR'06). LNAI, vol 4130
15. Baader F, Sattler U (2001) An overview of tableau algorithms for description logics. Stud Log 69:5–40
16. Brachman RJ, Schmolze JG (1985) An overview of the KL-ONE knowledge representation system. Cogn Sci 9(2)
17. Brandt S (2004) Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: Proc of the Eur conf on AI (ECAI'04)
18. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2005) DL-Lite: tractable description logics for ontologies. In: Proc of the nat conf on AI (AAAI'05)
19. Calvanese D, de Giacomo G, Lembo D, Lenzerini M, Rosati R (2006) Data complexity of query answering in description logics. In: Proc of the int conf on principles of knowledge representation and reasoning (KR'06)
20. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J Autom Reason 39(3):385–429
21. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Poggi A, Rosati R (2006) Linking data to ontologies: the description logic DL-Lite$_A$. In: Proc of the int WS on OWL: experiences and directions (OWLED'06). CEUR
22. Fitting M (1972) Tableau methods of proof for modal logics. Notre Dame J Form Log 13(2):237–247
23. Haarslev V, Möller R (2001) High performance reasoning with very large knowledge bases: a practical case study. In: Proc of the int joint conf on AI (IJCAI'01)
24. Haarslev V, Möller R (2001) RACER system description. In: Proc of the int joint conf on autom reasoning (IJCAR'01). LNAI, vol 2083
25. Haarslev V, Möller R (2008) On the scalability of description logic instance retrieval. J Autom Reason 41(2):99–142
26. Hollunder B, Baader F (1991) Qualifying number restrictions in concept languages. In: Proc of the int conf on the principles of knowledge representation and reasoning (KR'91)
27. Hollunder B, Nutt W, Schmidt-Schauß M (1990) Subsumption algorithms for concept description languages. In: Proc of the Eur conf on AI (ECAI'90)
28. Horrocks I (1998) Using an expressive description logic: FaCT or fiction? In: Proc of the int conf on principles of knowledge representation and reasoning (KR'98)
29. Horrocks I (2003) Implementation and optimization techniques. In: Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF (eds) The description logic handbook: theory, implementation, and applications. Cambridge Univ. Press, Cambridge
30. Horrocks I, Kutz O, Sattler U (2006) The even more irresistible $\mathcal{SROIQ}$. In: Proc of the int conf on principles of knowledge representation and reasoning (KR'06)
31. Horrocks I, Patel-Schneider PF (1999) Optimizing description logic subsumption. J Logic Comput 9(3):267–293
32. Horrocks I, Patel-Schneider PF (2004) Reducing OWL entailment to description logic satisfiability. J Web Sem 1(4):345–357
33. Horrocks I, Patel-Schneider PF, van Harmelen F (2003) From SHIQ and RDF to OWL: the making of a web ontology language. J Web Sem 1(1):7–26
34. Horrocks I, Sattler U (2005) A tableaux decision procedure for $\mathcal{SHOIQ}$. In: Proc of the int joint conf on AI (IJCAI'05)
35. Horrocks I, Sattler U, Tobies S (2000) Practical reasoning for very expressive description logics. J Interest Group Pure Appl Logic 8(3):239–264
36. Kazakov Y (2008) $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Proc of the int conf on principles of knowledge representation and reasoning (KR'08)
37. Kontchakov R, Lutz C, Toman D, Wolter F, Zakharyaschev M (2009) Combined FO rewritability for conjunctive query answering in DL-Lite. In: Proc of the description logic WS (DL'09)
38. Levesque HJ, Brachman RJ (1987) Expressiveness and tractability in knowledge representation and reasoning. Comput Intell 3:78–93
39. Lutz C (2008) The complexity of conjunctive query answering in expressive description logics. In: Proc of the int joint conf on autom reasoning (IJCAR'08). LNAI, vol 5195
40. Lutz C, Toman D, Wolter F (2008) Conjunctive query answering in $\mathcal{EL}$ using a database system. In: In Proc of the int WS on OWL: experiences and directions (OWLED'08)
41. Lutz C, Toman D, Wolter F (2009) Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: Proc of the int joint conf on AI (IJCAI'09)
42. MacGregor R (1991) The evolving technology of classification-based knowledge representation systems. In: Principles of semantic networks. Kaufmann, Los Altos
43. Mays E, Dionne R, Weida R (1991) K-REP system overview. SIGART Bull 2(3):93–97
44. Nebel B (1988) Computational complexity of terminological reasoning in BACK. Artif Intell 34(3):371–383
45. Nebel B (1990) Terminological reasoning is inherently intractable. Artif Intell 43(2):235–249
46. Ortiz M, Calvanese D, Eiter T (2008) Data complexity of query answering in expressive description logics via tableaux. J Autom Reason 41(1):61–98
47. Patel-Schneider PF (1984) Small can be beautiful in knowledge representation. In: Proc of the IEEE WS on knowledge-based systems
48. Peltason Ch (1991) The BACK system—an overview. SIGART Bull 2(3):114–119
49. Schmidt-Schauß M (1989) Subsumption in KL-ONE is undecidable. In: Proc of the int conf on the principles of knowledge representation and reasoning (KR'89)
50. Schmidt-Schauß M, Smolka G (1991) Attributive concept descriptions with complements. Artif Intell 48(1):1–26
51. Sirin E, Parsia B (2004) Pellet: an OWL DL reasoner. In: Proc of the description logic WS (DL'04)
52. Suntisrivaraporn B (2009) Polynomial-time reasoning support for design and maintenance of large-scale biomedical ontologies. PhD thesis, Fakultät Informatik, TU Dresden
53. Tsarkov D, Horrocks I (2006) FaCT++ description logic reasoner: system description. In: Proc of the int joint conf on autom reasoning (IJCAR'06). LNAI, vol 4130

Reasoning and Explanation in $\mathcal{EL}$ and in Expressive Description Logics

Anni-Yasmin Turhan

Theoretical Computer Science,
TU Dresden, Germany,
turhan@tcs.inf.tu-dresden.de

Abstract. Description Logics (DLs) are the formalism underlying the standard web ontology language OWL 2. DLs have formal semantics which are the basis for powerful reasoning services. In this paper, we introduce the basic notions of DLs and the techniques that realize subsumption—the fundamental reasoning service of DL systems. We discuss two reasoning methods for this service: the tableau method for expressive DLs such as $\mathcal{ALC}$ and the completion method for the light-weight DL $\mathcal{EL}$. We also present methods for generating explanations for computed subsumption relationships in these two DLs.

1 Introduction

The ontology language for the semantic web OWL provides means to describe entities of a application domain in an ontology. The underlying formalism for OWL are Description Logics, which have well-defined syntax and formal semantics. The recent version of the W3C standard OWL 2.0 has four language variants: the OWL 2 language itself and three profiles. The latter are light-weight ontology languages of relatively low expressivity and that are tailored to be efficient for specific reasoning tasks. We are interested in the reasoning task of computing subsumption, i.e., sub- and super-class relationships, and providing explanations for the obtained reasoning results. In this paper, we discuss reasoning techniques for computing subsumption relationships for the core description logics underlying the OWL 2 language: $\mathcal{ALC}$ and the core description logics underlying the EL profile: $\mathcal{EL}$. The EL profile is particularly suitable for applications with ontologies that define very large numbers of classes and that need subsumption as the main inference service. Based on the reasoning techniques for subsumption, we discuss methods to compute explanations for detected subsumption relationships in $\mathcal{ALC}$ and $\mathcal{EL}$. Before we turn to the reasoning techniques, we give general overview of Description Logics.

Description Logics (DLs) [6] are a family of knowledge representation formalisms that have formal semantics. This family of logics is tailored towards representing terminological knowledge of an application domain in a structured and formally well-understood way. Description logics allow users to define important notions, such as classes or relations of their application domain in terms of concepts and roles. These concepts (unary predicates) and roles (binary predicates) then restrict the way these classes and relations are interpreted. Based on these definitions, implicitly captured

knowledge can be inferred from the given descriptions of concepts and roles, as for instance sub-class or instance relationships.

The name *Description Logics* is motivated by the fact that classes and relations are defined in terms of concept *descriptions*. These concept descriptions are complex expressions built from atomic concepts and atomic roles using the concept constructors offered by the particular DL in use. Based on their formal semantics, a whole collection of inference services has been defined and investigated for different DLs. DLs have been employed in various domains, such as databases, biomedical or context-aware applications [3, 96]. Their most notable success so far is probably the adoption of the DL-based language OWL[1] as standard ontology language for the Semantic Web [53].

Historically, DLs stem from knowledge representation systems such as *semantic networks* [85, 94] or *frame systems* [73]. These early knowledge representation systems were motivated by linguistic applications and allow to specify information from the domain of discourse. They offer methods to compute inheritance relations between the specified notions. Early frame-based systems and semantic networks both have operational semantics, i.e., the semantics of reasoning is given by its implementation. As a consequence, the result of the reasoning process depends on the implementation of the reasoner and thus the result may differ from system to system for the same input [95]. To remedy this, DLs and their reasoning services are based on formal semantics. The information about the application domain is represented in a declarative and unambiguous way. More importantly, the formal semantics of the reasoning services ensure predictable and thus reliable behavior of the DL reasoning systems—independent of the implementation.

The investigation of algorithms for reasoning services and their complexity is the main focus of the DL research community. Typically, one can distinguish the following phases of DL research during the last decades. In the late eighties, reasoning algorithms have been devised for DL systems that mostly were sound, but incomplete, i.e., they would return correct answers, but would not find *all* correct answers. This development was led by the belief that terminological reasoning is inherently intractable [79, 80], and thus completeness was traded for tractability. These algorithms have been implemented in systems such as Classic [23, 22, 84] and Back [79, 81]. During the nineties, sound and complete reasoning methods were investigated for the core inferences of DL systems: consistency and subsumption. *Consistency* assures that the specification of the concepts, roles and individuals are free of contradictions. For *subsumption* one computes super- and sub-concept relations from the given specifications of concepts and roles. The use of incomplete algorithms for these inferences has largely been abandoned in the DL community since then, mainly because of the problem that the behavior of the systems is no longer determined by the semantics of the description language: an incomplete algorithm may claim that a subsumption relationship does not hold, although it should hold according to the semantics.

The underlying technique for computing the basic DL inferences is the tableau method [37], which was adapted to DLs in [91]. This method was extended to more and more expressive DLs (for an overview, see [17]). The gain in expressiveness came at the cost of higher complexity for the reasoning procedures—reasoning for the DLs in-

[1] http://www.w3.org/TR/owl-features/

vestigated is PSpace-complete or even ExpTime-complete [66, 54, 98] (for an overview see [17, 31]).

Despite the high complexity, highly optimized DL reasoning systems were implemented based on the tableau method—most prominently the FaCT system [49] and RACER [43]. These systems employed optimization methods developed for DL reasoning based on tableaux [7, 48, 58, 45] and demonstrated that the high worst case complexities would hardly be encountered in practice [49, 52, 58, 42, 50, 100]. In fact, it turned out that these highly optimized implementations of the reasoning methods do perform surprisingly well on DL knowledge bases from practical applications.

Encouraged by these findings and driven by application needs researchers investigated tableau algorithms for even more expressive DLs [55, 56, 51, 57] in the last decade. At the same time, the idea of the Semantic Web emerged and DLs became the basis for the W3C standardized web ontology language OWL [53, 44]. This brought DLs into the attention of new users from various application areas, which in turn necessitated automated support of ontology services and motivated research on various new inferences for DLs. For instance,

- the generation of *explanations* of consequences that the DL reasoner detected [90, 83, 63, 61, 15],
- support for building ontologies by computing *generalizations* [10, 27, 18, 101, 35],
- *conjunctive queries* as a means to access the instance data of an ontology [76, 29, 30, 39, 82, 36, 67], and
- computing *modularizations* of an ontology as means to facilitate their reuse [38, 69, 33, 32, 70].

All of them are currently investigated reasoning services for DLs and most of them are implemented in specialized reasoners. At the same time, the need for faster reasoners for the afore mentioned basic inferences for DLs led to two developments. On the one hand, the new tableau-based reasoners for expressive DL were developed such as PELLET [93], FaCT++ [99, 100] and RACERPRO [86] and new reasoning methods for expressive DLs were investigated and implemented such as resolution [74, 76] in KAON2 and hyper-tableau [77, 78] in HERMIT. On the other hand, *light-weight DLs*, which are DLs with relatively limited expressivity, but good computational properties for specific reasoning tasks were designed [13]. Reasoning even for large ontologies written in these DLs can be done efficiently, since the respective reasoning methods are tractable. There are two "families" of lightweight DLs: the $\mathcal{EL}$ family [25, 4, 5], for which the subsumption and the instance problem are polynomial, and the DL Lite family [28, 30], for which the instance problem and query answering are polynomial. A member of each of these families is the DL corresponding to one of the profiles of the OWL 2 standard.

In this paper, we examine the basic reasoning services for DLs for the light-weight DL $\mathcal{EL}$ and for expressive DLs. In the next section, we give basic definitions for the fundamental DLs $\mathcal{ALC}$ and $\mathcal{EL}$. We introduce basic notions such as concept descriptions, TBoxes and ABoxes and their semantics. Based on this, we define the central reasoning services common to most DL systems. In Section 3, we discuss the reasoning methods for basic reasoning problems: we describe the tableau method for $\mathcal{ALC}$ and the

completion-based approach for $\mathcal{EL}$. In Section 4, we turn to another reasoning service, namely the computation of explanations for (probably unexpected) reasoning results. Again, we consider methods for expressive DLs and for $\mathcal{EL}$ for this task.

2 Basic Definitions

The central notion for DLs are *concept descriptions*, which can be built from concept names and so-called *concept constructors*. For instance, one can describe a course as an event given by a lecturer in the following way by a concept description:

$$\mathsf{Event} \sqcap \exists\, \mathsf{given\text{-}by}.\mathsf{Lecturer} \sqcap \exists\, \mathsf{has\text{-}topic}.\top$$

This concept description is a conjunction (indicated by $\sqcap$) of the concept Event, the existential restriction $\exists$ given-by.Lecturer and the existential restriction $\exists$ has-topic.$\top$. The first existential restriction consists of the role name given-by and concept Lecturer, which relates the Lecturer to the course. The latter existential restriction states that there is a topic (which is not specified).

In general, concept descriptions are built from the set of concept names N_C and the set of role names N_R using concept constructors. Every DL offers a different set of concept constructors. The DL $\mathcal{EL}$ allows only for the concept constructors that were used in the example concept description above.

Definition 1 ($\mathcal{EL}$-concept descriptions). *Let N_C be a set of concept names and N_R a set of role names. The set of $\mathcal{EL}$-concept descriptions is the smallest set such that*

- *all concept names are $\mathcal{EL}$-concept descriptions;*
- *if C and D are $\mathcal{EL}$-concept descriptions, then $C \sqcap D$ is also an $\mathcal{EL}$-concept description;*
- *if C is an $\mathcal{EL}$-concept description and $r \in N_R$, then $\exists r.C$ is also an $\mathcal{EL}$-concept description.*

If this set of concept constructors is extended to all Boolean connectors, i.e., extended by disjunction ($\sqcup$) and full negation ($\neg$), one obtains the DL $\mathcal{ALC}$. We can define $\mathcal{ALC}$-concept descriptions inductively.

Definition 2 ($\mathcal{ALC}$-concept descriptions). *Let N_C be a set of concept names and N_R a set of role names. The set of $\mathcal{ALC}$-concept descriptions is the smallest set such that*

- *all concept names are $\mathcal{ALC}$-concept descriptions;*
- *if C and D are $\mathcal{ALC}$-concept descriptions, then $\neg C$, $C \sqcap D$ and $C \sqcup D$ are also $\mathcal{ALC}$-concept descriptions;*
- *if C is an $\mathcal{ALC}$-concept description and $r \in N_R$, then $\exists r.C$ and $\forall r.C$ are also $\mathcal{ALC}$-concept descriptions.*

We call concept descriptions of the form $\exists r.C$ existential restrictions and concept descriptions of the form $\forall r.C$ value restrictions. The semantics of DL concept descriptions is given by means of interpretations.

Definition 3 (Semantics of $\mathcal{ALC}$-concept descriptions). *Let C and D be $\mathcal{ALC}$-concept descriptions and r a role name. An interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where the* domain *$\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a function that assigns to every concept name A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every role name r a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This function is extended to complex $\mathcal{ALC}$-concept descriptions as follows:*

- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$;
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;
- $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid$ *there is a* $y \in \Delta^{\mathcal{I}}$ *with* $(x, y) \in r^{\mathcal{I}}$ *and* $y \in C^{\mathcal{I}}\}$; *and*
- $(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid$ *for all* $y \in \Delta^{\mathcal{I}}$, $(x, y) \in r^{\mathcal{I}}$ *implies* $y \in C^{\mathcal{I}}\}$.

This definitions clearly also captures the semantics of the less expressive DL $\mathcal{EL}$. Both, $\mathcal{EL}$ and $\mathcal{ALC}$ also offer the *top-concept* $\top$, which is always interpreted as the whole domain $\Delta^{\mathcal{I}}$. In addition $\mathcal{ALC}$ also offers the *bottom concept* $\bot$, which is always interpreted as the empty set. Now, with the $\mathcal{ALC}$-concept constructors at hand, one can, for instance, characterize a graduate CS student by the following concept description:

$$\exists \text{ studies-subject. CS} \sqcap (\text{Master-Student} \sqcup \text{PhD-Student})$$

Concept description like these are the main building blocks to model terminological knowledge.

2.1 Terminological Knowledge

A name can be assigned to a concept description by a *concept definition*. For instance, we can write $\text{Course} \equiv \text{Event} \sqcap \exists\, \text{given-by.Lecturer} \sqcap \exists\, \text{has-topic.}\top$ to supply a concept definition for the concept Course.

Definition 4 (Concept definition, general concept inclusion). *Let A be a concept name and C, D be (possibly) complex concept description.*

- *A* concept definition *is a statement of the form $A \equiv C$.*
- *A* general concept inclusion *(GCI for short) is a statement of the form $C \sqsubseteq D$.*

It is easy to see that every concept definition $A \equiv C$ can be expressed by two GCIs: $A \sqsubseteq C$ and $C \sqsubseteq A$. The terminological information expressed by GCIs is collected in the so-called TBox.

Definition 5 (TBox). *A finite set of GCIs is called a* TBox.

An interpretation is a model of a TBox $\mathcal{T}$, if it satisfies all GCIs, i.e., if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all $C \sqsubseteq D$ in $\mathcal{T}$.

If all concept descriptions in a TBox $\mathcal{T}$ are from a description logic $\mathcal{L}$, then we call $\mathcal{T}$ a $\mathcal{L}$-TBox.

If a concept definition $A \equiv C$ in a TBox uses a concept name B directly, i.e., B appears in C, or if B is used indirectly by the definitions of the names appearing in C, we say that the TBox is *cyclic*. Otherwise a TBox is *acyclic*.

Definition 6 (Unfoldable TBox). *A TBox $\mathcal{T}$ is a finite set of concept definitions that is acyclic and such that every concept name appears at most once on the left-hand side of the concept definitions in $\mathcal{T}$. Given a TBox $\mathcal{T}$, we call the concept name A a* defined concept*, if A occurs on the left-hand side of a concept definition in $\mathcal{T}$. All other concepts are called* primitive concepts*.*

One of the basic reasoning services in DL systems is to test for the *satisfiability* of a concept or a TBox, i.e., to test whether the information specified in it contains logical contradictions or not. In case the TBox contains a contradiction, any consequence can follow logically from the TBox. Moreover, if a TBox is not satisfiable, the specified information can hardly capture the intended meaning from an application domain. To test for satisfiability is often a first step for a user to check whether a TBox models something "meaningful".

Definition 7 (Concept satisfiability, TBox satisfiability). *Let C be a concept description and $\mathcal{T}$ a TBox. The concept description C is* satisfiable *iff it has a model, i.e., iff there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. A TBox $\mathcal{T}$ is* satisfiable *iff it has a model, i.e., an interpretation that satisfies all GCIs in $\mathcal{T}$.*

If a concept or TBox is not satisfiable, it is called *unsatisfiable*. Other typical reasoning services offered in DL systems test for equivalence or inclusion relations between concepts. In the latter case, if one concept of the TBox models a more general category than another one, we say that this concept *subsumes* the other one.

Definition 8 (Concept subsumption, concept equivalence). *Let C, D be two concept descriptions and $\mathcal{T}$ a (possibly empty) TBox. The concept description C is* subsumed *by the concept description D w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$), iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in every model $\mathcal{I}$ of $\mathcal{T}$. Two concepts C, D are* equivalent w.r.t. $\mathcal{T}$ *($C \equiv_{\mathcal{T}} D$), iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for every model $\mathcal{I}$ of $\mathcal{T}$.*

The computation of the subsumption relations for all named concepts mentioned in the TBox $\mathcal{T}$ is called *classification* of the TBox $\mathcal{T}$ and yields the *concept hierarchy* of the TBox $\mathcal{T}$.

2.2 Assertional Knowledge

Facts about individuals from the application domain can be stated by *assertions*. There are two basic kinds of assertions for DL systems—one expresses that an individual belongs to a concept and the other one specifies that two individuals are related via a role. The set N_I is the set of all individual names.

Definition 9 (Assertion, ABox). *Let C be a concept description, $r \in NR$ a role name and i, j ($\{i, j\} \subseteq N_I$) be two individual names, then*

- *$C(i)$ is called a* concept assertion *and*
- *$r(i, j)$ is called a* role assertion*.*

An ABox $\mathcal{A}$ is a finite set of concept assertions and role assertions.

For instance, we can express that Dresden is a city located at the river Elbe by the following ABox:

$$\{\ \mathsf{City(Dresden)},\ \mathsf{River(Elbe)},\ \mathsf{located\text{-}at(Dresden,\ Elbe)}\ \}$$

If all concept descriptions in an ABox $\mathcal{A}$ are from a Description Logic $\mathcal{L}$, then we call $\mathcal{A}$ a $\mathcal{L}$-ABox. In order to capture ABoxes, the interpretation function is now extended to individual names. Each individual name is mapped by the interpretation function to an element of the domain $\Delta^{\mathcal{I}}$.

Definition 10 (Semantics of assertions, semantics of ABoxes). *Let C be a concept description, r a role name and i, j two individual names, then an interpretation $\mathcal{I}$ satisfies*

- *the concept assertion $C(i)$ if $i^{\mathcal{I}} \in C^{\mathcal{I}}$ and*
- *the role assertion $r(i, j)$ if $(i^{\mathcal{I}}, j^{\mathcal{I}}) \in r^{\mathcal{I}}$.*

An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$, if $\mathcal{I}$ satisfies every assertion in $\mathcal{A}$.

A DL *knowledge base* $\mathcal{K}$ consists of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. We write $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. We can now test for the absence of contradictions in ABoxes.

Definition 11 (ABox consistency, instance of). *An ABox $\mathcal{A}$ is* consistent *w.r.t. a TBox $\mathcal{T}$, iff it has a model that is also a model for $\mathcal{T}$. The individual i is an* instance of *the concept description C w.r.t. an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ (we write $\mathcal{A} \models_{\mathcal{T}} C(i)$), iff $i^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$.*

ABox realization is a reasoning service that computes for each individual i of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ the set of all named concepts A appearing in $\mathcal{A}$ and $\mathcal{T}$ that (1) have i as an instance ($\mathcal{A} \models_{\mathcal{T}} A(i)$) and (2) that is least w.r.t. $\sqsubseteq_{\mathcal{T}}$.

Typically, all the reasoning services introduced in this section are implemented in DL systems. In Section 3, we discuss the reasoning algorithms for these inferences for $\mathcal{ALC}$ and in more detail for $\mathcal{EL}$. Before we do so, we survey some extensions of these two basic DLs.

2.3 Extensions of Basic DLs

The basic DL $\mathcal{ALC}$ has been extended in many ways and, as mentioned in the introduction, reasoning algorithms have been devised for many of these extensions, see [31]. We consider here now some of those extensions that are captured in the OWL 2 standard [102] and that are also covered in the OWL 2 EL profile [75]. The DLs underlying these standardized ontology languages are $\mathcal{SROIQ}$ [51] and $\mathcal{EL}^{++}$ [5], respectively. Both DLs allow to specify more information on roles.

A role r can be declared to be a *transitive role* in the TBox. The semantics is straight-forward. An interpretation $\mathcal{I}$ satisfies a transitive role declaration $\mathsf{transitive}(r)$ if $\{(a, b), (b, c)\} \subseteq r^{\mathcal{I}}$ implies $(a, c) \in r^{\mathcal{I}}$. Transitive roles can be used in concept descriptions. Assume that the role has-part is transitive, then the two axioms:

$$\mathsf{Summer\text{-}school} \equiv \exists\ \mathsf{has\text{-}part.\ Course}$$
$$\mathsf{Course} \equiv \exists\ \mathsf{has\text{-}part.\ Lesson}$$

imply that a Summer school has a part that is a lesson. The declaration of an *inverse role* applies to a role name r and yields its inverse r^{-1}, where the semantics is the obvious one, i.e.,

$$(r^{-1})^{\mathcal{I}} := \{(e, d) \mid (d, e) \in r^{\mathcal{I}}\}.$$

Using the inverse of the role attends, we can define the concept of a speaker giving a boring talk as

$$\mathsf{Speaker} \sqcap \exists \mathsf{gives}.(\mathsf{Talk} \sqcap \forall \mathsf{attends}^{-1}.(\mathsf{Bored} \sqcup \mathsf{Sleeping})).$$

Furthermore, it can be specified that a role is a super-role of another role by a *role inclusion axiom*. The set of all role inclusions form the *role hierarchy*. An interpretation $\mathcal{I}$ satisfies a role inclusion axiom $r \sqsubseteq s$ if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$.

For instance, we might capture the fact that everybody who is attending something (a course) is also interested in this (course) by a role inclusion axiom

$$\mathsf{attends} \sqsubseteq \mathsf{interested\text{-}in}.$$

DL researchers have introduced many additional constructors to the basic DL $\mathcal{ALC}$ and investigated various DLs obtained by combining such constructors. Here, we only introduce qualified number restrictions as example for additional concept constructors. This extension is covered also in the DL $\mathcal{SROIQ}$, but not in $\mathcal{EL}^{++}$. See [1] for an extensive list of additional concept and role constructors.

Qualified number restrictions are of the form $(\geq n\, r.C)$ (at-least restriction) and $(\leq n\, r.C)$ (at-most restriction), where $n \geq 0$ is a non-negative integer, $r \in N_R$ is a role name, and C is a concept description. The semantics of these additional constructors is defined as follows:

$$(\geq n\, r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \mathit{card}(\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}) \geq n\},$$
$$(\leq n\, r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \mathit{card}(\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}) \leq n\},$$

where $\mathit{card}(X)$ yields the cardinality of the set X. Using qualified number restrictions, we can define the concept of all persons that attend at most 20 talks, of which at least 3 have the topic DL:

$$\mathsf{Person} \sqcap (\leq 20\, \mathsf{attends}.\mathsf{Talk}) \sqcap (\geq 3\, \mathsf{attends}.(\mathsf{Talk} \sqcap \exists \mathsf{topic}.\mathsf{DL})).$$

2.4 Relations of DLs to Other Logics

Description logics are logic-based knowledge representation formalisms. A natural question is how they are related to other logics. In fact, it is easy to see, given their semantics, that most description logics are a fragment of first order logic (FOL). Concept descriptions can be translated into FOL formulae with one free variable. Concept names can be interpreted as unary predicates and role names as binary relations, see for example [88, 68, 59]. An arbitrary $\mathcal{ALC}$-concept description can be translated into a FOL formula τ_x, where x is a free variable in the following way:

- $\tau_x(A) := A(x)$ for a concept name A,

- $\tau_x(\neg C) := \neg\, \tau_x(C)$,
- $\tau_x(C \sqcap D) := \tau_x(C) \wedge \tau_x(D)$,
- $\tau_x(C \sqcup D) := \tau_x(C) \vee \tau_x(D)$,
- $\tau_x(\exists r.C) := \exists y.(r(x,y) \vee \tau_y(C))$, where y is a variable different from x, and
- $\tau_x(\forall r.C) := \forall y.(r(x,y) \rightarrow \tau_y(C))$, where y is a variable different from x.

The intuition of the translation to FOL is that the formula $\tau_x(C)$ describes all domain elements d from $\Delta^{\mathcal{I}}$ that make the formula τ_x true if x is replaced by d. This clearly coincides with the interpretation of the concept description $C^{\mathcal{I}}$. The translation does not yield arbitrary FOL formulae, but formulae from the two-variable fragment [41] and the guarded fragment [40]. Both of which are known to be decidable.

Description Logics are closely related to modal logics (see e.g. [37, 21]). For instance, the DL $\mathcal{ALC}$ is a syntactic variant of the multimodal logic K, see [89]. The multimodal logic K introduces several box and diamond operators that are indexed with the name of the corresponding transition relation, which can be directly translated into $\mathcal{ALC}$ using role names corresponding to the transition relations.

Any $\mathcal{ALC}$ interpretation $\mathcal{I}$ can be viewed as a Kripke structure $K_{\mathcal{I}}$. The elements of the domain $w \in \Delta^{\mathcal{I}}$ correspond to possible worlds in $K_{\mathcal{I}}$. A propositional variable A is true in world w, iff $w \in A^{\mathcal{I}}$. There is a transition relation r in the Kripke structure from world w_1 to world w_2 iff $(w_1, w_2) \in r^{\mathcal{I}}$. Many theoretical results on reasoning in modal logics carry directly over to standard inferences in DLs due to this direct translation.

3 DL Reasoning

In this section we present reasoning methods for the DL reasoning problems defined in the last section: satisfiability and subsumption. These problems are decision problems and we devise decision procedures for them. Before we do so, we recall some general requirements that we would like to hold for such decision procedures. Such a procedure must be:

- *sound*, i.e., the positive answers should be correct;
- *complete*, i.e., the negative answers should be correct; and
- *terminating*, i.e., it should always give an answer in finite time.

Together these properties ensure that we always obtain an answer and that every given answer of the procedure is correct. These properties guarantee that applications built on top of these procedures are predictable and reliable. To employ the decision procedures in real world applications, we also would like our decision procedure to be

- *efficient*, i.e., it should be optimal w.r.t. the (worst-case) complexity of the problem, and
- *practical*, i.e., easy to implement and optimize, and behave well for application cases.

DL research has mostly been dedicated to design decision procedures that fulfill these requirements. The underlying techniques to realize reasoning procedures that we are considering in the following are the tableaux method for expressive DLs and completion for $\mathcal{EL}$.

3.1 Reasoning in Expressive DLs

By expressive DLs we refer to DLs that offer at least all Boolean constructors and that are thus closed under negation. For this kind of DLs, it is not necessary to design and implement different algorithms for the different reasoning problems introduced in the last section, since there exist polynomial time reductions, which only require the availability of the concept constructors conjunction and negation in the description language. For the TBox reasoning problems there are the following reductions:

- Subsumption can be be reduced in polynomial time to equivalence:

$$C \sqsubseteq_{\mathcal{T}} D \text{ iff } C \sqcap D \equiv_{\mathcal{T}} C.$$

- Equivalence can be be reduced in polynomial time to subsumption:

$$C \equiv_{\mathcal{T}} D \text{ iff } C \sqsubseteq_{\mathcal{T}} D \text{ and } D \sqsubseteq_{\mathcal{T}} C.$$

- Subsumption can be be reduced in polynomial time to (un)satisfiability:

$$C \sqsubseteq_{\mathcal{T}} D \text{ iff } C \sqcap \neg D \text{ is unsatisfiable w.r.t. } \mathcal{T}.$$

- Satisfiability can be be reduced in polynomial time to (non-)subsumption:

$$C \text{ is satisfiable w.r.t. } \mathcal{T} \text{ iff not } C \sqsubseteq_{\mathcal{T}} \bot.$$

For reasoning problems w.r.t. ABoxes (and TBoxes) there are similar polynomial time reductions:

- Satisfiability can be be reduced in polynomial time to consistency:

$$C \text{ is satisfiable w.r.t. } \mathcal{T} \text{ iff the ABox } \{C(a)\} \text{ is consistent w.r.t. } \mathcal{T}.$$

- The instance problem can be reduced in polynomial time to (in)consistency:

$$\mathcal{A} \models_{\mathcal{T}} C(a) \text{ iff } \mathcal{A} \cup \{\neg C(a)\} \text{ is inconsistent w.r.t. } \mathcal{T}.$$

- Consistency can be reduced in polynomial time to the (non-)instance problem:

$$\mathcal{A} \text{ is consistent w.r.t. } \mathcal{T} \text{ iff } \mathcal{A} \not\models_{\mathcal{T}} \bot(a).$$

With these reductions at hand, it suffices to investigate a reasoning procedure for one of the reasoning problems. In this section, we restrict ourselves to unfoldable TBoxes, i.e., TBoxes without GCIs and cyclic definitions. We present a tableau algorithm for deciding ABox consistency in this setting. Such a tableau-based algorithm tries to construct a model for the ABox by breaking down the concept descriptions in the knowledge base and inferring new constraints on the elements of this model. The algorithm either stops because all attempts to build a model failed due to obvious contradictions, or it stops with a "canonical" model.

In a first step of the consistency test, negation is treated by transforming the concept description from the knowledge base into *negation normal form (NNF)*. This normal form pushes all negations into the description until they occur only in front of concept names, using de Morgan' rules.

The $\rightarrow_\sqcap$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but not both $C_1(x)$ and $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$.

The $\rightarrow_\sqcup$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{C_2(x)\}$.

The $\rightarrow_\exists$-rule
Condition: $\mathcal{A}$ contains $(\exists r.C)(x)$, but there is no individual name z such that $C(z)$ and $r(x, z)$ are in $\mathcal{A}$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y), r(x, y)\}$ where y is an individual name not occurring in $\mathcal{A}$.

The $\rightarrow_\forall$-rule
Condition: $\mathcal{A}$ contains $(\forall r.C)(x)$ and $r(x, y)$, but it does not contain $C(y)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$.

Fig. 1. Tableau rules of the consistency algorithm for $\mathcal{ALC}$.

Definition 12 ($\mathcal{ALC}$-negation normal form). *An $\mathcal{ALC}$-concept description is in $\mathcal{ALC}$-*negation normal form *(NNF) if the following rules have been applied exhaustively:*

$$\neg\bot \rightarrow \top \qquad \neg(C \sqcap D) \rightarrow (\neg C \sqcup \neg D) \qquad \neg(\exists r.C) \rightarrow (\forall r.\neg C)$$
$$\neg\top \rightarrow \bot \qquad \neg(C \sqcup D) \rightarrow (\neg C \sqcap \neg D) \qquad \neg(\forall r.C) \rightarrow (\exists r.\neg C)$$
$$\neg\neg C \rightarrow C$$

A TBox or an ABox is in NNF, if all concept descriptions appearing in it are in NNF.

The *size* of an $\mathcal{ALC}$-concept description is the number of occurrences of all concept and role names that appear in the concept description. The size of a TBox is the sum of the sizes of all the concept descriptions appearing in the TBox. Similarly, the size of an ABox is the sum of all the concept descriptions appearing the concept assertions plus the number of role assertions. Transforming an $\mathcal{ALC}$-concept description into NNF yields an equivalent concept description, TBox or ABox of the same size.

Let $\mathcal{A}_0$ be an $\mathcal{ALC}$-ABox that is to be tested for consistency. In a first preprocessing step the definitions from the TBox are expanded.[2] More precisely, names of defined concepts are replaced by the right-hand sides of their definitions in the TBox. This replacement is done exhaustively until only names of primitive concepts appear in the ABox $\mathcal{A}_0$. Next, this ABox is transformed into NNF. In order to test consistency of the normalized $\mathcal{A}_0$, the algorithm applies *tableau rules* to this ABox until no more rules apply. The tableau rules for $\mathcal{ALC}$ are depicted in Fig. 1. Tableau rules in general are consistency preserving transformation rules.

The tableau rule $\rightarrow_\sqcup$ that handles disjunction is *nondeterministic*. It transforms a given ABox into two new ABoxes such that the original ABox is consistent if *one* of the new ABoxes is so. For this reason, we will consider finite sets of ABoxes $\mathcal{S} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ instead of single ones. Such a set of ABoxes is *consistent* iff there is

[2] Recall, that we are dealing with unfoldable TBoxes (Def. 6).

some i, $1 \leq i \leq k$, such that $\mathcal{A}_i$ is consistent. A tableau rule of Fig. 1 is applied to a given finite set of ABoxes $\mathcal{S}$ as follows: it takes an element $\mathcal{A}$ of $\mathcal{S}$, and replaces it by one ABox $\mathcal{A}'$ or, in case of $\rightarrow_{\sqcup}$ by two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$.

Definition 13 (Clash, complete ABox, closed ABox). *An ABox $\mathcal{A}$ contains a* clash *iff $\{A(x), \neg A(x)\} \subseteq \mathcal{A}$ for some individual name x and some concept name A. An ABox $\mathcal{A}$ is called*

- complete *iff none of the tableau rules of Fig. 1 applies to it, and*
- closed *if it contains a clash, and* open *otherwise.*

The *consistency algorithm for $\mathcal{ALC}$* proceeds in the following steps. It starts with the singleton set of ABoxes $\{\mathcal{A}_0\}$, and applies the rules from Fig. 1 in arbitrary order until no more rules apply. The algorithm returns "consistent" if the set $\widehat{\mathcal{S}}$ of ABoxes obtained by exhaustively applying the tableau rules contains an open ABox, and "inconsistent" otherwise.

For this procedure, one can show that it is sound, complete and terminating by examining the individual tableau rules. For termination, it is easy to see that each rule application is monotonic in the sense that every rule application extends the number of concept assertions for the individuals in $\mathcal{A}$ and it never removes elements from $\mathcal{A}$. Furthermore, each concept description that appears in $\mathcal{A}$ due to the application of the tableau rules is a sub-concept description of a concept description that appears already in the initial ABox $\mathcal{A}_0$. These two facts together imply that the application of tableau rules terminates. Completeness of the procedure can easily be seen from the definition of a clash. Soundness can be shown by showing local correctness of the individual tableau rules. Local correctness means that the rules preserve consistency, i.e., if $\widehat{\mathcal{S}}'$ is obtained from the finite set of ABoxes $\widehat{\mathcal{S}}$ by application of a transformation rule, then $\widehat{\mathcal{S}}$ is consistent iff $\widehat{\mathcal{S}}'$ is consistent.

Due to space limitations, we refer the reader to [2, 6] for the proofs for soundness, completeness and termination of the tableau algorithm for $\mathcal{ALC}$.

For general TBoxes, the tableau algorithm needs to be extended by a rule for treating GCIs and a more complex mechanism to ensure termination. For a given general TBox $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$, it is easy to see that the general TBox consisting of the single GCI of the form

$$\top \sqsubseteq (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$$

is equivalent to $\mathcal{T}$, i.e., they have the same models. Thus, reasoning for general TBoxes can be done by taking a general TBox that consists of a single GCI of the form $\top \sqsubseteq C$, where C is a concept description constructed from the GCIs as above. This GCI states that every element in the model belongs to C. To capture this in the tableau method, we add a new rule: the $\rightarrow_{\mathcal{T} \sqsubseteq C}$-rule adds the concept assertion $C(x)$ in case the individual name x occurs in the ABox $\mathcal{A}$, and $C(x)$ is not yet present in $\mathcal{A}$. Local correctness, soundness, and completeness of this procedure can easily be shown. However, the procedure does not terminate, due to cyclic axioms. To regain termination, cyclic computations need to be detected and the application of the $\rightarrow_{\exists}$-rule must be blocked. For two individuals a and b, we say that a is *younger* than b, if a was introduced by an

application of the $\rightarrow_\exists$-rule after b was already present in the ABox. The application of the $\rightarrow_\exists$-rule to an individual x is *blocked* by an individual y in an ABox $\mathcal{A}$ iff

- x is younger than y, and
- $\{C \mid C(x) \in \mathcal{A}\} \subseteq \{C \mid C(y) \in \mathcal{A}\}$.

The main idea underlying blocking is that the blocked individual x can use the role successors of y instead of generating new role successors.

The complexity of the consistency problem in $\mathcal{ALC}$ w.r.t. unfoldable TBoxes is PSpace-complete [92, 66]. In case general TBoxes are used, the complexity of testing consistency is ExpTime-complete [89]. For the DLs underlying the OWL standard the complexity of testing consistency is even higher. Reasoning in the DL underlying the OWL 1.0 standard $\mathcal{SHOIQ}$ is NExpTime-complete [98] and for the DL $\mathcal{SROIQ}$, which is the basis for the OWL 2 standard, it is even N2ExpTime [64].

3.2 Reasoning in $\mathcal{EL}$

Since the DL $\mathcal{EL}$ does neither offer negation nor the bottom concept, contradictions cannot be expressed and thus testing satisfiability is trivial in $\mathcal{EL}$. For testing subsumption in $\mathcal{EL}$, it was shown in [25] that reasoning can be done in polynomial time. This result was rather surprising. For the very similar DL $\mathcal{FL}_0$, which allows for value restrictions instead of existential restrictions, reasoning w.r.t. general TBoxes is ExpTime-complete [46]. For a collection of extensions of $\mathcal{EL}$ it was investigated, whether they have the same nice computational properties [26, 4, 5]. These investigations identified extensions of $\mathcal{EL}$ that allow for efficient classification. The DL $\mathcal{EL}^{++}$ extends $\mathcal{EL}$ with the bottom concept ($\bot$), nominals, a restricted form of concrete domains, and a restricted form of so-called role-value maps. For this DL, it was shown in [5] that almost all additions of other typical DL constructors to $\mathcal{EL}$ make subsumption w.r.t. general TBoxes ExpTime-complete. The DL $\mathcal{EL}^{++}$ is the closest DL to the OWL 2 EL profile.

Despite its limited expressivity, $\mathcal{EL}$ is highly relevant for practical applications. In fact, both the large medical ontology SNOMED CT[3] and the Gene Ontology[4] can be expressed in $\mathcal{EL}$.

3.3 Subsumption in $\mathcal{EL}$

The polynomial time algorithm for computing subsumption w.r.t. a general TBox actually performs classification of the whole TBox, i.e., it computes the subsumption relationships between all named concepts of a given TBox simultaneously. This algorithm proceeds in four steps:

1. Normalize the TBox.
2. Translate the normalized TBox into completion sets.
3. Complete these sets using completion rules.
4. Read off the subsumption relationships from the normalized graph.

[3] http://www.ihtsdo.org/snomed-ct/
[4] http://www.geneontology.org/

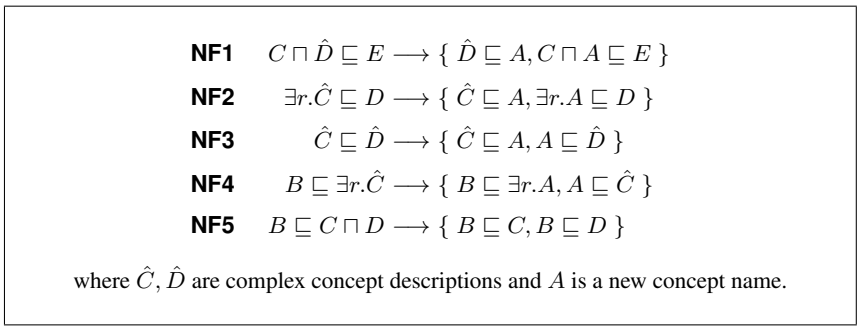

Fig. 2. $\mathcal{EL}$ normalization rules

The normal form for $\mathcal{EL}$-TBoxes required in the first step is defined as follows.

Definition 14 (Normal form for $\mathcal{EL}$-TBoxes). *An $\mathcal{EL}$-TBox $\mathcal{T}$ is in* normal form *if all concept inclusions have one of the following forms:*

$$A_1 \sqsubseteq B, \quad A_1 \sqcap A_2 \sqsubseteq B, \quad A_1 \sqsubseteq \exists r.A_2 \quad \text{or} \quad \exists r.A_1 \sqsubseteq B,$$

where A_1, A_2 and B are concept names appearing in $\mathcal{T}$ or the top-concept $\top$.

Any $\mathcal{EL}$-TBox $\mathcal{T}$ can be transformed into a normalized TBox $\mathcal{T}'$ by simply introducing new concept names. $\mathcal{EL}$-TBoxes can be transformed into normal form by applying the normalization rules displayed in Fig. 2 exhaustively. These rules replace the GCI on the left-hand side of the rule with the set of GCIs on the right-hand side of the rule. The idea behind the normalization rules is to introduce names for complex sub-concept descriptions. It suffices to obtain a TBox that is a subsumption-equivalent TBox to the original one, i.e., the original and the normalized TBox capture the same subsumption relationships for the named concepts from the original TBox. Thus it suffices to introduce the new concept names with GCIs instead of equivalences. The transformation into normal form can be done in linear time.

The completion algorithm works on a data-structure called *completion sets*. There are two kinds of completion sets used in the algorithm:

- $S(A)$ for each concept name A mentioned in the normalized TBox, and
- $S(A, r)$ for each concept name A and role name r mentioned in the normalized TBox.

Both kinds of completion sets contain concept names and $\top$. By $\mathsf{S}_\mathcal{T}$ we denote the set containing all completion sets of the TBox $\mathcal{T}$. In the completion algorithm, the completion sets are initialized as follows:

- $S(A) := \{A, \top\}$ for each concept name A mentioned in the normalized TBox, and
- $S(A, r) := \emptyset$ for each concept name A and role name r mentioned in the normalized TBox.

CR1	If $C' \sqsubseteq D \in \mathcal{T}$, $C' \in S(C)$, and $D \notin S(C)$ then add D to $S(C)$.
CR2	If $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, $C_1, C_2 \in S(C)$, and $D \notin S(C)$ then add D to $S(C)$.
CR3	If $C' \sqsubseteq \exists r.D \in \mathcal{T}$, $C' \in S(C)$, and $D \notin S(C, r)$ then add D to $S(C, r)$.
CR4	If $\exists r.D' \sqsubseteq E \in \mathcal{T}$, $D \in S(C, r)$, $D' \in S(D)$, and $E \notin S(C)$ then add E to $S(C)$.

Fig. 3. $\mathcal{EL}$ completion rules

The intuition is that the completion rules make implicit subsumption relationships explicit in the following sense:

- $B \in S(A)$ implies that $A \sqsubseteq_{\mathcal{T}} B$, i.e., $S(A)$ contains only subsumers of A, and
- $B \in S(A, r)$ implies that $A \sqsubseteq_{\mathcal{T}} \exists r.B$, i.e., $S(A, r)$ contains only concept names B s.t. A is subsumed by $\exists r.B$.

In fact, it can be shown that these properties of the completion sets are *invariants* and thus do not change during completion. Clearly, this holds for the initial elements of the completion. After initialization all completion sets in $\mathsf{S}_{\mathcal{T}}$ are extended by applying the completion rules that are shown in Fig. 3 exhaustively, i.e., until no more rule applies. It is easy to see that the rules preserve the above invariants. In each of the rules the last condition ensures that the rule is only applied once to the same concepts and completion sets. The first rule **CR1** propagates the transitivity of subsumption. The second **CR2** ensures that if a conjunction implies a concept C w.r.t. $\mathcal{T}$ and the conjuncts are already in the completion set of a concept, then C has to be in that completion set as well. The rule **CR3** is applicable if a concept name implies an existential restriction w.r.t. $\mathcal{T}$ and this concept name is contained in the completion set $S(C)$, then the existential restriction is implied by C as well. The most complicated rule is **CR4**. The axiom $\exists r.D' \sqsubseteq E \in \mathcal{T}$ implies $\exists r.D' \sqsubseteq_{\mathcal{T}} E$, and the assumption that the invariants are satisfied before applying the rule yields $D \sqsubseteq_{\mathcal{T}} D'$ and $C \sqsubseteq_{\mathcal{T}} \exists r.D$. The subsumption relationship $D \sqsubseteq_{\mathcal{T}} D'$ then implies $\exists r.D \sqsubseteq_{\mathcal{T}} \exists r.D'$. By applying transitivity of the subsumption relation $\sqsubseteq_{\mathcal{T}}$, we obtain $C \sqsubseteq_{\mathcal{T}} E$.

Once the completion process has terminated, the subsumption relation between two named concepts A and B can be tested by checking whether $B \in S(A)$. The fact that subsumption in $\mathcal{EL}$ w.r.t. general TBoxes can be decided in polynomial time follows from the following statements:

1. Rule application terminates after a polynomial number of steps.
2. If no more rules are applicable, then $A \sqsubseteq_{\mathcal{T}} B$ iff $B \in S(A)$.

The first statement holds, since the number of completion sets, of the kind $S(A)$ is linear in size of the TBox. In addition, the number of completion set of the kind $S(A, r)$ is quadratic in the size of $\mathcal{T}$. The size of the completion sets is bounded by the number of concept names and role names, and each rule application extends at least one label.

Theorem 1. *Subsumption in $\mathcal{EL}$ is polynomial w.r.t. general TBoxes.*

This nice computational property transfers also to $\mathcal{EL}^{++}$ [5], the DL corresponding closest to the OWL 2 EL profile.

The first implementation of the subsumption algorithm for $\mathcal{EL}$ sketched above is the CEL system [11, 71]. This system showed that the classification of the very large knowledge bases can be done in runtime acceptable for practical applications. For instance, classifying the knowledge base SNOMED CT, which contains more than 300.000 axioms takes less than half an hour and classification of the Gene Ontology, which contains more than 20.000 axioms, takes only 6 seconds [12].

4 Explanation of Reasoning Results

DL knowledge bases often contain thousands of axioms and have a complex structure due to the use of GCIs. These knowledge bases are developed by users who are experts in the domain to be modeled, but have little expertise in knowledge representation or logic. For this sort of applications, it is necessary that the development process of the knowledge base is supported by automated services implemented in the DL system.

Classical DL reasoning systems can detect that a certain consequence holds, such as an inconsistency or a subsumption relation, but they give no evidence *why* it holds. The reasoning service explanation facilitates better understanding of the knowledge base and gives a starting point to resolve an unwanted consequence in the knowledge base. For instance, the SNOMED ontology contains the subsumption relation:

$$\mathsf{Amputation\text{-}of\text{-}Finger} \sqsubseteq \mathsf{Amputation\text{-}of\text{-}Arm}.$$

A user who wants to correct this, faces the task of finding the axioms responsible for this unintended subsumption relation among 350.000 others. Clearly, automated support is needed for this task. A first step towards providing such support was described in [90], where an algorithm for computing all minimal subsets of a given knowledge base that have a given consequence is described. This approach was extended to expressive DLs in [83].

For a TBox $\mathcal{T}$ and a consequence c an *explanation* points to the "source" of the consequence, which is a subset of $\mathcal{T}$ that contributes to the consequence c. We call a *minimal axiom set* (MinA) a minimal subset (w.r.t. size) of a TBox $\mathcal{T}$, that has a certain consequence. *Axiom pinpointing* is the process of computing MinAs.

Example 1. Consider the following TBox:

$$\begin{aligned}
\mathcal{T}_{ex} = \{ \quad & \mathsf{Cat} \sqsubseteq \exists\, \mathsf{has\text{-}parent}.\, \mathsf{Cat}, && \text{I} \\
& \mathsf{Cat} \sqsubseteq \mathsf{Pet}, && \text{II} \\
& \exists\, \mathsf{has\text{-}parent}.\mathsf{Pet} \sqsubseteq \mathsf{Animal}, && \text{III} \\
& \mathsf{Pet} \sqsubseteq \mathsf{Animal} \quad \} && \text{IV}
\end{aligned}$$

For the TBox $\mathcal{T}_{ex}$, we find the consequence $\mathsf{Cat} \sqsubseteq_{\mathcal{T}_{ex}} \mathsf{Animal}$. The consequence holds since axiom I says that cats are pets and pets are in turn animals by axiom IV. This

consequence also follows from $\mathcal{T}_{ex}$ by using axiom I and axiom II, which together say that a cat has a parent that is a pet. Now from this together with axiom III it, follows that cats are animals. Thus, the one consequence has several MinAs, namely: $\{\text{I}, \text{IV}\}$ and $\{\text{I}, \text{II}, \text{III}\}$.

It turns out that there may be exponentially many MinAs, which shows that an algorithm for computing *all* MinAs needs exponential time in the size of the input TBox. In order to obtain an explanation for a consequence, we need to compute one single MinA of the consequence. There are two general approaches for pinpointing, i.e., computing a MinA of a consequence:

Black box approach, which uses a DL reasoner as an oracle, i.e, it repetitively queries the reasoner to compute a MinA.

Glass box approach, which modifies the internals of a DL reasoner s.t. it yields a MinA directly when computing an inference.

While the black box approach is independent of the reasoner, the glass box approach needs to be tailored to the reasoning method in use. We examine the black box approach first, which is the method of choice for expressive DLs, then we discuss the glass box approach for completion-based reasoning in $\mathcal{EL}$.

The task of computing explanations has also been considered in other research areas. For example, in the SAT community, people have considered the problem of computing minimally unsatisfiable subsets of a set of propositional formulae. Approaches for computing these sets developed there include algorithms that call a SAT solver as a black box [65, 20] but also algorithms that extend a resolution-based SAT solver directly [34, 103].

4.1 Black Box Method for Pinpointing

Assume we want to perform pinpointing for the consequence $A \sqsubseteq B$ w.r.t. the TBox $\mathcal{T}$. The basic idea underlying the black box method is a kind of uninformed search: Given a TBox $\mathcal{T}$ and the consequence $A \sqsubseteq B$: simply remove the first axiom from the TBox $\mathcal{T}$ and test whether the consequence still holds. If so, continue with the second axiom. If the consequence does *not* follow from the TBox with the first axiom removed, put the axiom back to the TBox and then test the second axiom. This naive method always performs as many subsumption tests as the number of axioms in the TBox. Since MinAs are often quite small, this is not a feasible method for very large TBoxes.

A more efficient method would not proceed axiom-wise, but first compute a not necessarily minimal subset of the TBox from which the consequence follows and then minimize this set using the naive procedure. This approach is only feasible if the algorithm for the first step produces fairly small sets of axioms and is efficient.

The black box method is independent of the DL in use and can be used to compute explanations for any DL, provided there is a DL reasoner for the DL and the consequence in question. This method can easily be implemented on top of a DL reasoner and does not require to change the internal structure of the reasoner. This is the reason why most implementations of pinpointing are based on the black box approach.

For $\mathcal{EL}$ the black box pinpointing algorithm has been implemented in the DL reasoning system CEL [16, 19, 97]. For a variant of the medical knowledge base GALEN [87] with 4000 axioms the overall run-time for computing a MinA with the non-naive method took 9:45 min. In contrast the naive method took seven hours for the same task. The first implementation of the black box method for pinpointing was done for the ontology editor SWOOP [62] based on the methods described in [83]. A more recent implementation of black box pinpointing was done in the ontology editor PROTÉGÉ. This implementation allows pinpointing even for *parts* of axioms that contribute to deriving a consequence [47].

4.2 Glass Box Pinpointing for $\mathcal{EL}$

The glass box approach for computing an explanation depends on the DL used and the reasoning method employed. It requires that the internals of a reasoner are modified by adding label sets to the reasoning procedure that collect the relevant axioms already during the computation of the consequence. For $\mathcal{EL}$, we modify the completion algorithm for subsumption from Section 3.3 to compute one explanation for a subsumption relationship. To this end, we annotate every element in the completion sets in S with a monotone Boolean formula that captures the MinAs.[5] The glass box algorithm for $\mathcal{EL}$ was described in [15] and extended in [16].

The *basic labeling* assigns to every GCI $t \in \mathcal{T}$ a unique propositional variable $lab(t)$ as a label. By $lab(\mathcal{T})$ we denote the set of all propositional variables labeling GCIs in the TBox $\mathcal{T}$. Now, a *monotone Boolean formula over $lab(\mathcal{T})$* is a Boolean formula using

- (some of) the variables in $lab(\mathcal{T})$, and
- only the connectives $\wedge$, $\vee$ and $true$ for truth.

Its propositional *valuation* (denoted ν) is the set of propositional variables that make the formula true when they are assigned the value true. For a valuation $\nu \subseteq lab(\mathcal{T})$, let $\mathcal{T}_\nu := \{t \in \mathcal{T} \mid lab(\mathcal{T}) \in \nu\}$. The idea is that the valuation characterizes a combination of axiom labels. These labels are mapped back to the actual axioms from the TBox $\mathcal{T}$ by $\mathcal{T}_\nu$.

Definition 15 (Pinpointing formula). *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and A and B concept names occurring in $\mathcal{T}$. The* monotone Boolean formula ϕ over $lab(\mathcal{T})$ *is a pinpointing formula for $\mathcal{T}$ w.r.t. $A \sqsubseteq_\mathcal{T} B$, if the following holds for every valuation $\nu \subseteq lab(\mathcal{T})$:*

$$A \sqsubseteq_{\mathcal{T}_\nu} B \text{ iff } \nu \text{ satisfies } \phi.$$

Consider Example 1 again. Take $lab(\mathcal{T}_{ex}) := \{\mathrm{I}, \mathrm{II}, \mathrm{III}, \mathrm{IV}\}$ as the set of propositional variables, then $\mathrm{II} \wedge (\mathrm{IV} \vee (\mathrm{I} \wedge \mathrm{III}))$ is a pinpointing formula for $\mathcal{T}_{ex}$ w.r.t. $A \sqsubseteq_{\mathcal{T}_{ex}} B$.

Lemma 1. *Let ϕ be a pinpointing formula for the TBox $\mathcal{T}$ w.r.t. $A \sqsubseteq_\mathcal{T} B$. If valuations are ordered by set inclusions, then*

[5] This method for generating explanations was first applied for default reasoning in [8].

$$M = \{\mathcal{T}_\nu \mid \nu \textit{ is a minimal valuation satisfying } \phi\}$$

is the set of all *MinAs for* $\mathcal{T}$ *w.r.t.* $A \sqsubseteq_\mathcal{T} B$.

Proof. We need to show the following claims:

1. M contains only MinAs.
2. There is no MinA m_1 s.t. $m_1 \notin M$.

Show claim 1.:
For each set of axioms $m \in M$ there is a valuation ν_m s.t. $\nu_m = lab(m)$, which is minimal in size and that satisfies ϕ. Since ϕ is satisfied, $A \sqsubseteq_\mathcal{T} B$ holds. Since ν_m is minimal there is no subset of ν_m satisfying ϕ, and thus m is a MinA.
Show claim 2.:
Assume m_1 is a MinA for $\mathcal{T}$ w.r.t. $A \sqsubseteq_\mathcal{T} B$ and $m_1 \notin M$. Since m_1 is a MinA, m_1 is minimal and $A \sqsubseteq_{m_1} B$ holds. Let ν_{m_1} be the valuation $\nu_{m_1} = lab(m_1)$. From $A \sqsubseteq_\mathcal{T} B$ follows ν_{m_1} satisfies the pinpointing formula ϕ. Thus, m_1 induces a minimal valuation satisfying ϕ, which is a contradiction to $m_1 \notin M$. ❑

Lemma 1 guarantees that it is enough to compute the pinpointing formula to obtain *all* MinAs, i.e., explanations for the consequence in question. However, to obtain one MinA from the pinpointing formula, one can transform the pinpointing formula into disjunctive normal form, remove those disjuncts that are implied by other disjuncts and then pick one disjunct as the explanation.

Next, we describe the computation algorithm for pinpointing formulae in $\mathcal{EL}$ based on completion. Again, we want to explain $A \sqsubseteq B$ w.r.t. the $\mathcal{EL}$-TBox $\mathcal{T}$. Since the completion algorithm starts by normalizing the TBox, we need to introduce the labels for the original TBox and labels for the normalized TBox $\mathcal{T}'$ as well. The labels of the normalized TBox $\mathcal{T}'$ need to "keep track" of the corresponding axioms in the original TBox.

The completion procedure needs to be adapted to propagate the labels and to construct the pinpointing formula. To this end, each element of the completion sets, say $X \in S(A)$, is labelled with a monotone Boolean formula: $lab(A, X)$. The initial elements of the completions sets $A \in S(A)$ and $\top \in S(A)$ are labelled with $true$, i.e., $lab(A, A) = lab(A, \top) = true$ for all concept names appearing in $\mathcal{T}$. Now, we need to modify the completion rules from Fig. 3. Let the precondition of a completion rule **CRi** be satisfied for a set of completion sets $\mathsf{S}_{\mathcal{T}'}$ w.r.t. the TBox $\mathcal{T}'$. The modified rule collects the labels of those GCIs and completion sets that make the rule **CRi** applicable. Let ϕ be the conjunction of :

- labels of GCIs in $\mathcal{T}'$ that appear in the precondition of **CRi**, and
- labels of elements in completion sets in $\mathsf{S}_{\mathcal{T}'}$ that appear in the precondition of **CRi**.

The conjunction collected in ϕ needs to be propagated to the consequence of the rule **CRi**. If the completion set element in the consequence of **CRi** is *not* in $\mathsf{S}_{\mathcal{T}'}$, then it is added with label ϕ. In case the consequence of **CRi** *is* already in $\mathsf{S}_{\mathcal{T}'}$ and has the label ψ, the completion algorithm has derived the consequence again. In this case, ψ and ϕ are compared. If $\psi \wedge \phi \not\equiv \psi$, the consequence of **CRi** is derived in an alternative way and

the label of this consequence is changed to $\phi \vee \psi$. The new label of the consequence is a more general Boolean formula. If $\psi \wedge \phi \equiv \psi$, then ϕ implies ψ. In this case the rule **CRi** is not applied.

Example 2. Consider Example 1 again. To compute the pinpointing formula for $\mathsf{Cat} \sqsubseteq_{\mathcal{T}_{ex}} \mathsf{Animal}$, the set of completion sets $\mathsf{S}_{\mathcal{T}_{ex}}$ is initialized as follows:

$$\begin{aligned}\mathsf{S}_{\mathcal{T}_{ex}} = \{\ & (\mathsf{Cat}, \top)^{true}, (\mathsf{Cat}, \mathsf{Cat})^{true}, \\ & (\mathsf{Pet}, \top)^{true}, (\mathsf{Pet}, \mathsf{Pet})^{true}, \\ & (\mathsf{Animal}, \top)^{true}, (\mathsf{Animal}, \mathsf{Animal})^{true} \ \}.\end{aligned}$$

Then we can apply the modified rules:

- Using axiom II: $\mathsf{Cat} \sqsubseteq \mathsf{Pet} \in \mathcal{T}_{ex}$ and $(\mathsf{Cat}, \mathsf{Cat})^{true} \in \mathsf{S}_{\mathcal{T}_{ex}}$, add $(\mathsf{Cat},\mathsf{Pet})^{\mathrm{II} \wedge true}$ to $\mathsf{S}_{\mathcal{T}_{ex}}$.
- Using axiom I: $\mathsf{Cat} \sqsubseteq \exists\, \mathsf{has\text{-}parent}.\, \mathsf{Cat} \in \mathcal{T}_{ex}$ and $(\mathsf{Cat}, \mathsf{Cat})^{true} \in \mathsf{S}_{\mathcal{T}_{ex}}$, add $(\mathsf{Cat}, \mathsf{has\text{-}parent}, \mathsf{Pet})^{\mathrm{I} \wedge true}$ to $\mathsf{S}_{\mathcal{T}_{ex}}$.
- Using axiom IV: $\mathsf{Pet} \sqsubseteq \mathsf{Animal} \in \mathcal{T}_{ex}$ and $(\mathsf{Cat},\mathsf{Pet})^{\mathrm{II} \wedge true} \in \mathsf{S}_{\mathcal{T}_{ex}}$, add $(\mathsf{Cat}, \mathsf{Animal})^{\mathrm{II} \wedge \mathrm{IV} \wedge true}$ to $\mathsf{S}_{\mathcal{T}_{ex}}$.
- Using axiom III: $\exists\, \mathsf{has\text{-}parent}.\mathsf{Pet} \sqsubseteq \mathsf{Animal} \in \mathcal{T}_{ex}$ and $\{(\mathsf{Cat}, \mathsf{Pet})^{\mathrm{II} \wedge true}, (\mathsf{Cat}, \mathsf{has\text{-}parent}, \mathsf{Pet})^{\mathrm{I} \wedge true})\} \subset \mathsf{S}_{\mathcal{T}_{ex}}$, modify $(\mathsf{Cat}, \mathsf{Animal})^{\mathrm{II} \wedge \mathrm{IV} \wedge true}$ to $(\mathsf{Cat}, \mathsf{Animal})^{(\mathrm{II} \wedge \mathrm{IV} \wedge true) \vee (\mathrm{III} \wedge \mathrm{II} \wedge \mathrm{I} \wedge true)}$.

Now, $lab(\mathsf{Cat}, \mathsf{Animal}) = (\mathrm{II} \wedge \mathrm{IV}) \vee (\mathrm{III} \wedge \mathrm{II} \wedge \mathrm{I})$ is the pinpointing formula for $\mathcal{T}_{ex}$ w.r.t. $\mathsf{Cat} \sqsubseteq_{\mathcal{T}_{ex}} \mathsf{Animal}$.

The modified completion algorithm always terminates, but not necessarily in polynomial time due to the possibility of repeated generalization of the label. Testing equivalence of monotone Boolean formulae is an NP-complete problem. However, given formulae over n propositional variables whose size is exponential in n, equivalence can be tested in time exponential in n. Thus, there are at most exponentially many rule applications and each of them takes at most exponential time. This yields an exponential time bound for the execution of the pinpointing algorithm.

However, the set of completion sets S obtained by the described process is identical to the one obtained by the unmodified algorithm. After the modified completion algorithm has terminated, the label $lab(A, B)$ is a pinpointing formula for $\mathcal{T}$ w.r.t. $A \sqsubseteq_{\mathcal{T}} B$.

Theorem 2. *Given an $\mathcal{EL}$-TBox $\mathcal{T}$ in normal form, the pinpointing algorithm terminates in time exponential in the size of $\mathcal{T}$. After termination, the resulting set of completion sets $\mathsf{S}_{\mathcal{T}}$ satisfies the following two properties for all concept names A,B occurring in $\mathcal{T}$:*

1. *$A \sqsubseteq_{\mathcal{T}} B$ iff $(S(A), B) \in \mathsf{S}_{\mathcal{T}}$, and*
2. *lab(A,B) is a pinpointing formula for $\mathcal{T}$ w.r.t $A \sqsubseteq_{\mathcal{T}} B$.*

This result was shown in [16] for the DL $\mathcal{EL}^{++}$. In the example, the TBox $\mathcal{T}_{ex}$ is already in normal form. In the general case, the TBox needs to be normalized and the pinpointing formula obtained by the modified completion needs to reconstruct the labels for the original axioms from the label of the normalized axioms.

The propositional variables from the normalized TBox in ϕ are replaced with those of the original one. More precisely, each label of a normalized GCI is replaced by the disjunction of its source GCIs. Once the de-normalized pinpointing formula is obtained, it is transformed into disjunctive normal form. One disjunct of this formula yields a MinA and thus an explanation of the consequence. To sum up, the *pinpointing extension* of the $\mathcal{EL}$ subsumption algorithm proceeds in the following steps:

1. Label all axioms in $\mathcal{T}$.
2. Normalize $\mathcal{T}$ according the rules from Fig. 2.
3. Label each axiom in the normalized TBox $\mathcal{T}'$ and keep the source GCI of every normalized GCI.
4. Apply the completion rules from Fig. 3 *modified* as described.
5. De-normalize the pinpointing formula.
6. Build the disjunctive normal form.
7. Pick one disjunct as explanation.

Note that the transformation into disjunctive normal form may cause an exponential blow-up, which means that, in some cases, the pinpointing formula provides us with a compact representation of the set of all MinAs. Also note that this blow-up is not in the size of the pinpointing formula but rather in the number of variables. Thus, if the size of the pinpointing formula is already exponential in the size of the TBox $\mathcal{T}$, computing all MinAs from it is still "only" exponential in the size of $\mathcal{T}$.

The glass box approach for pinpointing has also been investigated for more expressive DLs such as $\mathcal{ALC}$ in [72]. A more general view on tableaux and pinpointing was taken in [14].

We presented methods to obtain an explanation for a consequence. In order to actually repair a DL knowledge base, it is necessary to alleviate *all* causes of an unwanted consequence. In order to support users to repair a knowledge base, all MinAs need to be computed. The glass box method for $\mathcal{EL}$ computes all MinAs and can be employed for knowledge base repair directly. For the black box approach, a method for obtaining all MinAs is described in [90, 60]. This method computes the first MinA by the algorithms described above and then employs a method based on *hitting sets* to obtain the remaining MinAs.

The mechanism of pinpointing is not only useful for explanation or repair of DL knowledge bases. Access restrictions to knowledge bases can be supported as well [9]. If a user only has access to a part of the ontology, it is not obvious whether certain consequences can be accessed by the user as well. By computing all MinAs for the consequence, it can be tested whether the consequence follows from the accessible part alone. In that case access to the consequence does not violate the access restrictions.

Acknowledgement This article is based on the Description Logic tutorial by the author, which she taught at the 2009 Masters Ontology Spring School organized by the Meraka Institute in Tshwane (Pretoria), South Africa and it is based on the course material by Franz Baader at the 2009 Reasoning Web Summer School, see [2]. The author would like to thank the anonymous reviewers and Marcel Lippmann for valuable comments on earlier versions of this paper.

References

1. F. Baader. Description logic terminology. In *[6]*, pages 485–495. Cambridge University Press, 2003.
2. F. Baader. Description logics. In *Proceedings of Reasoning Web: Semantic Technologies for Information Systems*, volume 5689 of *Lecture Notes in Computer Science*, pages 1–39, 2009.
3. F. Baader, A. Bauer, P. Baumgartner, A. Cregan, A. Gabaldon, K. Ji, K. Lee, D. Rajaratnam, and R. Schwitter. A novel architecture for situation awareness systems. In M. Giese and A. Waaler, editors, *Proc. of the 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux 2009)*, volume 5607 of *Lecture Notes in Computer Science*, pages 77–92. Springer-Verlag, 2009.
4. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.
5. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In K. Clark and P. F. Patel-Schneider, editors, *In Proc. of the OWLED Workshop*, 2008.
6. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
7. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.
8. F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-92)*, pages 306–317. Morgan Kaufmann, Los Altos, 1992.
9. F. Baader, M. Knechtel, and R. Peñaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. *Proc. of the 8th International Semantic Web Conference (ISWC 2009)*, volume 5823 of *Lecture Notes in Computer Science*, pages 49–64, 2009.
10. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, 1999. Morgan Kaufmann, Los Altos.
11. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, volume 4130 of *Lecture Notes In Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006. CEL download page: `http://lat.inf.tu-dresden.de/systems/cel/`.
12. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Journal of Logic, Language and Information, Special Issue on Method for Modality (M4M)*, 2007.
13. F. Baader, C. Lutz, and A.-Y. Turhan. Small is again beautiful in description logics. *KI – Künstliche Intelligenz*, 24(1):25–33, April 2010.
14. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2010. Special Issue: Tableaux and Analytic Proof Methods.
15. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}$. In D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, S. Tessaris, and A.-Y. Turhan, editors, *Proc. of the 2007 Description Logic Workshop (DL 2007)*, CEUR-WS, 2007.

16. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of the 30th German Annual Conf. on Artificial Intelligence (KI'07)*, volume 4667 of *Lecture Notes In Artificial Intelligence*, pages 52–67, Osnabrück, Germany, 2007. Springer.
17. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
18. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logics*, 2007.
19. F. Baader and B. Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of the International Conference on Representing and Sharing Knowledge Using SNOMED (KR-MED'08)*, Phoenix, Arizona, 2008.
20. J. Bailey and P. J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In M. V. Hermenegildo and D. Cabeza, editors, *InProc. of Practical Aspects of Declarative Languages, 7th International Symposium (PADL 2005), USA*, LNCS, pages 174–186, 2005.
21. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, MA, USA, 2001.
22. A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.
23. R. J. Brachman, A. Borgida, D. L. McGuinness, and L. Alperin Resnick. The CLASSIC knowledge representation system, or, KL-ONE: the next generation. Preprints of the Workshop on Formal Aspects of Semantic Networks, Two Harbors, Cal., 1989.
24. R. J. Brachman and H. J. Levesque. *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos, 1985.
25. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. L. de Mantáras and L. Saitta, editors, *Proc. of the 16th European Conf. on Artificial Intelligence (ECAI-04)*, pages 298–302. IOS Press, 2004.
26. S. Brandt. Reasoning in $\mathcal{ELH}$ w.r.t. general concept inclusion axioms. LTCS-Report LTCS-04-03, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2004. See http://lat.inf.tu-dresden.de/research/reports.html.
27. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, D. McGuinness, and M.-A. Williams, editors, *Proc. of the 8th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-02)*, San Francisco, CA, 2002. Morgan Kaufmann Publishers.
28. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In M. M. Veloso and S. Kambhampati, editors, *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05)*, pages 602–607. AAAI Press/The MIT Press, 2005.
29. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.
30. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.
31. D. Calvanese and G. D. Giacomo. Expressive description logics. In *[6]*, pages 178–218. Cambridge University Press, 2003.
32. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, 31:273–318, 2008.

33. B. Cuenca Grau, Y. Kazakov, I. Horrocks, and U. Sattler. A logical framework for modular integration of ontologies. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, pages 298–303, 2007.
34. G. Davydov, I. Davydova, and H. K. Büning. An efficient algorithm for the minimal unsatisfiability problem for a subclass of cnf. *Ann. Math. Artif. Intell.*, 23(3-4):229–245, 1998.
35. F. M. Donini, S. Colucci, T. Di Noia, and E. Di Sciascio. A tableaux-based method for computing least common subsumers for expressive description logics. In *Proc. of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 739–745. AAAI, July 2009.
36. T. Eiter, C. Lutz, M. Ortiz, and M. Simkus. Query answering in description logics with transitive roles. In *Proc. of the 21st International Joint Conference on Artificial Intelligence IJCAI09*. AAAI Press, 2009.
37. M. Fitting. Basic modal logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 365–448. Oxford Science Publications, 1993.
38. S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? a case for conservative extensions in description logics. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-06)*, pages 187–197. AAAI Press, 2006.
39. B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In M. M. Veloso, editor, *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, pages 399–404, 2007.
40. E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.
41. E. Grädel, P. G. Kolaitis, and M. Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.
42. V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI-01)*, pages 161–166, 2001.
43. V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkov, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-01)*, Lecture Notes in Computer Science. Springer, 2001.
44. V. Haarslev and R. Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-04)*, pages 163–173, 2004.
45. V. Haarslev, R. Möller, and A.-Y. Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In R. Goré, A. Leitsch, and T. Nipkov, editors, *Proc. of the International Joint Conference on Automated Reasoning IJCAR'01*, LNAI. Springer Verlag, 2001.
46. M. Hofmann. Proof-theoretic approach to description-logic. In P. Panangaden, editor, *Proc. of the 20th Ann. IEEE Symp. on Logic in Computer Science (LICS-05)*, pages 229–237. IEEE Computer Society Press, 2005.
47. M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in OWL. In *ISWC 08 The International Semantic Web Conference 2008, Karlsruhe, Germany*, 2008.
48. I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
49. I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. Cohn, L. Schubert, and S. Shapiro, editors, *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 636–647, 1998.
50. I. Horrocks. Reasoning with expressive description logics: Theory and practice. In A. Voronkov, editor, *Proc. of the 19th Conf. on Automated Deduction (CADE-19)*, number 2392 in Lecture Notes In Artificial Intelligence, pages 1–15. Springer, 2002.

51. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-06)*, pages 57–67. AAAI Press, 2006.
52. I. Horrocks and P. Patel-Schneider. Optimizing description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
53. I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
54. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
55. I. Horrocks and U. Sattler. Optimised reasoning for $\mathcal{SHIQ}$. In *Proc. of the 15th European Conference on Artificial Intelligence*, 2002.
56. I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*. Morgan Kaufmann, Jan. 2005.
57. I. Horrocks and U. Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *J. of Automated Reasoning*, 39(3):249–276, 2007.
58. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.
59. U. Hustadt, R. A. Schmidt, and L. Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1:251–276, 2004.
60. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of owl dl entailments. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Korea, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280, 2007.
61. A. Kalyanpur, B. Parsia, E. Sirin, and B. Cuenca Grau. Repairing unsatisfiable concepts in owl ontologies. In Y. Sure and J. Domingue, editors, *Proc. of the 3rd European Semantic Web Conf. (ESWC'06)*, volume 4011 of *Lecture Notes in Computer Science*, pages 170–184. Springer, 2006.
62. A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca Grau, and J. Hendler. Swoop: A web ontology editing browser. *J. Web Sem.*, 4(2):144–153, 2006.
63. A. Kalyanpur, B. Parsia, E. Sirin, and J. A. Hendler. Debugging unsatisfiable classes in OWL ontologies. *J. Web Sem.*, 3(4):268–293, 2005.
64. Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In G. Brewka and J. Lang, editors, *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-08)*, pages 274–284. AAAI Press, 2008.
65. M. H. Liffiton and K. A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.
66. C. Lutz. Complexity of terminological reasoning revisited. In *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, Lecture Notes in Computer Science, pages 181–200. Springer, 1999.
67. C. Lutz. The complexity of conjunctive query answering in expressive description logics. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proc. of the 4th International Joint Conference on Automated Reasoning (IJCAR2008)*, number 5195 in LNAI, pages 179–193. Springer, 2008.
68. C. Lutz, U. Sattler, and F. Wolter. Description logics and the two-variable fragment. In D. McGuiness, P. Pater-Schneider, C. Goble, and R. Möller, editors, *Proc. of the 2001 International Workshop in Description Logics (DL-2001)*, pages 66–75, Stanford, California, USA, 2001.
69. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*. AAAI Press, 2007.

70. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation*, 45(2):194 – 228, 2010.
71. J. Mendez and B. Suntisrivaraporn. Reintroducing cel as an owl 2 el reasoner. In B. Cuenca Grau, I. Horrocks, B. Motik, and U. Sattler, editors, *Proc. of the 2008 Description Logic Workshop (DL 2009)*, volume 477 of *CEUR-WS*, 2009.
72. T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI'06)*. AAAI Press/The MIT Press, 2006.
73. M. Minsky. A framework for representing knowledge. Technical report, MIT-AI Laboratory, Cambridge, MA, USA, 1974.
74. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe, 2006.
75. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. Owl 2 web ontology language profiles. W3C Recommendation, 27 October 2009. `http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/`.
76. B. Motik and U. Sattler. A Comparison of Techniques for Querying Large Description Logic ABoxes. In M. Hermann and A. Voronkov, editors, *Proc. of the 13th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR'06)*, LNCS, Cambodia, 2006. Springer. KAON2 download page: `http://kaon2.semanticweb.org/`.
77. B. Motik, R. Shearer, and I. Horrocks. A hypertableau calculus for $\mathcal{SHIQ}$. In D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, S. Tessaris, and A.-Y. Turhan, editors, *Proc. of the 2007 Description Logic Workshop (DL 2007)*, 2007.
78. B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In F. Pfennig, editor, *Proc. of the 23th Conf. on Automated Deduction (CADE-23)*, LNAI, pages 67–83, Germany, 2007. Springer.
79. B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence Journal*, 34(3):371–383, 1988.
80. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence Journal*, 43:235–249, 1990.
81. B. Nebel and K. von Luck. Hybrid reasoning in BACK. In *Proc. of the 3rd Int. Sym. on Methodologies for Intelligent Systems (ISMIS-88)*, pages 260–269. North-Holland Publ. Co., Amsterdam, 1988.
82. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008.
83. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL Ontologies. In A. Ellis and T. Hagino, editors, *Proc. of the 14th Int. World Wide Web Conference (WWW2005)*, pages 633–640, Japan, 2005.
84. P. Patel-Schneider, D. L. McGuinness, R. J. Brachman, L. A. Resnick, and A. Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113, 1991.
85. M. R. Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967. Republished in [24].
86. Racer Systems GmbH & Co. KG. *RacerPro Reference Manual Version 1.9*, Dec. 2005. Available from: `http://www.racer-systems.com/products/racerpro/reference-manual-1-9.pdf`.
87. A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, 1997.
88. U. Sattler, D. Calvanese, and R. Molitor. Relationships with other formalisms. In *[6]*, pages 137–177. Cambridge University Press, 2003.

89. K. Schild. A correspondence theory for terminological logics: preliminary report. In J. Mylopoulos and R. Reiter, editors, *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, 1991.
90. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 355–362, Mexico, 2003. Morgan Kaufmann, Los Altos.
91. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with unions and complements. Technical Report SR-88-21, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1988.
92. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence Journal*, 48(1):1–26, 1991.
93. E. Sirin and B. Parsia. Pellet system description. In B. Parsia, U. Sattler, and D. Toman, editors, *Description Logics*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
94. J. F. Sowa, editor. *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, 1991.
95. J. F. Sowa. *Encyclopedia of Artificial Intelligence*, chapter Semantic Networks. John Wiley & Sons, New York, 1992.
96. T. Springer and A.-Y. Turhan. Employing description logics in ambient intelligence for modeling and reasoning about complex situations. *Journal of Ambient Intelligence and Smart Environments*, 1(3):235–259, 2009.
97. B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. PhD thesis, Fakultät Informatik, TU Dresden, 2009. http://lat.inf.tu-dresden.de/research/phd/#Sun-PhD-2008.
98. S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, May 2000.
99. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, 2006. FaCT++ download page: `http://owl.man.ac.uk/factplusplus/`.
100. D. Tsarkov, I. Horrocks, and P. F. Patel-Schneider. Optimising terminological reasoning for expressive description logics. *Journal of Automated Reasoning*, 2007.
101. A.-Y. Turhan. *On the Computation of Common Subsumers in Description Logics*. PhD thesis, TU Dresden, Institute for Theoretical Computer Science, 2007.
102. W3C OWL Working Group. Owl 2 web ontology language document overview. W3C Recommendation, 27th October 2009. `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`.
103. L. Zhang and S. Malik. Validating sat solvers using an independent resolution-based checker: Practical implementations and other applications. In *Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pages 10880–10885. IEEE Computer Society, 2003.

Journal of Ambient Intelligence and Smart Environments 1 (2009) 1–5
IOS Press

Employing Description Logics in Ambient Intelligence for Modeling and Reasoning about Complex Situations

Thomas Springer and Anni-Yasmin Turhan [a]
[a] *TU Dresden, Faculty of Computer Science, Chair for Computer Networks/Chair for Automata Theory*
E-mail: thomas.springer@tu-dresden.de, turhan@tcs.inf.tu-dresden.de

Abstract.
Ambient Intelligence systems need to represent information about their environment and recognize relevant situations to perform appropriate actions proactively and autonomously. The context information gathered by these systems comes with imperfections such as incompleteness or incorrectness. These characteristics need to be handled gracefully by the Ambient Intelligence system. Moreover, the represented information must allow for a fast and reliable recognition of the current situation.

To solve these problems we propose a method for situation modeling using the Description Logics based ontology language OWL DL and a framework for employing Description Logics reasoning services to recognize the current situation based on context. The benefits from the approach are manifold: the semantics of Description Logics allow for graceful handling of incomplete knowledge. The well-investigated reasoning services do not only allow recognizing the current situation, but also can add to the reliability of the overall system. Moreover optimized reasoning systems are freely available and ready to use.

We underpin the feasibility of our approach by providing a case study based on a smart home application conducting an evaluation of different Description Logics reasoners with respect to our application ontology as well as a discussion of Description Logics systems in Ambient Intelligence.

Keywords: Situation-Awareness, Description Logics, Reasoning services, OWL DL, Modeling context information

1. Introduction

Research on Ubiquitous Computing and especially Ambience Intelligence (AmI) aims at creating systems able to interact in an intelligent way with the environment, especially the user. Weiser characterized this kind of system as: "machines that fit the human environment instead of forcing humans to enter theirs will make using a computer as refreshing as a walk in the woods" [53].

A system able to recognize the environment's state can adjust its behavior according to that state. For instance, an application for supporting mobile workers during their tasks in the field could adapt the input and output modalities to improve the interaction with the user. Speech input and output could be used if the workers' hands are not free, or gesture input could be used if the surrounding noise level is very high. In a similar way an assistance application for elderly people could intelligently support planning of daily activities like selecting convenient connections of public transportation systems for carrying out shopping activities, visiting the doctor or meeting relatives or friends.

To realize such systems, they have to be able to capture information about the environment and the involved users based on heterogeneous and distributed information sources, mainly sensors but also extracted application data, user monitoring or other methods for gathering context information. The information captured in this way is usually low-level and has to be aggregated and abstracted to create a higher-level representation of the overall situation a system is currently in. Only the recognition of complex situations enables

systems of this kind to operate in a more autonomous, adaptive and intelligent way.

Thus, ambient intelligent systems should be aware of the current situation. We understand *situation-awareness* as the ability of a system to logically aggregate a type of a situation from a complex set of features of the system's context. The derived situation type should be meaningful to the system in the sense that it can adjust its behavior in a defined way to the current situation.

A generalized view on situation-aware systems based on knowledge-based systems as we use it in this paper is depicted in Figure 1 introducing four processing phases. The first phase of the recognition of a situation starts with the capturing of information about the environment using sensor devices. In the second phase, this information is aggregated and abstracted by different operations performed by the sensing devices or components of a context service. The third phase adds the preprocessed data in an adequate format to the knowledge base. Then situation types are inferred from an updated knowledge base. In the fourth phase, the application triggers appropriate actions based on the situation types inferred in phase three.

This process involves approaches from several areas of computer science, namely pervasive and ubiquitous computing, context awareness and artificial intelligence, as it was stated in [40] and [36]. Sensors, actuators and other miniaturized, low-cost computing devices installed in buildings, devices or even clothes and the human body help to capture usually low-level, physical information about the environment like temperature, light intensity, or blood pressure. These technologies are mainly adopted in the first phase. Systems developed for context-awareness aim at generalizing the access to these heterogeneous information sources and to apply technologies for aggregating and abstracting the sensed information to context information usable in applications [18,13,45]. Context models are created to establish a shared understanding between all providers of context information, the context middleware and the context-aware applications [19,48]. The cross-system sharing of context and the modeling of different context features, especially the quality of information are further goals of context models [14,41]. To sum up, context-aware systems provide a set of context information that has to match a context model.

The preprocessing of sensor data is only half the way to situation-aware systems. Classical AI methods come into play in phase two and three of situation-aware systems as described in Figure 1. The second phase uses AI methods as feature extraction and approaches for classification and aggregation of sensed data to obtain higher-level context information, as described in [27] and [32]. In the third phase, different AI techniques could be used to derive a situation type from a complex set of context features. While approaches like Neuronal Networks or Bayesian Networks could be adopted in the whole process from sensor data abstraction up to decision making, we focus on AI systems with explicit representation of knowledge as defined in Figure 1. A major advantage of such systems is the reproducibility of inferences and decisions.

In phase four, the AmI system derives a decision from the recognized situation types. The decoupling of situation recognition and decision making enables a clear separation of concerns and introduces flexibility for the AmI system. Especially, the association between situation types and actions to be triggered can be adjusted at run-time.

In this article, we focus on phase three, the recognition of situation types. This phase comprises the setup of a knowledge base containing a description of the situations to be considered in the application and the reasoning about the situation based on that description involving context as information about the current situation. To this end we discuss the role of context for this kind of systems.

1.1. The role of context in situation-awareness

Context is usually implicit information which has to be sensed or gathered from distributed and heterogeneous sources in order to be usable in applications. Caused by the way context is gathered and as well as by the fact that context reflects the state of a highly dynamic environment, it has special characteristics which influence the modeling and processing of context. A certain piece of information which models a certain aspect of the physical environment is called a *context feature*. We can identify certain characteristics of single context features:

Quality: Context represents a model of aspects of the environment abstracting from the real world. Dependent on the method used for gathering or abstracting context features, they have a certain quality. The quality comprises the relevance of the gathered information in a certain situation or related to a certain entity, the accuracy of a measured value and the probability of a derived context feature.

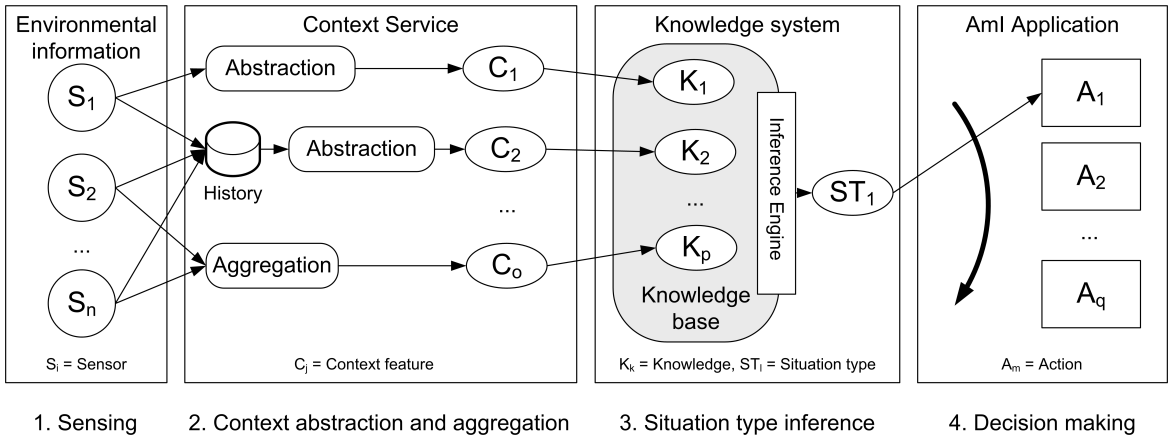

Fig. 1. Conceptual architecture of situation-aware systems.

Incorrectness: Since context features are sensed, extracted or derived, failures in measurement or wrong assumptions for derivation may lead to incorrect information. A system or user can also provide incorrect information to attack context-based authentication and authorization mechanisms.

Multiple sources: Moreover, a certain context feature can be gathered from multiple alternative sources. For instance, location information can be gathered by the client, either using a GPS device, a WLAN-based approach, or by the environment.

Heterogeneous sources: Because of its nature, context is in most cases not explicitly available but has to be gathered from heterogeneous sources. A context source can be any type of sensor system, database, or monitoring component. It can also be derived from user input or application data (see [45]).

In contrast to a single context feature, a set of context features has the following characteristics:

Inconsistency: A set of context features relevant for a certain application can be inconsistent because of contradictory information.

Incompleteness: It can further be incomplete due to the unavailability of certain context sources.

Multidimensional: Context information is multidimensional because it reflects heterogeneous information about the state of a highly dynamic environment comprising physical and technical information as well as personal information like activity, user preferences, social relations or business-related information.

Different abstraction levels: Information from different sources might be available at different levels of abstraction. While a sensor system provides low-level information sensed from the physical environment like temperature or light intensity, application data is usually available on a more abstract level like contact information of a person.

These characteristics of context features require AmI systems that can handle this kind of imperfect information gracefully. More precisely, an AmI system must be able to integrate context information and to deliver a "consistent" view of the current situation. Based on this view its main task is to recognize the type of the current situation in order to invoke the appropriate actions – even if the provided information is incomplete. While context information is usually provided at different abstraction levels, it has to be integrated into the knowledge base seamlessly. In addition, inconsistencies in the knowledge base could occur due to incorrect context and have to be detected by the system. Moreover, AmI systems have to be predictable, reliable and the selection of actions should be transparent to the developer and the user.

In this article, we present an approach to and a framework for modeling of complex situations and inferring the type of the current situation based on context information integrated into the situation model. We use the DL underlying OWL DL as the knowledge representation formalism and realize the task of situation type recognition by the use of Description Logic reasoning services. OWL DL is a formalism with clear semantics that offers reasoning services that remedy the above mentioned problems, especially the handling of incomplete knowledge, the discovery of inconsis-

tencies and the integration of context information at different levels of abstraction. The reasoning services in use are well-investigated. Moreover, sound, complete and terminating reasoning algorithms are available for the Description Logic underlying OWL DL. In addition, these methods are implemented in highly optimized reasoning systems which are freely available. By adopting a standard ontology language and off-the-shelf reasoning tools one can significantly reduce the development overhead and achieve faster prototyping. In addition to the already tested reasoning services of DL reasoners the AmI systems created according to the presented approach are very robust and reliable. Based on a case study and performance measurements we will demonstrate the advantages and limitations of our approach which employs DL systems for the realization of Ambient Intelligence.

In other research work OWL DL is mainly used for two purposes. Firstly, ontologies are created to establish a shared understanding between different components or even systems about the context information which is exchanged between them. Examples are the Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) and the CONtext ONtology (CONON) [14,19]. In both approaches a common context vocabulary is defined based on a hierarchy of ontologies. An upper ontology defines general terms while domain-specific ontologies define the details for certain application domains. Secondly, context or situation descriptions are employed in combination with different reasoning schemes to derive higher-level context or even situations from context.

Often these solutions adopt rule-based reasoning which could lead to undecidability as it is the case for [14], [15] and [19]. Other approaches for situation recognition adopt OWL DL as the base formalism for the knowledge base, but either adopt the formalism to their needs – as in [2] fuzzy logic and in [37] first order logic was used – or use other reasoning mechanisms than DL systems for the recognition task , e.g. in [47] decision trees were used. Closer to the work presented here comes the approach presented in [28], where TBox classification of OWL DL ontologies was applied to solve situation recognition. ABox realization as the means for recognizing situations is for instance used in [31].

While the use of a standard ontology language and off-the-shelf reasoning tools ensures soundness, completeness and decidability the approach has also some limitations. Situations are described by concepts in the TBox at design time and thus, have to be known to and explicitly modeled by the system developer. Only the context features relevant for situation detection are filled dynamically into the ABox at runtime. Hence, the situations which can be detected by the AmI system are limited to the situations modeled by the system developer at design time. Adding new situation descriptions or changing existing one's means changing the TBox at design time.

Systems like Bayesian networks or decision trees are able to model and especially to learn the associations between context features and situation types from a set of training examples and can thus, also be used to build situation-aware systems. A clear advantage of such approaches is that the developer does not have to specify the complete knowledge about situations manually, since it can be learned by the system based on examples. Major drawbacks of these approaches are that usually a large set of training examples is required to adjust the system and that the situation model is implicit so that system decisions are hardly reproducible.

In addition, probabilistic approaches are able to deal with uncertain knowledge by attaching a probability to all inferred values. With ontologies and Description Logics just facts can be handled.

In the following, we first give an overview of OWL DL, Description Logic and their reasoning services. Then we show how to model complex situations based on OWL DL. Especially we present a systematic modeling method by the composition of basic situations to complex scenarios. Next we describe how to apply DL reasoners for recognizing situation types from context features. By the use of a case study from the smart home domain we demonstrate the feasibility of our approach and give modeling examples. We assess the usefulness of DL systems for realizing AmI systems with respect to the expressivity of the formalism, the handling of imperfect context information and the performance of the DL reasoners and provide an evaluation of today's available Description Logic reasoners. We end the article with a summary and an outlook to future trends.

2. Description Logics for situation-aware systems

Description Logics (DLs) are a family of knowledge representation formalisms. The main asset of DLs is two-fold: on the one hand they are based on formal semantics and on the other they come with powerful reasoning services. These reasoning services make facts that are captured only implicitly in the represented

knowledge explicit. Most of the DL reasoning services are nowadays readily available in tools.

Historically, DLs stem from semantic networks [33, 44], which were introduced as a graphical knowledge representation formalism. Early versions of semantic networks lacked a clear definition of the meaning of the formalism. Thus reasoning algorithms developed for this kind of knowledge representation depended on the understanding of the developer. As a consequence implementations of the reasoning algorithms delivered different results for the same semantic network. To remedy this problem DLs were introduced as representation formalisms equipped with formal semantics [6], which then allowed to give formal definitions of their reasoning services. This in turn is the basis for implementations to be used in practical applications, which deliver predictable and reliable results.

DL research focuses on algorithms for DL reasoning services, the analysis of these algorithms in terms of computational complexity and the development of efficient implementations. For instance in case of subsumption tests, algorithms for a whole range of DLs has been investigated, see [7,25,23]. Moreover, there is a collection of very efficient DLs systems available that implement the investigated algorithms in optimized ways, see for instance [21,5,50,43]. These DL systems are employed in many different practical applications. The most prominent application area for DLs is the bio-medical domain, where the huge terminologies are built to represent facts from the biological domain as, for instance, genomic information [8,55] or from the medical domain, such as anatomy facts or medical procedures, see [38,17,16,39]. In the bio-medical domain, the modeling of the domain knowledge in a formal representation language is a benefit in itself, since this formalizes and to some extent standardizes what the community understands about certain terms in an unambiguous way. Thus by agreeing upon an ontology and the definition of concepts in it, a community can create a shared understanding of their domain of study as [56]. The obtained ontologies simply serve as a community knowledge reference and thereby remove heterogeneity in the community.

In the last couple of years the application area of the *Semantic Web* [11] brought more attention to DLs and their powerful reasoning systems. The semantic web is a future version of today's World Wide Web, where web content or services will be annotated with formal representation of their meaning. The annotation can state in which context a keyword is used. Based on a formal description of the keywords in an ontology, better search results or matching services can be obtained using reasoning methods. Despite a future vision, the semantic web is already a strong motivation for the development of powerful reasoning algorithms for very expressive DLs on the one hand and for the implementation of DL systems and DL tools on the other, see [21,20].

An important step towards realizing the semantic web was the standardization of the web ontology language OWL [10,24] and its DL-based dialect OWL DL, which we will discuss in more detail in Section 2.1.3 and which was used in our application.

In the remainder of this section we give a brief introduction to the main ingredients of a Description Logic system. We introduce some of the reasoning services employed in practice and emphasize those used in our application of context-aware systems.

2.1. Description Logics and their reasoning services

Typically, a Description Logic system consists of four parts:

1. *Description Language* formalism for capturing the notions from the domain.
2. *TBox*: a collection of concepts, which capture the main categories of the domain of interest,
3. *ABox*: a collection of facts about concrete instances in the application domain, and
4. the reasoning component.

The elements in TBox and ABox are formulated in the description language. Typically the TBox is also referred to as *ontology*. However, in the OWL lingo, where individuals are allowed in the TBox in the form of nominals, the term OWL ontology can refer to TBox *and* ABox collectively. We use both terms interchangeably throughout the paper. The TBox and ABox together are often referred to as the *knowledge base*. The concept descriptions in the knowledge base are given in the actual description logic. Next, we take a brief look at the syntax and semantics of the description logic $\mathcal{ALC}$. For a thorough introduction to Description Logics we refer the reader to [6].

2.1.1. DL knowledge bases

The main ingredients for representing terminological knowledge are *concept descriptions*. For example, such a concept description can characterize the category of 'mother' as a female person who has a person as a child, in the following way:

$$\texttt{Person} \sqcap \texttt{Female} \sqcap \exists \texttt{has-child}.\texttt{Person}.$$

In this expression `Person` and `Female` are *concepts* and `has-child` is a so-called *role* — a binary relation.

Starting from a set of primitive names, complex concept descriptions can be composed by using *concept constructors*. In Table 1 we see the concept constructors provided by the DL $\mathcal{ALC}$. $\mathcal{ALC}$ is the minimal propositionally closed DL. $\mathcal{ALC}$ provides negation, conjunction and disjunction of concepts. Furthermore, it provides existential restrictions and value restrictions. Intuitively, existential restrictions state that for every individual i_1 which belongs to the concept $\exists r.C$, there is an individual i_2 that is of concept C and i_1 and i_2 are related via the role r. Value restrictions state that for every individual i_1 which belongs to the concept $\forall r.C$, *all* individuals that are related to i_1 via the role r belong to concept C.

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

Starting from an interpretation of concept and role names, the extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is defined inductively, as shown in the third column of Table 1.

The TBox allows to introduce names for concept descriptions. For instance we can store

$$\texttt{Mother} \equiv \texttt{Person} \sqcap \texttt{Female} \sqcap \exists \texttt{has-child.Person}$$

as a concept definition in the TBox. Besides definitions there are other forms of TBox statements.

Definition 1 (TBox axioms) *Let A be a concept name and C and D be concept descriptions, then a*

primitive concept definition *is an expression of the form $A \sqsubseteq C$.*
concept definition *is an expression of the form $A \equiv C$.*
general concept inclusion (GCI) *is an expression of the form $C \sqsubseteq D$.*
concept equivalence *is an expression of the form $C \equiv D$.*

All of the above statements are called TBox axioms. *The semantics of TBox axioms are given by the interpretation function: $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, $A^{\mathcal{I}} = C^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ or $C^{\mathcal{I}} = D^{\mathcal{I}}$ respectively.*

Primitive concept definitions and GCIs give only necessary conditions while concept definitions and concept equivalence axioms state necessary *and* sufficient conditions for the concept. Obviously, GCIs are the most general type of concept axioms.

If in a concept definition $A \equiv D$ the concept description D refers directly or indirectly to the concept name A, we call such a concept A a *cyclic concept*. Based on this, we define different notions of a TBox.

Definition 2 (TBox) *Let A, B be concept names and let C, D be concept descriptions. A finite set of TBox axioms $\mathcal{T}$ is called a* TBox.

unfoldable TBoxes *only contain (primitive) concept definitions, where each name appears at most once on the left-hand side of a (primitive) definition and the TBox must be acyclic, i.e. without cyclic concepts.*
cyclic TBoxes *may contain cyclic (primitive) concept definitions.*
general TBoxes *may contain GCIs and concept equivalence.*

An interpretation is a model *of a TBox, if for all $A \sqsubseteq C \in \mathcal{T}$, $A \equiv C \in \mathcal{T}$, $C \sqsubseteq D \in \mathcal{T}$ and $C \equiv D \in \mathcal{T}$ it holds that $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, $A^{\mathcal{I}} \equiv C^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $C^{\mathcal{I}} = D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ iff $\mathcal{I}$ satisfies every axiom in $\mathcal{T}$. In this case $\mathcal{I}$ is a* model of $\mathcal{T}$.

OWL supports all of the above mentioned TBox axioms for modeling. The kind of TBox axioms a TBox contains has great effect on the computational complexity of reasoning methods for TBoxes – unfoldable TBoxes are often easier to handle than general ones.

Besides concept constructors many DLs provide means to declare properties of roles in the TBox. For instance, roles can be declared to be

- a *transitive role*, which is interpreted as a transitive relation.
- the *inverse role* of another role *inverse(R_1, R_2)*, which are interpreted as $R_2^{\mathcal{I}} = \{(a, b) \mid (b, a) \in R_1^{\mathcal{I}}\}$.
- a *super-role* of another one. Role inclusion axioms $R \sqsubseteq S$ enforce that every pair $(a, b) \in R^{\mathcal{I}}$ is also $(a, b) \in S^{\mathcal{I}}$. The set of these kind of statements forms the *role hierarchy*.

All of these role declarations are supported in OWL DL to build ontologies.

The knowledge about individual entities from the application domain can be expressed by so-called *ABox assertions*. For instance, we can state that the in-

Table 1
DL syntax and semantics of $\mathcal{ALC}$-concept descriptions and the corresponding OWL syntax.

constructor name	DL syntax	semantics	OWL syntax
negation	$\neg C$	$\Delta \setminus C^{\mathcal{I}}$	complementOf
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	intersectionOf
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	unionOf
existential restriction	$\exists r.C$	$\{x \in \Delta \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	someValuesFrom
value restriction	$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	allValuesFrom

dividual Alice is a software programmer and she has the colleague Fred in the following way:

`SoftwareProgrammer`(Alice),
`hasColleague`(Alice, Fred)

There are two kinds of ABox assertions used for DL systems—one kind expresses that an individual belongs to a concept and the other one specifies that two individuals are related via a role.

Definition 3 (ABox, ABox assertion) *Let C be an arbitrary concept description, r a role name and i, j two individual names be two individual names. Then a*

concept assertion *is a statement of the form $C(i)$.*
role assertion *is a statement of the form $r(i, j)$.*

An ABox $\mathcal{A}$ is a set of concept assertions and role assertions.

In order to capture ABoxes the interpretation function is extended to individual names, which are mapped to elements of the domain Δ.

Definition 4 (Semantics of ABox) *Let C be an arbitrary concept, r be a role name and i, j be two individuals. Then an interpretation $\mathcal{I}$ satisfies*

- *the concept assertion $C(i)$ iff $i^{\mathcal{I}} \in C^{\mathcal{I}}$ and*
- *the role assertion $r(i, j)$ iff $(i^{\mathcal{I}}, j^{\mathcal{I}}) \in r^{\mathcal{I}}$.*

An interpretation $\mathcal{I}$ satisfies an ABox $\mathcal{A}$ iff $\mathcal{I}$ satisfies every assertion in $\mathcal{A}$. In this case $\mathcal{I}$ is a model of $\mathcal{A}$.

Equipped with the formalisms for TBoxes and ABoxes and their meaning, we can turn to the reasoning services available once we have represented our information in this way.

2.1.2. DL reasoning services

Often the statements in the knowledge base capture other facts implicitly. To detect these facts and to make them explicit is the idea behind DL reasoning services. Furthermore the reasoning algorithms that infer new facts must fulfill certain requirements to ensure that applications using these services are reliable. To begin with, reasoning algorithms should be sound and complete, i.e. every answer returned by the service must be correct and we get all the answers. Furthermore the reasoning algorithm must be terminating so that an answer is always obtained. Based on the formal semantics of DLs, reasoning services can be defined and, more interestingly for our application, the requirements for reasoning algorithms that provide these services can be proven. These requirements hold for the inference algorithms developed for DLs, and they are implemented in today's typical DL systems. In the following we introduce some of the reasoning services that are readily available in DL systems and that we have used in our application of context-aware systems.

When adding a concept definition to a TBox, it is crucial to know whether the specified concept description contains a contradiction (w.r.t. the knowledge base) or whether it could be fulfilled by any individual and thus models something possibly meaningful. This leads to the formal notion of consistency.

Definition 5 (Consistency) *Let C be a concept description and $\mathcal{T}$ be a TBox. The concept description C is* consistent *iff it has a model, i.e., there exists an interpretation $\mathcal{I}$ where $C^{\mathcal{I}} \neq \emptyset$. In this case $\mathcal{I}$ is a model of C. A TBox $\mathcal{T}$ is* consistent *iff every concept in $\mathcal{T}$ is consistent.*

If a concept or TBox is not consistent, it is called *inconsistent*. Even for very expressive DLs inconsistencies can be detected automatically by the DL reasoner.

Another terminological inference task is to determine whether one concept description is more general than another, i.e., whether one concept description C is implied by another concept description D. This is the case, if every individual that is an instance of C also is an instance of D. The following definition formalizes this notion of subsumption.

Definition 6 (Subsumption) *Let C, D be concept descriptions and $\mathcal{T}$ a TBox. The concept description C*

is subsumed w.r.t. $\mathcal{T}$ *by the concept description* D *(*$C \sqsubseteq_{\mathcal{T}} D$*), iff* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *holds in every model* $\mathcal{I}$ *of* $\mathcal{T}$.

By *TBox classification* we denote the computation of all the subsumption relationships that hold for the named concepts in a TBox. By testing for subsumption relationships or by classifying the whole TBox the modeler can determine whether relations between notions from the domain are faithfully captured in the TBox. For instance, if two concepts, which stand for different notions in the application domain, turn out to be equivalent in the TBox, more information needs to be provided to distinguish them by means of their descriptions. Thus the subsumption tester in DL systems can help to spot these modeling deficiencies. This kind of test was helpful when we built our KB for the application scenario that we will describe in Section 3.1.

Beyond the concepts in the TBox there are also reasoning services available for the individuals in the ABox.

Definition 7 (Instance of, ABox realization) *Let* C *be an arbitrary concept description,* i *an individual name and (*$\mathcal{T}$*,*$\mathcal{A}$*) a DL knowledge base. The individual* i *is an* instance of C *w.r.t. (*$\mathcal{T}$*,*$\mathcal{A}$*), iff* $a^{\mathcal{I}} \in C^{\mathcal{I}}$ *for every model* $\mathcal{I}$ *of (*$\mathcal{T}$*,*$\mathcal{A}$*).*

ABox realization *of* i *w.r.t. (*$\mathcal{T}$*,*$\mathcal{A}$*) returns all concepts* C *defined in* $\mathcal{T}$ *for which* $i^{\mathcal{I}} \in C^{\mathcal{I}}$ *for every model* $\mathcal{I}$ *of (*$\mathcal{T}$*,*$\mathcal{A}$*).*

So, instance checking denotes the task of testing whether a given individual is an instance of a given concept. Realization of an individual, in turn denotes the retrieval of all (most specific) named concepts from the knowledge base that a given individual is an instance of. ABox retrieval realizes this task for all individuals described in the ABox. This reasoning service is the central one to realize the task of recognition of situation types in our approach.

Another ABox reasoning service that we only mention briefly here and that is a good alternative to realize situation type recognition are *conjunctive ABox queries*. Here one can pose more sophisticated queries to the knowledge base. The queries itself are complex expressions – typically conjunctions – containing variables. These variables are instantiated by the DL reasoner with ABox individuals to answer the query. Conjunctive ABox queries are a very powerful way to query the DL knowledge base. However, in the remainder of the paper we resort to ABox realization, since this service is provided by most of the current DL reasoner systems and thus is a better starting point for evaluation.

2.1.3. The web ontology language OWL

In 2004 the W3C made the web ontology language OWL a recommendation for the semantic web. The OWL standard incorporates ideas from the earlier ontology language DAML+OIL and from RDF Schema.

The W3C recommendation for OWL specifies three dialects. While the most expressive dialect *OWL full* is beyond the expressivity of DLs and reasoning in it is undecidable, the other two dialects correspond to DLs for which sound and complete reasoning procedures exist. *OWL DL* can express ontologies written in the DL $\mathcal{SHOIN}$[1]. The less expressive *OWL lite* can express ontologies written in the DL $\mathcal{SHIF}$[2].

In OWL lingo concepts are called classes and roles are referred to as object properties. In Table 1 on page 7 in the third column some of the concept constructors available in OWL DL are displayed in correspondence to the DL ones. The semantics of the OWL DL constructors is the same as for DLs. Thus all reasoning services introduced in the last section are applicable for OWL DL as well.

The standardization of OWL has brought DLs and their reasoning systems to the attention of people from many different application areas and the ontology language is used in many novel application areas – context-aware systems are some of them. Next, we turn to the usage of DLs (or OWL DL resp.) in this application domain.

2.2. DLs for context-aware systems

The main idea how to use DL systems for context-aware systems is to build a TBox with the main notions from the domain and the description of types of situations relevant to the application. This kind of TBox needs to be "hand-crafted" by a human modeler at design time. This task can be supported by the automated consistency tests and classification that DL systems provide. At run-time of the context-aware system a description of the current situation is generated by the context application and written into the ABox in form of ABox assertions. Based on this description, the DL system can determine automatically into which situation-type this situation falls by computing ABox realization for the situation ABox.

[1] $\mathcal{SHOIN}$ is the DL $\mathcal{ALC}$ augmented with number restrictions, nominals, functional roles, transitive and inverse roles and role hierarchies.

[2] $\mathcal{SHIF}$ is the DL $\mathcal{ALC}$ augmented with functional roles, transitive and inverse roles, and role hierarchies.

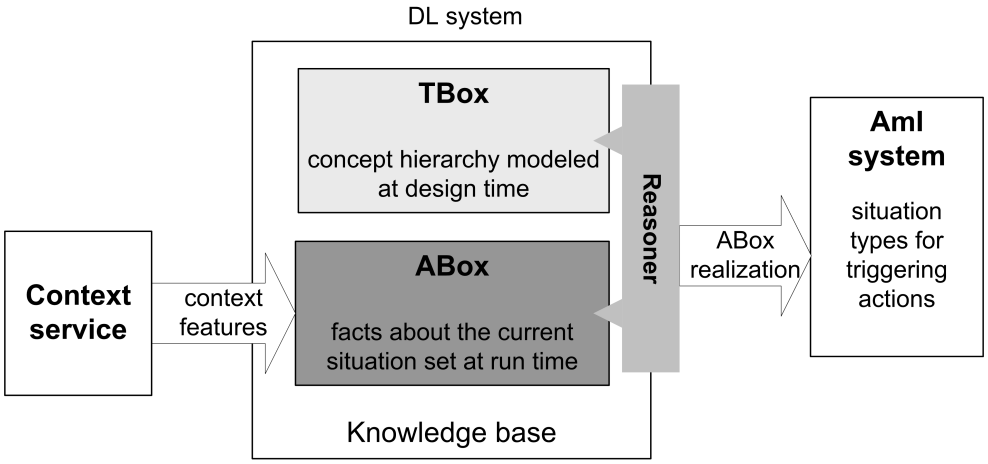

Fig. 2. Using DL systems for recognizing situation types.

Description Logics have several characteristics that make them a well-suited choice for context-aware systems. As knowledge representation formalisms DLs are designed to model hierarchies of notions from the respective domain. Thus DLs naturally support the requirement of context-aware systems to be able to model information on different levels of detail or abstraction as a means to deal with heterogeneous information sources. Moreover the formalism of DLs allows to combine information from different sources and combine them to aggregates in form of complex descriptions capturing different aspects of an informational entity.

However, the most important feature of DLs that makes them suitable for context-aware systems is their *open-world semantics*. Intuitively, this kind of semantics assume that the KB does not have complete information about the world, but that some information might be missing. So, from the absence of a fact in the KB, say `SoftwareProgrammer`(Alice) it cannot be inferred that the negation ¬`SoftwareProgrammer` (Alice) holds. In systems that adopt *closed world semantics*, such as databases, the absence of a fact means that the contrary holds, since these systems assume complete knowledge about the world. For context-aware systems open world semantics is clearly the better option, since context information is often incomplete. Thus DL systems offer a way to handle this kind of characteristics of context information in a graceful way.

As a collection of reasoning services DLs have even more to offer for context-aware systems. DL reasoning services support the building of the TBox by detecting inconsistencies or missing subsumption relations early. The reasoning task of classification gives the modeler an overview of how aggregated concepts relate to each other. Furthermore, ABox reasoning services can be used to infer the situation type of a certain situation and thus solve the problem of the actual context recognition. Since the methods for these reasoning tasks are implemented in highly optimized reasoner systems, these services are available to the ambient intelligence community directly.

To the best of our knowledge this approach of using DL systems for context-aware systems was first described by us in [52], but was also pursued by others in [54] and in [1]. The next section introduces the whole approach in detail.

3. Approach for ontology-based recognition of situation types

Our approach for situation-aware systems is based on DL systems. As described in the Section 2.1 a DL system consists of a knowledge base defined using the Description Logics based language OWL DL and a reasoning component providing a set of reasoning services. With OWL DL a *knowledge base* consists of a TBox and an ABox which together represent the knowledge of the AmI system about the current situation. The TBox contains *concepts* organized in a hierarchy which model the aspects of the situations relevant to the considered AmI system. The ABox contains a collection of *facts* describing the current situation of the AmI system.

At design time the TBox is created by the system developer. All conceptual knowledge about situations

which should be recognized by the system have to be expressed by OWL DL elements. Part of the presented approach is a modeling methodology to systematically decompose complex real world situations. The main idea of the methodology is to identify these aspects of situations which are relevant for decision making. Although the analysis of the application scenario and the identification of situation types remains an intuitive task, the modeler is assisted by the DL system when building the TBox. First, inconsistent descriptions can be detected automatically and second, the concept hierarchy can be computed automatically so that the modeler can discover unintended sub/super category-relations between concepts directly.

At run-time context information is added to the ABox as the information about the current situation. The reasoning service *ABox realization* is used as a means to recognize the type of situation from the situation description in the knowledge base. We propose a framework consisting of a context service, a DL system and an AmI system. The context management, especially the management and access of heterogeneous, alternative and distributed sources is covered by the context service. The AmI system specifies the relevant context information in a context profile. The concept definitions in the TBox are used as a common vocabulary specifying context to be exchanged between all system components. The DL system manages the knowledge base and provides the DL reasoning tasks. A summarized view on the proposed approach is depicted in Figure 2.

3.1. Intelligent door lock scenario

To illustrate our approach we use a scenario taken from the smart home domain. An automatic door lock should pick the next action to be taken depending on the person ringing at the door. We assume that the door system is equipped with a video camera and a microphone and provides information about the ringing person. Based on this information the door lock has to determine one of the following actions:

1. Open the door, if the person is authorized.
2. Ask a resident in case the person is unknown.
3. Do not respond at all or let the ringing person leave a message if no resident is available (similar to: nobody at home).

In a first step the person ringing is identified (e.g. as a resident of the house or a neighbour) or classified as a member of a group of persons (e.g. a fire fighter or a postman). Per definition the door opens for authorized persons only (e.g. a resident or family member). If the person at the door cannot be unambiguously classified, the decision whether to open is forwarded to a resident. The door tries to contact a resident taking her current situation into account (e.g. her activity and currently used devices). If a resident is watching TV, the image captured by the camera can be redirected to the TV set being used. If no resident is at home, the system tries to contact a resident via a cellular phone or another mobile device currently in use. If no resident can be reached within a short while, the system informs the person at the door, offers to leave a message and keeps the door closed.

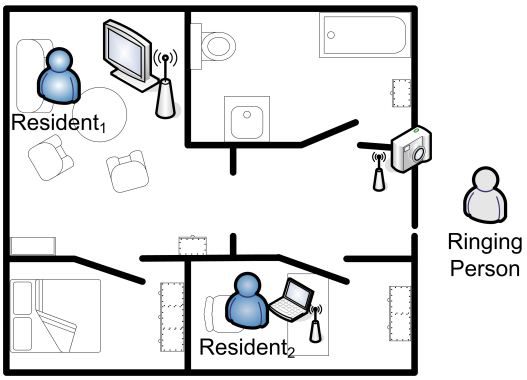

Fig. 3. Intelligent door-lock scenario.

Example: For the holiday season a neighbour is asked to water the flowers while the residents are on vacation. The door lock system identifies the person ringing as the neighbour. Furthermore, the door system checks whether the ringing neighbour is authorized by a resident to enter the house. If in addition all residents are on vacation, the neighbour can be recognized as an authorized person and the door opens. We decided to use this fairly simple scenario for our first case study to ensure that it is easy to model. Nevertheless, as the scenario description already shows, if modeled in detail, it becomes sufficiently complex to illustrate pitfalls of context modeling and the use of reasoning services.

3.2. Ontology-based situation modeling

Recalling the conceptual architecture in Figure 1, the DL system bridges the context-awareness and decision making phases. Thus, the goal of the modeling of situations is to recognize the current situation in a way that a decision can be made about the actions to be triggered based on the recognized situation. For instance, if our example AmI system is in the situation, that an

authorized person is ringing the doorbell, it should be able to derive the action to open the door immediately.

3.2.1. Decomposing the scenario

It is the task of the AmI system developer to analyze the application scenario, to identify relevant situations and to derive the conceptual knowledge about these situations. To handle this complex exercise the notion of situation decomposition is introduced as a systematic approach for creating ontology-based situation models.

The main idea is to use a task-based approach to identify major situations relevant to the triggering of actions in the AmI system so that the system can fulfill its foreseen tasks. Therefore, for all tasks the system should perform, situation types have to be identified, which allow an unambiguous selection of the actions to be triggered by the AmI systems to fulfill the appropriate task. The identified situation types can then be decomposed into a hierarchy of sub-situations. This method is derived from our observations, that these sub-situations can be modeled by independent aspects of a situation which can later be composed to model the identified situation types. For example, in the door-lock scenario the ringing person and the situation of the residents of the house are independent aspects of the overall situation.

For a systematic decomposition, the applications scenario can be analyzed according to so called *"aspects of interest"*. These aspects should be atomic in the ideal case, but at least fine-grained enough to be modeled by a small set of concepts. This usually results in a step-by-step decomposition with an increasing granularity in each step. For instance, in the door-lock scenario, the identity or role of the ringing person, and the presence of the residents are aspects of interest. Each of these aspects can be decomposed further. For instance, for the presence of the residents the aspects of reachability and the willingness to communicate are of interest, which again depend on the activity, location and the devices nearby. The following criteria can be used for scenario decomposition:

Spatial decomposition: In almost every scenario it will be possible to adopt a spatial decomposition. In our scenario, interesting locations are the door, where the ringing person is situated and the rooms of the house were the residents stay. Moreover, also outside locations might be of interest in the case that no resident is at home and should be contacted remotely due to urgency.

Temporal decomposition: Temporal aspects should also be taken into account when decomposing the scenario. Usually, different points in time contribute independently or in relation to the overall situation. In our example the resident might have left the house 10 minutes ago when a postman is ringing. Knowing that the resident went just out to buy a newspaper which usually takes him 15 minutes, the system could inform the postman that it expects the resident to be back in 5 minutes.

Acting persons: In many scenarios several persons play different roles. The roles they play can also be modeled as independent aspects of the overall scenario. In the door-lock example we distinguish between residents, relatives, neighbours and different types of professions like postman, fire fighter or police man.

In addition to these criteria further scenario-specific aspects can be identified. In our scenario the technical reachability, activity and presence of persons play an important role.

The goal of the decomposition is the identification of basic situations which can be unambiguously identified based on a small set of context features. Moreover, basic situations should also be processable independent from other basic situations. Therefore, in parallel to the decomposition of situations, the context features relevant for describing particular situations have to be identified. In the door-lock scenario for instance the location and the capabilities of the devices used by a resident are relevant context features to determine the reachability of the resident.

3.2.2. Modeling situations in the TBox

After this aspect-wise decomposition, the identified sub-situations have to be modeled in the TBox. Moreover, tests should be performed to validate the consistency and correctness of the created model. In addition, performance aspects play an important role, so performance issues at run-time might lead to changes of the TBox as well. According to our experiences gained in the process of modeling several scenarios, we encountered four main activities of knowledge base development:

1. Building of the TBox,
2. Building of the test ABox,
3. Testing the inferences for the ontology, and optionally
4. Performance tuning of the ontology w.r.t. the needed inferences

These activities, however, should be performed interlaced and should not be read as a strict sequence.

Building the TBox The set up of the TBox can be performed in a task-oriented and incremental way. That means, in the beginning one task of the AmI system can be selected and all identified sub-situations contributing to the decision for performing that task have to be modeled. The modeling should start with the most fine-grained sub-situations which are independent of each other, as stated above. More situations can be added step-by-step as soon as one task is completely covered. The sub-situations can also be modeled step-by-step and later on used to compose more complex situation.

In our modeling approach the resulting TBox is twofold. One part of the ontology contains situation descriptions and the second part consists of general concepts required for the definition of situations. Generic concepts of the door-lock example are for instance location, device and person. A situation is usually described based on several generic concepts. Situations are introduced into the TBox as named concepts which are ordered in the expected subsumption order manually. The situation concepts might be provided with a definition later on. As mentioned in the introduction, generic ontologies could be used as upper ontologies which are refined according to the scenario requirements. The basic relations from the scenario domain should be captured as roles in the TBox.

Once the basic situations are introduced as named concepts, the definition for these concepts should be provided (or successively refined). At the early stages these definitions should be "close to the intuition" of the notion to be modeled and all language constructs that express this intuition best should be used. Generally, it is a good strategy to model the TBox with as little redundancy as possible. This eases debugging of the ontology, i.e., tracking the cause of inconsistencies. Furthermore, keeping redundancy low is a good design principle when developing an ontology in a team. Furthermore, it is advisable to perform consistency checks often when extending the TBox. Sources of inconsistencies can be tracked more easily, if only a few definitions have been changed since the last consistency check.

For our application it is desirable to have a fine-grained concept hierarchy for the hierarchy of situation concepts and for collection of concepts from the sub-domain that will be central to distinguishing different context concepts (such as the collection of different resident concepts in the Doors scenario). So, the concept hierarchy should be computed and checked against the intended hierarchy. In case intended subsumption relations are missing the concept definitions must be revised and in case concepts collapse (i.e., are equivalent although intended as different concepts), the concepts must be refined. Often this involves the modeling of a new aspect, as devices or persons in our context applications. Each subdomain models a different aspect of the context concepts collection in a separate hierarchy.

At run-time of the application another benefit of the fine-grained hierarchy is to infer relatively specific context concepts via realization even in cases where information about the situation of the application is incomplete. For example, when the exact location of the resident is unknown, the situation that the Resident is out of home can still be inferred and appropriate actions can be taken. In contrast, if there is no fine-grained concept hierarchy, we can probably only infer a generic situation type, if this information does not suffice to derive that the resident is currently traveling, for example.

Building the test ABox In activity 2, we build an ABox for testing

1. whether the vocabulary in the TBox is already elaborated enough to be used for a detailed situation description and
2. whether concept definitions in the TBox are already precise enough to give the expected reasoning results for a situation description in the ABox.

The situation descriptions in this ABox should be similar to what is to be expected to occur in the application in terms of aspects that are supplied at run-time by the context service. If it turns out that the concepts defined in the TBox do not allow to describe a situation detailed enough, or if necessary "ingredients" for such a description are simply missing, the TBox must be extended appropriately. Thus, the activities 1 and 2 should be carried out in parallel.

Testing Activity 3 starts as soon as the TBox is filled with a few concept definitions. At the early stages it should be tested whether the TBox is consistent, i.e., whether it contains contradictions. These consistency tests should accompany the whole process of building the TBox. As soon as the concepts in the TBox are sufficiently elaborated to represent (sub-)situations from the application it is interesting to determine whether

the achieved level of detail is enough to infer the desired information. To this end classification tests and realization tests are performed. On the one hand it must be checked if the classification results meet the intuition of the modeler, for instance, if unintended subsumption relations are detected. On the other hand ABox realization has to be performed to check if the most specific concepts are detected for the situation. If the results do not meet the expectation of the modeler, the concept definitions should be refined.

Performance tuning Activity 4 is performance tuning. Now the knowledge base is analyzed w.r.t. the syntactic constructs in use and run-times for the inferences (needed at run-time of the application). The analysis of the syntactic constructs should yield what syntactic constructs are used and also how often they occur. If, for example, just one transitive role is used in the knowledge base, it should be carefully checked if the transitivity of this role is essential for intended result of the inferences. If syntactic constructs that are notorious for making reasoning harder and degrading the performance of DL reasoner, can be omitted or replaced by others without losing (important) inferences, these constructs should be deleted or replaced. Please refer also to Section 5.1.

Another way of enhancing performance of the reasoner is to add information to the TBox. One can add subsumption information so-called told subsumers to the (primitive) definition of concepts. For example, the concept C1 is a sub-concept of C2, but is not defined in terms of C2, then the reasoner has to "discover" this subsumption relation, which can be costly in terms of run-time. If this subsumption information is added to the definition of C1, the reasoner only needs to test for consistency. Similarly, one can also add "told non-subsumption" information, e.g. by adding more disjointness constraints, to avoid the effort of discovering non-subsumption by costly methods.

3.3. Filling the ABox

If the TBox is created the AmI system is ready to run. During the startup of the AmI system the TBox has to be loaded into the DL system and classified once before the first ABox realization service usage can be performed. As stated above context information is used as the information describing the current situation. It has to be added to the ABox as individuals to be processable in the DL system. To cover a subset of the characteristics of context named in Section 1.1 of the introduction, we assume the availability of a context service. This context service should be able to integrate and manage heterogeneous and geographically dispersed context sources. By accessing this context, context features representing facts about the current situation are written into the ABox.

To identify the context feature which have to be retrieved we apply the creation of a so called context profile. This context profile contains all concepts from the TBox for which the context service can provide information. This context profile have to be created manually or may be derived automatically. The latter can be achieved if the information about the managed information provided by the context service can be mapped to the concepts in the TBox.

The context profile is then used for either request all information at once in case that the AmI system is requested to perform an action. In our example this is the case if someone is ringing at the door. This is similar to taking a snapshot of all relevant information about the current situation at a certain point in time. If the AmI system should act proactively, it can subscribe for all situations in the profile to the context service. The system is then notified by the context service about the changes of any of the information in the context profile. If a change notification is received by the AmI system it can update the ABox and perform reasoning in order to check whether the situation has changed in a way that an action has to be performed.

3.4. Perform reasoning

As already described in the previous section reasoning can be performed on demand or event-based. In both cases all relevant and available context information have to be added as individuals to the ABox. To compute the types of the current situation the situation is itself modeled as an individual of the general situation concept. The context information is related to that situation based on roles which set different individuals in relation. To compute the types of a situation, ABox realization is performed on the situation individual. As a result the DL reasoner responds with a list of concept names. In particular, this list contains the most specific concepts for that situation individual.

Dependent on the available context information, the computed situation types, i.e., the concept names, are more or less specific. If not enough information about the current situation is available, the reasoner responds with a generic situation concept. For instance, if no information about the resident of the house are avail-

able, just a concept describing that the doorbell rings can be computed. If a large set of context information is available, a very specific situation type can be computed. As an example, if it is known that the location of the resident is the "living room", the TV is "on" and the ringing person is the postman, concepts describing that a postman is ringing and the resident should be informed by presenting a video on the TV can be computed. Thus, dependent on the amount of context information available, the computed situation types are more or less specific. A result is available in any case, even if the provided context information is incomplete.

Based on the computed situation types it is the task of the AmI system to determine the right action. This is a separate step in our approach to decouple the identification of the current situation and the decision process in order to enable a high flexibility within applications for context dependent decision making.

4. A case study of situation-awareness based on DL systems

We use the scenario introduced in Section 3.1 to implement a case study of validating the feasibility of our approach. Especially, a framework was developed which serves as the foundation for implementing various application scenarios. In the following we introduce a framework for DL-based situation-awareness and the situation model created for our scenario.

4.1. Framework architecture

The main components of our framework are the Context Service, responsible for providing the required context information, the AmI system which is interested in the current situation and the DL System, responsible for processing ontologies and performing ABox realization for situation recognition (see Figure 4.1). A further component is the Context Profile used by AmI systems for requesting context information from the context service as described in Section 3.3.

4.1.1. Context Service

In our implementation we use the distributed context service described in [45]. The context service is available to each application in the form of a local component which provides an access interface for contexts. The context service is able to manage a large set of highly distributed context sources and handles the access, transformation and distribution of applications to context information in a transparent manner. Where and how the information is gathered is not visible to the context application, it just uses a well-defined API for requesting context information based on context profiles.

The context management inside the context service is based on meta-model derived from topic maps. It contains entities which may have a set of attributes. Relations between entities are modeled as associations. The semantics of these elements is defined based on an ontology specified in OWL DL. Thus, we defined a mapping between the context domain model supported by the context service as described in [45] and our scenario ontology. Based on an extension of the API for accessing context information we support now a profile-based access of context information. Each time the Door-lock system should react on the ringing of the door bell, it sends the context profile with a request to fill it to the context service. The context service responds with the requested information integrated into the profile. The profile might be completely or partially filled according to the current availability of context sources.

4.1.2. DL system

The DL system is the second component of the framework. It is able to maintain multiple ontologies and provides the service to perform reasoning tasks on these ontologies. An AmI application can register a knowledge base (i.e., an OWL DL based ontology) at the DL system, which processes the TBox of the ontology and builds up an internal representation of the processed TBox. Each registered ontology is assigned with a unique identifier for later referral. During the run-time of context applications we assume a constant TBox allowing the reuse of the internal representation of the TBox built up at registration time. At run-time the AmI application requests reasoning tasks based on varying ABoxes.

Thus, the DL system is a system service which can be used by several context applications in parallel. It can be placed on a high-performance system server residing within the infrastructure. Especially, this enables the remote processing of reasoning tasks for applications running on mobile or resource limited devices. The DL system is represented by off-the-shelf DL reasoner systems, e.g., RACERPRO [22], the FACT++ system [51] or PELLET [42]. They all implement the DIG interface [9] and thus, can be exchanged on that basis.

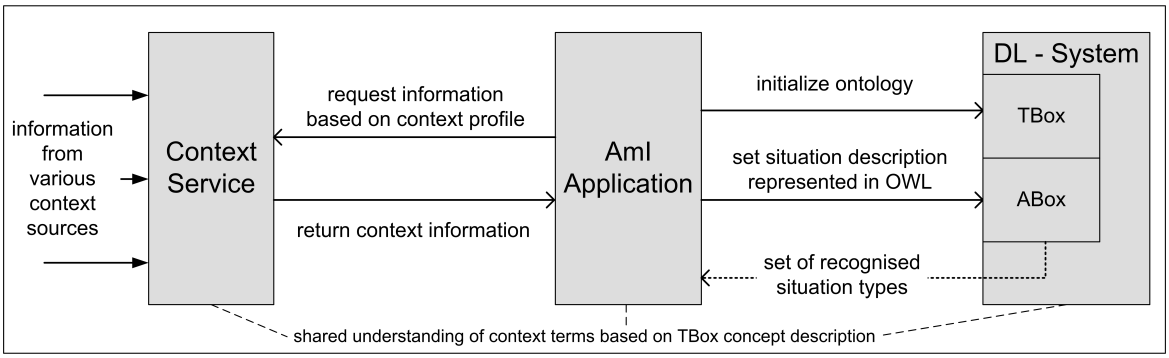

Fig. 4. Architecture of the framework for situation-awareness.

4.1.3. AmI Application

The AmI application is the component which coordinates the context access and reasoning. It registers an application specific ontology at the DL system and requests reasoning tasks from this DL system. The context is gathered from the context service component which can be local to the application component or remote on an infrastructure server. The context application uses a context profile to request the context values required for classifying the current situation or a certain aspect of a situation. The context service responds with a filled context profile containing at least a subset of the context values requested by the application.

The context values are used to create individuals of an ABox according to the defined context ontology. These individuals belong to a certain situation individual. The ABox is then sent to the DL system which performs realization on the situation individual to classify the situation. Based on the classification of the situation, the context application can determine what action has to be performed. Thus, the determination of the action to perform is separated from the situation detection.

The implementation of the AmI application is based on a set of up-to-date technologies. We used Java as implementation language for the framework according to the Java 5.0 specification. For the integration of ontologies and deterministic reasoning schemes we used the Semantic Web Framework Jena in Version 2.5.1. In addition to using the DIG interface we also integrated the DL reasoner Pellet in Version 1.5. via its provided Java interface. To publish our upper ontology we used Apache 2.2 as the web server. In ongoing work we also embedded this framework in a more general framework for situation-awareness. The extended framework is described in [46] and covers the integration of heterogeneous sensors (e.g., wireless sensor networks, microphones and other stand-alone sensing devices), the integration of classifiers for these sensors and the use of various reasoners for situation detection.

4.2. Implementing the intelligent door-lock scenario

We describe now how to model the ontology for the door-lock scenario according to the methodology introduced in Section 3. Starting with the decomposition of the scenario and the identification of relevant context information, the situation types for the door-lock scenario were identified and modeled in the TBox.

4.2.1. Decomposing the scenario

The scenario can be decomposed based on the criteria described in Section 3.2. Starting with the tasks of the system – opening the door, asking a resident or keeping the door closed – aspects of the overall situations can be identified. For all tasks a spatial decomposition is relevant, i.e., aspects are the situations in front of the door and in the house. Moreover, acting persons can be identified and their current situations can be modeled. For opening the door the identity and role of the ringing person is relevant. For instance for a resident or an authorized neighbour the door can be opened immediately, while if the ringing person is not authorized, the task of "asking a resident" has to be performed.

Relevant context. To reason about the situation of the door scenario, the following contextual information is relevant: The *identity and social relations* of and between persons or group of persons can be used to distinguish between residents, their relatives, friends and neighbours (e.g., while some of them may be authorized to enter the house, while others may only enter if another person is already at home) and categories of persons which can be determined by their clothes or

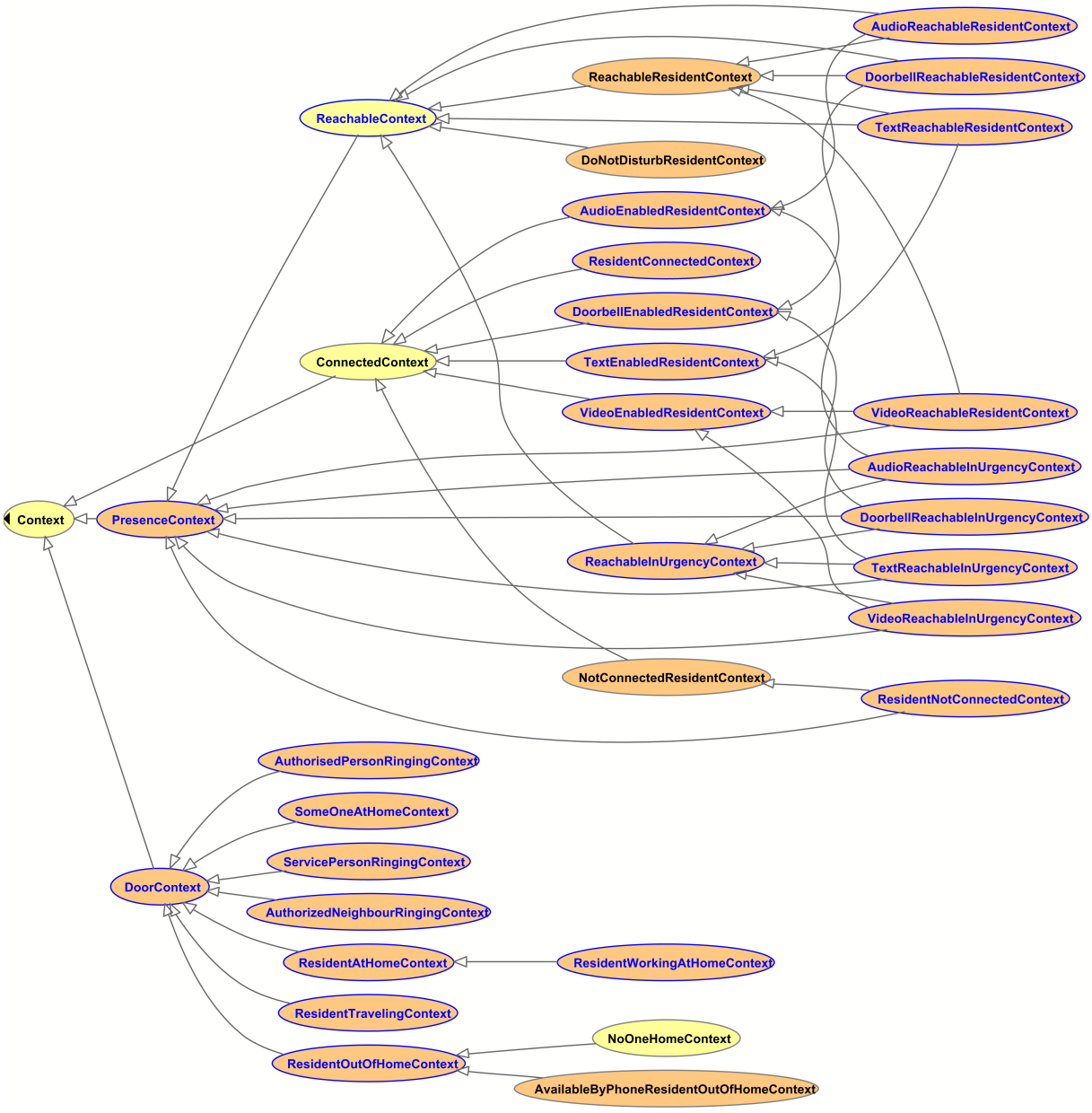

Fig. 5. Context concepts for the intelligent door scenario.

other characteristic features (e.g., the uniform of a police man or fire fighter, or the pizza boy carrying pizza boxes).

The *location* is relevant for persons who are residents or otherwise able to help the system to decide about the action related to the ringing person, i.e., to authorize ringing persons to enter. Especially of interest is the location of the residents. *Time* as contextual information is relevant in combination with other information, e.g. to determine the activity or current location of a person (e.g. person is working if it is 8 am and located at his office room). Information about the *presence* describes if and how persons/residents would like to be contacted (e.g. a resident is working at his office and do not want to be disturbed by anybody).

The concept `activity` of a person is relevant for determining the presence of a person. Information about the activity is usually not directly available but has to be extracted out of several information, e. g. the schedule, the activity, time, and persons nearby.

Information about the *devices* a person owns and which of them are currently active and in use (e.g. located close to the person) is relevant to determine how to contact a person. In combination with the current connectivity of the devices information can be extracted to describe how to contact a person (e. g. if the device and connection supports audio and video communication or email/text communication only).

4.2.2. Modeling situations in the TBox.

We modeled the doors ontology according to our approach described in Section 3.2 with the task based notion of context in mind. The resulting ontology contains concepts for describing the relevant context and roles for describing the interrelations between these concepts. Furthermore, the ontology contains concepts describing the situation itself in form of a hierarchy of contexts which describe certain aspects of the situation (e.g., who is ringing, the presence, activity and location of the residents and the interrelations between the residents and the ringing person).

Modeling locations. Location is a fundamental concept in the door scenario. We use the four basic concepts `InDoor`, `OutDoor`, `Mobile`, and `ImMobile` to describe the basic features of a location. Each location in the ontology is a sub-concept of either `InDoor` or `OutDoor` as well as either `Mobile` or `ImMobile` using multiple inheritance. The role `nearby` defines that two locations are close together. The properties `hasLightLevel` and `hasNoiseLevel` describe the features of locations relevant in our scenario to derive the activity of a person at a certain location.

`Building` is a sub-concept of `InDoor` and `Im−mobile` and consists of several parts (e.g., Rooms). A sub-concept of `Building` is `House` which enables the description of any house relevant to the situations of the door lock scenario. Another sub-concept `Home` describes the building which contains the "intelligent door lock" system. For `Home` and `House` a close distance can be defined using the `nearby` role to describe the neighbourhood of the home.

Each `Building` can consist of parts. We currently defined only `Rooms` as parts. The other way around, each room is part of a certain building described by the role `isPartOf`. Several sub-concepts of Room are defined to model the rooms relevant to the scenario (e.g. `LivingRoom`, `Kitchen`, `BathRoom`, `BedRoom`, and `OfficeRoom`). Locations of the category `OutDoor` and `Mobile` are also defined within the ontology (e.g. `Car`, `Plane`, `Bus`, and `Train`). These locations are relevant for describing activities of residents.

Modeling of identity and social context. The identity and social context of the persons relevant to the door scenario are modeled based on the `person` concept. Each person has a current location (role `locatedAt`) and lives at a certain location (role `livesAt`). A person which lives at `Home` is defined as a resident $\texttt{Resident} \equiv \texttt{Person} \sqcap \exists \texttt{livesAt.Home}$. A resident at home is described by: $\texttt{ResidentAtHome} \equiv \texttt{Resident} \sqcap \exists \texttt{locatedAt.}(\texttt{Home} \sqcup \exists \texttt{isPartOf.Home})$. Similarly a resident currently out of home is described by: $\texttt{ResidentOutOfHome} \equiv \texttt{Resident} \sqcap \exists \texttt{locatedAt.}(\texttt{Home} \sqcup (\exists \texttt{isPartOf.Home}))$. The social relations between residents and other persons are modeled based on the concepts neighbour and relative. A neighbour is a person who lives not at home but at a house nearby home: $\texttt{Neighbour} \equiv \texttt{Person} \sqcap \exists \texttt{livesAt.}(\neg \texttt{Home} \sqcap (\exists \texttt{nearby.Home}))$.

A relative is a relative of a resident by definition: $\texttt{Relative} \equiv \texttt{Person} \sqcap \exists \texttt{isRelativeOf.Resident}$. Service persons like police man or pizza boy are modeled as sub-concepts of person. In the ontology several service persons are modeled for example.

Modeling activity. The activity of residents is essential for determining the current presence of residents. Because activity usually can not be captured directly, it has to be derived from available information like time, location and schedule.

For modeling features of a location we use so called value partitions. A Value Partition is a design pattern to define an exhaustive list of values for a certain set. For instance, for describing the light level of a room the set of possible values can be defined by a value partition `LightLevel`. The possible values are: full, dimmed and off. Because Description Logics are based on the open world assumption, value partitions have to be used to restrict the valid values of a certain set (otherwise, further (currently undefined) values could be elements of this set). A covering axiom has to be defined to make the list of value types exhaustive. The value partition `Lightlevel` can be defined in form of a value partition as follows: $\texttt{LightLevel} \equiv \texttt{Off} \sqcup \texttt{Dimmed} \sqcup \texttt{Full}$ (where `Off`, `Dimmed` and `Full` are mutually disjoint).

1. AudioEnabledDevice ≡ Handy ⊔ SmartPhone ⊔ PDA ⊔ Laptop ⊔ Phone ⊔ DoorBell
2. SMSEnabledDevice ≡ Handy ⊔ SmartPhone ⊔ CellularPhone
3. AudioEnabledConnectedDevice ≡ AudioEnabledDevice ⊓ ∃isConnectedVia.(∃ hasBandwidthLevel.(LowBandwidth ⊔ MediumBandwidth ⊔ HighBandwidth))
4. SMSEnabledConnectedDevice ≡ SMSEnabledDevice ⊓ ∃isConnectedVia.(WAN ⊔ PSTN)
5. AudioEnabledResidentContext ≡ ∃ hasContextResident.(Resident ⊓ (∃uses.AudioEnabledConnectedDevice))
6. DoorbellEnabledResidentContext ≡ ResidentAtHomeContext
7. ReachableInUrgencyContext ≡ ∃hasContextAgent.(FireFighter ⊔ Policeman)
8. DoNotDisturbResidentContext ≡ ∃hasContextResident.(SleepingResident ⊔ VacationResident ⊔ (∃hasPresence.DoNotDisturb))
9. AudioReachableResidentContext ≡ AudioEnabledResidentContext ⊓ ReachableResidentContext

Fig. 6. Concept definitions from the Doors ontology.

The activity of a person can be set by definition or can be derived from contextual information. We have defined to classes of activities:

- Vacation containing the sub-concepts BusinessTrip and Holiday
- Working containing the sub-concepts Reading and Writing

To derive the activity of a resident the location and its features are used. For instance the activity sleeping of a resident is derived from the location BedRoom and the LightLevel of the BedRoom which have to be Dimmed or Off: SleepingResident ≡ Resident atHome ⊓ ∃locatedAt.(BedRoom ⊓ (∃hasLightLevel.(Dimmed ⊔ Off))).

Modeling devices, connectivity and presence. The reachability of a resident can be modeled based on the used devices and their connectivity. It describes how a resident can be contacted at a certain point in time from a technical point of view. The ontology contains concepts and properties to model the devices owned (role isOwnedBy) and used (role isUsedBy) by a certain person and the network links which are supported by these devices (role supportsLink).

Devices are modeled as sub-concepts of the root concept Device. A basic feature of each device is its mobility. Thus, each device is a sub-concept of either Stationary or Mobile. Stationary devices relevant for the scenario are PC and the Doorbell, relevant mobile devices are CellularPhone, PDA and Laptop. A device can be connected via a certain network link (role isConnectedVia).

Network links are discriminated into wired and wireless links. Wired links are for instance DSL and Ethernet, wireless links are subdivided into PAN, LAN and WAN links containing wireless links such as Bluetooth, WLAN, GSM and UMTS. Each network has a certain bandwidth level (role hasBandwidthLevel).

To describe the options of the door system to contact a resident the communication categories of text, audio and video are defined. Text communication can be further discriminated into SMS messages and instant messages. For each device the supported communication categories are defined, describing the communication capabilities of the devices (see definitions 1 and 2 in Figure 6.

To involve the bandwidth of the network connection the concept of the EnabledConnectedDevice is defined. Examples are the definitions 3 and 4 in Figure 6. The inferred concept hierarchy for the modeled devices is depicted in Figure 7.

Modeling situation types. A situation is described by the concept Context and its sub-concepts. Each situation is described by a person or group of persons ringing at the door (role hasContextPerson) and the residents of the house (role hasContextResident). In Figure 4.2 the complete hierarchy of situation concepts of the door-lock scenario are depicted.

To enable a decision of the door system to open the door automatically, ask a resident or let the ringing person leaving a message, two basic contexts are modeled which are called DoorContext and PresenceContext. The DoorContext contains concepts describing the aspects of the ringing person and if residents are at home or not. This supports the decision to open the door immediately if an authorized person is ringing (i.e., a resident or a person authorized by a resident). Furthermore, based on a black list the system can immediately decide to keep the door closed and to let the ringing person leave a message without contacting a resident. A necessary prerequisite therefore is to identify the person or their category.

If no clear decision is possible, one of the residents should be contacted. To decide how to contact a resident the system reasons about the PresenceContext.

The `PresenceContext` describes at one hand what communication modes can be used to contact a resident from the technical point of view with the concept `ConnectedContext`. On the other hand the willingness of a person to communicate is modeled with the concept `ReachableContext`. The `ConnectedContext` is defined in definition 5 and 6 in Figure 6.

To describe the presence of a resident we use for instance the definitions 7 and 8 in Figure 6. To combine the aspects of technical connectivity and reachability, the `PresenceContext` has been defined as in definition 9 in Figure 6.

4.2.3. Testing

To demonstrate the usage of the ontology we have defined several situations based on individuals of the concept hierarchy in the ABox. Two types of situations have been defined as instances of the concepts `DoorContext` and `PresenceContext`.

5. Evaluation

The proposed approach of modeling situations with OWL DL and the adoption of ABox realization for recognizing situation types focuses on the employment of DL systems. The choice of DL systems for the implementation of AmI systems has several consequences regarding the performance of DL reasoners, the expressivity of the formalism and the handling of imperfect context information. Therefore, we take a closer look at DL systems to assess their usefulness with respect to these requirements. Especially, we want to point out the advantages and limits of the presented approach.

5.1. Performance evaluation

The complexity of ABox reasoning for the DL employed in our case is NExpTime complete in the worst case, see [49]. However, these worst case complexities do not need to appear in concrete scenarios. Moreover, there exist a couple of highly optimized DL reasoners for OWL DL that behave well in practice. Since applications for Ambient Intelligence usually expect computation times no longer than a couple of seconds or even might require response within milliseconds in some application domains, it is interesting to get an impression of the performance of the reasoners for classification and realization. Some language constructs have higher worst case complexity and thus, might require longer computation times. A natural question is whether it is advisable to disallow these constructs from the ontology.

To answer these questions, benchmarks have been performed using Protegé-OWL 3.4 beta from Standford University running under Windows XP. As hardware platform a Lenovo T60 Laptop with 2 GByte of memory and an Intel Centrino Duo processor running at 2 GHz was used. To compare the performance of the reasoners the times for classifying the concepts defined in the TBox and for computing the inferred types from the individuals in the ABox of our ontology were measured. For the tests three different variants of our Doors ontology were used:

Doors: The knowledge base (KB) described in the last Section. contains 6 Situation individuals.

Doors-no-GCIs: Door KB without GCIs, i.e., disjointness and domain and range restrictions for roles were deleted.

Doors-easy-roles: transitivity, symmetric role and inverse roles statements as well as all domain and range restrictions were deleted. Functional roles were left in the TBox, since they do not increase the run-time dramatically.

The DL expressivity of the Doors and Doors-no-GCIs ontology variants is ALCHIF, while the DL expressivity of the Doors-easy-roles ontology is ALCF. All ontology variants contain 135 concepts, 98 individuals and 27 object properties. We tested four DL reasoners that implement ABox realization for OWL DL. We used the well-known system RACERPRO [22], the FACT++ system [51] and PELLET [42].

FACT++**:** is a successor system of the FACT system [26], that was a ground-breaking TBox reasoning system developed in the late nineties. FACT was the first DL system that could handle GCIs in an efficient way. FACT++ is implemented in C++. We used version 1.2.0 (25.9.2008) of FACT++ for our tests. In contrast to the other tableau-based systems, FACT++ implements an eager approach for realization. It sets up its data structures for realization already during the classification phase of the TBox. This results in longer run-times for classification, but speeds-up realization considerably, as we will see.

PELLET**:** The PELLET system is developed at the University of Maryland in 2003. It supports DL reasoning services for the DL SHOIQ(D) - where the qualified number restrictions are limited to 1 or 0

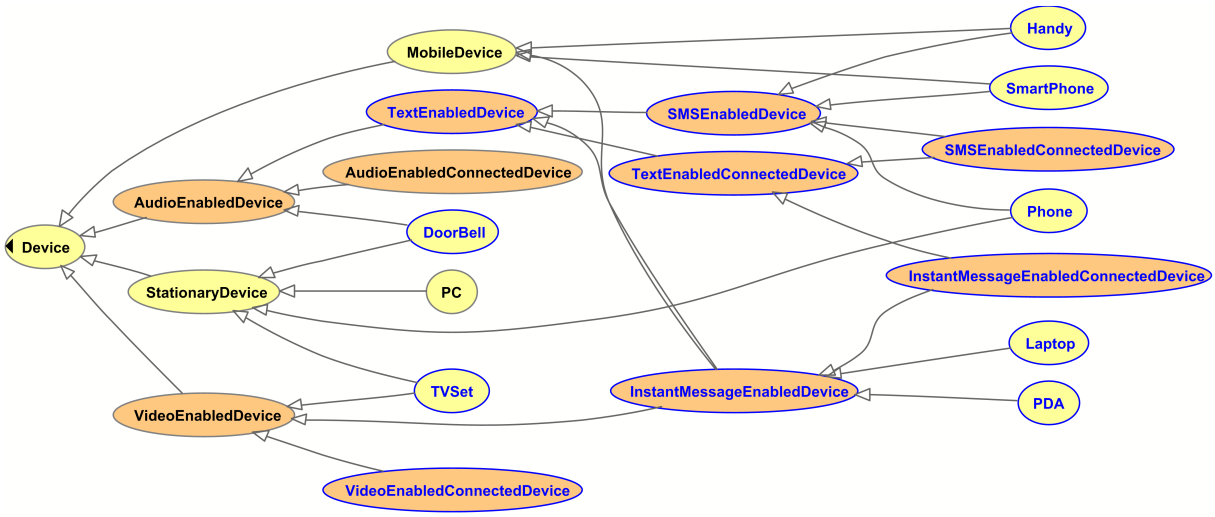

Fig. 7. Inferred concept hierarchy of device concept in the doors ontology.

and the concrete domains to what the OWL standard supports. Thus the PELLET system supports the full range of concept constructors of OWL DL. Besides the classical DL reasoning services, PELLET offers a range of additional functionality. For a full overview, please refer to the PELLET web pages [42]. In our evaluation of the systems we used PELLET 1.5.2 (1.5.2008).

RACERPRO**:** is the successor system of RACER provided by RacerSystems. This system is a commercial system with free licenses for academic institutions. In comparison to RACER, RACERPRO shows a much better performance. Furthermore, RACERPRO offers more functionality than most other DL reasoners [34,35], which helps to maintain and query the KB. As its successor RACERPRO also supports more expressive concrete domains than the OWL standard demands. For our evaluation we used the currently available stable version 1.9.0 of RACERPRO and the beta version of RACERPRO 1.9.2 beta.

With KAON2 [29] and HermiT [30] other DL reasoners are available. We have not included these reasoners in our test because of the lack of ABox realization reasoning service adopted in our approach.

The measured run-times are shown in Table 2. It contains measured times for the four reasoners. For each reasoner we measured the times for classifying all concepts of the TBox in our test ontologies and to perform ABox realization, as stated in Definition 7. The measured times for the classification task are below 1s, except for PELLET. While classification is only performed once during the start of the AmI system, these times are mainly relevant for the design time. Fast computation times especially allow for an interactive development of ontologies. The results for this reasoning tasks are much more heterogeneous. For the Doors ontology FACT++ needs orders of magnitude less time than PELLET. As stated in the description of FACT++ above, an eager approach was used for the implementation of the ABox realization. Thus, data structures for realization are already set up during the classification of the TBox. Compared to RACERPRO 1.9.2 beta, FACT++ needs a little more time for classification than RACERPRO. The ABox realization task is than performed much faster using FACT++ than using RACERPRO.

For the Doors-no-GCIs ontology the deletion of GCIs (see Definition 1) speeds-up the realization of the individuals. Especially, PELLET profits from that deletion. It performs the classification and realization tasks one order of magnitude faster than for the complete Doors ontology. But also both RACERPRO versions need less time for processing the Doors-no-GCIs ontology. The reduction of the Doors ontology to the Doors-easy-roles ontology causes an improvement of the computation times of classification for all reasoners. Anyway, only the both versions of RACERPRO can profit with respect to ABox realization. FACT++ and PELLET need more time for the realization of the Doors-easy-roles ontologies ABox than for the Doors ontology ABox.

Table 2
TBox classification and ABox realization run-times (in s)

	Doors		**Doors-no-GCIs**		**Doors-easy-roles**	
	TBox classification	ABox realization	TBox classification	ABox realization	TBox classification	ABox realization
FACT++	0,8	0,02	0,71	0,02	0,73	0,06
PELLET	3,79	18,45	0,66	2,19	2,57	30,85
RACERPRO 1.9.0	0,6	2,02	0,48	1,13	0,38	0,76
RACERPRO 1.9.2 beta	0,62	0,74	0,39	0,2	0,26	0,51

At one hand the reduction of language constructs used in the ontology may improve the performance of DL reasoners. On the other hand reduction comes at the cost of missing implicit subsumption or instance relations. For the Doors-no-GCIs ontology several of these relations can no longer be detected and not the most specific context can be recognized from the information available. So, it is clearly not advisable for this application to degrade expressivity to obtain better run-times. In general, this trade-off should be considered during the development of AmI systems following the proposed approach.

To sum up, today's DL reasoners can compute realization for ABoxes in a run time acceptable for the the intelligent door lock scenarios. FACT++ provides the best performance for ABox realization. While both newer version of RACERPRO also responds in times below 1s for ABox realization, PELLET is significantly slower with that task. Compared to previous measurements carried out with older versions of the reasoners but using the same ontologies [52], reasoners show up significant improvements in performance for both TBox classification and ABox reasoning. Because all reasoners are currently under development performance improvements can be expected in the future.

5.2. Expressivity Considerations

In addition to the performance of DL reasoners it is also important to what degree OWL DL supports the developer with the creation of situation descriptions. Major questions are how complex a situation description can be, what aspects of a situation can be modeled and which not, if it is transparent how situation types are recognized and to what degree situation models can be refined and adjusted at run-time and during the AmI systems life-cycle. Moreover, the modeling effort should be assessed.

From the experiences of the authors the modeling of aspects of situation types in the TBox is an intuitive task. Situation aspects are described as concepts, refinements of concepts can be expressed by defining a more specific context as a subconcept of an already defined concept. This way of thinking about the world is intuitive for developers, especially if they are familiar with object orientation. Similar to this approach, child concepts in OWL DL inherit all properties of their parent concepts. Based on the creation of a concept hierarchy even complex scenarios remain to be manageable when modeled with OWL DL.

In addition, the recognition of situation types is completely transparent for the system developer. If the design methodology proposed in Section 3 is adopted, during the activity of testing the developer creates a test ABox and performs ABox realization on the ontology to compute the most specific concepts characterizing a situation. If the TBox is specified correctly, the system behaves according to that specification. In contrast to DL systems, in case of adopting neuronal networks or Bayesian networks for situation recognition the phase for decision making is also performed by the network. For the developer it is not transparent how and why a certain situation was detected and a particular action was triggered.

Anyway, OWL DL also has some limitations a developer should be aware of. One of the most obvious limits of OWL DL was the limited expressivity and reasoning capabilities regarding numbers. One would like to specify ranges of numbers to map them to qualitative measures, like slow := has-velocity. (< 0) and (> 50). Another application of numbers would be to compare, for example the velocity of the user with the maximal velocity that is supported by a certain network technology. Although OWL DL offers XML data types for the use of numbers in ontologies, they do not allow to model the facts just mentioned, since there are only unary predicates available in OWL. One would like to model these facts by the use of concrete domains [4], which are supported by RACERPRO. We tried to emulate the ranges by number restrictions in an earlier version of the ontology, but since it was a very unintuitive

way of modeling and the performance of the reasoners was slowed down drastically, we gave up this approach immediately.

Often one would like to express facts such as "the provider of the available network is the same provider that the user has a contract with" – sometimes called agreements. The underlying DL concept constructor for this is called *role value maps* (or feature chain agreements). Unfortunately it is a well-known result that these constructors make reasoning for even small fragments of OWL DL (e.g., $\mathcal{ALC}$) undecidable. Thus, these concept constructors were not included in OWL DL. We tried to capture the above mentioned facts by the use of inverse roles. Sometimes the above mentioned agreements do not only refer to a single concept, but to roles. For example the role one would like to add "$\texttt{hasUncle} \equiv \texttt{hasParent} \circ \texttt{hasBrother}$" to the ontology. These kind of statements are not supported in OWL 1.0. However, it is planned to include them in the forthcoming OWL 1.1 standard.

In the step of the authoring of the knowledge base some typical effects can occur that lead to unexpected or unintuitive reasoning results.

Primitive definition vs. full definition: Sometimes when only necessary, but not the sufficient conditions are supplied for a concept C, by giving only a primitive definition and a subsumption relation seems to be missing.

Open world vs. closed world: Especially for users who have worked with systems that use closed world semantics (such as Prolog or Data bases) find it unintuitive, if from leaving out the fact that an individual is, for example located in the house, it cannot be derived that the individual is outside of the house. An insistent case of this are at-most restrictions, which can hardly ever be derived, but have to be supplied by the modeler in most cases.

Value restrictions vs. domain & range restrictions: often it is not clear whether the restriction of role-fillers to a certain type of concepts is only valid for a concept C or whether it is a property of a role.

Last but not least, the extensibility of the knowledge base at run-time plays an important role, especially in very dynamic scenarios. As a fact, situation descriptions are modeled at design time. That means, all relevant situations have to be known at design time. Moreover, situation descriptions depend on a static set of context features, i.e., new context types can't be involved in the situation model at run-time. Especially, new aspects of a situation can't be added at run-time even if our situation model is extensible at design time.

5.3. Handling of context characteristics

For our approach we see the handling of the described characteristics at different stages of context processing which we understand as a stepwise execution of operations for context interpretation, aggregation and derivation to produce higher-level context which can be used at application level. We assume that most of the processing is done within a context service (see Figure 4.1). While the application requests as set of context features according to a context profile at a certain point in time (what we call "to make a snapshot") the context service is responsible for handling dynamics to provide up to date information. This includes to provide the history if requested by the application. As for a certain context value multiple alternative sources can be available, they have to be maintained by the context service. These alternatives can be exploited to handle incorrectness and quality variations. The context service can apply operations for choosing the context value with the highest available quality out of several alternative values. Furthermore, it can compare alternative values to detect incorrect values.

Moreover, DL reasoning can handle *incomplete information* gracefully. Incompleteness can be detected by the context service based on the context profile. Nevertheless, we can handle it in the DL system because meaningful reasoning is also possible on incomplete data.Even if the context service can provide a subset of the requested context information, realization still is able to recognize concepts, even if they might be more general. However, even in such cases the system is still able to recognize context and the application can perform actions associated with that context.

Inconsistency can also be detected by the DL system and can cause the context application to request additional, clarifying information from the context service. We currently see the handling of data quality out of scope of the application level model, assuming that the handling is done by the context service as described above. We have so far focused on modeling the application domain, especially the context concepts useful for certain application tasks and reasoning about the situation the application is within.

Moreover, the approach to use DL reasoning for the recognition of contexts offers a graceful way of handling incomplete data. In such a case the realization

would simply return contexts that might be too general, but still a context is recognized and an action associated with the returned context will be performed. Furthermore, the separation of context recognition and choice of action allows adapting this association at run-time according to user preferences.

6. Conclusions and Future work

We proposed a method for situation modeling using the Description Logics based ontology language OWL DL and a framework for employing Description Logics reasoning services to recognize the current situation based on context. Especially, we employed the DL reasoning service ABox realization to identify the most specific types of a situation, representing the current situation. According to the introduced framework, the application then determines the actions to be performed in correspondence to the current situation.

The benefits from the approach are manifold: the semantics of Description Logics allow for graceful handling of incomplete knowledge. The well-investigated reasoning services do not only allow recognizing the current situation, but also can add to the reliability of the overall system. Moreover optimized reasoning systems are freely available and ready to use.

On the one hand the explicit specification of the situation model in the TBox may limit the ways in which an AmI system can adapt at run-time to changing situation types, but on the other hand the explicit specification clearly provides transparency to application developers and users and thus contributes to the reliability of the overall system. In addition, our approach requires neither a learning phase nor a large amount of training examples to set-up the initial knowledge base. However, it does not support full dynamic adaptation despite the separation into situation type inference and decision making, which allows a certain degree of flexibility at run-time.

The feasibility of the approach has been demonstrated with a case study based on a smart home application. The aspects of situations from that scenario could be modeled completely. The evaluation has shown that DL systems support the characteristics of context, especially the issue of incomplete knowledge can be handled very well. Further characteristics like heterogeneous and distributed context sources as well as multiple alternative context sources can also be handled in our framework, this is achieved by the context service adopted in our framework.

The performance of DL reasoners is appropriate for scenarios which can tolerate response times of a few seconds. Dependent on the chosen reasoner, the constructs used in the ontology and the number of concepts, roles and individuals run-times in the range of below 1s up to several second can be achieved by the currently available reasoning tools. While these tools are under development, mainly driven by the strong community of Semantic Web research, significant performance improvements can be expected for the future.

As pointed out in the evaluation in Section 5.2 there are also some limitations of the expressivity of DLs, a developer of AmI systems should be aware of. Solutions to overcome these limitations exists. Especially, mathematical or classification operations dealing with numbers can already be performed during the context-awareness phase. The approach presented in [46] supports a step-wise abstraction process of sensor measurements and can be seen as one possible solution for the issue of missing expressivity and reasoning capabilities regarding numbers in DLs. Another solution is to model these facts by the use of concrete domains [4], which are supported by RACERPRO.

To sum it up, OWL DL is extremely helpful as a standard, since it encourages the implementation of a lot of ontology tools and reasoners helpful for context applications. As a modeling language it offers wide range of language constructs that allow to model a lot of complex notions from context applications. However, some concept constructors central to modeling with numbers are missing in OWL DL.

The research work presented in this article was partially sponsored by Siemens AG. Moreover, the authors would like to thank Bootawee Suntisrivaraporn from TU Dresden and Michael Pirker from Siemens AG, Intelligent Autonomous Systems for their helpful comments on earlier versions of this paper.

References

[1] A. Agostini, C. Bettini, and D. Riboni. A performance evaluation of ontology-based context reasoning. In *Proc. of Fifth Annual IEEE International Conference on Pervasive Computing and Communications - Workshops*, pages 3–8. IEEE Computer Society, 2007.

[2] C. B. Anagnostopoulos, Y. Ntarladimas, and S. Hadjiefthymiades. Situational computing: An innovative architecture with imprecise reasoning. *Journal of Systems and Software*, 80(12):1993–2014, 2007.

[3] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook:*

Theory, Implementation, and Applications. Cambridge University Press, 2003.

[4] F. Baader and P. Hanschke. A Schema for Integrating Concrete Domains into Concept Languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, 1991.

[5] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, volume 4130 of *Lecture Notes In Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006. CEL download page: `http://lat.inf.tu-dresden.de/systems/cel/`.

[6] F. Baader and W. Nutt. *[3]*, chapter Basic Description Logics, pages 43–96. Cambridge University Press, 2003.

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

[8] P. Baker, C. Goble, S. Bechhofer, N. Paton, R. Stevens, and A. Brass. An ontology for bioinformatics applications. *Bioinformatics*, 15(6):510–520, 1999.

[9] S. Bechhofer. The dig description logic interface: Dig/1.1. Technical Report, 2003.

[10] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. W3C Recommendation, February 2004. `http://www.w3.org/TR/owl-ref/`.

[11] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[12] R. Brachman and H. Levesque. *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos, 1985.

[13] G. Chen, M. Li, and D. Kotz. Design and implementation of a large-scale context fusion network. pages 246 – 255, Aug. 2004.

[14] H. Chen, F. Perich, T. Finin, and A. Joshi. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, August 2004.

[15] E. Christopoulou, C. Goumopoulos, and A. Kameas. An ontology-based context management and reasoning process for ubicomp applications. In *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 265–270, New York, NY, USA, 2005. ACM Press.

[16] R. Cornet and A. Abu-Hanna. Using description logics for managing medical terminologies. In P. B. M. Dojat, E. Keravnou, editor, *Artificial Intelligence in Medicine: 9th Conference on Artificial Intelligence, in Medicine in Europe (AIME 2003)*, Lecture Notes in Computer Science, pages 61–70. Springer, 2003.

[17] R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL: College of American Pathologists, 1993.

[18] A. K. Dey, D. Salber, and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction (HCl) Journal*, 16(2–4):97–166, 2001.

[19] T. Gu, H. Pung, and D. Zhang. Toward an OSGi-based infrastructure for context-aware applications. *IEEE Pervasive Computing*, 3(4):66–74, Oct.-Dec. 2004.

[20] V. Haarslev, R. Möller, and M. Wessel. Querying the semantic web with racer + nrql. In *In Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADLŠ04*, 2004.

[21] V. Haarslev and R. Möller. Racer: A core inference engine for the semantic web. In *Proc. of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), located with ISWC*, pages 27–36, 2003.

[22] V. Haarslev, R. Möller, and M. Wessel. RacerPro reasoner, 2005. See `http://www.racer-systems.com/`.

[23] C. Haase and C. Lutz. Complexity of subsumption in the $\mathcal{EL}$ family of description logics: Acyclic and cyclic tboxes. In M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *Proc. of the 18th European Conference on Artificial Intelligence (ECAI08)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 25–29. IOS Press, 2008.

[24] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[25] I. Horrocks and U. Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *J. of Automated Reasoning*, 39(3):249–276, 2007.

[26] I. R. Horrocks. Using an expressive description logic: Fact or fiction. pages 636–647, 1998.

[27] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Kernen, and E.-J. Malm. Managing context information in mobile devices. *IEEE Pervasive Computing*, 2(3):42–51, 2003.

[28] M. Luther, Y. Fukazawa, M. Wagner, and S. Kurakake. Situational reasoning for task-oriented mobile service recommendation. *The Knowledge Engineering Review*, 23(Special Issue 01):7–19, 2008.

[29] B. Motik and U. Sattler. A Comparison of Techniques for Querying Large Description Logic ABoxes. In M. Hermann and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 4246 of *Lecture Notes in Computer Science*, pages 227–241, Phnom Penh, Cambodia, November 13–17 2006. Springer. KAON2 download page: `http://kaon2.semanticweb.org/`.

[30] B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. Lecture Notes in Artificial Intelligence, pages 67–83, Bremen, Germany, July 17–20 2007. Springer.

[31] B. Mrohs, M. Luther, R. Vaidya, M. Wagner, S. Steglich, W. Kellerer, and S. Arbanowski. OWL-SF–a distributed semantic service framework. In *Proc. of the Workshop on Context Awareness for Proactive Systems (CAPS'05), Helsinki*, pages 67–77, 2005.

[32] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa. Computational auditory scene recognition. In *In IEEE IntŠl Conf. on Acoustics, Speech, and Signal Processing*, pages 1941–1944, 2002.

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

[34] Racer Systems GmbH & Co. KG. Racerpro reference manual version 1.9, dec. 2005., 2005.

[35] Racer Systems GmbH & Co. KG. Racerpro User's guide version 1.9, dec. 2005., 2005.

[36] C. Ramos, J. C. Augusto, and D. Shapiro. Ambient intelligence – the next step for artificial intelligence. *IEEE Intelligent Systems*, 23(2):15–18, March-April 2008.

[37] A. Ranganathan and R. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, (7):353–364, 2003.

[38] A. Rector. Medical informatics. In *[3]*, pages 406–426. Cambridge University Press, 2003.

[39] S. Schulz, B. Suntisrivaraporn, and F. Baader. SNOMED CT's problem list: Ontologists' and logicians' therapy suggestions. In *Proceedings of The Medinfo 2007 Congress*, Studies in Health Technology and Informatics (SHTI-series). IOS Press, 2007.

[40] N. Shadbolt. Ambient intelligence. *IEEE Intelligent Systems*, 18(4):2–3, 2003.

[41] K. Sheikh, M. Wegdam, and M. van Sinderen. Middleware support for quality of context in pervasive context-aware systems. In *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 461–466, Washington, DC, USA, 2007. IEEE Computer Society.

[42] E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In V. Haarslev and R. Möller, editors, *Proc. of the 2004 Description Logic Workshop (DL 2004)*, number 104 in CEUR Workshop, 2004. See also `http://www.mindswap.org/2003/pellet/index.shtml`.

[43] E. Sirin and B. Parsia. Pellet system description. In B. Parsia, U. Sattler, and D. Toman, editors, *Description Logics*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[44] J. F. Sowa, editor. *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, 1991.

[45] T. Springer, K. Kadner, F. Steuer, and M. Yin. Middleware support for context-awareness in 4G environments. In *World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006. International Symposium on a*, pages 203–211, 26-29 June 2006.

[46] T. Springer, P. Wustmann, I. Braun, W. Dargie, and M. Berger. A comprehensive approach for situation-awareness based on sensing and reasoning about context. In F. E. Sandnes, Y. Zhang, C. Rong, L. T. Yang, and J. Ma, editors, *UIC*, volume 5061 of *Lecture Notes in Computer Science*, pages 143–157. Springer, 2008.

[47] V. Stankovski and J. Trnkoczy. Application of decision trees to smart homes. In *Designing Smart Homes*, volume 4008 of *Lecture Notes in Computer Science*, pages 132–145. Springer Berlin / Heidelberg, 2006.

[48] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[49] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, May 2000.

[50] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, 2006. FaCT++ download page: `http://owl.man.ac.uk/factplusplus/`.

[51] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4130:292–297, 2006.

[52] A.-Y. Turhan, T. Springer, and M. Berger. Pushing doors for modeling contexts with OWL DL – a case study. In J. Indulska and D. Nicklas, editors, *Proceedings of the Workshop on Context Modeling and Reasoning (CoMoRea'06)*, pages 13–17. IEEE Computer Society, March 2006.

[53] M. Weiser. The Computer for the 21st Century. *Scientific American*, pages 66–75, Sep 1991.

[54] M. Wessel, M. Luther, and M. Wagner. The difference a day makes – recognizing important events in daily context logs. In P. Bouquet, J. Euzenat, C. Ghidini, D. L. McGuinness, L. Serafini, P. Shvaiko, and H. Wache, editors, *Proc. of the International Workshop on Contexts and Ontologies: Representation and Reasoning (C&O:RR)*, volume 298 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[55] K. Wolstencroft, A. Brass, I. Horrocks, P. Lord, U. Sattler, R. Stevens, and D. Turi. A little semantic web goes a long way in biology. In *Proc. of the 2005 International Semantic Web Conference (ISWC 2005)*, number 3729 in Lecture Notes in Computer Science, pages 786–800. Springer, 2005.

[56] K. Wolstencroft, P. W. Lord, L. Tabernero, A. Brass, and R. Stevens. Protein classification using ontology classification. In *In Proceedings 14th International Conference on Intelligent Systems for Molecular Biology ISMB'06 (Supplement of Bioinformatics)*, pages 530–538, 2006.

Situation Recognition for Service Management Systems Using OWL 2 Reasoners

Waltenegus Dargie†, Eldora*, Julian Mendez*, Christoph Möbius†,
Kateryna Rybina†, Veronika Thost*, Anni-Yasmin Turhan*

*, Chair for Automata Theory
Institute for Theoretical Computer Science
Technische Universität Dresden

email: *lastname*@tcs.inf.tu-dresden.de

†, Chair of Computer Networks,
Institute of Systems Architecture
Technische Universität Dresden

email: *firstname*. *lastname*@tu-dresden.de

***Abstract*— For service management systems the early recognition of situations that necessitate a rebinding or a migration of services is an important task. To describe these situations on differing levels of detail and to allow their recognition even if only incomplete information is available, we employ the ontology language OWL 2 and the reasoning services defined for it. In this paper we provide a case study on the performance of state of the art OWL 2 reasoning systems for answering class queries and conjunctive queries modeling the relevant situations for service rebinding or migration in the differing OWL 2 profiles.**

I. Introduction

Service management systems (SMS) are systems that, broadly speaking, orchestrate the execution of complex services in distributed (cluster) computing environments. The prime goal of these systems is to ensure that computing resources are efficiently utilized while functional and non-functional requirements of individual services expressed in so-called *service level agreements* (SLAs) are respected. One of the resources managed by an SMS is power.

Unfortunately, a significant portion of the power consumption of Internet-based servers is wasted due to underutilization [6]—often Internet-based servers are utilized only between 30 to 70% of their full capacity even though their idle power consumption amounts up to 60% of their peak power consumption [2], [1]. An SMS can be employed in a cluster environment to ensure that power is frugally consumed by servers. In this regard, a key aspect of the SMS is its ability to adapt the use of hardware resources according to the present and anticipated workload. Depending on the type of services that are being hosted and the priority and magnitude of the processed workload, an SMS can carry out different forms of adaptations to save power.

A. Service Management

One of the essential adaptation strategies is to switch off underutilized machines as often as possible. This can be achieved by consolidating services running on different machines onto a selected number of machines, which are then optimally configured. Service consolidation in turn can be achieved by either carrying out runtime service migration [10] or service rebinding [11]. In a *service migration*, the main memory content of a service is transferred from one physical machine to another at runtime while the service is still executing. In virtualized environments, this can be achieved by encapsulating the service inside a virtual machine (VM) and then migrating the VM itself. If the service is stateless, an SMS may prefer the adaptation technique of *service-rebinding*. In this case, another instance (*service B*) of a service being run on an underutilized server (*service A*) will be started elsewhere and all future requests directed to *service A* will be redirected to *service B*. The aim is to gradually terminate *service A* and switch off the server on which *service A* was running.

Clearly, adaptation techniques can be carried out if there are 'symptoms' indicating a need for adaptation. These symptoms may refer to potential SLA violations or to some utilization thresholds that are being crossed or, to put it more generally, to some critical situations that can be sensed in the system. The time between recognizing a critical situation and finishing all necessary adaptation tasks can be considerably long – for example, the migration of a VM can take several seconds or even minutes, depending on its size and the available network bandwidth [23]. During this delay, the performance of the service may degrade and the power consumption of both the target and the source servers may increase.

Ideally, the SMS should be notified in advance about situations, where a server is overloaded or underutilized and carry out the appropriate adaptations. To this end, it is desirable to describe contextual information that sufficiently characterizes the execution environment (i.e., context pertaining to SLAs, workloads, services, and servers etc.), then recognize situations in the actual system that potentially lead to the violation of one of the predefined thresholds, and decide on the suitable adaptation strategy to alleviate this violation.

In order to do so the situations need to be represented on differing levels of detail—a task Description Logics (DLs) ontologies [5] are designed for. Moreover, the various aspects of the overall managed system's status are supplied by different information sources. For instance, the properties of the different implementations of services may be given by the

This work is supported in a part by the German Research Foundation (DFG) in the Collaborative Research Center 912 'Highly Adaptive Energy-Efficient Computing'.

software providers and can be stored in a database, while other information such as the layout of the hardware and the current system parameters can be retrieved from the OS directly. These information sources yield information of different levels of detail. Data integration can be performed by using DLs, see [9], [7]. Most importantly, DL systems can handle incomplete information gracefully, since they operate under the *open world assumption*, i.e. missing is neither regarded *true* or *false*. In contrast to this, in database systems missing information is regarded to be *false*.

We follow a common approach to ontology-based situation recognition: we use an ontology describing the managed system and servers and employ DL reasoning to identify situations of interest. This approach has been employed in several domains for context-aware applications, for instance, in the intelligent home domain [27], [22] and air surveillance [3]. In [19], [3] it was demonstrated that the expressivity and reasoning capabilities of DL systems suffice to model the domain at hand faithfully.

B. OWL 2 for Situation Recognition

DLs are the logical formalism underlying the W3C standard OWL. In OWL categories from the application domain can be described by *class expressions* and binary relations by so-called *(object) properties*. For example, the class Server, which is hardware that has a CPU as a part and has memory as a part can be characterized by the expression:[1]

$$\mathsf{Server} \equiv \mathsf{Hardware} \sqcap (\exists \mathsf{hasPart}.\mathsf{CPU}) \sqcap (\exists \mathsf{hasPart}.\mathsf{Memory}).$$

The definition assigns to the named class Server the complex class expression on the right-hand side and uses the property hasPart and the other named classes Hardware, CPU and Memory. Now, based on Server we can define an IdleServer as a server that has the power state 'idle' by writing:

$$\mathsf{IdleServer} \equiv \mathsf{Server} \sqcap \exists \mathsf{hasPowerState}.\mathsf{Idle}$$

Such definitions of classes are stored in the *TBox*. In addition, characteristics of properties can be stated in the TBox, e.g., that the property hasPart is transitive or that the property isPartOf is its inverse.[2]

The *ABox* stores concrete facts from the application, expressed by *class assertions*, which state that an individual belongs to a (possibly complex) class or *property assertions* that relate two individuals via a property.

Example 1: We can state in our ABox $\mathcal{A}_1$ that the individual named $Server1$ belongs to the class Server and that its related power state is individual $State2$, which belongs to the class IdleState by writing the statements:

$$\mathcal{A}_1 = \{\ \mathsf{Server}(Server1), \mathsf{hasPowerState}(Server1, State2), \mathsf{IdleState}(State2)\ \}.$$

The TBox and the ABox together constitute the *ontology*. For DL systems there are several *reasoning services* that can infer from the explicitly given information in the ontology the implicitly captured facts. *Subsumption* can compute super- and sub-class relationships for the classes defined in the TBox. For example, it can be derived that the class IdleServer is a subclass of the class $\exists$hasPart.CPU. *Class queries* compute for a given (complex) class $\mathsf{C_q}$ and an ontology all the individuals in the ABox that belong to the given class $\mathsf{C_q}$. For the query class IdleServer and $\mathcal{A}_1$ we can derive the individual $Server1$. A more powerful way to query the ABox are conjunctive queries. A *conjunctive query* is a conjunction of assertions that may also contain variables, of which some can be existentially quantified. For example, the conjunctive query

$$q_{ex} = \exists x, y.\mathsf{Server}(x) \wedge \mathsf{hasPart}(x, z) \wedge \mathsf{uses}(y, z) \wedge \mathsf{Process}(y)$$

asks for all pairs of servers and processes, where the process uses some part of the server. In contrast to class queries, conjunctive queries can return a tuple of individuals from the ABox.

We model the basic categories and relations of the SMS domain in a TBox, such as the hardware or the managed services. The current state of the system managed by the SMS is then captured at runtime in an ABox, similarly to [27], [22], [3]. To recognize the relevant situations for the SMS we employ answering of class queries or conjunctive queries w.r.t. the ABox. Once such a situation is detected for a (tuple of) ABox individual(s), the SMS invokes the appropriate adaptations on the returned individuals to ensure energy efficiency for the overall system.

The OWL 2 standard for ontology languages comprises so-called *OWL profiles* which differ w.r.t. expressivity [28]. Depending on the profile, more class constructors and property statements are allowed for the TBox.

- OWL 2 is the most expressive ontology language in the W3C standard. Reasoning in the corresponding DL $\mathcal{SROIQ}$ is 2NExpTime-complete [13], [14], i.e. class queries can take more than double exponential time.
- OWL 2 EL corresponds to the DL $\mathcal{EL}^{++}$, where class query answering is in P [4], i.e. can always be done in polynomial time. However, conjunctive query answering in the sublogic $\mathcal{EL}$ is already P-complete w.r.t. the size of the ABox alone.
- OWL 2 QL allows only for very limited class descriptions. For its corresponding DL DL-Lite$_\mathcal{R}$ query answering is in AC^0 (which is a proper subclass of the class of P), if measured w.r.t. the size of the ABox alone [8].

The motivation for the different profiles are the good computational properties of the lightweight DLs $\mathcal{EL}^{++}$ and DL-Lite$_\mathcal{R}$ for answering class queries or conjunctive queries respectively. There are non-commercial, optimized reasoners for answering class queries or conjunctive queries. Although the computational complexity of the implemented algorithms is promisingly low, it is not clear whether these implementations are yet fast enough to realize situation recognition for applications that deal with complex situations and require fast response times—such as SMSs. While [19], [3] argued that the reasoning capabilities of DL systems suffice to recognize complex

[1] We give the class expressions in DL syntax for better readability.

[2] For the exact syntax and semantics of DLs we refer the reader to [5].

situations, little is known about whether the performance of the implementation of DL reasoners' performance is yet good enough for this kind of task. This question was addressed in the study in [27] back in 2006 for class queries, where it turned out that for fairly small ontologies and only applications that require moderate response times (of about 20 seconds), the performance of the DL reasoners was barely adequate. Since then DL reasoners have evolved in terms of reasoning services offered and in terms of performance. This motivates our empiric study that measures the performance of today's reasoning systems for class queries and conjunctive queries for the different OWL profiles. The application is to recognize situations for an SMS that manages a video platform such that it runs in an energy efficient way.

The paper is structured as follows: Section II describes the ontology for the video platform use case and the modeling of the relevant situations. Section III presents the empirical evaluation how current DL reasoners perform on class and conjunctive queries w.r.t. the different OWL profiles. As it is custom our paper ends with some conclusions.

II. Use Case: Managing a Video Platform Server

For a proof of concept for our DL-based approach for SMS, we consider a video platform as application scenario, i.e., a distributed application over several servers, which allows users to search for, upload, and download videos. Internally, services for ranking and transcoding of videos (i.e., conversion of video encodings) are executed. The up- and downloading of videos are complex and resource-intensive processes. For that reason, we apply an elaborate service management to effectively exploit the available resources.

The two techniques considered to reduce the energy consumption of the video server platform are service migration and service rebinding. Service rebinding is performed in case one server is not optimally utilized, while another server still has available resources. Consider a server providing a downloading service. If several users request this service at the same time, the server becomes overloaded. To balance the load, this downloading service can be rebound – by starting an instance of this service on another server that has available resources and by 'redirecting' future requests to the new instance.

To recognize situations where the application of such adaptation techniques can be beneficial, we create ontologies capturing information about the system and then apply DL reasoners to detect situations apt for optimization. More precisely, at design time the general domain knowledge about video platforms (e.g., the kinds of services provided) and notions of SMS (e.g., when a server has available resources) are described in the TBox. The relevant situations to be recognized are modeled as query classes or conjunctive queries–depending on the reasoning task to be employed. The TBox and the queries are assumed to be fixed over the runtime of the SMS.

The ABox describes the architecture of the specific application managed by the SMS (e.g., available servers) and its current state (e.g., load of the servers, executed implementations, etc.). Most of the data in the ABox has to be collected at runtime. Due to the highly dynamic nature of the system, the ABox is refreshed several times a minute. Each ABox can be generated from many sources as, for instance, sensor data delivered by the OS or a database describing all implementations available to the SMS. For the task of converting numerical data, such as sensor data, *preprocessors* are applied, to convert the numeric data into named classes (following the approach used in [3], [24]). For example, if the load measured for a server $Server1$ has been constantly very low, the assertions

$$\mathsf{hasLoadAverage}(Server1, Load2),\ \mathsf{UnderUtilized}(Load2)$$

are added to the ABox created for the past interval. Once the ABox is refreshed, the DL reasoner performs answering of the class or conjunctive queries provided at design time.

A. Modeling the OWL 2 Video Platform Ontology

Our TBox contains basic notions of the video platform domain such as characteristics of a DownloadingService and notions specific for SMS as AvailableResourceServer written in the DL $\mathcal{ALCIQ}$, which is a proper sub-logic of OWL 2. For this DL, testing class queries is PSpace-complete [25], while answering of conjunctive queries is even 2ExpTime-complete [16]. Our ABox contains assertions describing the architecture of the video platform and its current state based on the available sensor data.

Example 2: Let's assume that $State1$ from ABox $\mathcal{A}_1$ has changed to 'operating' in the last interval. Now, the characterization of $Server1$, its resources, and states at runtime can be captured by:

$\mathsf{Server}(Server1)$, $\mathsf{CPU}(CPU1)$, $\mathsf{Memory}(Memory1)$, $\mathsf{Operating}(State1)$, $\mathsf{UnderUtilized}(Load1)$, $\mathsf{hasPart}(Server1, CPU1)$, $\mathsf{hasPart}(Server1, Memory1)$, $\mathsf{hasPowerState}(Server1, State1)$, $\mathsf{hasLoadAverage}(Server1, Load1)$

It turned out that even the expressivity of the lightweight profiles allows to describe at least the main characteristics of the domain knowledge of our application scenario. This is because the TBox primarily captures the conceptual model of the application, which is exactly the use-case DL-Lite has been developed for. If needed, complex class definitions, which cannot be represented in the lightweight profiles, can be captured alternatively using fine-granular conjunctive queries when modeling the rebinding situations.

Example 3: Consider the class definition for underutilized servers, which have an average load that is underutilized or that have a part that is an underutilized CPU or NIC:

$$\mathsf{UnderUtilizedServer} \equiv \exists \mathsf{hasLoadAverage}.\mathsf{UnderUtilized} \sqcup \exists \mathsf{hasPart}.(\mathsf{UnderUtilizedCPU} \sqcup \mathsf{UnderUtilizedNIC})$$

It cannot be expressed in an OWL 2 EL/QL ontology, due to disjunction ($\sqcup$). Thus, such a query concept would have to consist of the right-hand side of the definition.

$$\mathsf{RebindingDownloadingServiceSituation} =$$
$$\exists \mathsf{hasServer}.(\mathsf{AvailableResourceServer} \sqcap \exists \mathsf{runs}.\exists \mathsf{hosts}.\mathsf{DownloadingImplementation}) \sqcap$$
$$\exists \mathsf{hasServer}.(\neg \mathsf{OptimallyUtilizedServer} \sqcap \exists \mathsf{runs}.\exists \mathsf{hosts}.\exists \mathsf{bindsTo}.\mathsf{DownloadingService})$$
$$\mathsf{RebindingServiceSituation} =$$
$$\exists \mathsf{hasServer}.(\mathsf{AvailableResourceServer} \sqcap \exists \mathsf{runs}.\exists \mathsf{hosts}.\mathsf{Implementation}) \sqcap$$
$$\exists \mathsf{hasServer}.(\neg \mathsf{OptimallyUtilizedServer} \sqcap \exists \mathsf{runs}.\exists \mathsf{hosts}.\exists \mathsf{bindsTo}.\mathsf{Service})$$
$$q_{\mathsf{RebindingDownloadingServiceSituation}} =$$
$$\exists x, y.\ \mathsf{AvailableResourceServer}(x) \wedge \mathsf{runs}(x, z_1) \wedge \mathsf{hosts}(z_1, z_2) \wedge \mathsf{DownloadingImplementation}(z_2) \wedge$$
$$\mathsf{bindsTo}(z_2, z_3) \wedge \mathsf{DownloadingService}(z_3) \wedge$$
$$\neg \mathsf{OptimallyUtilizedServer}(y) \wedge \mathsf{runs}(y, z_4) \wedge \mathsf{hosts}(z_4, z_5) \wedge \mathsf{DownloadingImplementation}(z_5) \wedge$$
$$\mathsf{bindsTo}(z_5, z_6) \wedge \mathsf{isBoundTo}(z_6, z_5) \wedge \mathsf{DownloadingService}(z_6)$$
$$q_{\mathsf{RebindingServiceSituation}} =$$
$$\exists x, y.\ \mathsf{AvailableResourceServer}(x) \wedge \mathsf{runs}(x, z_1) \wedge \mathsf{hosts}(z_1, z_2) \wedge \mathsf{Implementation}(z_2) \wedge$$
$$\mathsf{bindsTo}(z_2, z_3) \wedge \mathsf{Service}(z_3) \wedge$$
$$\neg \mathsf{OptimallyUtilizedServer}(y) \wedge \mathsf{runs}(y, z_4) \wedge \mathsf{hosts}(z_4, z_5) \wedge \mathsf{Implementation}(z_5) \wedge \mathsf{bindsTo}(z_5, z_6) \wedge$$
$$\mathsf{isBoundTo}(z_6, z_5) \wedge \mathsf{Service}(z_6)$$

Fig. 1. The situation when to rebind a (downloading) service captured as query classes and conjunctive queries.

B. Modeling the Rebinding Situations

To recognize critical situations, we apply either answering of class queries or of conjunctive queries. For the former, the situations need to be specified as classes, while for the latter, the situations need to be described by conjunctive queries.

Example 4: A situation apt for rebinding a downloading service considers two servers, one with available resources and one that is not optimally utilized. The first one hosts the corresponding implementation and the second one hosts the same implementation currently bound by the service. The resulting query class is displayed in Figure 1 in the upper half as the class RebindingDownloadingServiceSituation and the corresponding conjunctive query $q_{\mathsf{RebindingDownloadingServiceSituation}}$ in the lower half of the figure. Note, that the fact that the same implementation is used by the servers cannot be expressed by a class description, since they only allow to describe tree-like structures. Furthermore, conjunctive queries retrieve tuples from the ABox, while a query concept can only retrieve a single individual.

In Figure 1 a situation that generalizes the above one is characterized in the query class RebindingServiceSituation and in the query $q_{\mathsf{RebindingDownloadingServiceSituation}}$, respectively. In this situation the service and the implementation are not further specified, otherwise these situations are the same. Clearly, this situation is refined by the first one.

It is fruitful to model such refinement of situations in order to allow for graceful handling of incomplete information. Assume, that it is stated in the TBox that every DownloadingServer is a Server and that every DownloadingImplementation is an Implementation. Furthermore, assume that for a particular downloading implementation it cannot be retrieved that it is an implementation of that kind, but only that it is an implementation (of some kind). Thus the next ABox is incomplete. In such a case a situation that might necessitate the rebinding of a downloading service cannot be recognized. More precisely, the class RebindingDownloadingServiceSituation does not have an instance in the current ABox and the query $q_{\mathsf{RebindingDownloadingServiceSituation}}$ yields no tuples. However, the more general class RebindingServiceSituation would have an instance and the query $q_{\mathsf{RebindingServiceSituation}}$ would yield a result tuple. Thus a counter measure could be invoked at least for this kind of situation.

Class and conjunctive queries differ in the expressive power for specifying the situations. While the former are limited by the expressivity of the ontology language, the latter can in addition make use of the variables to describe arbitrary structures to describe the details of the situations. This addition comes at the cost of higher computational complexity.

III. Evaluation for the OWL 2 profiles

The goal of our evaluation is to see whether current OWL 2 reasoners are appropriate for situation recognition in SMSs. However, to adopt DL reasoning for this kind of scenario, the reasoners have to be able to detect situations by processing realistic amounts of data within short time. We consider OWL 2 and the two profiles OWL 2 EL and OWL 2 QL in our evaluation. However, the syntactic restrictions of the lightweight profiles allow only for coarser modeling than full OWL 2—some information simply cannot be modeled. An interesting question is whether this leads to missing inferences in our scenario.

A. Test Data and Reasoning Systems

a) Test ontologies: Our base TBox from Section II-A contains 113 class and 66 property definitions and uses $\mathcal{ALCIQ}$ a sub-logic of OWL 2. For both lightweight profiles, we built variations of the base TBox manually—keeping as much information as possible. Since the OWL 2 QL profile does not support truly complex class descriptions, the situations in the OWL 2 QL TBox cannot be modeled as classes.

However, the necessary information can be captured in the conjunctive queries. Thus we only test answering conjunctive queries for the QL profile.

The ABoxes model a video platform running on four servers and providing the services described in Section II. Since the class assertions use only named classes, the ABoxes do not vary for the profiles. We consider two different ABoxes modeling two different states of the system. In order to reflect realistic scenarios, the test ABoxes do not only contain information about the situation to be detected, but model the overall system state. We added data about other users requesting video services, which are carried out on other servers. This roughly doubles the sizes of both ABoxes. Each of the test ABoxes contains about 380 individuals, more than 770 class, and more than 545 property assertions.

b) Test Queries: We modeled 13 situations as OWL 2 classes. Since OWL 2 EL does not offer universal quantification, only 11 of them are modeled as OWL 2 EL classes. For these 11 situations we formulated the corresponding conjunctive queries included in our test suite. The class queries have a size of about 10 counting the class and property names. The conjunctive queries are formulated in the query languages SPARQL and nrql. They contain on average 15 disjuncts of conjunctions with 8 conjuncts each.

c) Reasoner Systems: The tests were run for seven DL reasoners, which differ w.r.t. the DL they support and the reasoning services provided. Table I depicts the tested reasoners, the used version and the closest DL of the respective profile they implement ('x' stands for full coverage). Besides the tableaux-based reasoners for expressive DLs in the first group of Table I, we tested reasoners specialized on lightweight profiles, which are listed in the second and third group of the table. QUEST can be used for ontology based data access (i.e., a data base functions as ABox and can be queried directly). We used QUEST with classical ABoxes here.

B. Evaluation

The tests were carried out on an Intel Core 2 Duo workstation with 2 GB RAM using Java 1.6.0. on Ubuntu. Besides recording the mere runtimes, we checked whether the reasoners delivered the same results for a query. For our class and conjunctive queries, all reasoners agree on the result tuples. However, when comparing the results for conjunctive queries w.r.t. differing expressiveness of the profiles, it shows that

		Query type		Profile		
Reasoner	Version	Class	Conj.	OWL	EL	QL
FACT++ [26]	v1.6.1	x		x	x	x
HERMIT [18]	v1.3.6	x		$\mathcal{SHOIQ}$	x	x
PELLET [21]	v2.3.0	x	x	$\mathcal{SHOIN}(\mathcal{D})$	x	x
RACERPRO [12]	v2.0	x	x	$\mathcal{SHIQ}(\mathcal{D})$	x	x
ELK [15]	v0.3.1	x			$\mathcal{EL}^+$	
JCEL [17]	v0.18.0	x			$\mathcal{EL}^+$	
QUEST [20]	v1.7-alpha		x			x

TABLE I

REASONERS AND THEIR SUPPORTED QUERIES AND PROFILES.

		Load.	Reason.	Avg/Query	Total
OWL	HERMIT	0.180	1.832	0.148	2.012
	PELLET	0.203	0.434	0.033	0.637
	FACT++	0.208	0.225	0.017	0.433
	RACERPRO	0.199	24.163	1.985	24.362
EL	ELK	0.228	0.078	0.004	0.306
	FACT++	0.245	0.063	0.002	0.308
	HERMIT	0.212	0.120	0.004	0.332
	JCEL	0.230	0.199	0.012	0.429
	PELLET	0.197	0.576	0.045	0.773
	RACERPRO	0.342	1.675	0.112	2.018

TABLE II

RUNTIMES FOR CLASS QUERIES IN SECONDS.

		Load.	Reason.	Avg/Query	Total
OWL	RACERPRO	1.302	40.035	3.336	41.336
EL	PELLET	0.522	2.344	0.195	2.866
	RACERPRO	0.503	6.773	0.564	7.277
QL	PELLET	0.541	1.918	0.160	2.460
	QUEST	0.453	93.208	7.767	93.661
	RACERPRO	0.349	6.604	0.550	6.953

TABLE III

RUNTIMES FOR CONJUNCTIVE QUERIES IN SECONDS.

RACERPRO detects all of the (expected) tuples for OWL 2, while less tuples are returned for the lightweight profiles. As to be expected, this is due to the loss in expressivity when using a lightweight profile. We observed the same effect for the lightweight profiles in the results of PELLET and QUEST.

1) Performance for Class Queries: For class queries we ran tests for the OWL 2 and the OWL 2 EL profile. We used the OWL API (version 3.4.1) to access the reasoners. The results are displayed in Table II, sorted by profiles. The first column depicts the time spent on loading the ontology. The next one displays the time for answering *all* the queries. The average runtime per query is displayed next. The last column contains the runtime for the overall process and is thus the most interesting for our application of situation recognition. As expected, it shows that the overall runtime is 6-10 times higher for OWL 2 than for the OWL 2 EL profile with the exception of PELLET, which performs slight better for OWL.

OWL 2: Apart from RACERPRO, which took about 25 seconds, all reasoners delivered a full situation recognition within 2 seconds.

OWL 2 EL: With an overall runtime of about 0.3 seconds, ELK, FACT++, and HERMIT outperform the other systems. All systems can perform situation recognition within 0.8 seconds besides RACERPRO, which, again, takes considerably more time.

2) Performance for Conjunctive Queries: For the conjunctive queries, the results for all of the three profiles are displayed in Table III. As for class querying, reasoning in the lightweight profiles is much faster.

OWL 2: Here RACERPRO needs about 41 seconds overall runtime. Interestingly, compared to the corresponding class query, it takes nearly twice as long, due to one outlier query.

OWL 2 EL: PELLET answers all queries in less than 3 seconds, but takes about four times as long for class queries. With

7.2 seconds RACERPRO takes more than twice as long than PELLET.

OWL 2 QL: The times of PELLET and RACERPRO are similar to the OWL 2 EL case. QUEST, in contrast, shows a significantly worse performance by taking more than 1.5 minutes. We conjecture that this is attributed to running a first alpha version of it and with a traditional ABox (i.e., instead of using a database).

All in all, the experiments show that most state of the art reasoners can be applied for situation recognition in our SMS application, since response times of half a minute would be acceptable. Especially by the use of the lightweight profiles, we achieve very good runtimes for reasoning. Surprisingly, the loss of information, when using a light weight profile, turned out to be only marginal for our video platform use case.

IV. CONCLUSIONS AND FUTURE WORK

We have supplied a study on employing state of the art DL reasoners to perform situation recognition for service management applied to a video platform. The task was to recognize complex situations that might invoke rebinding of services in order to achieve energy efficiency. To solve this task the domain was modeled in an OWL 2 ontology, where the ABox reflected realistic situations in the application. The actual recognition of critical situations, was realized by class or conjunctive query answering. Our experiments w.r.t. the different OWL 2 profiles gave evidence that the performance of today's DL systems is sufficient to detect complex situations fast enough. In particular, for the OWL 2 EL and the OWL 2 QL profile it can be done within 2 seconds.

Future work on the practical side includes to run QUEST in the ODBA mode and to realize the whole situation recognition more tightly coupled to a DB, such that the data collected there can be queried directly, instead of generating and loading an ABox. On the theoretical side, we would like to lift the limitation of OWL regarding the modeling of fuzzy or even temporal information by investigating query answering for sequences of ABoxes, which contain this kind of information.

REFERENCES

[1] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy proportional datacenter networks. *SIGARCH Comput. Archit. News*, 38(3):338–347, 2010.

[2] F. Ahmad and T. N. Vijaykumar. Joint optimization of idle and cooling power in data centers while maintaining response time. In *Proc. of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*, ASPLOS '10, p. 243–256, USA, 2010. ACM.

[3] F. Baader, A. Bauer, P. Baumgartner, A. Cregan, A. Gabaldon, K. Ji, K. Lee, D. Rajaratnam, and R. Schwitter. A novel architecture for situation awareness systems. In *Proc. of the 18th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux'09)*, vol. 5607 of *LNCS*, p. 77–92. Springer, 2009.

[4] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In K. Clark and P. F. Patel-Schneider, eds. *In Proc. of the OWLED Workshop*, 2008.

[5] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds.˙ *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[6] L. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, Dec. 2007.

[7] A. Borgida, M. Lenzerini, and R. Rosati. Description logics for databases. In *[5]*, p. 462–484. Cambridge University Press, 2003.

[8] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

[9] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Knowledge representation approach to information integration. In *Proc. of AAAI Workshop on AI and Information Integration*, p. 58–65. AAAI Press/The MIT Press, 1998.

[10] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Vol. 2*, p. 273–286. USENIX Association, 2005.

[11] W. Dargie, A. Strunk, and A. Schill. Energy-aware service execution. In *Proc. of the 36th Annual IEEE Conference on Local Computer Networks*, 2011.

[12] V. Haarslev, K. Hidde, R. Möller, and M. Wessel. The RacerPro knowledge representation and reasoning system. *Semantic Web Journal*, 3(3):267–277, 2012.

[13] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In P. Doherty, J. Mylopoulos, and C. Welty, eds. *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-06)*, p. 57–67. AAAI Press, 2006.

[14] Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In G. Brewka and J. Lang, eds. *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-08)*, p. 274–284. AAAI Press, 2008.

[15] Y. Kazakov, M. Krötzsch, and F. Simančík. ELK reasoner: Architecture and evaluation. In I. Horrocks, M. Yatskevich, E. Jimenez-Ruiz, editor, *Proc. of the OWL Reasoner Evaluation Workshop (ORE'12)*, vol. 858 of *CEUR*, 2012.

[16] C. Lutz. The complexity of conjunctive query answering in expressive description logics. In A. Armando, P. Baumgartner, and G. Dowek, eds. *Proc. of the 4th International Joint Conference on Automated Reasoning (IJCAR'08)*, nr. 5195 in LNAI, p. 179–193. Springer, 2008.

[17] J. Mendez. jCel: A modular rule-based reasoner. In *In Proc. of the 1st Int. Workshop on OWL Reasoner Evaluation (ORE'12)*, vol. 858 of *CEUR*, 2012.

[18] B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In F. Pfennig, editor, *Proc. of the 23th Conf. on Automated Deduction (CADE-23)*, LNAI, p. 67–83, Germany, 2007. Springer.

[19] B. Neumann and R. Möller. On scene interpretation with description logics. In H. Christensen and H.-H. Nagel, eds. *Cognitive Vision Systems: Samping the Spectrum of Approaches*, nr. 3948 in LNCS, p. 247–278. Springer, 2006.

[20] M. Rodriguez-Muro and D. Calvanese. Quest, an OWL 2 QL reasoner for ontology-based data access. In *Proc. of the 9th Int. WS on OWL: Experiences and Directions (OWLED'12)*, vol. 849 of *CEUR*, 2012.

[21] E. Sirin and B. Parsia. Pellet system description. In B. Parsia, U. Sattler, and D. Toman, eds. *Description Logics*, vol. 189 of *CEUR*, 2006.

[22] T. Springer and A.-Y. Turhan. Employing description logics in ambient intelligence for modeling and reasoning about complex situations. *J. of Ambient Intelligence and Smart Environments*, 1(3):235–259, 2009.

[23] A. Strunk and W. Dargie. Does live migration of virtual machines cost energy? In *Proc. of the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013)*, 2013.

[24] K. Taylor and L. Leidinger. Ontology-driven complex event processing in heterogeneous sensor networks. In *Proc. of 8th Extended Semantic Web Conference (ESWC'11)*, vol. 6644 of *LNCS*, p. 285–299. Springer, 2011.

[25] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.

[26] D. Tsarkov, I. Horrocks, and P. F. Patel-Schneider. Optimising terminological reasoning for expressive description logics. *J. of Automated Reasoning*, 2007.

[27] A.-Y. Turhan, T. Springer, and M. Berger. Pushing doors for modeling contexts with OWL DL – a case study. In J. Indulska and D. Nicklas, eds. *Proc. of the Workshop on Context Modeling and Reasoning (CoMoRea'06)*. IEEE Computer Society, 2006.

[28] W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation, 27th October 2009. `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`.

A Practical Approach for Computing Generalization Inferences in $\mathcal{EL}$

Rafael Peñaloza and Anni-Yasmin Turhan

TU Dresden, Germany,
Institute of Theoretical Computer Science
email: *last name*@tcs.inf.tu-dresden.de

Abstract. We present methods that compute generalizations of concepts or individuals described in ontologies written in the Description Logic $\mathcal{EL}$. These generalizations are the basis of methods for ontology design and are the core of concept similarity measures. The reasoning service least common subsumer (lcs) generalizes a set of concepts. Similarly, the most specific concept (msc) generalizes an individual into a concept description. For $\mathcal{EL}$ with general $\mathcal{EL}$-TBoxes, the lcs and the msc may not exist. However, it is possible to find a concept description that is the lcs (msc) up to a certain role-depth.
In this paper we present a practical approach for computing the lcs and msc with a bounded depth, based on the polynomial-time completion algorithm for $\mathcal{EL}$ and describe its implementation.

1 Introduction

Ontologies have become a commonly used means to describe controlled vocabularies, most prominently, in life sciences. Categories that form these vocabularies are sometimes only described in terms of specializations, i.e. by the "is-a" relation. Since the standardization of the web ontology language OWL [25], more applications have begun using this richer modeling language for describing notions from their domain in a more precise and detailed way. The formalism underlying OWL are Description Logics (DLs) [3], which are a family of logics with formal semantics. The formal semantics of DLs are the basis for the definition of reasoning services such as *subsumption* or *instance checking*. Subsumption tests whether a sub- / super-concept relationship holds between a pair of concept descriptions. Instance checking answers the question whether it follows from the ontology that a given individual must belong to a concept. The reasoning algorithms for these reasoning services are well-investigated for a range of DLs and implemented in powerful reasoner systems. In this paper we want to devise computation methods for inferences that can be employed to derive generalizations. These inferences turn out to be useful for range of ontology-based applications such as e.g. the life sciences [21, 9] or context-aware systems [22].

The newest version of the OWL standard [25] offers several *OWL profiles*, which correspond to DLs with varying expressivity. We are interested in the OWL EL profile, which corresponds to the DL $\mathcal{EL}$++, an extension of the DL $\mathcal{EL}$ where reasoning is still tractable. $\mathcal{EL}$-concept descriptions are composed from conjunctions or existential restrictions. Despite its limited expressivity, $\mathcal{EL}$ has turned out to be useful to model

notions from life science applications. Most prominently, the medical ontology SnoMed [21] and the Gene Ontology [9] are written in $\mathcal{EL}$. For instance, it is possible to express by

$$\mathsf{Myocarditis} \sqsubseteq \mathsf{inflammation} \sqcap \exists \mathsf{has-location.heart}$$

that myocarditis is a kind of inflammation that is located in the heart.

In fact, medical and context-aware applications deal with very large ontologies, which are often *light-weight*, in the sense that they can be formulated in $\mathcal{EL}$ or one of its extensions from the so-called $\mathcal{EL}$-family. Members of the $\mathcal{EL}$-family allow for reasoning in polynomial time [2]. In particular, subsumption and instance checking are tractable for $\mathcal{EL}$ and $\mathcal{EL}$++, which was the main reason to standardize it in an own OWL 2 profile [25]. The reasoning algorithms for the $\mathcal{EL}$-family are based on a completion method and have been implemented in optimized reasoners such as CEL [16].

We investigate here two inferences that generalize different entities from DL knowledge bases. The first one is the *least common subsumer* (lcs) [7], which generalizes a collection of concept descriptions into a single concept description that is the least w.r.t. subsumption. Intuitively, the lcs yields a new (complex) concept description that captures all the commonalities of the input concept descriptions. The second inference is the *most specific concept* (msc) [4], which generalizes an individual into a concept description. Intuitively, the msc delivers the most specific concept description that is capable of describing the individual.

Applying Generalization Inferences

In the following we describe some of the most prominent applications of the lcs and the msc.

Similarity measures. Concept similarity measures compute, given a pair of concept descriptions, a numerical value between 0 and 1 that lies closer to 1 the more similar the concepts are. Similarity measures are an important means to discover, for instance, functional similarities of genes modeled in ontologies. In [13] and, more recently, in [19] several similarity measures were evaluated for the Gene Ontology and it was concluded that the similarity measure from Resnik [20] performed well, if not best. This similarity measure is an edge-based approach, which finds the most specific common ancestor (msa) [1] of the concepts to be compared in the concept hierarchy and computes a similarity value based on the number of edges between the concepts in question and their msa. Clearly, the msa can only yield a named concept from the TBox and thus captures possibly only *some* of the commonalities of the concepts to be compared. The lcs, in contrast, captures *all* commonalities and is thus a more faithful starting point for a similarity measure. In fact, the lcs was employed for similarity measures for DLs in [6] already. In a similar fashion a similarity measure for comparing individuals can be based on the msc [10]

[1] Sometimes also called *least common ancestor* (lca)

Building ontologies. In [11] it was observed that users working with biological ontologies would like to develop the description of the application categories in an example-driven way. More precisely, users would like to start by modeling individuals which are then generalized into a concept description. In fact, in the bottom-up approach for the construction of knowledge bases [4], a collection of individuals is selected for which a new concept definition is to be introduced in the ontology. Such a definition can be generated automatically by first generalizing each selected individual into a concept description (by computing the msc for each of them) and then applying the lcs to these concept descriptions.

The lcs can also be employed to enrich unbalanced concept hierarchies by adding new intermediate concepts [23].

Reconciling heterogeneous sources. The bottom-up procedure sketched before can also be employed in applications that face the problem that different information sources provide differing observations for the same state of affairs. For instance, in context-aware systems a GPS sensor or a video camera can provide differing information on a the location of a user. Alternatively, in medical applications, different diagnosing methods may yield differing results. It can be determined what the different sources agree on by representing this information as distinct ABox individuals and then by finding a common generalization of them by the bottom-up approach.

Information retrieval. The msc inference can be employed to obtain a query concept from an individual to search for other, similar individuals in an ontology [15, 8].

In order to support all these ontology services for practical applications automatically, computation algorithms for the generalization inferences in $\mathcal{EL}$ are needed. Unfortunately, the lcs in $\mathcal{EL}$ does not always exist, when computed w.r.t. cyclic TBoxes [1]. Similarly, the msc in $\mathcal{EL}$ does not always exist, if the ABox is cyclic [12], mainly because cyclic structures cannot be captured in $\mathcal{EL}$-concept descriptions. In [12] the authors propose to use an approximation of the msc by limiting the role-depth of the concept description computed. We pursue this approach here for the lcs and the msc and thus would obtain only "common subsumers" and "specific concepts" that are still generalizations of the input, but not necessarily the least ones w.r.t. subsumption. However, by our proposed method we obtain *the* lcs or *the* msc w.r.t. the given role depth bound. We argue that such approximations are still useful in practice.

Recently, a different approach for obtaining the lcs (or the msc) in presence of cyclic knowledge bases was proposed in [14] by extending $\mathcal{EL}$ with concept constructors for greatest fixpoints. In the so obtained DL $\mathcal{EL}^{\nu}$ reasoning stays polynomial and the lcs and msc w.r.t. cyclic knowledge bases can be computed. However, the DL obtained by adding constructors for greatest fixpoints is possibly not easy to comprehend for naive users of ontologies.

For medical or context-aware applications knowledge bases can typically grow very large in practice. Thus, in order to support the computation of the (role-depth bounded) lsc or the msc for such applications, efficient computation of these generalizations for $\mathcal{EL}$ is desirable. Our computation methods build directly on the completion method for subsumption and instance checking for $\mathcal{EL}$ [2] for which optimizations already exists

and are employed in modern reasoner systems. This enables the implementation of the role-depth bounded lcs and msc on top of existing reasoner systems. More precisely, in our completion-based approach, we obtain the role-depth bounded lcs by traversing the data-structures built during the computation of the subsumption hierarchy of the ontology. The role-depth bounded msc can be obtained from the data-structures generated during the computation of all instance relations for the knowledge base. We have recently implemented the completion-based computation of the role-depth bounded lcs and msc in our system GEL.

This paper is structured as follows: after introducing basic notions of DLs, we discuss the completion algorithms for classification and instance checking in $\mathcal{EL}$ in Section 3. We extend these methods to computation algorithms for the role-depth bounded lcs in Section 4.1 and for the role-depth bounded msc in Section 4.2 and we describe our initial implementation of the presented methods in Section 5. We conclude the paper with an outline of possible future work.

2 Preliminaries

We now formally introduce the DL $\mathcal{EL}$. Let N_I, N_C and N_R be disjoint sets of *individual names*, *concept names* and *role names*, respectively. *$\mathcal{EL}$-concept descriptions* are built according to the syntax rule

$$C ::= \top \mid A \mid C \sqcap D \mid \exists r.C$$

where $A \in N_C$, and $r \in N_R$.

A *general concept inclusion* (GCI) is a statement of the form $C \sqsubseteq D$, where C, D are $\mathcal{EL}$- concept descriptions. An *$\mathcal{EL}$-TBox* is a finite set of GCIs. Observe that TBoxes can be cyclic and allow for multiple inheritance. An *$\mathcal{EL}$-ABox* is a set of assertions of the form $C(a)$, or $r(a, b)$, where C is an $\mathcal{EL}$-concept description, $r \in N_R$, and $a, b \in N_I$. An *ontology* or *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

The semantics of $\mathcal{EL}$ is defined by means of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names, subsets of $\Delta^{\mathcal{I}}$ to concepts and elements of $\Delta^{\mathcal{I}}$ to individual names. The interpretation function $\cdot^{\mathcal{I}}$ is extended to concept descriptions in the usual way. For a more detailed description of the semantic of DLs see [3].

An interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; it *satisfies* an assertion $C(a)$ (or $r(a, b)$), denoted as $\mathcal{I} \models C(a)$ ($\mathcal{I} \models r(a, b)$, resp.) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ($(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, resp.). An interpretation $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it satisfies all GCIs in $\mathcal{T}$ and all assertions in $\mathcal{A}$.

We say that C is *subsumed* by D w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$. The computation of the subsumption hierarchy of all named concepts in a TBox is called *classification*.

Finally, an individual $a \in N_I$ is an *instance* of a concept description C w.r.t. $\mathcal{K}$ (written $\mathcal{K} \models C(a)$) if $\mathcal{I} \models C(a)$ for all models $\mathcal{I}$ of $\mathcal{K}$. *ABox realization* is the task of computing, for each individual a in $\mathcal{A}$, the set of named concepts from $\mathcal{K}$ that have a as an instance and that are least (w.r.t. $\sqsubseteq$).

In this paper we are interested in computing generalizations by least common subsumers and most specific concepts, which we now formally define. Notice that our definition is general for any DL and not necessarily specific for $\mathcal{EL}$.

Definition 1 (least common subsumer). *Let $\mathcal{L}$ be a DL, $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB. The* least common subsumer *(lcs) w.r.t. $\mathcal{T}$ of a collection of concepts $C_1, \ldots, C_n$ is the $\mathcal{L}$-concept description C such that*

1. *$C_i \sqsubseteq_{\mathcal{T}} C$ for all $1 \leq i \leq n$, and*
2. *for each $\mathcal{L}$-concept description D holds: if $C_i \sqsubseteq_{\mathcal{T}} D$ for all $1 \leq i \leq n$, then $C \sqsubseteq_{\mathcal{T}} D$.*

We will mostly consider the DL $\mathcal{EL}$ in this paper. Although defined as an n-ary operation, we will often write the lcs as a binary operation in the remainder of the paper for simplicity.

Definition 2 (most specific concept). *Let $\mathcal{L}$ be a DL, $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB. The* most specific concept *(msc) w.r.t. $\mathcal{K}$ of an individual a from $\mathcal{A}$ is the $\mathcal{L}$-concept description C such that*

1. *$\mathcal{K} \models C(a)$, and*
2. *for each $\mathcal{L}$-concept description D holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

Both inferences depend on the DL in use. For the DLs with conjunction as concept constructor the lcs and msc are, if exist, unique up to equivalence. Thus it is justified to speak of *the* lcs or *the* msc. Our computation methods for generalizations are based on the completion method, which we introduce in the following section.

3 Completion Algorithms for $\mathcal{EL}$

In principle, completion algorithms try to construct minimal models of the knowledge base. In case of classification algorithms such a model is constructed for the TBox and in case of ABox realization for the whole knowledge base. We describe the completion algorithm for ABox realization in $\mathcal{EL}$, originally described in [2], which can be easily restricted to obtain algorithms for classification. While the former is the basis for computing the role-depth bounded msc, the latter is used to obtain the role-depth bounded lcs.

For an $\mathcal{EL}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ we want to test whether $\mathcal{K} \models D(a)$ holds. The completion algorithm first adds to $\mathcal{K}$ a concept name for the complex concept description D used in the instance check, i.e., $\mathcal{K} = (\mathcal{T} \cup \{A_q \equiv D\}, \mathcal{A})$, where A_q is a fresh concept name in $\mathcal{K}$. The instance checking algorithm for $\mathcal{EL}$ normalizes the knowledge base in two steps: first the ABox is transformed into a simple ABox. An ABox is a *simple ABox*, if it only contains concept names in concept assertions. An $\mathcal{EL}$-ABox $\mathcal{A}$ can be transformed into a simple ABox by first replacing each complex assertion $C(A)$ in $\mathcal{A}$ by $A(a)$ with a fresh name A and, second, introduce $A \equiv C$ in the TBox.

To describe the second normalization step, we need some notation. Let X be a concept description, a TBox, an ABox or a knowledge base. $\mathsf{CN}(X)$ denotes the set

$$\textbf{NF1}\ C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\ \hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E\ \}$$
$$\textbf{NF2}\quad \exists r.\hat{C} \sqsubseteq D \longrightarrow \{\ \hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D\ \}$$
$$\textbf{NF3}\quad \hat{C} \sqsubseteq \hat{D} \longrightarrow \{\ \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D}\ \}$$
$$\textbf{NF4}\quad B \sqsubseteq \exists r.\hat{C} \longrightarrow \{\ B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C}\ \}$$
$$\textbf{NF5}\ B \sqsubseteq C \sqcap D \longrightarrow \{\ B \sqsubseteq C, B \sqsubseteq D\ \}$$
where $\hat{C}, \hat{D} \notin \mathsf{CN}(\mathcal{T}) \cup \{\top\}$ and A is a new concept name.

Fig. 1. $\mathcal{EL}$ normalization rules

of all concept names and $\mathsf{RN}(X)$ denotes the set of all role names that appear in X. The *signature of* X (denoted $\mathsf{sig}(X)$) is then $\mathsf{CN}(X) \cup \mathsf{RN}(X)$. Now, an $\mathcal{EL}$-TBox $\mathcal{T}$ is in *normal form* if all concept axioms have one of the following forms, where $C_1, C_2 \in \mathsf{sig}(\mathcal{T})$ and $D \in \mathsf{sig}(\mathcal{T}) \cup \{\bot\}$:

$$C_1 \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C_1 \sqsubseteq \exists r.C_2 \quad \text{or} \quad \exists r.C_1 \sqsubseteq D.$$

Any $\mathcal{EL}$-TBox can be transformed into normal form by introducing new concept names and by simply applying the normalization rules displayed in Figure 1 exhaustively. These rules replace the GCI on the left-hand side of the rules with the set of GCIs on the right-hand side.

Clearly, for a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ the signature of $\mathcal{A}$ may be changed only during the first of the two normalization steps and the signature of $\mathcal{T}$ may be extended during both of the normalization steps. The normalization of the KB can be done in linear time.

The completion algorithm for instance checking is based on the one for classifying $\mathcal{EL}$-TBoxes introduced in [2]. Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a normalized $\mathcal{EL}$-KB, i.e., with a simple ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ in normal form. The completion algorithm works on four kinds of *completion sets*: $S(a), S(a,r), S(C)$ and $S(C,r)$ for each $a \in \mathsf{IN}(\mathcal{A})$, each $C \in \mathsf{CN}(\mathcal{K})$, and each $r \in \mathsf{RN}(\mathcal{K})$. The sets of the kind $S(a)$ and $S(a,r)$ contain individuals and concept names. The completion algorithm for classification uses only the latter two kinds of completion sets: $S(C)$ and $S(C,r)$, which contain only concept names from $\mathsf{CN}(\mathcal{K})$. Intuitively, the completion rules make implicit subsumption and instance relationships explicit in the following sense:

- $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{T}} D$,
- $D \in S(C,r)$ implies that $C \sqsubseteq_{\mathcal{T}} \exists r.D$.
- $D \in S(a)$ implies that a is an instance of D w.r.t. $\mathcal{K}$,
- $D \in S(a,r)$ implies that a is an instance of $\exists r.D$ w.r.t. $\mathcal{K}$.

$\mathsf{S}_{\mathcal{K}}$ denotes the set of all completion sets of a normalized $\mathcal{K}$. The completion sets are initialized for each $C \in \mathsf{CN}(\mathcal{K})$, each $r \in \mathsf{RN}(\mathcal{K})$, and each $a \in \mathsf{IN}(\mathcal{A})$ as follows:

- $S(C) := \{C, \top\}$
- $S(C,r) := \emptyset$
- $S(a) := \{C \in \mathsf{CN}(\mathcal{A}) \mid C(a) \text{ appears in } \mathcal{A}\} \cup \{\top\}$
- $S(a,r) := \{b \in \mathsf{IN}(\mathcal{A}) \mid r(a,b) \text{ appears in } \mathcal{A}\}$.

CR1 If $C \in S(X)$, $C \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR2 If $C_1, C_2 \in S(X)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR3 If $C \in S(X)$, $C \sqsubseteq \exists r.D \in \mathcal{T}$, and $D \notin S(X, r)$
then $S(X, r) := S(X, r) \cup \{D\}$

CR4 If $Y \in S(X, r)$, $C \in S(Y)$, $\exists r.C \sqsubseteq D \in \mathcal{T}$, and
$D \notin S(X)$ then $S(X) := S(X) \cup \{D\}$

Fig. 2. $\mathcal{EL}$ completion rules

Then these sets are extended by applying the completion rules shown in Figure 2 until no more rule applies. In these rules C, C_1, C_2 and D are concept names and r is a role name, while X and Y can refer to concept or individual names in the algorithm for instance checking. In the algorithm for classification, X and Y refer to concept names. After the completion has terminated, the following relations hold between an individual a, a role r and named concepts A and B:

- subsumption relation between A and B from $\mathcal{K}$ holds iff $B \in S(A)$
- instance relation between a and B from $\mathcal{K}$ holds iff $B \in S(a)$,

which has been shown in [2].

To decide the initial query: $\mathcal{K} \models D(a)$, one has to test now, whether A_q appears in $S(a)$. In fact, instance queries for all individuals and all named concepts from the KB can be answered now; the completion algorithm does not only perform one instance check, but complete ABox realization. The completion algorithm for $\mathcal{EL}$ runs in polynomial time in size of the knowledge base.

4 Computing Role-depth Bounded Generalizations

We employ the completion method now to compute first the role-depth bounded lcs and then the role-depth bounded msc in $\mathcal{EL}$.

4.1 Computing the Role-depth Bounded LCS

As mentioned in the introduction, the lcs does not need to exist for cyclic TBoxes. Consider the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A \sqcap C,\ B \sqsubseteq \exists r.B \sqcap C\}$. The lcs of A and B is then

$$C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \cdots$$

and cannot be expressed by a finite concept description. To avoid such infinite nestings, we limit the role-depth of the concept description to be computed. The *role-depth* of a concept description C (denoted $rd(C)$) is the maximal number of nested quantifiers of C. Now we can define the lcs with limited role-depth.

Definition 3 (Role-depth bounded $\mathcal{L}$-lcs). *Let $\mathcal{T}$ be an $\mathcal{L}$-TBox and $C_1, \ldots, C_n$ $\mathcal{L}$-concept descriptions and $k \in \mathbb{N}$. Then the $\mathcal{L}$-concept description C is the* role-depth bounded $\mathcal{L}$-least common subsumer *of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ and role-depth k (written k-$lcs(C_1, \ldots, C_n)$) iff*

1. $rd(C) \leq k$,
2. $C_i \sqsubseteq_{\mathcal{T}} C$ *for all* $1 \leq i \leq n$, *and*
3. *for each $\mathcal{L}$-concept descriptions D with $rd(D) \leq k$ it holds that, $C_i \sqsubseteq_{\mathcal{T}} D$ for all $1 \leq i \leq n$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

The computation algorithm for the role-depth bounded lcs w.r.t. general $\mathcal{EL}$-TBoxes, constructs the concept description from the set of completion sets. More precisely, it combines and intersects the completion sets in the same fashion as in the cross-product computation in the lcs algorithm for $\mathcal{EL}$-concept descriptions (without TBoxes) from [4]. The method we present here to compute the role-depth bounded lcs was described in [17].

However, the completion sets may contain concept names that were introduced during normalization. The returned lcs-concept description should only contain concept names that appear in the initial TBox, thus we need to "de-normalize" the concept descriptions obtained from the completion sets. However, the extension of the signature by normalization according to the normalization rules from Figure 1 does not affect subsumption tests for $\mathcal{EL}$-concept descriptions formulated w.r.t. the initial signature of $\mathcal{T}$. The following Lemma has been shown in [17].

Lemma 1. *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $\mathcal{T}'$ the TBox obtained from $\mathcal{T}$ by applying the $\mathcal{EL}$ normalization rules, C, D be $\mathcal{EL}$-concept descriptions with $\mathsf{sig}(C) \subseteq \mathsf{sig}(\mathcal{T})$ and $\mathsf{sig}(D) \subseteq \mathsf{sig}(\mathcal{T}')$ and D' be the concept description obtained by replacing all names $A \in \mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$ from D with $\top$. Then $C \sqsubseteq_{\mathcal{T}'} D$ iff $C \sqsubseteq_{\mathcal{T}} D'$.*

Lemma 1 guarantees that subsumption relations w.r.t. the normalized TBox $\mathcal{T}'$ between C and D, also hold w.r.t. the original TBox $\mathcal{T}$ for C and D', which is basically obtained from D by removing the names introduced by normalization, i.e., concept names from $\mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$.

We assume that the role-depth of each input concept of the lcs has a role-depth less or equal to k. This assumption is motivated by the applications of the lcs on the one hand and on the other by the simplicity of presentation, rather than a technical necessity. The algorithm for computing the role-depth bounded lcs of two $\mathcal{EL}$-concept descriptions is depicted in Algorithm 1.

The procedure k-lcs first adds concept definitions for the input concept descriptions to (a copy of) the TBox and transforms this TBox into the normalized TBox $\mathcal{T}'$. Next, it calls the procedure apply-completion-rules, which applies the $\mathcal{EL}$ completion rules exhaustively to the TBox $\mathcal{T}'$, and stores the obtained set of completion sets in S. Then it calls the function k-lcs-r with the concept names A and B for the input concepts, the set of completion sets S, and the role-depth limit k. The result is then de-normalized and returned (lines 4 to 6). More precisely, in case a complex concept description is returned from k-lcs-r, the procedure remove-normalization-names removes concept names that were added during the normalization of the TBox.

Algorithm 1 Computation of a role-depth bounded $\mathcal{EL}$-lcs.

Procedure k-lcs $(C, D, \mathcal{T}, k)$
Input: C, D: $\mathcal{EL}$-concept descriptions; $\mathcal{T}$: $\mathcal{EL}$-TBox; k: natural number
Output: $k\text{-}lcs(C, D)$: role-depth bounded $\mathcal{EL}$-lcs of C and D w.r.t $\mathcal{T}$ and k.

1: $\mathcal{T}' :=$ normalize($\mathcal{T} \cup \{A \equiv C, B \equiv D\}$)
2: $\mathsf{S}_{\mathcal{T}'}$:= apply-completion-rules($\mathcal{T}'$)
3: L := k-lcs-r $(A, B, \mathsf{S}_{\mathcal{T}'}, k)$
4: **if** $L = A$ **then return** C
5: **else if** $L = B$ **then return** D
6: **else return** remove-normalization-names(L)
7: **end if**

Procedure k-lcs-r (A, B, S, k)
Input: A, B: concept names; S: set of completion sets; k: natural number
Output: $k\text{-}lcs(A, B)$: role-depth bounded $\mathcal{EL}$-lcs of A and B w.r.t $\mathcal{T}$ and k.

1: **if** $B \in S(A)$ **then return** B
2: **else if** $A \in S(B)$ **then return** A
3: **end if**
4: common-names := $S(A) \cap S(B)$
5: **if** $k = 0$ **then return** $\prod_{P \in \text{common-names}} P$
6: **else return** $\prod_{P \in \text{common-names}} P \sqcap \prod_{r \in \mathsf{RN}(\mathcal{T})} \Big(\prod_{(E,F) \in S(A,r) \times S(B,r)} \exists r.\ \text{k-lcs-r}\ (E, F, \mathsf{S}, k-1) \Big)$
7: **end if**

The function k-lcs-r gets a pair of concept names, a set of completion sets and a natural number as inputs. First, it tests whether one of the input concepts subsumes the other w.r.t. $\mathcal{T}'$. In that case the name of the subsuming concept is returned. Otherwise the set of concept names that appear in the completion sets of both input concepts is stored in common-names (line 4).[2] In case the role-depth bound is reached ($k = 0$), the conjunction of the elements in common-names is returned. Otherwise, the elements in common-names are conjoined with a conjunction over all roles $r \in \mathsf{RN}(\mathcal{T})$, where for each r and each element of the cross-product over the r-successors of the current A and B a recursive call to k-lcs-r is made with the role-depth bound reduced by 1 (line 6). This conjunction is then returned to k-lcs.

For $L = \text{k-lcs}(C, D, \mathcal{T}, k)$ it holds by construction that $rd(L) \leq k$.[3] We now show that the result of the function k-lcs is a common subsumer of the input concept descriptions. It was shown in [17] that all conditions of Definition 3 are fulfilled for k-lcs$(C, D, \mathcal{T}, k)$.

Theorem 1. *Let C and D be $\mathcal{EL}$-concept descriptions, $\mathcal{T}$ an $\mathcal{EL}$-TBox, $k \in \mathbf{N}$, then* k-lcs$(C, D, \mathcal{T}, k) \equiv k\text{-}lcs(C, D)$.

[2] Note, that the intersection $S(A) \cap S(B)$ is never empty, since both sets contain $\top$.
[3] Recall our assumption: the role-depth of each input concept is less or equal to k.

For cases where k-lcs returns a concept description with role-depth of less than k we conjecture that it is the exact lcs.

The complexity of the overall method is exponential. However, if a compact representation of the lcs with structure sharing is used, the lcs-concept descriptions can be represented polynomially.

If a k-lcs is too general and a bigger role depth of the k-lcs is desired, the completion of the TBox does not have to be redone for a second computation. The completion sets can simply be "traversed" further.

4.2 Computing the Role-depth Bounded MSC

The msc was first investigated for $\mathcal{EL}$-concept descriptions and w.r.t. unfoldable TBoxes and possibly cyclic ABoxes in [12]. Similar to the lcs, the msc does not need to exist, since cyclic structures cannot be expressed by $\mathcal{EL}$-concept descriptions. Now we can define the msc with limited role-depth.

Definition 4 (role-depth bounded $\mathcal{L}$-msc). *Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then the $\mathcal{L}$-concept description C is the* role-depth bounded $\mathcal{EL}$-most specific concept *of a w.r.t. $\mathcal{K}$ and role-depth k (written k-$msc_{\mathcal{K}}(a)$) iff*

1. $rd(C) \leq k$,
2. $\mathcal{K} \models C(a)$, *and*
3. *for each $\mathcal{EL}$-concept description D with $rd(D) \leq k$ holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

In case the exact msc has a role-depth less than k the role-depth bounded msc is the exact msc.

Again, we construct the msc by traversing the completion sets to "collect" the msc. More precisely, the set of completion sets encodes a graph structure, where the sets $S(X)$ are the nodes and the sets $S(X, r)$ encode the edges. Traversing this graph structure, one can construct an $\mathcal{EL}$-concept. To obtain a finite concept in the presence of cyclic ABoxes or TBoxes one has to limit the role-depth of the concept to be obtained.

Definition 5 (traversal concept). *Let $\mathcal{K}$ be an $\mathcal{EL}$-KB, $\mathcal{K}''$ be its normalized form, $\mathsf{S}_{\mathcal{K}}$ the completion set obtained from $\mathcal{K}$ and $k \in \mathbb{N}$. Then the* traversal concept of a named concept A *(denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(A)$) with $\mathsf{sig}(A) \subseteq \mathsf{sig}(\mathcal{K}'')$ is the concept obtained from executing the procedure call* traversal-concept-c*(A, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm 2.*

The traversal concept of an individual a *(denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(a)$) with $a \subseteq \mathsf{sig}(\mathcal{K})$ is the concept description obtained from executing the procedure call* traversal-concept-i*(a, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm 2.*

The idea is that the traversal concept of an individual yields its msc. However, the traversal concept contains names that were introduced during normalization. The returned msc should be formulated w.r.t. the signature of the original KB, thus the normalization names need to be removed or replaced.

Algorithm 2 Computation of a role-depth bounded $\mathcal{EL}$-msc.

Procedure k-msc $(a, \mathcal{K}, k)$
Input: a: individual from $\mathcal{K}$; $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) an $\mathcal{EL}$-KB; $k \in \mathbb{N}$
Output: role-depth bounded $\mathcal{EL}$-msc of a w.r.t. $\mathcal{K}$ and k.

1: $(\mathcal{T}', \mathcal{A}')$:= simplify-ABox($\mathcal{T}$, $\mathcal{A}$)
2: $\mathcal{K}''$:= (normalize($\mathcal{T}'$), $\mathcal{A}'$)
3: $\mathsf{S}_\mathcal{K}$:= apply-completion-rules($\mathcal{K}$)
4: **return** Remove-normalization-names (traversal-concept-i$(a, \mathsf{S}_\mathcal{K}, k)$)

Procedure traversal-concept-i (a, S, k)
Input: a: individual name from $\mathcal{K}$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth traversal concept (w.r.t. $\mathcal{K}$) and k.

1: **if** $k = 0$ **then return** $\bigsqcap_{A \in S(a)} A$
2: **else return** $\bigsqcap_{A \in \mathsf{S}(a)} A \sqcap$
$$\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{A \in \mathsf{CN}(\mathcal{K}'') \cap S(a,r)} \exists r.\ \text{traversal-concept-c}\ (A, \mathsf{S}, k-1) \sqcap$$
$$\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{b \in \mathsf{IN}(\mathcal{K}'') \cap S(a,r)} \exists r.\ \text{traversal-concept-i}\ (b, \mathsf{S}, k-1)$$
3: **end if**

Procedure traversal-concept-c (A, S, k)
Input: A: concept name from $\mathcal{K}''$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth bounded traversal concept.

1: **if** $k = 0$ **then return** $\bigsqcap_{B \in S(A)} B$
2: **else return** $\bigsqcap_{B \in S(A)} B \sqcap \bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{B \in S(A,r)} \exists r.\text{traversal-concept-c}\ (B, \mathsf{S}, k-1)$
3: **end if**

Lemma 2. *Let $\mathcal{K}$ be an $\mathcal{EL}$-KB, $\mathcal{K}''$ its normalized version, $\mathsf{S}_\mathcal{K}$ be the set of completion sets obtained for $\mathcal{K}$, $k \in \mathbb{N}$ a natural number and $a \in \mathsf{IN}(\mathcal{K})$. Furthermore let $C = k$-$\mathsf{C}_{\mathsf{S}_\mathcal{K}}(a)$ and $\widehat{C}$ be obtained from C by removing the normalization names. Then*

$$\mathcal{K}'' \models C(a) \text{ iff } \mathcal{K} \models \widehat{C}(a).$$

This lemma guarantees that removing the normalization names from the traversal concept preserves the instance relationships. Intuitively, this lemma holds since the construction of the traversal concept conjoins exhaustively all named subsumers and all subsuming existential restrictions to a normalization name up to the role-depth bound. Thus removing the normalization name does not change the extension of the conjunction. The proof can be found in [18]. We are now ready to devise a computation algorithm for the role-depth bounded msc: procedure k-msc as displayed in Algorithm 2.

The procedure k-msc has an individual a from a knowledge base $\mathcal{K}$, the knowledge base $\mathcal{K}$ itself and number k for the role depth-bound as parameter. It first performs the two normalization steps on $\mathcal{K}$, then applies the completion rules from Figure 2 to the normalized KB $\mathcal{K}''$ and stores the set of completion sets in $\mathsf{S}_\mathcal{K}$. Afterwards it computes the traversal-concept of a from $\mathsf{S}_\mathcal{K}$ w.r.t. role-depth bound k. In a post-processing step it applies Remove-normalization-names to the traversal concept.

Obviously, the concept description returned from the procedure k-msc has a role-depth less or equal to k. The other conditions of Definition 4 are fulfilled as well, which has been shown in [18] yielding the correctness of the overall procedure.

Theorem 2. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then* k-msc$(a, \mathcal{K}, k) \equiv k\text{-}msc_{\mathcal{K}}(a)$.

The k-msc can grow exponential in the size of the knowledge base.

5 Implementation of GEL

The completion algorithm for classifying $\mathcal{EL}$ TBoxes was first implemented in the CEL reasoner [5]. We used its successor system JCEL [16] as a starting point for our implementation for the computation of the role-depth bounded lcs and msc. The implementation was done in Java and provides a simple GUI for the ontology editor PROTÉGÉ as can be seen in the screen-shot in Figure 3.

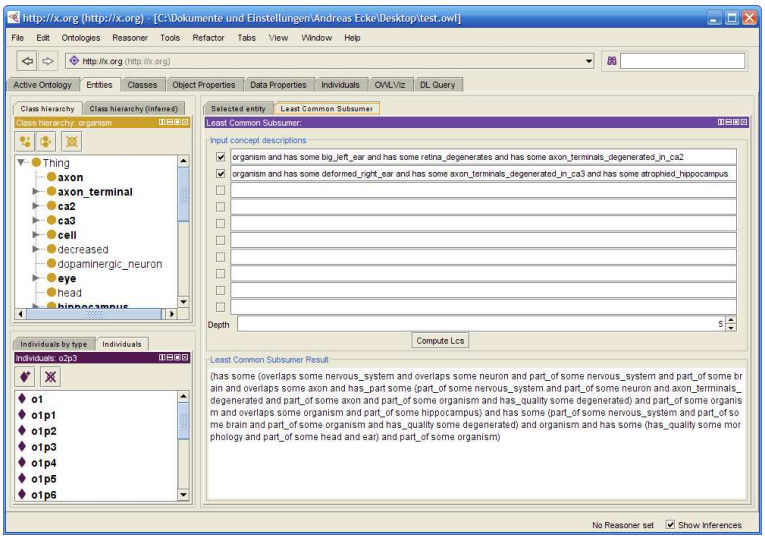

Fig. 3. LCS plugin

Our implementation of the methods presented here accesses the internal data structures of JCEL directly, providing a full integration of GEL into JCEL. The reasoning methods in GEL are in this first version realized in a naive way and are still in need of optimizations in order to handle the large knowledge bases that can be encountered in practice.

The concept descriptions returned by the lcs and the msc can grow exponentially in the worst case. On top of that, the returned concept descriptions are quite redundant in our current implementation, which might be acceptable if used as an input for a

similarity measure, but surely not if presented to a human reader. It is future work to investigate methods for minimal rewritings of concept descriptions w.r.t. a general $\mathcal{EL}$ knowledge base in order to be able to present redundancy-free concept descriptions. Our tool will be made available as a plug-in for the ontology editor PROTÉGÉ and an API for the k-limited lcs and -msc is planned. The former system sonic [24] implemented the lcs and msc as well, but allowed only for acyclic, unfoldable TBoxes.

6 Conclusions

In this paper we have presented a practical method for computing the role-depth bounded lcs and the role-depth bounded msc of $\mathcal{EL}$-concepts w.r.t. a general TBox. We have argued that such generalization inferences are useful for ontology-based applications in many ways. Our approach for computing (approximations of) these inferences is based on the completion sets that are computed during classification of a TBox or realization of an ABox. Thus, any of the available implementations of the $\mathcal{EL}$ completion algorithm can be easily extended to an implementation of the two generalization inferences considered here. The same idea can be adapted for the computation of generalizations in the probabilistic DL Prob-$\mathcal{EL}_c^{01}$ [17, 18].

These theoretical results complete the (approximative) bottom-up approach for general $\mathcal{EL}$- (and Prob-$\mathcal{EL}_c^{01}$-) KBs. Continuing on the theoretical side, we want to investigate the bottom-up constructions (i.e. lcs and msc computations) in more expressive members of the $\mathcal{EL}$-family. We want to extend the approximative methods to $\mathcal{EL}$++, which extends $\mathcal{EL}$, for example, by transitive roles and role hierarchies. Such an extension would enable generalization reasoning services for the OWL 2 EL profile. Another interesting extension is to allow for more expressive means for probabilities.

Although a non-redundant representation of the concept descriptions obtained by the approximative lcs and msc is desirable when presented to a human reader, it is not clear whether a minimal representation of the obtained concept descriptions is favorable in every case. It might depend on the similarity measures employed whether a redundant representation of a concept is preferable over a compact one.

On the practical side, our future work will include evaluations of the usefulness of the offered reasoning services for biomedical applications and the development and testing of optimizations regarding the performance of the implementation.

Acknowledgments: We would like to thank Andreas Ecke and Julian Mendez for their implementation effort.

References

1. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 325–330. Morgan Kaufmann, 2003.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.

3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
4. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann, Los Altos.
5. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, volume 4130 of *Lecture Notes In Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006. CEL download page: `http://lat.inf.tu-dresden.de/systems/cel/`.
6. A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*, volume 147 of *CEUR Workshop Proceedings*, 2005.
7. W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In W. Swartout, editor, *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI-92)*, pages 754–760, San Jose, CA, 1992. AAAI Press/The MIT Press.
8. S. Colucci, E. Di Sciascio, F. M. Donini, and E. Tinelli. Partial and informative common subsumers in description logics. In *proc. of 18th European Conference on Artificial Intelligence (ECAI 2008)*. IOS Press, 2008.
9. T. G. O. Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
10. C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In *Pro.c of Convegno Italiano di Logica Computazionale, CILC05*, 2005.
11. M. C. Keet, M. Roos, and M. S. Marshall. A survey of requirements for automated reasoning services for bio-ontologies in OWL. In *Proceedings of Third international Workshop OWL: Experiences and Directions (OWLED 2007)*, 2007.
12. R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. *AI Communications*, 15(1):47–59, 2002.
13. P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble. Investigating semantic similarity measures across the gene ontology: The relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–1283, 2003.
14. C. Lutz, R. Piro, and F. Wolter. Enriching el-concepts with greatest fixpoints. In *Proc. of the 19th European Conf. on Artificial Intelligence (ECAI-10)*. IOS Press, 2010.
15. T. Mantay and R. Möller. Content-based information retrieval by computation of least common subsumers in a probabilistic description logic. In *Proc. International Workshop on Intelligent Information Integration, ECAI'98, Aug. 23-28, Brighton UK, 1998,*, 1998.
16. J. Mendez and B. Suntisrivaraporn. Reintroducing CEL as an OWL 2 EL reasoner. In B. Cuenca Grau, I. Horrocks, B. Motik, and U. Sattler, editors, *Proc. of the 2008 Description Logic Workshop (DL 2009)*, volume 477 of *CEUR-WS*, 2009.
17. R. Peñaloza and A.-Y. Turhan. Role-depth bounded least common subsumers by completion for $\mathcal{EL}$- and Prob-$\mathcal{EL}$-TBoxes. In V. Haarslev, D. Toman, and G. Weddell, editors, *Proc. of the 2010 Description Logic Workshop (DL'10)*, 2010.
18. R. Peñaloza and A.-Y. Turhan. Towards approximative most specific concepts by completion for $\mathcal{EL}^{01}$ with subjective probabilities. In T. Lukasiewicz, R. Peñaloza, and A.-Y. Turhan, editors, *Proceedings of the First International Workshop on Uncertainty in Description Logics (UniDL'10)*, 2010.
19. C. Pesquita, D. Faria, A. O. Falco, P. Lord, and F. M. Couto. Semantic similarity in biomedical ontologies. *PLoS Comput Biol*, 5, 2009.

20. P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI-95)*, pages 448–453, 1995.
21. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. *Journal of the American Medical Informatics Assoc.*, 2000. Fall Symposium Special Issue.
22. T. Springer and A.-Y. Turhan. Employing description logics in ambient intelligence for modeling and reasoning about complex situations. *Journal of Ambient Intelligence and Smart Environments*, 1(3):235–259, 2009.
23. A.-Y. Turhan. *On the Computation of Common Subsumers in Description Logics*. PhD thesis, TU Dresden, Institute for Theoretical Computer Science, 2007.
24. A.-Y. Turhan and C. Kissig. SONIC — Non-standard inferences go OILED. In D. Basin and M. Rusinowitch, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-04)*, volume 3097 of *Lecture Notes in Computer Science*, pages 321–325. Springer, 2004. SONIC is available from `http://wwwtcs.inf.tu-dresden.de/~sonic/`.
25. W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation, 27th October 2009. `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`.

Computing Role-depth Bounded Generalizations in the Description Logic $\mathcal{ELOR}$

Andreas Ecke[1⋆], Rafael Peñaloza[1,2⋆⋆], and Anni-Yasmin Turhan[1⋆⋆⋆]

[1] Institute for Theoretical Computer Science,
Technische Universität Dresden
[2] Center for Advancing Electronics Dresden
{ecke,penaloza,turhan}@tcs.inf.tu-dresden.de

Abstract. Description Logics (DLs) are a family of knowledge representation formalisms, that provides the theoretical basis for the standard web ontology language OWL. Generalization services like the least common subsumer (lcs) and the most specific concept (msc) are the basis of several ontology design methods, and form the core of similarity measures. For the DL $\mathcal{ELOR}$, which covers most of the OWL 2 EL profile, the lcs and msc need not exist in general, but they always exist if restricted to a given role-depth. We present algorithms that compute these role-depth bounded generalizations. Our method is easy to implement, as it is based on the polynomial-time completion algorithm for $\mathcal{ELOR}$.

1 Introduction

Description logics (DLs) are knowledge representation formalisms with formal and well-understood semantics [4]. They supply the foundation for the web ontology language OWL 2 standardized by the W3C [20]. Since then, DLs became more widely used for the representation of knowledge from several domains.

Each DL offers a set of concept constructors by which complex concepts can be built. These concepts describe categories from the application domain at hand. A DL knowledge base consists of two parts: the TBox captures the terminological knowledge about categories and relations, and the ABox captures the assertional knowledge, i.e., individual facts, from the application domain. Prominent inferences are *subsumption*, which determines subconcept relationships and *instance checking*, which tests for a given individual and concept whether the individual belongs to the concept.

The lightweight DL $\mathcal{EL}$ offers limited expressivity but allows for polynomial time reasoning [3]. These good computational properties are maintained by several extensions of $\mathcal{EL}$—most prominently by $\mathcal{EL}^{++}$, the DL underlying the OWL 2 EL profile [15], which allows for the use of nominals, i.e., singleton concepts, when building complex concept descriptions. The reasoning algorithms

⋆ Supported by DFG Graduiertenkolleg 1763 (QuantLA).
⋆⋆ Partially supported by DFG within the Cluster of Excellence 'cfAED'
⋆⋆⋆ Partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing".

for (fragments of) $\mathcal{EL}^{++}$ have been implemented in highly optimized reasoner systems such as jCEL [14] and ELK [10]. It is worth pointing out that the initial reasoning algorithm for handling nominals in $\mathcal{EL}$ [3] turned out to be incomplete, but a complete method has been recently devised in [11].

In this paper we describe methods for computing generalizations in the $\mathcal{EL}$-family with the help of the standard reasoning algorithms. We consider the following two inferences: The *least common subsumer* (lcs), which computes for a given set of concepts a new concept that subsumes the input concepts and is the least one w.r.t. subsumption; and the *most specific concept* which provides a concept which has a given individual as an instance and is the least one w.r.t. subsumption. Both inferences have been employed for several applications. Most prominently the lcs and the msc can be employed in the 'bottom-up approach' for generating TBoxes, where modellers can generate a new concept from picking ABox individuals that instantiate the desired concept and then generalizing this set into a single concept automatically–first by applying the msc to each of the individuals and then generalizing the obtained concepts by applying the lcs [5]. Other applications of the lcs and the msc include similarity measures [8, 6, 13], which are the core of ontology matching algorithms and more (see [7, 16]). In particular for large bio-medical ontologies the lcs can be used effectively to support construction and maintenance. Many of these bio-medical ontologies, notably SNOMED CT [19] and the FMA Ontology [18], are written in the $\mathcal{EL}$-family of lightweight DLs.

It is known that for concepts captured in a general TBox or even just a cyclic TBox, the lcs does not need to exist [1], since cycles cannot be captured in an $\mathcal{EL}$-concept. Therefore, an approximation has been introduced in [16], that limits the maximal nesting of quantifiers of the resulting concept descriptions. These so-called role-depth bounded lcs (k-lcs), can be computed for $\mathcal{EL}$ and for $\mathcal{EL}$ extended by role inclusions using completion sets produced by the subsumption algorithm [16, 9]. In this paper, we describe a subsumption algorithm for the DL $\mathcal{ELOR}$—building on the one for $\mathcal{ELO}$ ($\mathcal{EL}$ extended by nominals) from [11]. Our algorithm is given in terms of the completion algorithm in order to extend the methods for the k-lcs to $\mathcal{ELOR}$.

Recently, necessary and sufficient conditions for the existence of the lcs w.r.t. general $\mathcal{EL}$-TBoxes have been devised [21]. By the use of these conditions the bound k for which the role-depth bounded lcs and the lcs coincide can be determined, if the lcs exists; i.e., if such k is finite.

Similarly to the lcs, the msc does not need to exist, if the ABox [12] contain cycles. To obtain an approximative solution, the role-depth of the resulting concept can be limited as suggested in [12]. A computation algorithm for the role-depth bounded msc has been proposed in [17] for $\mathcal{EL}$. If nominals are allowed, the computation of the msc is trivial, since the msc of an individual a is simply the nominal that contains a (i.e., $\{a\}$). Thus, we consider the computation of the role-depth bounded msc in $\mathcal{EL}$ w.r.t. an $\mathcal{ELOR}$ knowledge base.

We introduce the basic notions of DL and the reasoning services considered in the next section. In Section 3 we give a completion-based classification algo-

	Syntax	Semantics
concept name	$A \quad (A \in N_C)$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
top concept	$\top$	$\Delta^{\mathcal{I}}$
nominal	$\{a\} \quad (a \in N_I)$	$\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C \quad (r \in N_R)$	$(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RIA	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
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. Concept constructors and TBox axioms for $\mathcal{ELOR}$.

rithm for $\mathcal{ELOR}$, which serves as a basis for the computation algorithms of the role-depth bounded lcs and msc presented subsequently. The paper ends with conclusions and future work.[3]

2 Preliminaries

$\mathcal{ELOR}$-concepts are built from mutually disjoint sets N_C of *concept names*, N_R of *role names* and N_I of *individual names* using the syntax rule:

$$C, D ::= \top \mid A \mid \{a\} \mid C \sqcap D \mid \exists r.C,$$

where $A \in N_C$, $r \in N_R$ and $a \in N_I$. The individuals appearing in concepts are also called *nominals*. The sub-logic of $\mathcal{ELOR}$ that does not allow for individuals in concepts is called $\mathcal{ELR}$.

As usual, the semantics of $\mathcal{ELOR}$-concepts is defined through interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of an *interpretation domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that maps concept names A to subsets $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and role names to binary relations on the domain $\Delta^{\mathcal{I}}$. This function is extended to complex concepts as shown in the upper part of Table 1.

Concepts can be used to model notions from the application domain in the TBox. Given two concepts C and D, a *general concept inclusion axiom* (GCI) is of the form $C \sqsubseteq D$. We use $C \equiv D$ as an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. Given the roles $r_1, \ldots, r_n$ and s, a *role inclusion axiom* (RIA) is an expression of the form $r_1 \circ \cdots \circ r_n \sqsubseteq s$. An $\mathcal{ELOR}$-TBox is a finite set of GCIs and RIAs. An interpretation is a *model* for a TBox $\mathcal{T}$ if it satisfies all GCIs and RIAs in $\mathcal{T}$, as shown in the middle part of Table 1. An $\mathcal{EL}$-TBox is an $\mathcal{ELR}$-TBox (i.e., without the nominal constructor) that does not contain any RIAs.

Knowledge about individual facts of the application domain can be captured by assertions. Let $a, b \in N_I$, $r \in N_R$ and C a concept, then $C(a)$ is a *concept*

[3] Because of space constraints, some proofs are deferred to the appendix of long version of this paper at `http://lat.inf.tu-dresden.de/research/papers.html`.

assertion and $r(a,b)$ a *role assertion*. An *ABox* $\mathcal{A}$ is a finite set of (concept or role) assertions. An interpretation is a *model* for an ABox $\mathcal{A}$ if it satisfies all concept and role assertions in $\mathcal{A}$, as shown in the lower part of Table 1.

A *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. An interpretation is a model of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it is a *model* of both $\mathcal{T}$ and $\mathcal{A}$. With $\mathsf{Sig}(\mathcal{T})$ we denote the *signature* of a TBox $\mathcal{T}$, i.e. the set of all concept names, role names, and individual names that appear in $\mathcal{T}$. By $\mathsf{Sig}(\mathcal{A})$ and $\mathsf{Sig}(\mathcal{K})$ we denote the analogous notions for ABoxes and KBs, respectively.

Important reasoning tasks considered for DLs are *subsumption* and *instance checking*. A concept C *is subsumed by* a concept D w.r.t. a TBox $\mathcal{T}$ (denoted $C \sqsubseteq_\mathcal{T} D$) if $C^\mathcal{I} \subseteq D^\mathcal{I}$ holds in all models $\mathcal{I}$ of $\mathcal{T}$. A concept C is *equivalent* to a concept D w.r.t. a TBox $\mathcal{T}$ (denoted $C \equiv_\mathcal{T} D$) if $C \sqsubseteq_\mathcal{T} D$ and $D \sqsubseteq_\mathcal{T} C$ hold. The reasoning service *classification* of a TBox $\mathcal{T}$ computes all subsumption relationships between the named concepts occurring in $\mathcal{T}$. A reasoning service dealing with a whole KB is *instance checking*. An individual a *is an instance of* a given concept C w.r.t. $\mathcal{K}$ (denoted $\mathcal{K} \models C(a)$) if $a^\mathcal{I} \in C^\mathcal{I}$ holds in all models $\mathcal{I}$ of $\mathcal{K}$. *ABox realization* computes, for every concept name in $\mathcal{K}$, the set of individuals from the ABox that belong to that concept. These reasoning problems can all be decided for $\mathcal{ELOR}$, and hence also in $\mathcal{EL}$, in polynomial time [3].

There are two central inferences discussed in this paper that compute generalizations. The first is called the *least common subsumer* (lcs); it computes, for two given concepts, a (possibly complex) concept that subsumes both input concepts and that is the least concept with this property w.r.t. subsumption. The second is called the *most specific concept* (msc), which computes for a given individual a the least concept w.r.t. subsumption that has a as an instance w.r.t. $\mathcal{K}$.

The lcs does not need to exist if computed w.r.t. general $\mathcal{EL}$-TBoxes, i.e., TBoxes that use complex concepts in the left-hand sides of GCIs, or even just cyclic TBoxes [2]. The reason is that the resulting concept cannot capture cycles. Thus, we follow here the idea from [16] and compute only approximations of the lcs and of the msc by limiting the nesting of quantifiers of the resulting concept.

The *role depth* $rd(C)$ of a concept C denotes the maximal nesting depth of the existential quantifier in C. Sometimes it is convenient to write the resulting concept in a different DL than the one the inputs concepts are written in. Thus we distinguish a 'source DL' $\mathcal{L}_s$ and a 'target DL' $\mathcal{L}_t$. With these notions at hand, we can define the first generalization inference.

Definition 1 (lcs, role-depth bounded lcs). *The* least common subsumer *of two $\mathcal{L}_s$-concepts C_1, C_2 w.r.t. an $\mathcal{L}_s$-TBox $\mathcal{T}$ (written: $lcs_\mathcal{T}(C_1, C_2)$) is the $\mathcal{L}_t$-concept description D s.t.:*

1. *$C_1 \sqsubseteq_\mathcal{T} D$ and $C_2 \sqsubseteq_\mathcal{T} D$, and*
2. *for all $\mathcal{L}_t$-concepts E, $C_1 \sqsubseteq_\mathcal{T} E$ and $C_2 \sqsubseteq_\mathcal{T} E$ implies $D \sqsubseteq_\mathcal{T} E$.*

Let $k \in \mathbb{N}$. If the concept D has a role-depth up to k and Condition 2 holds for all such E with role-depth up to k, then D is the role-depth bounded lcs *(k-$lcs_\mathcal{T}(C_1, C_2)$) of C_1 and C_2 w.r.t. $\mathcal{T}$ and k.*

The role-depth bounded lcs is unique up to equivalence, thus we speak of *the k-lcs*. In contrast, common subsumers need not be unique. Note that for target DLs that offer disjunction, the lcs is always trivial: $lcs(C_1, C_2) = C_1 \sqcup C_2$. Thus target DLs without disjunction may yield more informative lcs.

Similarly to the lcs, the msc does not need to exist if computed w.r.t. cyclic ABoxes. Again we compute here approximations of the msc by limiting the role-depth of the resulting concept as suggested in [12].

Definition 2. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB written in $\mathcal{L}_s$ and a be an individual from $\mathcal{A}$. An $\mathcal{L}_t$-concept description C is the* most specific concept of a w.r.t. $\mathcal{K}$ *(written $msc_{\mathcal{K}}(a)$) if it satisfies:*

1. *$\mathcal{K} \models C(a)$, and*
2. *for all $\mathcal{L}_t$-concepts D, $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

If the concept C has a role-depth up to k and Condition 2 holds for all such D with role-depth up to k, then C is the role depth bounded msc *of a w.r.t. $\mathcal{K}$ and k (k-$msc_{\mathcal{K}}(a)$).*

The msc and the *k-msc* are unique up to equivalence in $\mathcal{EL}$ and $\mathcal{ELOR}$. In $\mathcal{ELOR}$ the msc is trivial, since $\mathrm{msc}_{\mathcal{K}}(a) = \{a\}$. Thus we consider in this paper a more interesting case, where the target DL $\mathcal{L}_t$ for the resulting concept is a less expressive one without nominals, namely $\mathcal{EL}$ or $\mathcal{ELR}$.

3 The *k*-lcs in $\mathcal{ELOR}$

The algorithms to compute the role-depth bounded lcs are based on completion-based classification algorithms for the corresponding DL. For the DL $\mathcal{ELOR}$, a consequence-based algorithm for classification of TBoxes was presented in [11], building upon the completion algorithm developed in [3]. The completion algorithm presented next adapts the ideas of the complete algorithm.

3.1 Completion Algorithm for $\mathcal{ELOR}$-TBoxes

The completion algorithms work on normalized TBoxes. We define for $\mathcal{ELOR}$ the set of *basic concepts* for a TBox $\mathcal{T}$:

$$\mathsf{BC}_{\mathcal{T}} = (\mathsf{Sig}(\mathcal{T}) \cap (N_C \cup N_I)) \cup \{\top\}.$$

Let $\mathcal{T}$ be an $\mathcal{ELOR}$-TBox and $A, A_1, A_2, B \in \mathsf{BC}_{\mathcal{T}}$; then $\mathcal{T}$ is in *normal form* if

- each GCI in $\mathcal{T}$ is of the form: $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$, or $\exists r.A \sqsubseteq B$.
- each RIA in $\mathcal{T}$ is of the form: $r \sqsubseteq s$ or $r_1 \circ r_2 \sqsubseteq s$.

Every $\mathcal{ELOR}$-TBox can be transformed into normal form in linear time by applying a set of normalization rules given in [3]. These normalization rules essentially introduce new named concepts for complex concepts used in GCIs or new roles used in complex RIAs.

Before describing the completion algorithm in detail, we introduce the reachability relation $\rightsquigarrow_R$, which plays a fundamental role in the correct treatment of nominals in TBox classification algorithms [3, 11].

Definition 3 ($\rightsquigarrow_R$). *Let $\mathcal{T}$ be an $\mathcal{ELOR}$-TBox in normal form, $G \in N_C$ a concept name, and $D \in \mathsf{BC}_\mathcal{T}$. $G \rightsquigarrow_R D$ iff there exist roles $r_1, \ldots, r_n \in N_R$ and basic concepts $A_0, \ldots, A_n, B_0, \ldots, B_n \in \mathsf{BC}_\mathcal{T}$, $n \geq 0$ such that $A_i \sqsubseteq_\mathcal{T} B_i$ for all $0 \leq i \leq n$, $B_{i-1} \sqsubseteq \exists r_i.A_i \in \mathcal{T}$ for all $1 \leq i \leq n$, A_0 is either G or a nominal, and $B_n = D$.*

Informally, the concept name D is reachable from G if there is a chain of existential restrictions starting from G or a nominal and ending in D. This implies that, for $G \rightsquigarrow_R D$, if the interpretation of G is not empty, then the interpretation of D cannot be empty either. This in turn causes additional subsumption relationships to hold. Note that, if D is reachable from a nominal, then $G \rightsquigarrow_R D$ holds for all concept names G, since the interpretation of D can never be empty.

The basic idea of completion algorithms in general is to generate canonical models of the TBox. To this end, the elements of the interpretation domain are represented by named concepts or nominals from the normalized TBox. These elements are then related via roles according to the existential restrictions derived for the TBox. More precisely, let $\mathcal{T}$ be a normalized TBox, $G \in \mathsf{Sig}(\mathcal{T}) \cap N_C \cup \{\top\}$ and $A \in \mathsf{BC}_\mathcal{T}$, we introduce a *completion set* $S^G(A)$. We store all basic concepts that subsume a basic concept A in the completion set $S^A(A)$ and all basic concepts B for which $\exists r.B$ subsumes A in the completion set $S^A(A, r)$. These completion sets are then extended using a set of rules. However, the algorithm needs to keep track also of completion sets of the form $S^G(A)$ and $S^G(A, r)$ for every $G \in (\mathsf{Sig}(\mathcal{T}) \cap N_C) \cup \{\top\}$, since the non-emptiness of an interpretation of a concept G may imply additional subsumption relationships for A. The completion set $S^G(A)$ therefore stores all basic concepts that subsume A under the assumption that G is not empty. Similarly $S^G(A, r)$ stores all concepts B for which $\exists r.B$ subsumes A under the same assumption.

For every $G \in (\mathsf{Sig}(\mathcal{T}) \cap N_C) \cup \{\top\}$, every basic concept A and every role name r, the completion sets are initialized as $S^G(A) = \{A, \top\}$ and $S^G(A, r) = \emptyset$. These sets are then extended by applying the completion rules shown in Figure 1 (adapted from [11]) exhaustively.

To compute the reachability relation $\rightsquigarrow_R$ used in rule **OR7**, the algorithm can use Definition 3 with all previously derived subsumption relationships; that is, $A_i \sqsubseteq B_i$ if it finds $B_i \in S^{A_i}(A_i)$. Thus the computation of $\rightsquigarrow_R$ and the application of the completion rules need to be carried out simultaneously.

It can be shown that the algorithm terminates in polynomial time, and is sound and complete for classifying the TBox $\mathcal{T}$. In particular, when no rules are applicable anymore the completion sets have the following properties.

Proposition 1. *Let $\mathcal{T}$ be an $\mathcal{ELOR}$-TBox in normal form, $C, D \in \mathsf{BC}_\mathcal{T}$, $r \in \mathsf{Sig}(\mathcal{T}) \cap N_R$, and $G = C$ if $C \in N_C$ and $G = \top$ otherwise. Then, the following properties hold:*

$C \sqsubseteq_\mathcal{T} D$ iff $D \in S^G(C)$, and
$C \sqsubseteq_\mathcal{T} \exists r.D$ iff there exists $E \in \mathsf{BC}_\mathcal{T}$ such that $E \in S^G(C, r)$ and $D \in S^G(E)$.

We now show how to use these completion sets for computing the role-depth bounded lcs for $\mathcal{ELOR}$-concept w.r.t. a general $\mathcal{ELOR}$-TBox.

OR1 If $A_1 \in S^G(A)$, $A_1 \sqsubseteq B \in \mathcal{T}$ and $B \notin S^G(A)$,
then $S^G(A) := S^G(A) \cup \{B\}$

OR2 If $A_1, A_2 \in S^G(A)$, $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $B \notin S^G(A)$,
then $S^G(A) := S^G(A) \cup \{B\}$

OR3 If $A_1 \in S^G(A)$, $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$ and $B \notin S^G(A, r)$,
then $S^G(A, r) := S^G(A, r) \cup \{B\}$

OR4 If $B \in S^G(A, r)$, $B_1 \in S^G(B)$, $\exists r.B_1 \sqsubseteq C \in \mathcal{T}$ and $C \notin S^G(A)$,
then $S^G(A) := S^G(A) \cup \{C\}$

OR5 If $B \in S^G(A, r)$, $r \sqsubseteq s \in \mathcal{T}$ and $B \notin S^G(A, s)$,
then $S^G(A, s) := S^G(A, s) \cup \{B\}$

OR6 If $B \in S^G(A, r_1), C \in S^G(B, r_2)$, $r_1 \circ r_2 \sqsubseteq s \in \mathcal{T}$ and $C \notin S^G(A, s)$,
then $S^G(A, s) := S^G(A, s) \cup \{C\}$

OR7 If $\{a\} \in S^G(A_1) \cap S^G(A_2)$, $G \rightsquigarrow_R A_2$, and $A_2 \notin S^G(A_1)$,
then $S^G(A_1) := S^G(A_1) \cup \{A_2\}$

Fig. 1. Completion rules for $\mathcal{ELOR}$

3.2 Computing the Role-depth Bounded $\mathcal{ELOR}$-lcs

In order to compute the role-depth bounded lcs of two $\mathcal{ELOR}$-concepts C and D, we extend the methods from [16] for $\mathcal{EL}$-concepts and from [9] for $\mathcal{ELR}$-concepts, where we compute the cross-product of the tree unravelings of the canonical model represented by the completion sets for C and D up to the role-depth k. Clearly, in the presence of nominals, the right completion sets need to be chosen that preserve the non-emptiness of the interpretation of concepts derived by $\rightsquigarrow_R$.

An algorithm that computes the role-depth bounded $\mathcal{ELOR}$-lcs using completion sets is shown in Figure 2. In the first step, the algorithm introduces two new concept names A and B as abbreviations for the (possibly complex) concepts C and D, and the augmented TBox is normalized. The completion sets are then initialized and the completion rules from Figure 1 are applied exhaustively, yielding the saturated completion sets $\mathsf{S}_\mathcal{T}$. In the recursive procedure k-lcs-r for concepts A and B, we first obtain all the basic concepts that subsume both A and B from the sets $S^A(A)$ and $S^B(B)$. For every role name r, the algorithm then recursively computes the $(k-1)$-lcs of the concepts A' and B' in the subsumer sets $S^A(A, r)$ and $S^B(B, r)$, i.e. for which $A \sqsubseteq_\mathcal{T} \exists r.A'$ and $B \sqsubseteq_\mathcal{T} \exists r.B'$. These concepts are added as existential restrictions to the k-lcs.

The algorithm only introduces concept and role names that occur in the original TBox $\mathcal{T}$. Therefore those names introduced by the normalization are not used in the concept for the k-lcs and an extra denormalization step as in [16, 9] is not necessary.

Notice that for every pair (A', B') of r-successors of A and B it holds that $A \rightsquigarrow_R A'$ and $B \rightsquigarrow_R B'$. Intuitively, we are assuming that the interpretation of both A and B is not empty. This in turn causes the interpretation of $\exists r.A'$ and $\exists r.B'$ to be not empty, either. Thus, it suffices to consider the completion

Procedure k-lcs$(C, D, \mathcal{T}, k)$
Input: C, D: $\mathcal{ELOR}$-concepts; $\mathcal{T}$: $\mathcal{ELOR}$-TBox; $k \in \mathbf{N}$
Output: role-depth bounded $\mathcal{ELOR}$-lcs of C, D w.r.t. $\mathcal{T}$ and k

1: $\mathcal{T}' :=$ normalize$(\mathcal{T} \cup \{A \equiv C, B \equiv D\})$
2: $\mathsf{S}_\mathcal{T} :=$ apply-completion-rules$(\mathcal{T}')$
3: **return** k-lcs-r$(A, B, \mathsf{S}_\mathcal{T}, k, A, B, \mathsf{Sig}(\mathcal{T}))$

Procedure k-lcs-r$(X, Y, \mathsf{S}_\mathcal{T}, k, A, B, \mathsf{Sig}(\mathcal{T}))$
Input: A, B: concept names, X, Y: basic concepts with $A \rightsquigarrow_R X, B \rightsquigarrow_R Y$; $k \in \mathbf{N}$;
$\mathsf{S}_\mathcal{T}$: set of saturated completion sets; $\mathsf{Sig}(\mathcal{T})$: signature of $\mathcal{T}$
Output: role-depth bounded $\mathcal{ELOR}$-lcs of X, Y w.r.t. $\mathcal{T}$ and k

1: common-names $:= S^A(X) \cap S^B(Y) \cap \mathsf{BC}_\mathcal{T}$
2: **if** $k = 0$ **then**
3: **return** $\displaystyle\prod_{P \in \text{common-names}} P$
4: **else**
5: **return** $\displaystyle\prod_{P \in \text{common-names}} P \sqcap$

$$\prod_{r \in \mathsf{Sig}(\mathcal{T}) \cap N_R} \Bigg(\prod_{\substack{C \in S^A(X,r), \\ D \in S^B(Y,r)}} \exists r.\text{k-lcs-r}\big(C, D, \mathsf{S}_\mathcal{T}, k-1, A, B, \mathsf{Sig}(\mathcal{T})\big) \Bigg)$$

Fig. 2. Computation algorithm for role-depth bounded $\mathcal{ELOR}$-lcs.

sets S^A and S^B, without the need to additionally compute $S^{A'}$ and $S^{B'}$, or the completion sets S^C for any other basic concept C encountered during the recursive computation of the k-lcs. This allows for a goal-oriented optimization in cases where there is no need to classify the full TBox.

3.3 Computing the Role-depth Bounded msc w.r.t. $\mathcal{ELOR}$-KBs

We now turn our attention to the other generalization inference: the computation of the most specific concept representing a given individual. Recall that, since $\mathcal{ELOR}$ allows the use of nominals, computing the (exact) $\mathcal{ELOR}$-msc for a given individual is a trivial task: the most specific $\mathcal{ELOR}$-concept describing an individual $a \in N_I$ is simply the nominal $\{a\}$. However, it may be of interest to compute the msc w.r.t. a less expressive target DL. Next, we describe how to compute the depth-bounded $\mathcal{EL}$-msc of an individual w.r.t. an $\mathcal{ELOR}$-KB.

As we have defined them, KBs consist of two parts: the TBox, which represents the conceptual knowledge of the domain, and the ABox, which states information about individuals. In the presence of nominals, this division between concepts and individuals is blurred. In fact, it is possible to simulate ABox assertions using GCIs as described by the following proposition.

Lemma 1. *An interpretation $\mathcal{I}$ satisfies the concept assertion $C(a)$ iff it satisfies the GCI $\{a\} \sqsubseteq C$. It satisfies the role assertion $r(a, b)$ iff it satisfies the GCI $\{a\} \sqsubseteq \exists r.\{b\}$.*

Procedure k-msc $(a, \mathcal{K}, k)$
Input: a: individual from $\mathcal{K}$; $\mathcal{K}$ =($\mathcal{T}$,$\mathcal{A}$) an $\mathcal{ELOR}$-KB; $k \in \mathbf{N}$
Output: role-depth bounded $\mathcal{EL}$-msc of a w.r.t. $\mathcal{K}$ and k.

1: $\mathcal{T}' := \mathcal{T} \cup \mathsf{absorb\text{-}ABox}(\mathcal{K})$
2: $\mathcal{T}'' := \mathsf{normalize}(\mathcal{T}')$
3: $\mathsf{S}_\mathcal{K} := \mathsf{apply\text{-}completion\text{-}rules}(\mathcal{T}'')$
4: **return** $\mathsf{traversal\text{-}concept}(\{a\}, \mathsf{S}_\mathcal{K}, k, \mathsf{Sig}(\mathcal{K}))$

Procedure traversal-concept $(A, \mathsf{S}_\mathcal{K}, k, \mathsf{Sig}(\mathcal{K}))$
Input: A: basic concept from $\mathcal{T}'$; $\mathsf{S}_\mathcal{K}$: set of completion sets; $k \in \mathbf{N}$;
$\mathsf{Sig}(\mathcal{K})$: signature of original KB $\mathcal{K}$
Output: role-depth bounded traversal concept w.r.t. $\mathcal{K}$ and k.

1: **if** $k = 0$ **then**
2: **return** $\bigsqcap_{B \in S^\top(A) \cap (\mathsf{BC}_\mathcal{T} \setminus N_I)} B$
3: **else**
4: **return** $\bigsqcap_{B \in S^\top(A) \cap (\mathsf{BC}_\mathcal{T} \setminus N_I)} B \;\sqcap$
$\bigsqcap_{r \in \mathsf{Sig}(\mathcal{K}) \cap N_R} \; \bigsqcap_{B \in S^\top(A,r)} \exists r.\mathsf{traversal\text{-}concept}(B, \mathsf{S}_\mathcal{K}, k{-}1, \mathsf{Sig}(\mathcal{K}))$

Fig. 3. Computation algorithm for the role-depth bounded $\mathcal{EL}$-msc w.r.t. $\mathcal{ELOR}$-KBs.

Using this result, we can 'absorb' the ABox into the TBox and restrict our attention to reasoning w.r.t. TBoxes only, without losing generality. Figure 3 describes the algorithm for computing the $\mathcal{EL}$-k-msc w.r.t. an $\mathcal{ELOR}$-KB.

As before, correctness of this algorithm is a consequence of the invariants described by Proposition 1. The set $S^\top(\{a\})$ contains all the basic concepts that subsume the nominal $\{a\}$; that is, all concepts whose interpretation must contain the individual $a^\mathcal{I}$. Likewise, $S^\top(\{a\}, r)$ contains all the existential restrictions subsuming $\{a\}$. Thus, a recursive conjunction of all these subsumers provides the most specific representation of the individual a.

Since the target language is $\mathcal{EL}$, no nominals may be included in the output. However, the recursion includes also the $\mathcal{EL}$-msc of the removed nominals, hence indeed providing the most specific $\mathcal{EL}$ representation of the input individual. As in the computation of the lcs presented above, the only completion sets relevant for computing the msc are those of the form $S^\top(A)$ and $S^\top(A, r)$. Once again, this means that it is possible to implement a goal-oriented approach that computes only these sets, as needed, when building the msc for a given individual.

In this section we have shown how to compute generalization inferences with a bounded role-depth in the DL $\mathcal{ELOR}$ that extends $\mathcal{EL}$ by allowing nominals and complex role inclusion axioms in the KB. With the exception of data-types and disjointness axioms, this covers the full expressivity of the OWL 2 EL profile of the standard ontology language OWL 2. Given its status as W3C standard, it is likely that more and bigger ontologies built using this profile, thus the gener-

alization inferences investigated in this paper and their computation algorithms for approximation will become more useful to ontology engineers. In fact, there already exist ontologies that use nominals in their representation. For example, the FMA ontology [18] is written in $\mathcal{ELOR}$ and currently contains 85 nominals.

4 Conclusions

We have studied reasoning services in extensions of the light-weight description logic $\mathcal{EL}$ by nominals and role inclusions, which yields the DL $\mathcal{ELOR}$. One of the characterizing features of $\mathcal{EL}$ and its extension $\mathcal{ELOR}$ is that they allow for polynomial time reasoning. Efficient reasoning becomes expedient when dealing with huge knowledge bases such as the biomedical ontologies SNOMED and the Gene Ontology. Additionally, $\mathcal{ELOR}$ covers a large part of the OWL 2 EL profile. Given its status as a W3C recommendation, it is likely that the usage of the $\mathcal{EL}$-family of DLs becomes more widespread in the future.

Especially for the huge ontologies written in extensions of $\mathcal{EL}$, tools that aid the user with the construction and maintenance of the knowledge base are necessary. As previous work has shown, the generalization inferences lcs and msc can be effectively used for such tasks. Besides this, the generalizations can be used as a basis for other inferences, like the construction of semantic similarity measures and information retrieval procedures.

The contributions of the paper are manyfold. First, we have given a completion algorithm for $\mathcal{ELOR}$ knowledge bases, inspired by a consequence-based classification algorithm for $\mathcal{EL}$ with nominals [11]. This completion algorithm is then employed to extend the algorithms for computing approximations of the lcs and of the msc for the DL $\mathcal{ELOR}$. In general, the lcs and msc do not need to exist, even for $\mathcal{EL}$, thus we approximate them by limiting the role-depth of the resulting concept description, up to a maximal bound specified by the user.

We extended here the computation algorithm of the k-lcs to the DL $\mathcal{ELOR}$, using the new completion algorithm, by allowing nominals as part of the resulting concept. Since the k-msc is trivial in $\mathcal{ELOR}$ due to nominals, we give a computation algorithm for the k-msc for the target language $\mathcal{EL}$, which works for $\mathcal{ELOR}$-KBs. Using these algorithms, the generalization inferences can be used for a large set of ontologies built for the OWL 2 EL profile. Both algorithms have the property that, if the exact lcs or msc exist, then our algorithms compute the exact solution for a sufficiently large role-depth bound k. Such a k can be computed for $\mathcal{EL}$ using the necessary and sufficient conditions for the existence of the lsc and msc given in [21].

As future work we intend to study methods of finding these generalizations in further extensions of $\mathcal{EL}$. Initial steps in this direction have been made by considering $\mathcal{EL}$ with subjective probability constructors [17]. In a different direction, we also intend to implement a system that can compute the lcs and the msc, by modifying and improving existing completion-based reasoners.

References

1. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 325–330. Morgan Kaufmann, 2003.
2. F. Baader. Terminological cycles in a description logic with existential restrictions. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 319–324. Morgan Kaufmann, 2003.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.
4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
5. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann, Los Altos.
6. A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*, volume 147 of *CEUR Workshop*, 2005.
7. S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics — what does it buy me? In G. Görz, V. Haarslev, C. Lutz, and R. Möller, editors, *Proc. of the 2001 Applications of Description Logic Workshop (ADL 2001)*, number 44 in CEUR Workshop, Vienna, Austria, September 2001. RWTH Aachen. See `http://CEUR-WS.org/Vol-44/`.
8. C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In *Pro.c of Convegno Italiano di Logica Computazionale, CILC05*, 2005.
9. A. Ecke and A.-Y. Turhan. Role-depth bounded least common subsumers for $\mathcal{EL}^+$ and $\mathcal{ELI}$. In Y. Kazakov, D. Lembo, and F. Wolter, editors, *Proc. of the 2012 Description Logic Workshop (DL 2012)*, volume 846 of *CEUR Workshop Proceedings*. CEUR Workshop, 2012.
10. Y. Kazakov, M. Krötzsch, and F. Simančík. ELK reasoner: Architecture and evaluation. In I. Horrocks, M. Yatskevich, and E. Jimenez-Ruiz, editors, *Proceedings of the OWL Reasoner Evaluation Workshop (ORE'12)*, volume 858 of *CEUR Workshop*. CEUR-WS.org, 2012.
11. Y. Kazakov, M. Krötzsch, and F. Simančík. Practical reasoning with nominals in the $\mathcal{EL}$ family of description logics. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-12)*, pages 264–274. AAAI Press, 2012.
12. R. Küsters and R. Molitor. Approximating most specifc concepts in description logics with existential restrictions. In F. Baader, G. Brewka, and T. Eiter, editors, *Proc. of the 24th German Annual Conf. on Artificial Intelligence (KI'01)*, volume 2174 of *Lecture Notes In Artificial Intelligence*, pages 33–47. Springer, 2001.
13. K. Lehmann and A.-Y. Turhan. A framework for semantic-based similarity measures for $\mathcal{ELH}$-concepts. In L. F. del Cerro, A. Herzig, and J. Mengin, editors, *Proceedings of the 13th European Conference on Logics in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 307–319. Springer Verlag, 2012.

14. J. Mendez. jCel: A modular rule-based reasoner. In *In Proc. of the 1st Int. Workshop on OWL Reasoner Evaluation (ORE'12)*, volume 858 of *CEUR Workshop*, 2012.
15. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 web ontology language profiles. W3C Recommendation, 27 October 2009. `http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/`.
16. R. Peñaloza and A.-Y. Turhan. A practical approach for computing generalization inferences in $\mathcal{EL}$. In M. Grobelnik and E. Simperl, editors, *Proc. of the 8th European Semantic Web Conf. (ESWC'11)*, Lecture Notes in Computer Science. Springer, 2011.
17. R. Peñaloza and A.-Y. Turhan. Instance-based non-standard inferences in $\mathcal{EL}$ with subjective probabilities. In F. Bobillo, P. C. G. Costa, C. d'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool, editors, *Uncertainty Reasoning for the Semantic Web II*, number 7123 in Lecture Notes in Computer Science, pages 80–98. Springer, 2013.
18. C. Rosse and J. L. V. Mejino. A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of Biomedical Informatics*, 36:478–500, 2003.
19. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Assoc.*, 2000. Fall Symposium Special Issue.
20. W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation, 27th October 2009. `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`.
21. B. Zarrieß and A.-Y. Turhan. Most specific generalizations w.r.t. general $\mathcal{EL}$-tboxes. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, Beijing, China, 2013. AAAI Press. To appear.

Instance-based Non-standard Inferences in $\mathcal{EL}$ with Subjective Probabilities

Rafael Peñaloza and Anni-Yasmin Turhan*

Institute for Theoretical Computer Science, TU Dresden, Germany,
email: *last name*@tcs.inf.tu-dresden.de

Abstract. For practical ontology-based applications representing and reasoning with probabilities is an essential task. For Description Logics with subjective probabilities reasoning procedures for testing instance relations based on the completion method have been developed.
In this paper we extend this technique to devise algorithms for solving non-standard inferences for $\mathcal{EL}$ and its probabilistic extension Prob-$\mathcal{EL}_c^{01}$: computing the most specific concept of an individual and finding explanations for instance relations.

1 Introduction

The ontology language recommended for the semantic web OWL [11, 25] is based on Description Logics (DLs) [4]. Description logics are knowledge representation formalisms with formal semantics. Based on these semantics, powerful reasoning services have been defined and reasoning algorithms have been investigated. In recent years, so-called lightweight DLs have been devised; these DLs have a limited expressiveness, which allows for efficient reasoning [6]. For the lightweight DL $\mathcal{EL}$, typical DL reasoning services such as classification of TBoxes, i.e., computation of all sub- / superconcept relations of named concepts, or the realization of ABoxes, i.e., computation of the named concepts each of the ABox individuals belongs to, can be done in polynomial time. The basis for ABox realization is *instance checking*, which tests whether a given individual from the ABox belongs to a given concept. In the so-called $\mathcal{EL}$-family of DLs, which are the tractable extensions of $\mathcal{EL}$, this inference can be computed using completion algorithms, which extend the ones for concept subsumption [2, 3].

The DLs from the $\mathcal{EL}$-family are employed most prominently in the medical field, for instance in the well-known knowledge base SNOMED CT [23], as well as in context-aware applications. In both of these application areas, the need for characterizing uncertain observations, which are only known to hold with some probability, has been long recognized. While several probabilistic extensions of DLs have been proposed—see [14] for a survey—these are typically very expressive and thus no longer tractable and they cannot handle subjective probabilities. A simple probabilistic variant of $\mathcal{EL}$ that can express subjective probabilities is Prob-$\mathcal{EL}_c^{01}$, recently introduced in [15]. This logic allows only a fairly limited use of uncertainty. More precisely, it is only possible to

* Partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing".

express that a concept *may* hold ($P_{>0}C$), or that it holds *almost surely* ($P_{=1}C$). Despite its limited expressivity, this logic is interesting due to its nice algorithmic properties; as shown in [15], subsumption and instance checking can also be performed in polynomial time.

In this paper we employ the above mentioned completion algorithms to compute two non-standard inferences for DLs that allow to express subjective probability: the most specific concept and explanation of instance relations in Prob-$\mathcal{EL}_c^{01}$.

Many practical applications that need to represent observed information, such as medical applications or context-aware applications, need to characterize that these observations only hold with certain probability. Furthermore, these applications face the problem that information from different sources does not coincide, e.g., different diagnoses yield differing results. These applications need to "integrate" differing observations for the same state of affairs [24]. A way to determine what information the different information sources agree upon is to represent this information in the ABox by different individuals and then to find a common generalization of these individuals. A description of such a generalization of a group of ABox individuals can be obtained by applying the so-called *bottom-up approach* for constructing knowledge bases [5]. In this approach a set of individuals is generalized into a single concept description by first generating the most specific concept (msc) of each individual and then applying the least common subsumer (lcs) to the set of obtained concept descriptions to extract their commonalities.

The second step, i.e., a computation procedure for the approximate lcs has been investigated for $\mathcal{EL}$ and Prob-$\mathcal{EL}_c^{01}$ in [21]. In this paper we present a similar procedure for the msc. For the Description Logic $\mathcal{EL}$ the msc need not exist [1], if computed with respect to general $\mathcal{EL}$-TBoxes. However, it is still possible to find a concept description that is the msc up to a fixed role-depth. This so-called k-msc is still a generalization of the input, but not necessarily the least one—in this sense, it is only an approximation of the msc. We first describe a practical approach for computing the role-depth bounded msc, based on the polynomial-time completion algorithm for $\mathcal{EL}$, and then extend it to the probabilistic variant Prob-$\mathcal{EL}_c^{01}$. Our algorithms are based upon the completion algorithms for ABox realization in $\mathcal{EL}$ and in Prob-$\mathcal{EL}_c^{01}$ and thus can be easily implemented on top of reasoners of these DLs. All the proofs can be found in [18].

The second non-standard inference that we explore in this paper is the *explanation* of a given consequence. In case a large knowledge base is edited by hand, it is not trivial for the developer to see why a particular consequence holds [10, 12]. In our case of instance checking, we want to identify those statements in the TBox and the ABox that cause an instance relationship to follow from the knowledge base. More precisely, we want to compute *minimal axiom sets* (MinAs) that entail the consequence. We compute these sets using a glass-box approach for axiom-pinpointing [22, 7]. Even for ontology-based context-aware systems, which may operate on automatically generated ABoxes, the identification of MinAs that cause an unwanted consequence is crucial, since it is the first step to edit the knowledge base such that the consequence is resolved. More than in the crisp case, finding the axioms that entail a consequence for a knowledge base written in a DL with probabilities is a difficult task to do by hand.

A method to compute MinAs for subsumptions in $\mathcal{EL}$ was devised in [9] as an extension of the completion algorithm for TBox classification. In this paper we devise a method to compute MinAs for instance relationships as an extension of the completion algorithm for ABox realization for Prob-$\mathcal{EL}_c^{01}$.

This paper extends earlier work presented in [20, 21] by algorithms for computing explanations for instance relationships in $\mathcal{EL}$ and Prob-$\mathcal{EL}_c^{01}$. To the best of our knowledge, explanation has not yet been investigated for DLs that allow to express probabilities. We start this undertaking by giving the basic notions in Section 2. In Section 3 we recall the completion algorithms for ABox realization. Section 4 discusses the computation algorithm for the role-depth bounded msc. In Section 5 we introduce the algorithm for computing explanations.

2 $\mathcal{EL}$ and Prob-$\mathcal{EL}$

In this section we introduce the DL $\mathcal{EL}$ and its probabilistic variant Prob-$\mathcal{EL}_c^{01}$. Let N_I, N_C and N_R be disjoint sets of *individual-*, *concept-* and *role names*, respectively. *Prob-$\mathcal{EL}_c^{01}$-concept descriptions* are built using the syntax rule

$$C, D ::= \top \mid A \mid C \sqcap D \mid \exists r.C \mid P_{>0}C \mid P_{=1}C,$$

where $A \in N_C$, and $r \in N_R$. *$\mathcal{EL}$-concept descriptions* are Prob-$\mathcal{EL}_c^{01}$-concept description that do not contain the constructors $P_{>0}$ or $P_{=1}$.

A *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. An $\mathcal{EL}$- (Prob-$\mathcal{EL}_c^{01}$-) *TBox* is a finite set of *general concept inclusions* (GCIs) of the form $C \sqsubseteq D$, where C, D are $\mathcal{EL}$- (Prob-$\mathcal{EL}_c^{01}$-) concept descriptions. An *$\mathcal{EL}$-ABox* is a set of assertions of the form $C(a)$ or $r(a, b)$, where C is an $\mathcal{EL}$-concept description, $r \in N_R$, and $a, b \in N_I$. A *Prob-$\mathcal{EL}_c^{01}$-ABox* is a set of assertions of the form $C(a)$, $r(a, b)$, $P_{>0}r(a, b)$, or $P_{=1}r(a, b)$, where C is a Prob-$\mathcal{EL}_c^{01}$-concept description, $r \in N_R$, and $a, b \in N_I$.

The semantics of $\mathcal{EL}$ is defined by means of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names, subsets of $\Delta^{\mathcal{I}}$ to concepts and elements of $\Delta^{\mathcal{I}}$ to individual names. For a more detailed description of this semantics, see [4].

We say that the interpretation $\mathcal{I}$ *satisfies* a general concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; it *satisfies* an assertion $C(a)$, denoted as $\mathcal{I} \models C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and it *satisfies* an assertion $r(a, b)$, denoted as $\mathcal{I} \models r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. It is a *model* of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it satisfies all GCIs in $\mathcal{T}$ and all assertions in $\mathcal{A}$.

The semantics of Prob-$\mathcal{EL}_c^{01}$ is a generalization of the semantics of $\mathcal{EL}$, that considers a set of possible worlds. A *probabilistic interpretation* is of the form

$$\mathcal{I} = (\Delta^{\mathcal{I}}, W, (\mathcal{I}_w)_{w \in W}, \mu),$$

where $\Delta^{\mathcal{I}}$ is the (non-empty) *domain*, W is a (non-empty) set of *worlds*, μ is a discrete probability distribution on W, and for each world $w \in W$, $\mathcal{I}_w$ is a classical $\mathcal{EL}$ interpretation with domain $\Delta^{\mathcal{I}}$, where $a^{\mathcal{I}_w} = a^{\mathcal{I}_{w'}}$ for all $a \in N_I$, $w, w' \in W$. The probability

that a given element of the domain $d \in \Delta^{\mathcal{I}}$ belongs to the interpretation of a concept name A is

$$p_d^{\mathcal{I}}(A) := \mu(\{w \in W \mid d \in A^{\mathcal{I}_w}\}).$$

The functions $\mathcal{I}_w$ and $p_d^{\mathcal{I}}$ are extended to complex concepts in the usual way for the classical $\mathcal{EL}$-constructors, where the extension to the new constructors P_* is defined as

$$(P_{>0}C)^{\mathcal{I}_w} := \{d \in \Delta^{\mathcal{I}} \mid p_d^{\mathcal{I}}(C) > 0\},$$
$$(P_{=1}C)^{\mathcal{I}_w} := \{d \in \Delta^{\mathcal{I}} \mid p_d^{\mathcal{I}}(C) = 1\}.$$

The probabilistic interpretation $\mathcal{I}$ *satisfies* a general concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$, if for every $w \in W$ it holds that $C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w}$. It is a *model* of a TBox $\mathcal{T}$ if it satisfies all general concept inclusions in $\mathcal{T}$. Let C, D be two Prob-$\mathcal{EL}_c^{01}$ concepts and $\mathcal{T}$ a TBox. We say that C is *subsumed* by D w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$. The concepts c and D are equivalent, if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$ holds. The probabilistic interpretation $\mathcal{I}$ *satisfies* the assertion $P_{>0}r(a,b)$ if $\mu(\{w \in W \mid \mathcal{I}_w \models r(a,b)\}) > 0$, and analogously for $P_{=1}r(a,b)$. $\mathcal{I}$ *satisfies* the ABox $\mathcal{A}$ if there is a $w \in W$ such that $\mathcal{I}_w \models \mathcal{A}$.

Finally, an individual $a \in N_I$ is an *instance* of a concept description C w.r.t. $\mathcal{K}$ ($\mathcal{K} \models C(a)$) if $\mathcal{I} \models C(a)$ for all models $\mathcal{I}$ of $\mathcal{K}$. The *ABox realization problem* is to compute for each individual a in $\mathcal{A}$ the set of named concepts from $\mathcal{K}$ that have a as an instance and that are least (w.r.t. $\sqsubseteq$). One of our main interests in this paper is to compute most specific concepts.

Definition 1 (most specific concept). *Let $\mathcal{L}$ be a DL, $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-knowledge base. The* most specific concept *(msc) of an individual a from $\mathcal{A}$ is the $\mathcal{L}$-concept description C s. t.*

1. *$\mathcal{K} \models C(a)$, and*
2. *for each $\mathcal{L}$-concept description D holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

The msc depends on the DL in use. For the DLs with conjunction as concept constructor the msc is, if it exists, unique up to equivalence. Thus it is justified to speak of *the* msc.

3 Completion Algorithms for ABox Realization

In this section we briefly sketch the completion algorithms for instance checking in the DLs $\mathcal{EL}$ [2] and Prob-$\mathcal{EL}_c^{01}$ [15].

3.1 The Completion Algorithm for $\mathcal{EL}$

Assume we want to test for an $\mathcal{EL}$-knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ whether $\mathcal{K} \models D(a)$ holds. The completion algorithm first augments the knowledge base by introducing a concept name for the complex concept description D for the instance check; that is, it redefines the knowledge base to $\mathcal{K} = (\mathcal{T} \cup \{A_q \equiv D\}, \mathcal{A})$, where A_q is a new concept name not appearing in $\mathcal{K}$. The instance checking algorithm for $\mathcal{EL}$ works on knowledge

NF1 $C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\ \hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E\ \}$
NF2 $\exists r.\hat{C} \sqsubseteq D \longrightarrow \{\ \hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D\ \}$
NF3 $\hat{C} \sqsubseteq \hat{D} \longrightarrow \{\ \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D}\ \}$
NF4 $B \sqsubseteq \exists r.\hat{C} \longrightarrow \{\ B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C}\ \}$
NF5 $B \sqsubseteq C \sqcap D \longrightarrow \{\ B \sqsubseteq C, B \sqsubseteq D\ \}$
where $\hat{C}, \hat{D} \notin \mathsf{BC}_{\mathcal{T}}$ and A is a new concept name.

Fig. 1. $\mathcal{EL}$ normalization rules (from [2])

bases containing only axioms in a structured *normal form*. Every knowledge base can be transformed into a normalized one via a two-step procedure.

First the ABox is transformed into a simple ABox. An ABox $\mathcal{A}$ is a *simple ABox*, if for every concept assertion $C(a) \in \mathcal{A}$, C is a concept name. An arbitrary $\mathcal{EL}$-ABox $\mathcal{A}$ can be transformed into a simple ABox by first replacing each complex assertion $C(A)$ in $\mathcal{A}$ by $A(a)$ where A is a fresh concept name and, second, introducing $A \equiv C$ into the TBox.

After this step, the TBox is transformed into a normal form as well. For a concept description C let $\mathsf{CN}(C)$ denote the set of all concept names and $\mathsf{RN}(C)$ denote the set of all role names that appear in C. The *signature of a concept description* C (denoted $\mathsf{sig}(C)$) is given by $\mathsf{CN}(C) \cup \mathsf{RN}(C)$. Similarly, the set of concept (respectively role) names that appear in a TBox is denoted by $\mathsf{CN}(\mathcal{T})$ (respectively $\mathsf{RN}(\mathcal{T})$). The *signature of a TBox* $\mathcal{T}$ (denoted $\mathsf{sig}(\mathcal{T})$) is $\mathsf{CN}(\mathcal{T}) \cup \mathsf{RN}(\mathcal{T})$. The *signature of an ABox* $\mathcal{A}$ (denoted $\mathsf{sig}(\mathcal{A})$) is the set of concept (role / individual) names $\mathsf{CN}(\mathcal{A})$ ($\mathsf{RN}(\mathcal{A})$/$\mathsf{IN}(\mathcal{A})$ resp.) that appear in $\mathcal{A}$. The signature of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (denoted $\mathsf{sig}(\mathcal{K})$) is $\mathsf{sig}(\mathcal{T}) \cup \mathsf{sig}(\mathcal{A})$.

An $\mathcal{EL}$-TBox $\mathcal{T}$ is in *normal form* if all concept axioms have one of the following forms, where $C_1, C_2 \in \mathsf{sig}(\mathcal{T})$ and $D \in \mathsf{sig}(\mathcal{T}) \cup \{\bot\}$:

$$C_1 \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C_1 \sqsubseteq \exists r.C_2 \quad \text{or} \quad \exists r.C_1 \sqsubseteq D.$$

Any $\mathcal{EL}$-TBox can be transformed into normal form by introducing new concept names and by applying the normalization rules displayed in Figure 1 exhaustively, where $\mathsf{BC}_{\mathcal{T}}$ is the set containing all the concept names appearing in $\mathcal{T}$ and the concept $\top$. These rules replace the GCI on the left-hand side of the rules with the set of GCIs on the right-hand side. Clearly, for a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ the signature of $\mathcal{A}$ may be changed only during the first of the two normalization steps and the signature of $\mathcal{T}$ may be extended during both of them. The normalization of the knowledge base can be done in linear time.

The completion algorithm for instance checking is based on the one for classifying $\mathcal{EL}$-TBoxes introduced in [2]. The completion algorithm constructs a representation of the minimal model of $\mathcal{K}$. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a normalized $\mathcal{EL}$-knowledge base, i.e., with a simple ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ in normal form. The completion algorithm works on four kinds of *completion sets*: $S(a), S(a, r), S(C)$ and $S(C, r)$ for each $a \in \mathsf{IN}(\mathcal{A})$, $C \in \mathsf{CN}(\mathcal{K})$ and $r \in \mathsf{RN}(\mathcal{K})$. These completion sets contain concept names

CR1 If $C \in S(X), C \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR2 If $C_1, C_2 \in S(X), C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR3 If $C \in S(X), C \sqsubseteq \exists r.D \in \mathcal{T}$, and $D \notin S(X, r)$
then $S(X, r) := S(X, r) \cup \{D\}$

CR4 If $Y \in S(X, r), C \in S(Y), \exists r.C \sqsubseteq D \in \mathcal{T}$, and
$D \notin S(X)$ then $S(X) := S(X) \cup \{D\}$

Fig. 2. $\mathcal{EL}$ completion rules

from $\mathsf{CN}(\mathcal{K})$. Intuitively, the completion rules make implicit subsumption and instance relationships explicit in the following sense:

- $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{T}} D$,
- $D \in S(C, r)$ implies that $C \sqsubseteq_{\mathcal{T}} \exists r.D$.
- $D \in S(a)$ implies that a is an instance of D w.r.t. $\mathcal{K}$,
- $D \in S(a, r)$ implies that a is an instance of $\exists r.D$ w.r.t. $\mathcal{K}$.

$\mathsf{S}_{\mathcal{K}}$ denotes the set of all completion sets of a normalized $\mathcal{K}$. The completion sets are initialized for each $a \in \mathsf{IN}(\mathcal{A})$ and each $C \in \mathsf{CN}(\mathcal{K})$ as follows:

- $S(C) := \{C, \top\}$ for each $C \in \mathsf{CN}(\mathcal{K})$,
- $S(C, r) := \emptyset$ for each $r \in \mathsf{RN}(\mathcal{K})$,
- $S(a) := \{C \in \mathsf{CN}(\mathcal{A}) \mid C(a) \text{ appears in } \mathcal{A}\} \cup \{\top\}$, and
- $S(a, r) := \{b \in \mathsf{IN}(\mathcal{A}) \mid r(a, b) \text{ appears in } \mathcal{A}\}$ for each $r \in \mathsf{RN}(\mathcal{K})$.

Then these sets are extended by applying the completion rules shown in Figure 2 until no more rule applies. In these rules X and Y can refer to concept or individual names, while C, C_1, C_2 and D are concept names and r is a role name. After the completion has terminated, the following relations hold between an individual a, a role r and named concepts A and B:

- subsumption relation between A and B from $\mathcal{K}$ holds iff $B \in S(A)$
- instance relation between a and B from $\mathcal{K}$ holds iff $B \in S(a)$,

as shown in [2]. To decide the initial query: $\mathcal{K} \models D(a)$, one has to test whether A_q appears in $S(a)$. In fact, instance queries for all individuals and all named concepts from the knowledge base can be answered from the resulting completion sets; the completion algorithm does not only perform one instance check, but complete ABox realization. The completion algorithm runs in polynomial time in size of the knowledge base.

3.2 The Completion Algorithm for Prob-$\mathcal{EL}_c^{01}$

Before describing the completion algorithm for Prob-$\mathcal{EL}_c^{01}$, we modify the notion of basic concepts. The set $\mathsf{BC}_{\mathcal{T}}$ of Prob-$\mathcal{EL}_c^{01}$ *basic concepts* for a knowledge base $\mathcal{K}$ is the smallest set that contains

PR1	If $C' \in S_*(X, v)$ and $C' \sqsubseteq D \in \mathcal{T}$, then $S_*(X, v) := S_*(X, v) \cup \{D\}$
PR2	If $C_1, C_2 \in S_*(X, v)$ and $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, then $S_*(X, v) := S_*(X, v) \cup \{D\}$
PR3	If $C' \in S_*(X, v)$ and $C' \sqsubseteq \exists r.D \in \mathcal{T}$, then $S_*(X, r, v) := S_*(X, r, v) \cup \{D\}$
PR4	If $D \in S_*(X, r, v)$, $D' \in S_{\gamma(v)}(D, \gamma(v))$ and $\exists r.D' \sqsubseteq E \in \mathcal{T}$, then $S_*(X, v) := S_*(X, v) \cup \{E\}$
PR5	If $P_{>0}A \in S_*(X, v)$, then $S_*(X, P_{>0}A) := S_*(X, P_{>0}A) \cup \{A\}$
PR6	If $P_{=1}A \in S_*(X, v)$, $v \neq 0$, then $S_*(X, v) := S_*(X, v) \cup \{A\}$
PR7	If $A \in S_*(X, v)$ and $v \neq 0$, $P_{>0}A \in \mathcal{P}_0^{\mathcal{T}}$, then $S_*(X, v') := S_*(X, v') \cup \{P_{>0}A\}$
PR8	If $A \in S_*(X, 1)$ and $P_{=1}A \in \mathcal{P}_1^{\mathcal{T}}$, then $S_*(X, v) := S_*(X, v) \cup \{P_{=1}A\}$
PR9	If $r(a, b) \in \mathcal{A}, C \in S(b, 0), \exists r.C \sqsubseteq D \in \mathcal{T}$, then $S(a, 0) := S(a, 0) \cup \{D\}$
PR10	If $P_{>0}r(a, b) \in \mathcal{A}$, $C \in S(b, P_{>0}r(a, b))$ and $\exists r.C \sqsubseteq D \in \mathcal{T}$, then $S(a, P_{>0}r(a, b)) := S(a, P_{>0}r(a, b)) \cup \{D\}$
PR11	If $P_{=1}r(a, b) \in \mathcal{A}$, $C \in S(b, v)$ with $v \neq 0$ and $\exists r.C \sqsubseteq D \in \mathcal{T}$, then $S(a, v) := S(a, v) \cup \{D\}$

Fig. 3. Prob-$\mathcal{EL}_c^{01}$ completion rules

1. the concept $\top$,
2. all concept names used in $\mathcal{K}$, and
3. all concepts of the form $P_{>0}A$ or $P_{=1}A$,

where A is a concept name in $\mathcal{K}$. A Prob-$\mathcal{EL}_c^{01}$-TBox $\mathcal{T}$ is in *normal form* if all its axioms are of one of the following forms

$$C \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C \sqsubseteq \exists r.A, \quad \exists r.A \sqsubseteq D,$$

where $C, C_1, C_2, D \in \mathsf{BC}_{\mathcal{T}}$ and A is a concept name. The normalization rules in Figure 1 can also be used to transform a Prob-$\mathcal{EL}_c^{01}$-TBox into this extended normal form. We still assume that the ABox $\mathcal{A}$ is a simple ABox; that is, for all assertions $C(a)$ in $\mathcal{A}$, C is a concept name. We denote as $\mathcal{P}_0^{\mathcal{T}}$ and $\mathcal{P}_1^{\mathcal{T}}$ the set of all concepts of the form $P_{>0}A$ and $P_{=1}A$ respectively, occurring in a normalized knowledge base $\mathcal{K}$. Analogously, $\mathcal{R}_0^{\mathcal{T}}$ denotes the set of all assertions of the form $P_{>0}r(a, b)$ appearing in $\mathcal{K}$.

The completion algorithm for Prob-$\mathcal{EL}_c^{01}$ follows the same idea as the algorithm for $\mathcal{EL}$, but uses several completion sets to deal with the information of what needs to be satisfied in the different worlds of a model. Intuitively, we will build a general description of all models, using the set of worlds $V := \{0, \varepsilon, 1\} \cup \mathcal{P}_0^{\mathcal{T}} \cup \mathcal{R}_0^{\mathcal{T}}$, where the probability distribution μ assigns a probability of 0 to the world 0, and the uniform probability $1/(|V|-1)$ to all other worlds. The main idea is that the world 1 will include all the entailments that hold with probability 1, and ε those that hold with probability greater than 0.

For each individual name a, concept name A, role name r and world v, we store the completion sets $S_0(A, v), S_\varepsilon(A, v), S_0(A, r, v), S_\varepsilon(A, r, v), S(a, v)$, and $S(a, r, v)$.

The algorithm initializes the sets as follows for every $A \in \mathsf{BC}_{\mathcal{T}}, r \in \mathsf{RN}(\mathcal{K})$, and $a \in \mathsf{IN}(\mathcal{A})$:

- $S_0(A,0) = \{\top, A\}$ and $S_0(A,v) = \{\top\}$ for all $v \in V \setminus \{0\}$,
- $S_\varepsilon(A,\varepsilon) = \{\top, A\}$ and $S_\varepsilon(A,v) = \{\top\}$ for all $v \in V \setminus \{\varepsilon\}$,
- $S(a,0) = \{\top\} \cup \{A \mid A(a) \in \mathcal{A}\}$, $S(a,v) = \{\top\}$ for all $v \neq 0$,
- $S_0(A,r,v) = S_\varepsilon(A,r,v) = \emptyset$ for all $v \in V$, $S(a,r,v) = \emptyset$ for $v \neq 0$,
- $S(a,r,0) = \{b \in \mathsf{IN}(\mathcal{A}) \mid r(a,b) \in \mathcal{A}\}$.

These sets are then extended by exhaustively applying the rules shown in Figure 3, where X ranges over $\mathsf{BC}_\mathcal{T} \cup \mathsf{IN}(\mathcal{A})$, $S_*(X,v)$ stands for $S(X,v)$ if X is an individual and for $S_0(X,v), S_\varepsilon(X,v)$ if $X \in \mathsf{BC}_\mathcal{T}$, and $\gamma : V \to \{0,\varepsilon\}$ is defined by $\gamma(0) = 0$, and $\gamma(v) = \varepsilon$ for all $v \in V \setminus \{0\}$.

This algorithm terminates in polynomial time. After termination, the completion sets store all the information necessary to decide subsumption of concept names, as well as checking whether an individual is an instance of a given concept name [15]. For the former decision, it holds that for every pair A, B of concept names: $B \in S_0(A,0)$ iff $A \sqsubseteq_\mathcal{K} B$. In the case of instance checking, we have that $\mathcal{K} \models A(a)$ iff $A \in S(a,0)$.

4 Computing the k-MSC using Completion

The msc was first investigated for $\mathcal{EL}$-concept descriptions and w.r.t. unfoldable TBoxes and possibly cyclic ABoxes in [13]. It was shown that the msc does not need to exists for cyclic ABoxes. Consider the ABox $\mathcal{A} = \{r(a,a), C(a)\}$. The msc of a is then

$$C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \cdots$$

and cannot be expressed by a finite concept description. For cyclic TBoxes it has been shown in [1] that the msc does not need to exists even if the ABox is acyclic.

To avoid infinite nestings in presence of cyclic ABoxes it was proposed in [13] to limit the role-depth of the concept description to be computed. This limitation yields an approximation of the msc, which is still a concept description with the input individual as an instance, but it does not need to be the least one (w.r.t. $\sqsubseteq$) with this property. We follow this idea to compute approximations of the msc also in presence of general TBoxes.

The *role-depth* of a concept description C (denoted $rd(C)$) is the maximal number of nested quantifiers of C. This allows us to define the msc with limited role-depth for $\mathcal{EL}$.

Definition 2 (role-depth bounded $\mathcal{EL}$-msc). *Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be an $\mathcal{EL}$-knowledge base and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then the $\mathcal{EL}$-concept description C is the* role-depth bounded $\mathcal{EL}$-most specific concept *of a w.r.t. $\mathcal{K}$ and role-depth k (written k-$msc_\mathcal{K}(a)$) iff*

1. *$rd(C) \leq k$,*
2. *$\mathcal{K} \models C(a)$, and*
3. *for all $\mathcal{EL}$-concept descriptions E with $rd(E) \leq k$ holds: $\mathcal{K} \models E(a)$ implies $C \sqsubseteq_\mathcal{T} E$.*

Notice that in case the exact msc has a role-depth less or equal to k the role-depth bounded msc is the exact msc.

Example 3. As an example we consider the labeled knowledge base $\mathcal{K}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$. In this labeled knowledge base each axiom and assertion is associated with a label (printed in the same line), which will be used later.

$$\mathcal{T}_{ex} = \{\, \exists r.\top \sqsubseteq A, \quad ax1 \qquad B \sqsubseteq \exists r.C, \quad ax2 \qquad D \sqsubseteq E\} \quad ax3$$

and

$$\mathcal{A}_{ex} = \{\, B(a), \quad as1 \qquad D(b), \quad as2 \qquad r(a,b), \quad as3 \qquad s(a,c), \quad as4 \qquad r(c,a)\ \} \quad as5$$

Obviously the ABox $\mathcal{A}_{ex}$ is cyclic due to the last two assertions. Note, that c is an instance of A due to $as5$ and $ax1$. Now, for $k = 3$ we obtain the following role-depth bounded msc for a:

$$\begin{aligned} 3\text{-}msc_{\mathcal{K}_{ex}}(a) = \; & B \sqcap \\ & \exists r.D \sqcap \\ & \exists s.(A \sqcap \exists r.(B \sqcap \exists r.D \sqcap \exists s.A))). \end{aligned}$$

Next we describe how to obtain the k-msc in general.

4.1 Computing the k-msc in $\mathcal{EL}$ by Completion

The computation of the msc relies on a characterization of the instance relation. While in earlier works this was given by homomorphisms [13] or simulations [1] between graph representations of the knowledge base and the concept in question, we use the completion algorithm as such a characterization. Moreover, we construct the msc by traversing the completion sets to "collect" the msc. More precisely, the set of completion sets encodes a graph structure, where the sets $S(X)$ are the nodes and the sets $S(X, r)$ encode the edges. Traversing this graph structure, one can construct an $\mathcal{EL}$-concept. To obtain a finite concept in the presence of cyclic ABoxes or TBoxes one can limit the number of edges than can be traversed during this construction.

Definition 4 (traversal concept). *Let $\mathcal{K}$ be an $\mathcal{EL}$-knowledge base, $\mathcal{K}'$ be its normalized form, $\mathsf{S}_{\mathcal{K}}$ the completion set obtained from $\mathcal{K}$ and $k \in \mathbb{N}$. Then the* traversal concept of a named concept A *(denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(A)$) with $\mathsf{sig}(A) \subseteq \mathsf{sig}(\mathcal{K}')$ is the concept obtained from executing the procedure call* traversal-concept-c*(A, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm 1.*

The traversal concept of an individual a *(denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(a)$) with $a \in \mathsf{sig}(\mathcal{K})$ is the concept description obtained from executing the procedure call* traversal-concept-i*(a, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm 1.*

The idea is that the traversal concept of an individual yields its msc. However, the traversal concept contains names from $\mathsf{sig}(\mathcal{K}') \setminus \mathsf{sig}(\mathcal{K})$, i.e., concept names that were introduced during normalization—we call this kind of concept names *normalization names* in the following. The returned msc should be formulated w.r.t. the signature of the original knowledge base, thus the normalization names need to be removed or replaced.

Algorithm 1 Computation of a role-depth bounded $\mathcal{EL}$-msc.

Procedure k-msc $(a, \mathcal{K}, k)$
Input: a: individual from $\mathcal{K}$; $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) an $\mathcal{EL}$-knowledge base; $k \in \mathbb{N}$
Output: role-depth bounded $\mathcal{EL}$-msc of a w.r.t. $\mathcal{K}$ and k.

1: $(\mathcal{T}', \mathcal{A}')$:= simplify-ABox($\mathcal{T}$, $\mathcal{A}$)
2: $\mathcal{K}'$:= (normalize($\mathcal{T}'$), $\mathcal{A}'$)
3: $\mathsf{S}_{\mathcal{K}}$:= apply-completion-rules($\mathcal{K}$)
4: **return** Remove-normalization-names (traversal-concept-i$(a, \mathsf{S}_{\mathcal{K}}, k)$)

Procedure traversal-concept-i (a, S, k)
Input: a: individual name from $\mathcal{K}$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth traversal concept (w.r.t. $\mathcal{K}$) and k.

1: **if** $k = 0$ **then return** $\bigsqcap_{A \,\in\, S(a)} A$
2: **else return** $\bigsqcap_{A \,\in\, \mathsf{S}(a)} A \sqcap$
$$\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}')} \; \bigsqcap_{A \,\in\, \mathsf{CN}(\mathcal{K}') \cap S(a,r)} \exists r.\, \text{traversal-concept-c}\ (A, \mathsf{S}, k-1) \;\sqcap$$
$$\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}')} \; \bigsqcap_{b \,\in\, \mathsf{IN}(\mathcal{K}') \cap S(a,r)} \exists r.\, \text{traversal-concept-i}\ (b, \mathsf{S}, k-1)$$
3: **end if**

Procedure traversal-concept-c (A, S, k)
Input: A: concept name from $\mathcal{K}'$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth bounded traversal concept.

1: **if** $k = 0$ **then return** $\bigsqcap_{B \in S(A)} B$
2: **else return** $\bigsqcap_{B \in S(A)} B \;\sqcap \bigsqcap_{r \in \mathsf{RN}(\mathcal{K}')} \; \bigsqcap_{B \in S(A,r)} \exists r.\text{traversal-concept-c}\ (B, \mathsf{S}, k-1)$
3: **end if**

Lemma 5. *Let $\mathcal{K}$ be an $\mathcal{EL}$-knowledge base, $\mathcal{K}'$ its normalized version, $\mathsf{S}_{\mathcal{K}}$ be the set of completion sets obtained for $\mathcal{K}$, $k \in \mathbb{N}$ a natural number and $a \in \mathsf{IN}(\mathcal{K})$. If $C = k\text{-}\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(a)$ and $\widehat{C}$ is obtained from C by removing the normalization names, then*

$$\mathcal{K}' \models C(a) \textit{ iff } \mathcal{K} \models \widehat{C}(a).$$

This lemma guarantees that removing the normalization names from the traversal concept preserves the instance relationships. Intuitively, this lemma holds since the construction of the traversal concept conjoins exhaustively all named subsumers and all subsuming existential restrictions to a normalization name up to the role-depth bound. Thus removing the normalization name does not change the extension of the conjunction. The proof can be found in [18].

The procedure k-msc uses an individual a from a knowledge base $\mathcal{K}$, the knowledge base $\mathcal{K}$ itself and a number k for the role depth-bound as parameters. It first performs the two normalization steps on $\mathcal{K}$, then applies the completion rules from Figure 2 to the normalized knowledge base $\mathcal{K}'$, and then stores the set of completion sets in $\mathsf{S}_{\mathcal{K}}$. Afterwards it computes the traversal-concept of a from $\mathsf{S}_{\mathcal{K}}$ w.r.t. role-depth bound k. In a post-processing step it applies Remove-normalization-names to the traversal concept obtained in the previous step.

Example 6. We use the knowledge base from Example 3, to apply the algorithm k-msc to the individual a from $\mathcal{A}_{ex}$ again for $k = 3$. Since the TBox $\mathcal{T}_{ex}$ is in normal form and the ABox $\mathcal{A}_{ex}$ is simple, completion can be applied directly. After completion we have the following elements in the completion sets:

$$\begin{array}{lll} S(A) = \{\top, A\} & S(a) = \{\top, A, B\} & S(B, r) = \{\top, C\} \\ S(B) = \{\top, A, B\} & S(b) = \{\top, D, E\} & S(a, r) = \{\top, D, E\} \\ S(C) = \{\top, C\} & S(c) = \{\top, A\} & S(a, s) = \{\top, A\} \\ S(D) = \{\top, D, E\} & & S(c, r) = \{\top, A\} \end{array}$$

The here omitted completion sets do not change after initialization and are empty. We obtain:

$$\begin{aligned} \mathsf{k\text{-}msc}(a, \mathcal{K}_{ex}, 3) = \; & \top \sqcap A \sqcap B \sqcap \\ & \exists r.(\top \sqcap D \sqcap E) \sqcap \\ & \exists s.(\top \sqcap A \sqcap \exists r.(\top \sqcap A \sqcap B \sqcap \exists r.(\top \sqcap D \sqcap E) \sqcap \exists s.(\top \sqcap A))). \end{aligned}$$

The resulting concept description is larger than the k-msc derived in Example 3, since all the elements from the completion set are conjoined to the result concept description in traversal-concept-i and traversal-concept-c. However, it is easy to see that the result is a concept description equivalent to the k-msc w.r.t. $\mathcal{K}_{ex}$.

Obviously, the concept description returned from the procedure k-msc has a role-depth less or equal to k. Thus the first condition of Definition 2 is fulfilled. As we prove next, the concept description obtained from the procedure k-msc fulfills also the second condition from Definition 2.

Lemma 7. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-knowledge base and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. If $C = \mathsf{k\text{-}msc}(a, \mathcal{K}, k)$, then $\mathcal{K} \models C(a)$.*

The claim can be shown by induction on k. Each name in C is from a completion set of (1) an individual or (2) a concept, which is connected via existential restrictions to an individual. The full proof can be found in [18].

Lemma 8. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-knowledge base, a an individual appearing in $\mathcal{A}$, and $k \in \mathbb{N}$. If $C = \mathsf{k\text{-}msc}(a, \mathcal{K}, k)$, then for every $\mathcal{EL}$-concept description E with $rd(E) \leq k$ the following holds: $\mathcal{K} \models E(a)$ implies $C \sqsubseteq_{\mathcal{T}} E$.*

Again, the full proof can be found in [18]. Together, these two Lemmas yield the correctness of the overall procedure.

Theorem 9. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-knowledge base and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$.*
Then $\mathsf{k\text{-}msc}(a, \mathcal{K}, k) \equiv k\text{-}msc_{\mathcal{K}}(a)$.

It is important to notice that, while the completion sets can be computed in polynomial time, the k-msc can grow exponential in the size of the knowledge base. In addition to that and as the example already indicated, the concept description obtained from k-msc contains a lot of redundant information and thus is quite larger. However for practical usability it is necessary to rewrite the concept to an equivalent, but smaller one. A heuristic for this has been proposed in [16]. The algorithm and the rewriting heuristic are implemented in the GEL system[1].

[1] See `http://gen-el.sourceforge.net/`

4.2 Computing the k-msc in Prob-$\mathcal{EL}_c^{01}$ by Completion

The role-bounded msc for a Prob-$\mathcal{EL}_c^{01}$-knowledge base can be computed in a similar fashion to the one described before for $\mathcal{EL}$. The knowledge base is first normalized and the completion procedure is executed to obtain all the completion sets.

In order to compute the msc, we simply accumulate all concepts to which the individual a belongs, given the information stored in the completion sets. This process needs to be done recursively in order to account for both, the successors of a explicitly encoded in the ABox, and the nesting of existential restrictions masked by normalization names. In the following we use the abbreviation $S^{>0}(a,r) := \bigcup_{v \in V \setminus \{0\}} S(a,r,v)$. We then define $\mathsf{traversal\text{-}concept\text{-}i}(a, \mathsf{S}, k)$ as

$$\prod_{B \in S(a,0)} B \sqcap \prod_{r \in \mathsf{RN}(\mathcal{K}'')} \Big(\prod_{r(a,b) \in \mathcal{K}''} \exists r.\mathsf{traversal\text{-}concept\text{-}i}(b, \mathsf{S}, k-1) \sqcap \prod_{B \in \mathsf{CN}(\mathcal{K}'') \cap S(a,r,0)} \exists r.\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, k-1) \sqcap \prod_{B \in \mathsf{CN}(\mathcal{K}'') \cap S(a,r,1)} P_{=1}(\exists r.\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, k-1)) \sqcap \prod_{B \in \mathsf{CN}(\mathcal{K}'') \cap S^{>0}(a,r)} P_{>0}(\exists r.\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, k-1))\Big),$$

where $\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, k+1)$ is

$$\prod_{C \in S_0(B,0)} B \sqcap \prod_{r \in \mathsf{RN}} \Big(\prod_{C \in S_0(B,r,0)} \exists r.\mathsf{traversal\text{-}concept\text{-}c}(C, \mathsf{S}, k) \sqcap \prod_{C \in S_0(B,r,1)} P_{=1}(\exists r.\mathsf{traversal\text{-}concept\text{-}c}(C, \mathsf{S}, k)) \sqcap \prod_{C \in S_0^{>0}(B,r)} P_{>0}(\exists r.\mathsf{traversal\text{-}concept\text{-}c}(C, \mathsf{S}, k))\Big)$$

and $\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, 0) = \prod_{C \in S_0(B,0)} B$. Once the traversal concept has been computed, it is possible to remove all normalization names preserving the instance relation, which gives us the msc in the original signature of $\mathcal{K}$. As in the case for $\mathcal{EL}$, the proof of correctness of this method can be found in [18].

Theorem 10. *Let $\mathcal{K}$ a Prob-$\mathcal{EL}_c^{01}$-knowledge base, $a \in \mathsf{IN}(\mathcal{A})$, and $k \in \mathbb{N}$; then* $\mathsf{Remove\text{-}normalization\text{-}names}(\mathsf{traversal\text{-}concept\text{-}i}(a, \mathsf{S}, k)) \equiv k\text{-}msc_{\mathcal{K}}(a)$.

5 Computing Explanations for Instance Relations in Prob-$\mathcal{EL}_c^{01}$

By definition, an individual a is always an instance of its (role-depth bounded) msc. However, it is not always obvious why this is the case. We thus provide a method for describing the axiomatic causes for a to be an instance of a concept name A.

Definition 11 (MinA). *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an Prob-$\mathcal{EL}_c^{01}$-knowledge base, a an individual in $\mathcal{A}$ and A a concept name such that $\mathcal{K} \models A(a)$. A* minimal axiom set (MinA) *for $\mathcal{K}$ w.r.t. $A(a)$ is a sub-knowledge base $\mathcal{K}' = (\mathcal{S}, \mathcal{B})$, with $\mathcal{S} \subseteq \mathcal{T}, \mathcal{B} \subseteq \mathcal{A}$ such that*

– $\mathcal{K}' \models A(a)$ *and*
– *for all strict subsets* $\mathcal{S}' \subset \mathcal{S}, \mathcal{B}' \subset \mathcal{B}$*, it holds that (i)* $(\mathcal{S}', \mathcal{B}) \not\models A(a)$ *and (ii)* $(\mathcal{S}, \mathcal{B}') \not\models A(a)$.

Intuitively, a MinA is a sub-ontology that still entails the instance relationship between a and A, and that is minimal (w.r.t. set inclusion) with this property. As the following example illustrates there may be several MinAs for one consequence.

Example 12. Continuing with our running example, we have that $\mathcal{K}_{ex} \models A(a)$, and there are two MinAs for $\mathcal{K}_{ex}$ w.r.t. this instance relationship, namely

$$\mathcal{K}_1 = (\{\exists r.\top \sqsubseteq A, B \sqsubseteq \exists r.C\}, \{B(a)\}), \text{ and}$$
$$\mathcal{K}_2 = (\{\exists r.\top \sqsubseteq A\}, \{r(a,b)\}).$$

It is a simple task to verify that indeed these two knowledge bases entail $A(a)$, and that they satisfy the minimality requirement w.r.t. set inclusion.

The process of computing MinAs is called *pinpointing*. As it has been done before for other kinds of reasoning problems, we show that the completion algorithm for Prob-$\mathcal{EL}_c^{01}$ can be modified into a *pinpointing algorithm*. Rather than directly computing the MinAs, we will construct a monotone Boolean formula—called the *pinpointing formula*—that encodes all these MinAs. To define this formula, we first assume that every axiom and every assertion α in $\mathcal{K}$ is labeled with a *unique* propositional variable $\mathsf{lab}(\alpha)$ and denote as $\mathsf{lab}(\mathcal{K})$ the set of all propositional variables labeling axioms and assertions in $\mathcal{K}$. A *monotone Boolean formula* over $\mathsf{lab}(\mathcal{K})$ is a Boolean formula that uses only variables from $\mathsf{lab}(\mathcal{K})$, the binary connectives conjunction ($\wedge$) and disjunction ($\vee$), and the constant t (for "truth"). As customary in propositional logic, we identify a *valuation* with the set of propositional variables that it makes true. Finally, given a valuation $\mathcal{V} \subseteq \mathsf{lab}(\mathcal{K})$, we define

$$\mathcal{K}_\mathcal{V} := (\{\alpha \in \mathcal{T} \mid \mathsf{lab}(\alpha) \in \mathcal{V}\}, \{\alpha \in \mathcal{A} \mid \mathsf{lab}(\alpha) \in \mathcal{V}\}).$$

Definition 13 (pinpointing formula). *Given a Prob-*$\mathcal{EL}_c^{01}$*-knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$*, an individual name* a *occurring in* $\mathcal{A}$ *and a concept name* A*, the monotone Boolean formula* ϕ *over* $\mathsf{lab}(\mathcal{K})$ *is a* pinpointing formula *for* $\mathcal{K}$ *w.r.t.* $A(a)$ *if for every valuation* $\mathcal{V} \subseteq \mathsf{lab}(\mathcal{K})$ *it holds that*

$$\mathcal{K}_\mathcal{V} \models A(a) \textit{ iff } \mathcal{V} \textit{ satisfies } \phi.$$

Example 14. Recall that we have given every axiom and assertion of $\mathcal{K}_{ex}$ a unique label, depicted in Example 3. Hence, for instance $\mathsf{lab}(\exists r.\top \sqsubseteq A) = ax1$. The following is a pinpointing formula for $\mathcal{K}_{ex}$ w.r.t. $A(a)$:

$$ax1 \wedge (as3 \vee (ax2 \wedge as1)).$$

The MinAs for an instance relation can be obtained from the pinpointing formula ϕ by computing the minimal valuations that satisfy ϕ.

Proposition 15. *If ϕ is a pinpointing formula for $\mathcal{K}$ w.r.t. $A(a)$, then the set*

$$\{\mathcal{K}_{\mathcal{V}} \mid \mathcal{V} \text{ is a minimal valuation satisfying } \phi\}$$

is the set of all MinAs for $\mathcal{K}$ w.r.t. $A(a)$.

We take advantage of this proposition and describe an algorithm that computes a pinpointing formula for a given instance relationship.[2] If one is interested in the specific MinAs, it is only necessary to find the minimal valuations that satisfy this formula. This can be done by e.g. bringing the pinpointing formula to disjunctive normal form first and then removing all the non-minimal disjuncts. In general, a pinpointing formula may yield a more compact representation of the set of all MinAs, and hence be of more practical use.

We will use a so-called glass-box approach for computing pinpointing formulas for all the instance relationships that follow from a knowledge base $\mathcal{K}$. The idea is to extend the completion algorithm for deciding instances in Prob-$\mathcal{EL}_c^{01}$ with a tracing mechanism that encodes all the axiomatic causes for a consequence—in this case, either a subsumption or an instance relation—to follow. Since $\mathcal{EL}$ is a sub-logic of Prob-$\mathcal{EL}_c^{01}$ and classification can be reduced to instance checking,[3] our approach can also find the pinpointing formulas for the different subsumption relations that follow from the knowledge base. Thus, we generalize previous results on axiom-pinpointing in $\mathcal{EL}$ [9] in two ways by developing explanations also for the entailed instance relationships and include the probabilistic concept constructors from Prob-$\mathcal{EL}_c^{01}$.

In order to describe the pinpointing algorithm, we assume first that the knowledge base $\mathcal{K}$ is already in normal form; recall that our example knowledge base $\mathcal{K}_{ex}$ is in normal form. The *pinpointing extension* of the completion algorithm for Prob-$\mathcal{EL}_c^{01}$ also stores completion sets $S(a, v), S(a, r, v), S_0(C, v), S_0(A, r, v), S_\varepsilon(A, v)$, and, $S_\varepsilon(A, r, v)$ for the different individual-, and role names a, r, respectively, and basic concept A appearing in the knowledge base. However, the elements of these sets are not only concept names from $\mathsf{CN}(\mathcal{K})$ as in Section 3, but rather pairs of the form (D, φ), where $D \in \mathsf{CN}(\mathcal{K})$ and φ is a monotone Boolean formula. Intuitively, $(D, \varphi) \in S(C)$ means that D is a subsumer of C w.r.t. $\mathcal{K}$, and φ stores information of the axioms responsible for this fact. For the other three kinds of completion sets the idea is analogous.

The pinpointing algorithm initializes these completion sets as follows: for every $A \in \mathsf{BC}_{\mathcal{T}}, r \in \mathsf{RN}(\mathcal{K})$, and $a \in \mathsf{IN}(\mathcal{A})$

- $S_0(A, 0) = \{(\top, \mathsf{t}), (A, \mathsf{t})\}$ and $S_0(A, v) = \{(\top, \mathsf{t})\}$ for all $v \in V \setminus \{0\}$,
- $S_\varepsilon(A, \varepsilon) = \{(\top, \mathsf{t}), (A, \mathsf{t})\}$ and $S_\varepsilon(A, v) = \{(\top, \mathsf{t})\}$ for all $v \in V \setminus \{\varepsilon\}$,
- $S(a, 0) = \{(\top, \mathsf{t})\} \cup \{(A, p) \mid A(a) \in \mathcal{A}, p = \mathsf{lab}(A(a))\}$,
- $S(a, v) = \{(\top, \mathsf{t})\}$ for all $v \neq 0$,
- $S_0(A, r, v) = S_\varepsilon(A, r, v) = \emptyset$ for all $v \in V$, $S(a, r, v) = \emptyset$ for $v \neq 0$,
- $S(a, r, 0) = \{(b, p) \in \mathsf{IN}(\mathcal{A}) \mid r(a, b) \in \mathcal{A}, p = \mathsf{lab}(A(a))\}$.

[2] In fact, our method produces pinpointing formulas for *all* instance relationships that follow from the knowledge base at once.

[3] Indeed, $A \sqsubseteq_{\mathcal{K}} B$ iff $\mathcal{K} \cup \{A(a)\} \models B(a)$, where a is an individual name not appearing in $\mathcal{K}$.

PpR1 If $(C', \varphi) \in S_*(X, v)$, $\alpha = C' \sqsubseteq D \in \mathcal{T}$, and $\mathsf{lab}(\alpha) = p$
then $S_*(X, v) := S_*(X, v) \Cup (D, \varphi \wedge p)$

PpR2 If $(C_1, \varphi_1), (C_2, \varphi_2) \in S_*(X, v)$, $\alpha = C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $\mathsf{lab}(\alpha) = p$
then $S_*(X, v) := S_*(X, v) \Cup (D, \varphi_1 \wedge \varphi_2 \wedge p)$

PpR3 If $(C', \varphi) \in S_*(X, v)$, $\alpha = C' \sqsubseteq \exists r.D \in \mathcal{T}$, and $\mathsf{lab}(\alpha) = p$
then $S_*(X, r, v) := S_*(X, r, v) \Cup (D, \varphi \wedge p)$

PpR4 If $(D, \varphi) \in S_*(X, r, v)$, $(D', \varphi') \in S_{\gamma(v)}(D, \gamma(v))$, $\alpha = \exists r.D' \sqsubseteq E \in \mathcal{T}$, and
$\mathsf{lab}(\alpha) = p$ then $S_*(X, v) := S_*(X, v) \Cup (E, \varphi \wedge \varphi' \wedge p)$

PpR5 If $(P_{>0}A, \varphi) \in S_*(X, v)$, then $S_*(X, P_{>0}A) := S_*(X, P_{>0}A) \Cup (A, \varphi)$

PpR6 If $(P_{=1}A, \varphi) \in S_*(X, v)$, $v \neq 0$, then $S_*(X, v) := S_*(X, v) \Cup (A, \varphi)$

PpR7 If $(A, \varphi) \in S_*(X, v)$ and $v \neq 0$, $P_{>0}A \in \mathcal{P}_0^{\mathcal{T}}$
then $S_*(X, v') := S_*(X, v') \Cup (P_{>0}A, \varphi)$

PpR8 If $(A, \varphi) \in S_*(X, 1)$ and $P_{=1}A \in \mathcal{P}_1^{\mathcal{T}}$, then $S_*(X, v) := S_*(X, v) \Cup (P_{=1}A, \varphi)$

PpR9 If $\alpha_1 = r(a, b) \in \mathcal{A}$, $(C, \varphi) \in S(b, 0)$, $\alpha_2 = \exists r.C \sqsubseteq D \in \mathcal{T}$,
$\mathsf{lab}(\alpha_1) = p_1$, and $\mathsf{lab}(\alpha_2) = p_2$ then $S(a, 0) := S(a, 0) \Cup (D, \varphi \wedge p_1 \wedge p_2)$

PpR10 If $\alpha_1 = P_{>0}r(a, b) \in \mathcal{A}$, $(C, \varphi) \in S(b, P_{>0}r(a, b))$, $\alpha_2 = \exists r.C \sqsubseteq D \in \mathcal{T}$,
$\mathsf{lab}(\alpha_1) = p_1$, and $\mathsf{lab}(\alpha_2) = p_2$
then $S(a, P_{>0}r(a, b)) := S(a, P_{>0}r(a, b)) \Cup (D, \varphi \wedge p_1 \wedge p_2)$

PpR11 If $\alpha_1 = P_{=1}r(a, b) \in \mathcal{A}$, $(C, \varphi) \in S(b, v)$ with $v \neq 0$, $\alpha_2 = \exists r.C \sqsubseteq D \in \mathcal{T}$,
$\mathsf{lab}(\alpha_1) = p_1$, and $\mathsf{lab}(\alpha_2) = p_2$ then $S(a, v) := S(a, v) \Cup (D, \varphi \wedge p_1 \wedge p_2)$

Fig. 4. Prob-$\mathcal{EL}_c^{01}$ completion rules for axiom-pinpointing

For describing the extended completion rules, we need some more notation. For a set S and a pair (D, φ), the operation $S \Cup (D, \varphi)$ is defined as follows: if there exists a ψ such that $(D, \psi) \in S$, then $S \Cup (D, \varphi) := S \setminus \{(D, \psi)\} \cup \{(D, \psi \vee \varphi)\}$; otherwise, $S \Cup (D, \varphi) := S \cup \{(D, \varphi)\}$. In other words, if the concept name D already belongs to S with some associated formula ψ, we modify the formula by adding φ to it as a disjunct; otherwise, we simply add the pair (D, φ) to S.

The completion sets are then extended by exhaustively applying the rules shown in Figure 4, where X ranges over $\mathsf{BC}_{\mathcal{T}} \cup \mathsf{IN}(\mathcal{A})$, $S_*(X, v)$ stands for $S(X, v)$ if X is an individual and for $S_0(X, v), S_\varepsilon(X, v)$ if $X \in \mathsf{BC}_{\mathcal{T}}$, and $\gamma : V \to \{0, \varepsilon\}$ is defined by $\gamma(0) = 0$, and $\gamma(v) = \varepsilon$ for all $v \in V \setminus \{0\}$.

To ensure termination of this algorithm, the completion can only be applied if their application modifies at least one of the completion sets; that is, if either a new pair is added, or the second element of an existing pair is modified to a (strictly) more general Boolean formula. Under this applicability condition, this modified algorithm always terminates, although not necessarily in polynomial time. In fact, every completion set can contain at most as many pairs as there are concept names in $\mathcal{K}$, and hence polynomially many. Whenever the formula of a pair is changed, it is done so by generalizing it in the sense that it has more models than the previous one. As there are exponentially

many models, such changes can only be done an exponential number of times. Thus, in total we can have at most exponentially many rule applications, which take each at most exponential time; that is, the pinpointing algorithm runs in exponential time in the size of $\mathcal{K}$.

As stated before, these completion sets make the subsumption and instance relationships explicit, together with a formula that describe which axioms are responsible for each of these relationships. It is easy to see that the concepts appearing in the completion sets are exactly the same that will be obtained by applying the *standard* completion rules from Section 3. We thus know that $A \sqsubseteq_{\mathcal{K}} B$ iff there is some ψ with $(B, \psi) \in S_0(A, 0)$ and $\mathcal{K} \models A(a)$ iff $(A, \psi) \in S(a, 0)$ for some monotone Boolean formula ψ. Moreover, the pinpointing algorithm maintains the following invariants:

- if $(B, \psi) \in S_0(A, 0)$, then for every valuation $\mathcal{V}$ satisfying ψ, $A \sqsubseteq_{\mathcal{K}_\mathcal{V}} B$,
- if $(A, \psi) \in S(a, 0)$, then for every valuation $\mathcal{V}$ satisfying ψ, $\mathcal{K}_\mathcal{V} \models A(a)$.

It can also be shown that when the algorithm has terminated, the converse implications also hold; this is a consequence of the results from [8].

Theorem 16. *Given a Prob-$\mathcal{EL}_c^{01}$-knowledge base in normal form, the pinpointing algorithm terminates in exponential time. After termination, the following holds for every concept name A and individual name a appearing in $\mathcal{K}$:*

$$\textit{if } (A, \psi) \in S(a, 0)\textit{, then } \psi \textit{ is a pinpointing formula for } \mathcal{K} \textit{ w.r.t. } A(a).$$

We have so far described how to find the MinAs of a normalized knowledge base w.r.t. instance and subsumption relations. We now show how to extend this method to deal also with non-normalized knowledge bases; that is, to obtain the MinAs referring to the original axioms of the knowledge base and not to their normalized versions. Before going into the details, it is worth noticing that the relationship between original axioms and normalized axioms is many-to-many: one axiom in the original knowledge base may produce several axioms in the normalized one, while one axiom in the normalized knowledge base can be due to the presence of several axioms from the original one. An example of the latter can be given by the two axioms $A \sqsubseteq B$, $A \sqsubseteq B \sqcap C$. The normalization rules change these axioms into $A \sqsubseteq B, A \sqsubseteq C$, but the first axiom has two sources; that is, it will appear in the normalized knowledge base whenever *any* of the two original axioms is present.

Let $\hat{\mathcal{K}}$ be an arbitrary Prob-$\mathcal{EL}_c^{01}$-knowledge base and $\mathcal{K}$ its normalized version. If ϕ is a pinpointing formula for $\mathcal{K}$ w.r.t. an instance or subsumption relation, that uses only basic concepts appearing in $\hat{\mathcal{K}}$, then we can modify ϕ into a pinpointing formula *for the original knowledge base* $\hat{\mathcal{K}}$ as follows. As in the case of normalized knowledge bases, each axiom in $\hat{\mathcal{K}}$ is associated with a unique propositional variable. Each normalized axiom in $\mathcal{K}$ has a finite number of original axioms that created it—at most as many as there were in the original knowledge base. We modify the pinpointing formula ϕ by replacing the propositional formula associated to each normalized axiom by the disjunction of the labels of all its sources. We thus obtain a new pinpointing formula that speaks of the original ontology $\hat{\mathcal{K}}$. In the above example, let $\mathsf{lab}(A \sqsubseteq B) = p_1$ and $\mathsf{lab}(A \sqsubseteq B \sqcap C) = p_2$, and suppose that the labels of the normalized ontology are $\mathsf{lab}(A \sqsubseteq B) = q_1, \mathsf{lab}(A \sqsubseteq C) = q_2$, and that the knowledge base also contains

an assertion $A(a)$ with label q_3. The pinpointing formula for the normalized ontology w.r.t. $B(a)$ is $q_1 \wedge q_3$. For the original ontology, this formula is changed to $(p_1 \vee p_2) \wedge q_3$.

It is worth commenting on the execution time of the pinpointing algorithm and the complexity of finding *all* MinAs. Recall that computing all MinAs is crucial when resolving an unwanted consequence of a knowledge base. As described before, the algorithm takes exponential time to compute all instance and subsumption relations between concept names and individual names, with their respective pinpointing formulas. These formulas may be exponential in the size of the knowledge base $\mathcal{K}$, however finding one or all the minimal valuations satisfying a formula is only exponential on the number of propositional variables appearing in that formula, hence, we can compute one or all MinAs from each of these pinpointing formulas in exponential time in the size of $\mathcal{K}$. Since classification of an $\mathcal{EL}$ TBox is a special case of our setting—where the ABox $\mathcal{A}$ is empty and no probabilistic concepts are used—our algorithm yields an optimal upper bound on the complexity of pinpointing for Prob-$\mathcal{EL}_c^{01}$. Indeed, it has been shown that finding all MinAs for one subsumption relation in $\mathcal{EL}$ requires already exponential time [17]. Additionally, other kinds of tasks like finding a MinA of least cardinality or the first MinA w.r.t. some underlying ordering, can be also solved by computing the related valuations over the pinpointing formula; this is in particular beneficial, as the various optimizations developed in the SAT community, and in particular the very efficient modern SAT/SMT-solvers, can be exploited.

6 Conclusions

In this paper we have presented a practical method for computing the role-depth bounded msc in $\mathcal{EL}$- and in Prob-$\mathcal{EL}_c^{01}$- w.r.t. a general TBox or cyclic ABoxes. Our approach is based on the completion sets that are computed during realization of a knowledge base. Thus, any of the available implementations of the $\mathcal{EL}$ completion algorithm, as for instance JCEL[4] [16] can be easily extended to an implementation of the (approximative) msc computation algorithm – as it is provided in the GEL system[5]. We also showed that the same idea can be adapted for the computation of the msc in the probabilistic DL Prob-$\mathcal{EL}_c^{01}$.

Together with the completion-based computation of role-depth bounded (least) common subsumers given in [19] these results complete the bottom-up approach for general $\mathcal{EL}$- and Prob-$\mathcal{EL}_c^{01}$-knowledge bases. This approach yields a practical method to compute commonalities for differing observations regarding individuals. To the best of our knowledge this has not been investigated for DLs that can express uncertainty.

We have also applied the ideas of axiom-pinpointing to compute explanations for instance relationships that follow from a Prob-$\mathcal{EL}_c^{01}$-knowledge base. To the best of our knowledge this is also the first time that axiom-pinpointing has been applied to instance relationships, even for crisp DLs. The glass-box approach proposed modifies the computation of the completion sets to include an encoding of the axiomatic causes for a concept to be added to each set. Understanding the causes for some unexpected instance relationships is an important first step towards correcting a knowledge base,

[4] `http://jcel.sourceforge.net/`
[5] `http://gen-el.sourceforge.net/`

specially in the case of automatically generated ones, as done through the bottom-up approach described before. In general, finding out the precise axioms responsible for an unwanted consequence is a very hard task, even for experts, due to the large number of axioms available. When dealing with uncertainty, the difficulty grows, as the probabilities may interact in unexpected ways. Thus, being able to explain the consequences of a Prob-$\mathcal{EL}_c^{01}$ ontology automatically is of special importance.

References

1. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 325–330. Morgan Kaufmann, 2003.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In K. Clark and P. F. Patel-Schneider, editors, *In Proc. of the OWLED Workshop*, 2008.
4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
5. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann, Los Altos.
6. F. Baader, C. Lutz, and A.-Y. Turhan. Small is again Beautiful in Description Logics. *KI – Künstliche Intelligenz*, 24(1):25–33, April 2010.
7. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2010. Special Issue: Tableaux and Analytic Proof Methods.
8. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2010. Special Issue: Tableaux and Analytic Proof Methods.
9. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of the 30th German Annual Conf. on Artificial Intelligence (KI'07)*, volume 4667 of *Lecture Notes In Artificial Intelligence*, pages 52–67, Osnabrück, Germany, 2007. Springer.
10. F. Baader and B. Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proceedings of the International Conference on Representing and Sharing Knowledge Using SNOMED (KR-MED'08)*, Phoenix, Arizona, 2008.
11. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. W3C Recommendation, February 2004. `http://www.w3.org/TR/owl-ref/`.
12. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280, 2007.
13. R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. *AI Communications*, 15(1):47–59, 2002.
14. T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *J. Web Sem.*, 6(4):291–308, 2008.

15. C. Lutz and L. Schröder. Probabilistic description logics for subjective probabilities. In F. Lin and U. Sattler, editors, *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-10)*, 2010.
16. J. Mendez, A. Ecke, and A.-Y. Turhan. Implementing completion-based inferences for the $\mathcal{EL}$-family. In R. Rosati, S. Rudolph, and M. Zakharyaschev, editors, *Proc. of the 2011 Description Logic Workshop (DL 2011)*, volume 745. CEUR, 2011.
17. R. Peñaloza and B. Sertkaya. On the complexity of axiom pinpointing in the el family of description logics. In F. Lin, U. Sattler, and M. Truszczynski, editors, *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning (KR 2010)*. AAAI Press, 2010.
18. R. Peñaloza and A.-Y. Turhan. Completion-based computation of most specific concepts with limited role-depth for $\mathcal{EL}$ and prob-$\mathcal{EL}^{01}$. LTCS-Report LTCS-10-03, Chair f. Automata Theory, Inst. for Theoretical Computer Science, TU Dresden, Germany, 2010.
19. R. Peñaloza and A.-Y. Turhan. Role-depth bounded least common subsumers by completion for $\mathcal{EL}$- and Prob-$\mathcal{EL}$-TBoxes. In V. Haarslev, D. Toman, and G. Weddell, editors, *Proc. of the 2010 Description Logic Workshop (DL'10)*, 2010.
20. R. Peñaloza and A.-Y. Turhan. Towards approximative most specific concepts by completion for $\mathcal{EL}^{01}$ with subjective probabilities. In T. Lukasiewicz, R. Peñaloza, and A.-Y. Turhan, editors, *Proceedings of the First International Workshop on Uncertainty in Description Logics (UniDL'10)*, 2010.
21. R. Peñaloza and A.-Y. Turhan. A practical approach for computing generalization inferences in $\mathcal{EL}$. In M. Grobelnik and E. Simperl, editors, *Proc. of the 8th European Semantic Web Conf. (ESWC'11)*, Lecture Notes in Computer Science. Springer, 2011.
22. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.
23. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. *Journal of the American Medical Informatics Assoc.*, 2000. Fall Symposium Special Issue.
24. T. Springer and A.-Y. Turhan. Employing description logics in ambient intelligence for modeling and reasoning about complex situations. *Journal of Ambient Intelligence and Smart Environments*, 1(3):235–259, 2009.
25. W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation, 27th October 2009. `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`.

Most Specific Generalizations w.r.t. General $\mathcal{EL}$-TBoxes

Benjamin Zarrieß and **Anni-Yasmin Turhan***
Institute for Theoretical Computer Science, Technische Universität Dresden, Germany
{zarriess,turhan}@tcs.inf.tu-dresden.de

Abstract

In the area of Description Logics the least common subsumer (lcs) and the most specific concept (msc) are inferences that generalize a set of concepts or an individual, respectively, into a single concept. If computed w.r.t. a general $\mathcal{EL}$-TBox neither the lcs nor the msc need to exist. So far in this setting no exact conditions for the existence of lcs- or msc-concepts are known. This paper provides necessary and sufficient conditions for the existence of these two kinds of concepts. For the lcs of a fixed number of concepts and the msc we show decidability of the existence in PTime and polynomial bounds on the maximal role-depth of the lcs- and msc-concepts. This bound allows to compute the lcs and the msc, respectively.

1 Introduction

Description Logics (DL) allow to model application domains in a structured and well-understood way. Due to their formal semantics, DLs can offer powerful reasoning services.

In recent years the lightweight DL $\mathcal{EL}$ became popular as an ontology language for large-scale ontologies. $\mathcal{EL}$ provides the logical underpinning of the OWL 2 EL profile of the W3C web ontology language [W3C OWL Working Group, 2009], which is used in important life science ontologies, as for instance, SNOMED CT [Spackman, 2000] and the thesaurus of the US national cancer institute (NCI) [Sioutos *et al.*, 2007], which contain ten thousands of concepts. The reason for the success of $\mathcal{EL}$ is that it offers limited, but sufficient expressive power, while reasoning can still be done in polynomial time [Baader *et al.*, 2005].

In DLs basic categories from an application domain can be captured by *concepts* and binary relations by *roles*. Implications between concepts can be specified in the so-called *TBox*. A *general TBox* allows complex concepts on both sides of implications. Facts from the application domain can be captured by *individuals* and their relations in the *ABox*.

*Partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing".

Classical inferences for DLs are *subsumption*, which computes the sub- and super-concept relationships of named concepts and *instance checking*, which determines for a given individual whether it belongs to a given concept. Reasoning support for the design and maintenance of large ontologies can be provided by the *bottom-up approach*, which allows to derive a new concept from a set of example individuals, see [Baader *et al.*, 1999]. For this kind of task the generalization inferences *least common subsumer* (lcs) and *most specific concept* (msc) are investigated for lightweight DLs like $\mathcal{EL}$. The lcs of a collection of concepts is a complex concept that captures all commonalities of these concepts. The msc generalizes an individual into a complex concept, that is the most specific one (w.r.t. subsumption) of which the individual is an instance of.

Unfortunately, neither the lcs nor the msc need to exist, if computed w.r.t. general $\mathcal{EL}$-TBoxes [Baader, 2003] or cyclic ABoxes written in $\mathcal{EL}$ [Küsters and Molitor, 2002]. Let's consider the TBox statements:

$$\begin{aligned}
\mathsf{Penicillin} &\sqsubseteq \mathsf{Antibiotic} \sqcap \exists \mathsf{kills.S\text{-}aureus},\\
\mathsf{Carbapenem} &\sqsubseteq \mathsf{Antibiotic} \sqcap \exists \mathsf{kills.E\text{-}coli},\\
\mathsf{S\text{-}aureus} &\sqsubseteq \mathsf{Bacterium} \sqcap \exists \mathsf{resistantMutant.Penicillin},\\
\mathsf{E\text{-}coli} &\sqsubseteq \mathsf{Bacterium} \sqcap \exists \mathsf{resistantMutant.Carbapenem}
\end{aligned}$$

We want to compute the lcs of Penicillin and Carbapenem. Now, both concepts are defined by the type of bacterium they kill. These, in turn, are defined by the substance a mutant of theirs is resistant to. This leads to a cyclic definition and thus the common subsumer cannot be captured by a finite $\mathcal{EL}$-concept, since this would need to express the cycle. If computed w.r.t. a TBox that extends the above one by the axioms:

$$\begin{aligned}
\mathsf{Antibiotic} &\sqsubseteq \exists \mathsf{kills.Bacterium},\\
\mathsf{Bacterium} &\sqsubseteq \exists \mathsf{resistantMutant.Antibiotic},
\end{aligned}$$

then the lcs of Penicillin and Carbapenem is just Antibiotic. We can observe that the existence of the lcs does not merely depend on whether the TBox is cyclic. In fact, for cyclic $\mathcal{EL}$-TBoxes exact conditions for the existence of the lcs have been devised [Baader, 2004]. However, for the case of general $\mathcal{EL}$-TBoxes such conditions are unknown.

There are several approaches to compute generalizations even in this setting. In [Lutz *et al.*, 2010] an extension of

$\mathcal{EL}$ with greatest fixpoints was introduced, where the generalization concepts always exist. Computation algorithms for approximative solutions for the lcs were devised in [Baader *et al.*, 2007; Peñaloza and Turhan, 2011b] and for the msc in [Küsters and Molitor, 2002]. The last two methods simply compute the generalization concept up to a given k, a bound on the maximal nestings of quantifiers. If the lcs or msc exists and a large enough k was given, then these methods yield the exact solutions. However, to obtain the *least* common subsumer and the *most* specific concept by these methods in practice, a decision procedure for the existence of the lcs or msc, resp., and a method for computing a sufficient k are still needed. This paper provides these methods for the lcs and the msc.

In this paper we first introduce basic notions for the DL $\mathcal{EL}$ and its canonical models, which serve as a basis for the characterization of the lcs introduced in the subsequent section. There we show that the characterization can be used to verify whether a given generalization is the most specific one and that the size of the lcs, if it exists, is polynomially bounded in the size of the input, which yields a decision procedure for the existence problem. In Section 4 we show the corresponding results for the msc. We end with some conclusions.

2 Preliminaries

2.1 The Description Logic $\mathcal{EL}$

Let N_C, N_R and N_I be disjoint sets of *concept*, *role* and *individual names*. Let $A \in N_C$ and $r \in N_R$. *$\mathcal{EL}$-concepts* are built according to the syntax rule

$$C ::= \top \mid A \mid C \sqcap D \mid \exists r.C$$

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ that assigns subsets of $\Delta^{\mathcal{I}}$ to concept names, binary relations on $\Delta^{\mathcal{I}}$ to role names and elements of $\Delta^{\mathcal{I}}$ to individual names. The function is extended to complex concepts in the usual way. For a detailed description of the semantic of DLs see [Baader *et al.*, 2003].

Let C, D denote $\mathcal{EL}$-concepts. A *general concept inclusion* (GCI) is an expression of the form $C \sqsubseteq D$. A *(general) TBox* $\mathcal{T}$ is a finite set of GCIs. A GCI $C \sqsubseteq D$ is satisfied in an interpretation $\mathcal{I}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies all GCIs in $\mathcal{T}$.

Let $a, b \in N_I$, $r \in N_R$ and C a concept, then $C(a)$ is a *concept assertion* and $r(a, b)$ a *role assertion*. An interpretation $\mathcal{I}$ satisfies an assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ holds. An *ABox* $\mathcal{A}$ is a finite set of assertions. An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ if it satisfies all assertions in $\mathcal{A}$. A *knowledge base* (KB) $\mathcal{K}$ consists of a TBox and an ABox ($\mathcal{K} = (\mathcal{T}, \mathcal{A})$). An interpretation is a model of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$.[1]

Important reasoning tasks considered for DLs are *subsumption* and *instance checking*. A concept C *is subsumed by* a concept D w.r.t. a TBox $\mathcal{T}$ (denoted $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in all models $\mathcal{I}$ of $\mathcal{T}$. A concept C is *equivalent* to a concept D w.r.t. a TBox $\mathcal{T}$ (denoted $C \equiv_{\mathcal{T}} D$) if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$ hold. A reasoning service dealing with a KB is instance checking. An individual a *is instance of* the concept C w.r.t. $\mathcal{K}$ (denoted $\mathcal{K} \models C(a)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds in all models $\mathcal{I}$ of $\mathcal{K}$. These two reasoning problems can be decided for $\mathcal{EL}$ in polynomial time [Baader *et al.*, 2005].

Based on subsumption and instance checking our two inferences of interest *least common subsumer* (lcs) and *most specific concept* (msc) are defined.

Definition 1. Let C, D be concepts and $\mathcal{T}$ a TBox. The concept E is the *lcs of C, D w.r.t. $\mathcal{T}$* ($\text{lcs}_{\mathcal{T}}(C, D)$) if the properties

1. $C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$, and
2. $C \sqsubseteq_{\mathcal{T}} F$ and $D \sqsubseteq_{\mathcal{T}} F$ implies $E \sqsubseteq_{\mathcal{T}} F$.

are satisfied. If a concept E satisfies Property 1 it is a *common subsumer of C and D w.r.t. $\mathcal{T}$*.

The lcs is unique up to equivalence, while common subsumers are not unique, thus we write $G \in \text{cs}_{\mathcal{T}}(C, D)$.

The *role depth* $rd(C)$ of a concept C denotes the maximal nesting depth of $\exists$ in C. If, in Definition 1 the concepts E and F have a role-depth up to k, then E is the *role-depth bounded lcs* (k-$\text{lcs}_{\mathcal{T}}(C, D)$) of C and D w.r.t. $\mathcal{T}$.

$N_{I,\mathcal{A}}$ is the set of individual names used in an ABox $\mathcal{A}$.

Definition 2. Let $a \in N_{I,\mathcal{A}}$ and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a KB. A concept C is the *most specific concept of a w.r.t. $\mathcal{K}$* ($\text{msc}_{\mathcal{K}}(a)$) if it satisfies:

1. $\mathcal{K} \models C(a)$, and 2. $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.

If in the last definition the concepts C and D have a role-depth limited to k, then C is the *role depth bounded msc* of a w.r.t. $\mathcal{K}$ (k-$\text{msc}_{\mathcal{K}}(a)$). The msc and the k-msc are unique up to equivalence in $\mathcal{EL}$.

2.2 Canonical Models and Simulation Relations

The correctness proof of the computation algorithms for the lcs and msc depends on the characterization of subsumption and instance checking, respectively. In case of an empty TBox, homomorphisms between syntax trees of concepts [Baader *et al.*, 1999] were used. A characterization w.r.t. general TBoxes using *canonical models* and *simulations* was given in [Lutz and Wolter, 2010], which we want to use in the following.

Let X be a concept, TBox, ABox or KB, then $N_{C,X}$ ($N_{R,X}$) denotes the set of concept names (role names) occurring in X and $\mathsf{sub}(X)$ denotes the subconcepts in X.

Definition 3. Let C be a concept and $\mathcal{T}$ a TBox.
The *canonical model $\mathcal{I}_{C,\mathcal{T}}$ of C and $\mathcal{T}$* is defined as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_{C,\mathcal{T}}} &:= \{d_C\} \cup \{d_D \mid \exists r.D \in \mathsf{sub}(C) \cup \mathsf{sub}(\mathcal{T})\};\\
A^{\mathcal{I}_{C,\mathcal{T}}} &:= \{d_D \mid D \sqsubseteq_{\mathcal{T}} A\}, \text{ for all } A \in N_{C,\mathcal{T}}\\
r^{\mathcal{I}_{C,\mathcal{T}}} &:= \{(d_D, d_E) \mid D \sqsubseteq_{\mathcal{T}} \exists r.E \text{ for } \exists r.E \in \mathsf{sub}(\mathcal{T})\\
&\qquad \text{or } \exists r.E \text{ is conjunct in } D\}, \text{ for all } r \in N_{R,\mathcal{T}}.
\end{aligned}$$

The notion of a canonical model can be extended to a KB.

[1] Since we only use the DL $\mathcal{EL}$, we write 'concept' instead of '$\mathcal{EL}$-concept' and assume all TBoxes, ABoxes and KBs to be written in $\mathcal{EL}$ in the following.

Definition 4. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB. The *canonical model* $\mathcal{I}_\mathcal{K}$ *w.r.t.* $\mathcal{K}$ is defined as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_\mathcal{K}} &:= \{d_a \mid a \in N_{I,\mathcal{A}}\} \cup \{d_C \mid \exists r.C \in \mathsf{sub}(\mathcal{K})\} \\
A^{\mathcal{I}_\mathcal{K}} &:= \{d_a \mid \mathcal{K} \models A(a)\} \cup \{d_C \mid C \sqsubseteq_\mathcal{T} A\}, \\
&\quad \text{for all } A \in N_{C,\mathcal{K}}; \\
r^{\mathcal{I}_\mathcal{K}} &:= \{(d_C, d_D) \mid C \sqsubseteq_\mathcal{T} \exists r.D, \exists r.D \in \mathsf{sub}(\mathcal{T})\} \\
&\quad \cup \{(d_a, d_b) \mid r(a,b) \in \mathcal{A}\}, \text{ for all } r \in N_{R,\mathcal{K}}; \\
a^{\mathcal{I}_\mathcal{K}} &:= d_a, \text{ for all } a \in N_{I,\mathcal{A}}.
\end{aligned}$$

To identify some properties of canonical models we use *simulation relations* between interpretations.

Definition 5. Let $\mathcal{I}_1, \mathcal{I}_2$ be interpretations. $\mathcal{S} \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ is a *simulation between* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ if the following conditions are satisfied for all $A \in N_C$ and for all $r \in N_R$:

(*S*1) If $(e_1, e_2) \in \mathcal{S}$ and $e_1 \in A^{\mathcal{I}_1}$, then $e_2 \in A^{\mathcal{I}_2}$.

(*S*2) If $(e_1, e_2) \in \mathcal{S}$ and $(e_1, e_1') \in r^{\mathcal{I}_1}$, then there exists $e_2' \in \Delta^{\mathcal{I}_2}$ s.t. $(e_2, e_2') \in r^{\mathcal{I}_2}$ and $(e_1', e_2') \in \mathcal{S}$.

The tuple $(\mathcal{I}, d)$ denotes an interpretation $\mathcal{I}$ with $d \in \Delta^\mathcal{I}$. If there exists a simulation $\mathcal{S} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{J}$ with $(d, e) \in \mathcal{S}$, we write $(\mathcal{I}, d) \lesssim (\mathcal{J}, e)$ and say $(\mathcal{J}, e)$ *simulates* $(\mathcal{I}, d)$. We write $(\mathcal{I}, d) \simeq (\mathcal{J}, e)$ if $(\mathcal{I}, d) \lesssim (\mathcal{J}, e)$ and $(\mathcal{J}, e) \lesssim (\mathcal{I}, d)$ holds. We summarize some important properties of canonical models.

Lemma 6. *[Lutz and Wolter, 2010] Let C be a concept and $\mathcal{T}$ a TBox.*

1. *$\mathcal{I}_{C,\mathcal{T}}$ is a model of $\mathcal{T}$.*
2. *For all models $\mathcal{I}$ of $\mathcal{T}$ and all $d \in \Delta^\mathcal{I}$ holds: $d \in C^\mathcal{I}$ iff $(\mathcal{I}_{C,\mathcal{T}}, d_C) \lesssim (\mathcal{I}, d)$.*
3. *$C \sqsubseteq_\mathcal{T} D$ iff $d_C \in D^{\mathcal{I}_{C,\mathcal{T}}}$ iff $(\mathcal{I}_{D,\mathcal{T}}, d_D) \lesssim (\mathcal{I}_{C,\mathcal{T}}, d_C)$.*

This Lemma gives us a characterization of subsumption. A similar Lemma was shown for the instance relationship.

Lemma 7. *[Lutz and Wolter, 2010] Let $\mathcal{K}$ be a KB. $\mathcal{I}_\mathcal{K}$ satisfies: 1. $\mathcal{I}_\mathcal{K}$ is a model of $\mathcal{K}$. 2. $\mathcal{K} \models C(a)$ iff $d_a \in C^{\mathcal{I}_\mathcal{K}}$.*

Next, we recall some known operations on interpretations. Taking an element d of the domain of an interpretation as the root, the interpretation can be unraveled into a possibly infinite tree. The nodes of the tree are words that correspond to paths starting in d. We have that $\pi = dr_1d_1r_2d_2r_3...$ is a *path* in an interpretation $\mathcal{I}$, if the domain elements d_i and d_{i+1} are connected via $r_{i+1}^\mathcal{I}$ for all i.

Definition 8. Let $\mathcal{I}$ be an interpretation with $d \in \Delta^\mathcal{I}$. The *tree unraveling* $\mathcal{I}_d$ of $\mathcal{I}$ in d is defined as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_d} &:= \{dr_1d_1r_2...r_nd_n \mid (d_i, d_{i+1}) \in r_{i+1}^\mathcal{I}, i \geq 0, d_0 = d\} \\
A^{\mathcal{I}_d} &:= \{\sigma d' \mid \sigma d' \in \Delta^{\mathcal{I}_d} \wedge d' \in A^\mathcal{I}\} \\
r^{\mathcal{I}_d} &:= \{(\sigma, \sigma r d') \mid (\sigma, \sigma r d') \in \Delta^{\mathcal{I}_d} \times \Delta^{\mathcal{I}_d}\}.
\end{aligned}$$

The *length* of an element $\sigma \in \Delta^{\mathcal{I}_d}$, denoted by $|\sigma|$, is the number of role names occurring in σ. If σ is of the form $dr_1d_1r_2...r_md_m$, then d_m is the *tail* of σ denoted by $\mathsf{tail}(\sigma) = d_m$. The interpretation $\mathcal{I}_d^\ell$ denotes the finite subtree of the tree unraveling $\mathcal{I}_d$ up to depth ℓ. Such a tree can be translated into an ℓ*-characteristic concept* of an interpretation $(\mathcal{I}, d)$.

Definition 9. Let $(\mathcal{I}, d)$ be an interpretation. The ℓ-characteristic concept $X^\ell(\mathcal{I}, d)$ is defined as follows:

- $X^0(\mathcal{I}, d) := \bigsqcap \{A \in N_C \mid d \in A^\mathcal{I}\}$
- $X^\ell(\mathcal{I}, d) :=$
$$X^0(\mathcal{I}, d) \sqcap \bigsqcap_{r \in N_R} \bigsqcap \{\exists r.X^{\ell-1}(\mathcal{I}, d') \mid (d, d') \in r^\mathcal{I}\}$$

3 Existence of Least Common Subsumers

In this section we develop a decision procedure for the problem whether for two given concepts and a given TBox the least common subsumer of these two concepts exists w.r.t. the given TBox. If not stated otherwise, the two input concepts are denoted by C and D and the TBox by $\mathcal{T}$.

Similar to the approach used in [Baader, 2004] we proceed by the following steps:

1. Devise a method to identify lcs-candidates for the lcs. The set of lcs-candidates is a possibly infinite set of common subsumers of C and D w.r.t. $\mathcal{T}$, such that if the lcs exists then one of these lcs-candidates actually is the lcs.

2. Characterize the existence of the lcs. Find a condition such that the problem whether a given common subsumer of C and D w.r.t. $\mathcal{T}$ is least (w.r.t. $\sqsubseteq_\mathcal{T}$), can be decided by testing this condition.

3. Establish an upper bound on the role-depth of the lcs. We give a bound ℓ such that *if the lcs exists*, then it has a role-depth less or equal ℓ. By such an upper bound one needs to check only for finitely many of the lcs-candidates if they are least (w.r.t. $\sqsubseteq_\mathcal{T}$).

The next subsection addresses the first two problems, afterwards we show that such a desired upper bound exists.

3.1 Characterizing the Existence of the lcs

The characterization presented here is based on the product of canonical models. This product construction is adopted from [Baader, 2003; Lutz *et al.*, 2010] where it was used to compute the lcs in $\mathcal{EL}$ with gfp-semantics and in the DL $\mathcal{EL}^\nu$, respectively.

To obtain the k-lcs$_\mathcal{T}(C, D)$ we build the product of the canonical models $(\mathcal{I}_{C,\mathcal{T}}, d_C)$ and $(\mathcal{I}_{D,\mathcal{T}}, d_D)$ and then take the k-characteristic concept of this product model.

Lemma 10. *Let $k \in \mathbb{N}$.*

1. *$X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \in cs_\mathcal{T}(C, D)$.*
2. *Let $E \in cs_\mathcal{T}(C, D)$ with $rd(E) \leq k$. It holds that $X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \sqsubseteq_\mathcal{T} E$.*

This and all the proofs omitted in this paper due to space constraints can be found in [Zarrieß and Turhan, 2013].

In the following we take $X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ as a representation of the k-lcs$_\mathcal{T}(C, D)$. It is implied by Lemma 10 that the set of k-characteristic concepts of the product model $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ for all k is the set of lcs-candidates for the lcs$_\mathcal{T}(C, D)$, which can be stated as follows.

Corollary 11. *The lcs$_\mathcal{T}(C, D)$ exists iff there exists a $k \in \mathbb{N}$ such that for all $\ell \in \mathbb{N}$: k-lcs$_\mathcal{T}(C, D) \sqsubseteq_\mathcal{T} \ell$-lcs$_\mathcal{T}(C, D)$.*

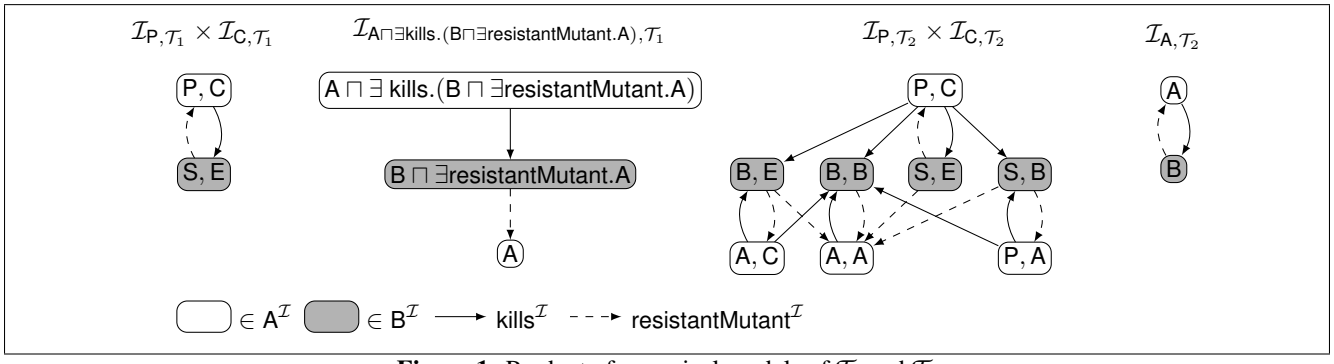

Figure 1: Product of canonical models of $\mathcal{T}_1$ and $\mathcal{T}_2$

Obviously, this doesn't yield a decision procedure for the problem whether the k-$\mathrm{lcs}_{\mathcal{T}}(C, D)$ is the lcs, since subsumption cannot be checked for infinitely many ℓ in finite time.

Next, we address step 2 and show a condition on the common subsumers that decides whether a common subsumer is least or not. The main idea is that the product model captures all commonalities of the input concepts by means of canonical models. Thus we compare the canonical models of the common subsumers and the product model using simulation-equivalence $\simeq$.

Lemma 12. *Let E be a concept. $E \equiv_{\mathcal{T}} lcs_{\mathcal{T}}(C, D)$ iff $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \simeq (\mathcal{I}_{E,\mathcal{T}}, d_E)$.*

Proof sketch. For any $F \in \mathrm{cs}_{\mathcal{T}}(C, D)$ it holds by Lemma 6, Claim 3 that $(\mathcal{I}_{F,\mathcal{T}}, d_F)$ is simulated by $(\mathcal{I}_{C,\mathcal{T}}, d_C)$ and $(\mathcal{I}_{D,\mathcal{T}}, d_D)$ and therefore also by $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$.

Assume $(\mathcal{I}_{E,\mathcal{T}}, d_E)$ is simulation-equivalent to the product model. We need to show that $E \equiv_{\mathcal{T}} \mathrm{lcs}_{\mathcal{T}}(C, D)$. By transitivity of $\lesssim$ it is implied that $(\mathcal{I}_{F,\mathcal{T}}, d_F) \lesssim (\mathcal{I}_{E,\mathcal{T}}, d_E)$ and $E \sqsubseteq_{\mathcal{T}} F$ by Lemma 6. Therefore $E \equiv_{\mathcal{T}} \mathrm{lcs}_{\mathcal{T}}(C, D)$.

For the other direction assume $E \equiv_{\mathcal{T}} \mathrm{lcs}_{\mathcal{T}}(C, D)$. It has to be shown that $(\mathcal{I}_{E,\mathcal{T}}, d_E)$ simulates the product model. Let $\mathcal{J}_{(d_C,d_D)}$ be the tree unraveling of the product model. Since E is more specific than the k-characteristic concepts of the product model for all k (by Corollary 11), $(\mathcal{I}_{E,\mathcal{T}}, d_E)$ simulates the subtree $\mathcal{J}^k_{(d_C,d_D)}$ of $\mathcal{J}_{(d_C,d_D)}$ limited to elements up to depth k, for all k. For each k we consider the maximal simulation from $\mathcal{J}^k_{(d_C,d_D)}$ to $(\mathcal{I}_{E,\mathcal{T}}, d_E)$. Note that $((d_C, d_D), d_E)$ is contained in any of these simulations. Let σ be an element of $\Delta^{\mathcal{J}_{(d_C,d_D)}}$ at an arbitrary depth ℓ. We show how to determine the elements of $\Delta^{\mathcal{I}_{E,\mathcal{T}}}$, that simulate this fixed element σ. Let $\mathcal{S}_n(\sigma)$ be the maximal set of elements from $\Delta^{\mathcal{I}_{E,\mathcal{T}}}$ that simulate σ in each of the trees $\mathcal{J}^n_{(d_C,d_D)}$ with $n \geq \ell$. We can observe that the infinite sequence $(\mathcal{S}_{\ell+i}(\sigma))_{i=0,1,2,\ldots}$ is decreasing (w.r.t. $\supseteq$). Therefore at a certain depth we reach a fixpoint set. This fixpoint set exists for any σ. It can be shown that the union of all these fixpoint sets yields a simulation from the product model to $(\mathcal{I}_{E,\mathcal{T}}, d_E)$. □

By the use of this Lemma it can be verified whether a given common subsumer is the least one or not, which we illustrate by an example.

Example 13. Consider again the TBox from the introduction (now displayed with abbreviated concept names)

$$\mathcal{T}_1 = \{\mathsf{P} \sqsubseteq \mathsf{A} \sqcap \exists\mathsf{kills}.\mathsf{S}, \quad \mathsf{S} \sqsubseteq \mathsf{B} \sqcap \exists\mathsf{resistantMutant}.\mathsf{P},$$
$$\mathsf{C} \sqsubseteq \mathsf{A} \sqcap \exists\mathsf{kills}.\mathsf{E}, \quad \mathsf{E} \sqsubseteq \mathsf{B} \sqcap \exists\mathsf{resistantMutant}.\mathsf{C}\}$$

and the following extended TBox

$$\mathcal{T}_2 = \mathcal{T}_1 \cup \{\mathsf{A} \sqsubseteq \exists\mathsf{kills}.\mathsf{B}, \ \ \mathsf{B} \sqsubseteq \exists\mathsf{resistantMutant}.\mathsf{A}\}.$$

In Figure 1 we can see that

$$\mathsf{A} \sqcap \exists\mathsf{kills}.(\mathsf{B} \sqcap \exists\mathsf{resistantMutant}.\mathsf{A}) \in \mathrm{cs}_{\mathcal{T}_1}(\mathsf{P}, \mathsf{C}),$$

but it is not the lcs, because its canonical model cannot simulate the product model $(\mathcal{I}_{\mathsf{P},\mathcal{T}_1} \times \mathcal{I}_{\mathsf{C},\mathcal{T}_1}, (d_\mathsf{P}, d_\mathsf{C}))$. The concept A, however, is the lcs of P and C w.r.t. $\mathcal{T}_2$. We have $(\mathcal{I}_{\mathsf{P},\mathcal{T}_2} \times \mathcal{I}_{\mathsf{C},\mathcal{T}_2}, (d_\mathsf{P}, d_\mathsf{C})) \lesssim (\mathcal{I}_{\mathsf{A},\mathcal{T}_2}, d_\mathsf{A})$ since any element from $\Delta^{\mathcal{I}_{\mathsf{P},\mathcal{T}_2} \times \mathcal{I}_{\mathsf{C},\mathcal{T}_2}}$ in $\mathsf{A}^{\mathcal{I}_{\mathsf{P},\mathcal{T}_2} \times \mathcal{I}_{\mathsf{C},\mathcal{T}_2}}$ or $\mathsf{B}^{\mathcal{I}_{\mathsf{P},\mathcal{T}_2} \times \mathcal{I}_{\mathsf{C},\mathcal{T}_2}}$ is simulated by Ⓐ or Ⓑ, respectively.

The characterization of the existence of the lcs given in Corollary 11 can be reformulated using Lemma 12.

Corollary 14. *The $lcs_{\mathcal{T}}(C, D)$ exists iff there exists a k such that the canonical model of $X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ w.r.t. $\mathcal{T}$ simulates $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$.*

This corollary still doesn't yield a decision procedure for the existence problem, since the depth k is still unrestricted. Such a restriction will be developed in the next section.

3.2 A Polynomial Upper Bound on the Role-depth of the lcs

In this section we show that, if the lcs exists, its role-depth is bounded by the size of the product model. First, consider again the TBox $\mathcal{T}_2$ from Example 13, where $\mathsf{A} \sqsubseteq_{\mathcal{T}_2} \exists\mathsf{kills}.(\mathsf{B} \sqcap \exists\mathsf{resistantMutant}.\mathsf{A})$ holds, which results in a loop in the product model through the elements $\overline{\mathsf{A}, \mathsf{A}}$ and $\overline{\mathsf{B}, \mathsf{B}}$. Furthermore, the cycles in the product model involving the roles kills and resistantMutant are captured by the canonical model $\mathcal{I}_{\mathsf{A},\mathcal{T}_2}$. Therefore $\mathsf{A} \equiv_{\mathcal{T}_2} \mathrm{lcs}_{\mathcal{T}_2}(\mathsf{P}, \mathsf{C})$. On this observation we build our general method.

We call elements $(d_F, d_{F'}) \in \Delta^{\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}}$ *synchronous* if $F = F'$ and *asynchronous* otherwise. The structure of $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ can now be simplified by considering only synchronous successors of synchronous elements.

In order to find a number k, such that the product model is simulated by the canonical model of

$K = X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$, we first represent the model $(\mathcal{I}_{K,\mathcal{T}}, d_K)$ as a subtree of the tree unraveling of the product model $\mathcal{J}_{(d_C,d_D)}$ with root (d_C, d_D). We construct this representation by extending the subtree $\mathcal{J}^k_{(d_C,d_D)}$ by new tree models at depth k. We need to ensure that the resulting interpretation, denoted by $\widehat{\mathcal{J}}^k_{(d_C,d_D)}$, is a model of $\mathcal{T}$, that is simulation-equivalent to $(\mathcal{I}_{K,\mathcal{T}}, d_K)$. The elements $\sigma \in \Delta^{\mathcal{J}^k_{(d_C,d_D)}}$ with $|\sigma| = k$ that we extend and the corresponding trees we append to them are selected as follows: Let M be a conjunction of concept names and $\exists r.F \in \mathsf{sub}(\mathcal{T})$. If $\sigma \in M^{\mathcal{J}^k_{(d_C,d_D)}}$ and $M \sqsubseteq_{\mathcal{T}} \exists r.F$, then we append the tree unraveling of the canonical model $\mathcal{I}_{\exists r.F,\mathcal{T}}$. Furthermore, we consider elements that have a tail that is a synchronous element. If $\mathsf{tail}(\sigma) = (d_F, d_F)$, then F is called *tail concept* of σ. To select the elements with a synchronous tail, that we extend by the canonical model of their tail concept, we use embeddings of $\mathcal{J}^k_{(d_C,d_D)}$ into $(\mathcal{I}_{K,\mathcal{T}}, d_K)$. Let $\mathcal{H} = \{Z_1, ..., Z_n\}$ be the set of all functional simulations Z_i from $\mathcal{J}^k_{(d_C,d_D)}$ to $(\mathcal{I}_{K,\mathcal{T}}, d_K)$ with $Z_i((d_C, d_D)) = d_K$. We say that σ with tail concept F is *matched* by Z_i if $Z_i(\sigma) \in F^{\mathcal{I}_{K,\mathcal{T}}}$. The set of elements $\sigma \in \Delta^{\mathcal{J}^k_{(d_C,d_D)}}$ with $|\sigma| = k$, that are matched by a functional simulation Z_i is called *matching set*, denoted by $\mathcal{M}(Z_i)$. Now consider the set $\mathcal{M}(\mathcal{H}) := \{\mathcal{M}(Z_1), ..., \mathcal{M}(Z_n)\}$. If σ is contained in *all* maximal matching sets from $\mathcal{M}(\mathcal{H})$, then we extend σ by the tree unraveling of the canonical model of its tail concept w.r.t. $\mathcal{T}$.

We can show that the resulting interpretation $\widehat{\mathcal{J}}^k_{(d_C,d_D)}$ has the desired properties.

Lemma 15. *Let $K = X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$. $\widehat{\mathcal{J}}^k_{(d_C,d_D)}$ is a model of $\mathcal{T}$ and $\widehat{\mathcal{J}}^k_{(d_C,d_D)} \simeq (\mathcal{I}_{K,\mathcal{T}}, d_K)$.*

Having this representation of the canonical model of the k-lcs$_\mathcal{T}(C, D)$ we first show a sufficient condition for the existence of the lcs.

Corollary 16. *If all cycles in $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$, that are reachable from (d_C, d_D) consist of synchronous elements, then the lcs$_\mathcal{T}(C, D)$ exists.*

Proof sketch. There exists an $\ell \in \mathbb{N}$ such that all paths in the tree unraveling $\mathcal{J}_{(d_C,d_D)}$ of $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ starting in (d_C, d_D) have a maximal asynchronous prefix up to length ℓ, i.e., if there exists an element at depth $\geq \ell + 1$, then it is a synchronous element. Consider the number

$$m := \mathsf{max}(\{rd(F) \mid F \in \mathsf{sub}(\mathcal{T}) \cup \{C, D\}\}).$$

We unravel $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ up to depth $\ell + m + 1$ such that we get $\mathcal{J}^{\ell+m+1}_{(d_C,d_D)}$. Now it is ensured that the corresponding model $\widehat{\mathcal{J}}^{\ell+m+1}_{(d_C,d_D)}$ contains all paths with a maximal asynchronous prefix up to length ℓ. It is implied that $\widehat{\mathcal{J}}^{\ell+m+1}_{(d_C,d_D)} = \mathcal{J}_{(d_C,d_D)}$. From Lemma 15 and Corollary 14 it follows that $X^{\ell+m+1}(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ is the lcs. □

As seen in Example 13 for $\mathcal{T}_2$, this is not a necessary condition for the existence of the lcs.

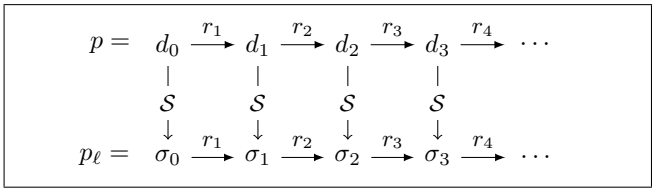

Figure 2: simulation chain of p and p_ℓ

Another consequence of Lemma 15 is, that if the product model $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ has only asynchronous cycles reachable from (d_C, d_D), then the lcs$_\mathcal{T}(C, D)$ does not exist. Since in this case $\mathcal{J}_{(d_C,d_D)}$ is infinite but $\widehat{\mathcal{J}}^k_{(d_C,d_D)}$ is finite for all $k \in \mathbb{N}$, a simulation from $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ to $\widehat{\mathcal{J}}^k_{(d_C,d_D)}$ never exists for all k. For instance, this case applies to Example 13 w.r.t. to $\mathcal{T}_1$.

The interesting case is where we have both asynchronous and synchronous cycles reachable from (d_C, d_D) in the product model. In this case we choose a k that is large enough and then check whether the canonical model of $X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ w.r.t. $\mathcal{T}$ simulates the product model.

We show in the next Lemma that the role-depth of the lcs$_\mathcal{T}(C, D)$, if it exists, can be bounded by a polynomial, that is quadratic in the size of the product model.

Lemma 17. *Let $n := |\Delta^{\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}}|$ and $m := \mathsf{max}(\{rd(F) \mid F \in \mathsf{sub}(\mathcal{T}) \cup \{C, D\}\})$. If lcs$_\mathcal{T}(C, D)$ exists then $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \lesssim \widehat{\mathcal{J}}^{n^2+m+1}_{(d_C,d_D)}$.*

Proof sketch. Assume lcs$_\mathcal{T}(C, D)$ exists. From Corollary 14 and Lemma 15 it follows that there exists a number ℓ such that

$$(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \lesssim \widehat{\mathcal{J}}^{\ell}_{(d_C,d_D)}. \quad (1)$$

Every path in $\widehat{\mathcal{J}}^{\ell}_{(d_C,d_D)}$ has a maximal asynchronous prefix of length $\leq \ell$. From depth $\ell + 1$ on there are only synchronous elements in the tree $\widehat{\mathcal{J}}^{\ell}_{(d_C,d_D)}$. From (1) it follows that every path p in $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ starting in (d_C, d_D), is simulated by a corresponding path p_ℓ in $\widehat{\mathcal{J}}^{\ell}_{(d_C,d_D)}$ also starting in (d_C, d_D). The *simulation chain* of p and p_ℓ is depicted in Figure 2. The idea is to use the simulating path p_ℓ to construct a simulating path in $\widehat{\mathcal{J}}^{\ell}_{(d_C,d_D)}$ (also starting in (d_C, d_D)) with a maximal asynchronous prefix of length $\leq n^2$, where n^2 is the number of pairs of elements from $\Delta^{\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}}$. Intuitively, if p_ℓ has a maximal asynchronous prefix that is longer than n^2, then there are pairs in the simulation chain that occur more than once. This is used to construct a simulating path with a shorter maximal asynchronous prefix step-wise. After a finite number of steps the result is a simulating path, such that all pairs consisting of asynchronous elements in the corresponding simulation chain are pairwise distinct. Therefore we need only asynchronous elements from $\widehat{\mathcal{J}}^{\ell}_{(d_C,d_D)}$ up to depth n^2 to simulate the product model. Then we add $m+1$ to n^2 to ensure that $\widehat{\mathcal{J}}^{n^2+m+1}_{(d_C,d_D)}$ contains *all* paths from $\mathcal{J}_{(d_C,d_D)}$ starting in (d_C, d_D), that have a maximal asynchronous pre-

fix of length $\leq n^2$. As argued above $\widehat{\mathcal{J}}^{n^2+m+1}_{(d_C,d_D)}$ simulates $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$. □

Using Lemma 12 and Lemma 17 we can now show the main result of this paper.

Theorem 18. *Let C, D be concepts and $\mathcal{T}$ a general TBox. It is decidable in polynomial time whether the $\text{lcs}_\mathcal{T}(C, D)$ exists. If the $\text{lcs}_\mathcal{T}(C, D)$ exists it can be computed in polynomial time.*

Proof. First we compute the bound k as given in Lemma 17 and then the k-characteristic concept K of $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$. The canonical model of K can be build according to Definition 3 in polynomial time [Baader *et al.*, 2005]. Next we check whether $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \lesssim (\mathcal{I}_{K,\mathcal{T}}, d_K)$ holds, which can be done in polynomial time. If yes, K is the lcs by Lemma 12 and if no, the lcs doesn't exist by Lemma 17. □

The results from this section can be easily generalized to the lcs of an arbitrary set of concepts $M = \{C_1, ..., C_m\}$ w.r.t. a TBox $\mathcal{T}$. But in this case the size of the lcs is already exponential w.r.t. an empty TBox [Baader *et al.*, 1999]. In this general case we have to take the product model

$$(\mathcal{I}_{C_1,\mathcal{T}} \times \cdots \times \mathcal{I}_{C_m,\mathcal{T}}, (d_{C_1}, \cdots, d_{C_m})),$$

whose size is exponential in the size of M and $\mathcal{T}$, as input for the methods introduced in this section. Then the same steps as for the binary version can be applied.

4 Existence of Most Specific Concepts

We show now that the results obtained for the lcs, can be easily applied to the existence problem of the msc.

Example 19 (From [Küsters and Molitor, 2002])**.** The msc of the individual a w.r.t. the following KB

$$\mathcal{K}_1 = (\emptyset, \mathcal{A}_1), \text{ with } \mathcal{A}_1 = \{r(a, a)\}$$

doesn't exist, whereas w.r.t. the modified KB

$$\mathcal{K}_2 = (\{C \sqsubseteq \exists r.C\}, \mathcal{A}_2), \text{ with } \mathcal{A}_2 = \mathcal{A}_1 \cup \{C(a)\}$$

C is the msc of a.

To decide existence of the msc of an individual a w.r.t. a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we again start with defining the set of msc-candidates for the msc by taking the k-characteristic concept of the canonical model $(\mathcal{I}_\mathcal{K}, d_a)$ of $\mathcal{K}$.

Lemma 20. *Let $k \in \mathbb{N}$. It holds that $\mathcal{K} \models X^k(\mathcal{I}_\mathcal{K}, d_a)(a)$ and for a concept E with $rd(E) \leq k$, $\mathcal{K} \models E(a)$ implies $X^k(\mathcal{I}_\mathcal{K}, d_a) \sqsubseteq_\mathcal{T} E$.*

Therefore $X^k(\mathcal{I}_\mathcal{K}, d_a) \equiv_\mathcal{T} k\text{-msc}_\mathcal{K}(a)$. Now we use the canonical model of $X^k(\mathcal{I}_\mathcal{K}, d_a)$ w.r.t. the TBox component $\mathcal{T}$ of $\mathcal{K}$ and the model $(\mathcal{I}_\mathcal{K}, d_a)$ to check whether $X^k(\mathcal{I}_\mathcal{K}, d_a)$ is the *most* specific concept.

Lemma 21. *For a concept C it holds that $C \equiv_\mathcal{T} \text{msc}_\mathcal{K}(a)$ iff $(\mathcal{I}_\mathcal{K}, d_a) \simeq (\mathcal{I}_{C,\mathcal{T}}, d_C)$.*

By this Lemma the existence of the msc can be characterized as follows.

Corollary 22. *The $\text{msc}_\mathcal{K}(a)$ exists iff there exists a k such that the canonical model of $X^k(\mathcal{I}_\mathcal{K}, d_a)$ w.r.t. $\mathcal{T}$ simulates $(\mathcal{I}_\mathcal{K}, d_a)$.*

To decide whether an appropriate k exists such that $X^k(\mathcal{I}_\mathcal{K}, d_a)$ simulates $(\mathcal{I}_\mathcal{K}, d_a)$, we further examine the structure of $(\mathcal{I}_\mathcal{K}, d_a)$. In Example 19 d_a has a self-loop in the model $(\mathcal{I}_{\mathcal{K}_1}, d_a)$, but the canonical models of $X^k(\mathcal{I}_{\mathcal{K}_1}, d_a)$ are finite for all $k \in \mathbb{N}$, because the TBox is empty. Therefore a simulation never exists. In comparison, the model $(\mathcal{I}_{\mathcal{K}_2}, d_a)$ has additionally a self-loop at d_C and the canonical models of $X^k(\mathcal{I}_{\mathcal{K}_2}, d_a)$ w.r.t. $\mathcal{T}_2$ also contain this loop.

Intuitively, in the general case, the elements in $\Delta^{\mathcal{I}_\mathcal{K}}$, that are elements in $b^{\mathcal{I}_\mathcal{K}}$ (for $b \in N_{I,\mathcal{A}}$), correspond to the asynchronous elements of the product of canonical models and the elements $d_C \in \Delta^{\mathcal{I}_\mathcal{K}}$ for some concept C, correspond to the synchronous elements. The model $(\mathcal{I}_\mathcal{K}, d_a)$ also has an analogous structure compared to the product model $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ in the sense that elements in $\Delta^{\mathcal{I}_\mathcal{K}}$, that belong to concepts only have successor elements that belong to concepts. Therefore similar arguments as presented in Section 3.2 can be used to show, that a representation of the canonical model of $X^k(\mathcal{I}_\mathcal{K}, d_a)$ as a subtree of the tree unraveling of $(\mathcal{I}_\mathcal{K}, d_a)$ can be obtained. This representation is denoted by $\widehat{\mathcal{J}}^k_{d_a}$. This model is used to show an upper bound on the role-depth k of the msc.

Lemma 23. *Let $m := \max(\{rd(F) \mid F \in \text{sub}(\mathcal{K})\})$ and $n := |N_{I,\mathcal{A}}|$. If the $\text{msc}_\mathcal{K}(a)$ exists, then $(\mathcal{I}_\mathcal{K}, d_a) \lesssim \widehat{\mathcal{J}}^{n^2+m+1}_{d_a}$.*

The results of this section can be summarized in the following theorem.

Theorem 24. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB and $a \in N_{I,\mathcal{A}}$. It is decidable in polynomial time whether the $\text{msc}_\mathcal{K}(a)$ exists. If the $\text{msc}_\mathcal{K}(a)$ exists, it can be computed in polynomial time.*

Proof sketch. First we compute the bound k as given in Lemma 23 and then the k-characteristic concept $X^k(\mathcal{I}_\mathcal{K}, d_a)$. The canonical model of $\mathcal{K}$ can be build according to Definition 4 in polynomial time [Baader *et al.*, 2005]. Then we check whether $(\mathcal{I}_\mathcal{K}, d_a) \lesssim (\mathcal{I}_{C,\mathcal{T}}, d_C)$ holds, which can be done in polynomial time. If yes, C is the msc and if no, the msc doesn't exist by Corollary 22. □

All the proofs omitted here due to space constraints are given in [Zarrieß and Turhan, 2013].

5 Conclusions

In this paper we have studied the conditions for the existence of the lcs and of the msc, if computed w.r.t. general TBoxes or cyclic ABoxes, respectively, written in the DL $\mathcal{EL}$. In this setting neither the lcs nor the msc need to exist. It was an open problem to give necessary and sufficient conditions for their existence. We showed that the existence problem of the msc and the lcs of two concepts is decidable in polynomial time. Furthermore, we showed that the role-depth of these most specific generalizations can be bounded by a polynomial. This upper bound k can be used to compute the msc or lcs, if it exists. Otherwise the computed concept can still serve as an approximation [Peñaloza and Turhan, 2011b].

References

[Baader *et al.*, 1999] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann, Los Altos.

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Baader *et al.*, 2005] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05*. Morgan-Kaufmann Publishers, 2005.

[Baader *et al.*, 2007] F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5(3):392–420, 2007.

[Baader, 2003] F. 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 Kaufman, 2003.

[Baader, 2004] F. Baader. A graph-theoretic generalization of the least common subsumer and the most specific concept in the description logic $\mathcal{EL}$. In J. Hromkovic and M. Nagl, editors, *Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2004)*, volume 3353 of *Lecture Notes in Computer Science*, pages 177–188. Springer-Verlag, 2004.

[Küsters and Molitor, 2002] R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. *AI Communications*, 15(1):47–59, 2002.

[Lutz and Wolter, 2010] C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation*, 45(2):194–228, 2010.

[Lutz *et al.*, 2010] C. Lutz, R. Piro, and F. Wolter. Enriching $\mathcal{EL}$-concepts with greatest fixpoints. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*. IOS Press, 2010.

[Peñaloza and Turhan, 2011b] R. Peñaloza and A.-Y. Turhan. A practical approach for computing generalization inferences in $\mathcal{EL}$. In M. Grobelnik and E. Simperl, editors, *Proc. of the 8th European Semantic Web Conf. (ESWC'11)*, Lecture Notes in Computer Science. Springer, 2011.

[Sioutos *et al.*, 2007] N. Sioutos, S. de Coronado, M. W. Haber, F. W. Hartel, Wen-Ling Shaiu, and Lawrence W. Wright. NCI thesaurus: A semantic model integrating cancer-related clinical and molecular information. *J. of Biomedical Informatics*, 40(1):30–43, 2007.

[Spackman, 2000] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Assoc.*, 2000. Fall Symposium Special Issue.

[W3C OWL Working Group, 2009] W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation, 27th October 2009. `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`.

[Zarrieß and Turhan, 2013] B. Zarrieß and A.-Y. Turhan. Most specific generalizations w.r.t. general $\mathcal{EL}$-TBoxes. LTCS-Report 13-06, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany, 2013. See `http://lat.inf.tu-dresden.de/research/reports.html`.

A Framework for Semantic-based Similarity Measures for $\mathcal{ELH}$-Concepts

Karsten Lehmann[1,2] and Anni-Yasmin Turhan[3⋆]

[1] Optimisation Research Group, NICTA; Karsten.Lehmann@nicta.com.au
[2] Artificial Intelligence Group, Australian National University
[3] Institute for Theoretical Computer, TU Dresden, Germany;
turhan@tcs.inf.tu-dresden.de

Abstract. Similarity measures for concepts written in Description Logics (DLs) are often devised based on the syntax of concepts or simply by adjusting them to a set of instance data. These measures do not take the semantics of the concepts into account and can thus lead to unintuitive results. It even remains unclear how these measures behave if applied to new domains or new sets of instance data.
In this paper we develop a framework for similarity measures for $\mathcal{ELH}$-concept descriptions based on the semantics of the DL $\mathcal{ELH}$. We show that our framework ensures that the measures resulting from instantiations fulfill fundamental properties, such as equivalence invariance, yet the framework provides the flexibility to adjust measures to specifics of the modelled domain.

1 Introduction

Concept similarity measures map a pair of concepts from an ontology to a value between 0 and 1 indicating how similar the concepts are. These measures are an important means to discover similar concepts in ontologies. In bio-medical ontology-based applications, for example the Gene ontology [5], they are employed to discover functional similarities of genes. Furthermore, concept similarity measures are used in ontology alignment algorithms [9].

A common approach to find and evaluate similarity measures is to have test data and to tune a similarity measure until it matches the results of a human expert. The disadvantage of this approach is that the behavior of such a measure is hard to predict when applied to new test data, or when used for ontologies modeling a different domain. As a consequence an ontology developer cannot competently decide whether a measure obtained in this way is suitable for a particular task.

Description Logics (DLs) are a family of knowledge representation formalisms with formal semantics. A good similarity measure for DL concepts should take the semantics of the underlying formalism into account, instead of assessing

⋆ Partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

similarity in a purely syntactical way. Similarity measures are often tailored for particular applications. Thus, one similarity measure will hardly meet the needs of all applications.

In [8] the intended behavior of a measure was discussed and partially captured in terms of properties. These properties were adapted from metric spaces which are related to similarity measures. We follow this approach to address the problems mentioned above. We extend this set of properties by including DL specific ones and mathematically describe those from [8] in terms of DL. The formalization of the properties allows us to prove whether or not an obtained measure has the desired properties. Additionally, we investigate existing DL similarity measures to determine which of the properties they fulfill. We then propose the framework *simi* for similarity measures for $\mathcal{ELH}$-concepts. If instantiated with the right functions and operators as building blocks, *simi* yields measures for which (most of) the formalized properties can be guaranteed. At the same time the framework retains flexibility as it allows users to choose from the list which properties the resulting measure should have and to build their measure accordingly. Furthermore, the resulting similarity measures can be computed efficiently, provided that functions employed can be computed efficiently as well.

Our choice for the DL $\mathcal{ELH}$ is motivated by the fact that large, well-known biomedical ontologies such as the Gene Ontology [5] or SNOMED [21] are written in (extensions of) $\mathcal{ELH}$. Furthermore, $\mathcal{ELH}$ is a fragment of the DL that corresponds to the OWL 2 EL profile, which is part of the W3C standard for an ontology language for the Semantic Web [23, 19].

The paper is structured as follows: we start with preliminaries on DLs. In Section 3, we introduce the set of properties desirable for similarity measures and in Section 4 we devise a framework for constructing similarity measures that fulfill (most of) the introduced properties. The paper ends with conclusions and directions for future work.

2 Preliminaries

In this section we introduce the basic notions of DLs. For a thorough introduction see [1]. Starting from a finite set of concept names N_C and a finite set of role names N_R, complex concepts can be defined using *concept constructors*. Let A, $B \in N_C$, then $\mathcal{EL}$*-concepts* are formed according to the following syntax rule:

$$C ::= \top \mid A \mid C \sqcap D \mid \exists r.C$$

where $r \in N_R$ and C, D denote arbitrary $\mathcal{EL}$-concepts. A concept of the form $\exists r.C$ is called an *existential restriction* and one of the from $C \sqcap D$ is called a *conjunction*. We call the DL, that only offers conjunction as a concept constructor, $\mathcal{L}_0$. The semantics of concepts is given in terms of interpretations. An *interpretation* $\mathcal{I} = (\Delta, \cdot)$ consists of the *interpretation domain* $\Delta^{\mathcal{I}}$ a non-empty set and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns role names to binary relations on $\Delta^{\mathcal{I}}$ and concepts to subsets of $\Delta^{\mathcal{I}}$. The top-concept $\top$ is mapped to $\Delta^{\mathcal{I}}$. The

extension of the interpretation function to conjunctions is $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$ and to existential restrictions $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}}$ and $e \in C^{\mathcal{I}}\}$.

A *concept definition* assigns a concept name to a complex concept. We call $A = C$ a *concept definition* and $A \sqsubseteq C$ a *primitive concept definition*. A finite set of (possibly primitive) concept definitions is a *TBox* $\mathcal{T}$. If the (primitive) definitions in a TBox are acyclic and do not contain multiple definitions we call the TBox *unfoldable*. Concept names occurring on the left-hand side of a definition are called *defined concepts*. All other concept names are called *primitive concepts*. Let $s, r \in N_R$. A *role inclusion axiom* (RIA) is a statement of the form: $r \sqsubseteq s$. The DL that extends $\mathcal{EL}$ by RIAs is called $\mathcal{ELH}$. An interpretation is a model for $s \sqsubseteq r$ iff $s^{\mathcal{I}} \subseteq r^{\mathcal{I}}$. A finite set of RIAs is called *RBox* $\mathcal{R}$. An interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ (RBox $\mathcal{R}$) iff it satisfies all its concept definitions (RIAs). We write $s \sqsubseteq_{\mathcal{R}} r$, if $s^{\mathcal{I}} \subseteq r^{\mathcal{I}}$ holds in all models of $\mathcal{R}$ and $s \equiv_{\mathcal{R}} r$, if $s \sqsubseteq_{\mathcal{R}} r$ and $r \sqsubseteq_{\mathcal{R}} s$ hold.

A DL *knowledge base* (KB) $\mathcal{K}$ consists of the *TBox* and the *RBox* and we say that an interpretation $\mathcal{I}$ is a *model* of $\mathcal{K}$, if it is a model for the corresponding TBox and RBox.

Based on the semantics of concepts, reasoning problems can be defined. The concept C is *subsumed* by the concept D w.r.t. the KB $\mathcal{K}$ ($C \sqsubseteq_{\mathcal{K}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{K}$. C and D are *equivalent* w.r.t. $\mathcal{K}$ ($C \equiv_{\mathcal{K}} D$) iff $C \sqsubseteq_{\mathcal{K}} D$ and $D \sqsubseteq_{\mathcal{K}} C$.

For a given concept C, *expansion* replaces exhaustively all occurrences of defined concepts in C by the right-hand sides of their concept definitions. For unfoldable TBoxes all reasoning problems can be reduced to reasoning for concepts by using expansion of concepts w.r.t. the TBox [1].

We denote the set of concepts for a specific DL $\mathcal{L}$ with $\mathcal{C}(\mathcal{L})$, e.g., $\mathcal{C}(\mathcal{EL})$ is the set of all $\mathcal{EL}$-concepts. We call concepts that are either concept names or existential restrictions *atoms* and denote the set of atoms by N_A.

For $\mathcal{EL}$-concepts a unique normal form (modulo associativity and commutativity), was given in [2], which we extend to $\mathcal{ELH}$-concepts in presence of RBoxes. To treat equivalent roles, we define $[r] = \{s \in N_R \mid r \equiv_{\mathcal{R}} s\}$ and fix a function f that picks one role r_i from each equivalence class and replaces each occurrence of a role from $[r_i]$ with r_i. Given an RBox $\mathcal{R}$ and an $\mathcal{ELH}$-concept C, C is in *$\mathcal{ELH}$-normal form*, if the following 4 rules have been applied exhaustively to the concept C and its subconcepts:

1. $A \sqcap \top \longrightarrow A$, 2. $A \sqcap A \longrightarrow A$, 3. $\exists r.C' \longrightarrow \exists f([r]).C'$,

4. $\exists r.C' \sqcap \exists s.D' \longrightarrow \exists r.C'$ if $r \sqsubseteq_{\mathcal{R}} s$ and $C' \sqsubseteq D'$

The transformation of $\mathcal{ELH}$-concepts into $\mathcal{ELH}$-normal form can be done in polynomial time.

3 Properties for Concept Similarity Measures

Formally, a *concept similarity measure sim* is a function mapping from pairs of $\mathcal{ELH}$-concepts to the interval $[0, 1]$. To identify properties of similarity measures for concepts, [8] used *metric spaces* as a starting point, which was also done in other areas of similarity research (see [22, 16, 17, 20]). A metric can be interpreted as a *dissimilarity measure*. The distance represents the dissimilarity between two objects—the lower their distance, the higher the similarity. Using a metric d, we can obtain a similarity function s by defining $s(a, b) := 1 - d(a, b)$. If we adapt the properties of a metric accordingly, we obtain the following properties for similarity functions.

Definition 1. *Let D be a set. A function $s : D \times D \longrightarrow [0, 1]$ is called a* similarity function *for D iff for all $a, b, c \in D$ holds*

1. $s(a, b) = 1 \iff a = b$, (identity of indiscernibles)
2. $s(a, b) = s(b, a)$, *and* (symmetry)
3. $1 + s(a, b) \geq s(a, c) + s(c, b)$ (triangle inequality).

Next we present definitions of properties of concept similarity measures and the underlying intuitions for these properties. We start with a formal definition of the properties and discuss each of them afterwards.

Definition 2. *Let $C, D, E \in \mathcal{C}(\mathcal{ELH})$. A similarity measure sim is*

1. symmetric *iff* $sim(C, D) = sim(D, C)$.
2. *fulfilling the* triangle inequality *property iff*

$$1 + sim(D, E) \geq sim(D, C) + sim(C, E).$$

3. equivalence invariant *iff* $C \equiv D \implies sim(C, E) = sim(D, E)$.
4. equivalence closed *iff* $sim(C, D) = 1 \iff C \equiv D$.
5. subsumption preserving *iff* $C \sqsubseteq D \sqsubseteq E \implies sim(C, D) \geq sim(C, E)$.
6. reverse subsumption preserving *iff* $C \sqsubseteq D \sqsubseteq E \implies sim(C, E) \leq sim(D, E)$.
7. structurally dependent *iff for all sequences $(C_n)_n$ of atoms with $\forall i, j \in \mathbb{N}, i \neq j : C_i \not\sqsubseteq C_j$ the concepts*

$$D_n := \bigsqcap_{i \leq n} C_i \sqcap D \quad \text{and} \quad E_n := \bigsqcap_{i \leq n} C_i \sqcap E$$

fulfill the condition $\lim_{n \to \infty} sim(D_n, E_n) = 1$.

The properties 1. to 4. are adopted from the literature, whereas to the best of our knowledge the properties 5. to 7. are introduced for DLs in this paper.

Symmetry is a rather controversial property for similarity in general—while some consider it essential [18], cognitive sciences seems to favor an asymmetric notion of similarity [22, 4]. Even for DL concepts Janowicz et al. [13, 12] prefer asymmetry (but devise symmetric measures), whereas most [3, 7, 6, 10, 8] consider it a fundamental property of similarity of concepts.

Triangle property is inherited from metrics. Two papers mentioned triangle inequality in the context of DLs: [8] argues in favor, while [12] argue against it, because of Tversky's [22] work.

DLs allow the same thing to be described in different ways. Two concepts can be syntactically different and yet semantically equivalent. A similarity measure for complex concepts should depend on the semantics rather than the syntax of the concepts to measure.

Equivalence invariance ensures that two equivalent concepts have the same similarity towards a third concept. Equivalence invariance is widely accepted as a necessary property for measures for DL concepts ([13, 12, 6, 8]). Yet we found that the methods used to ensure equivalence invariance were not always sound (see Section 3.1).

Equivalence closure holds for a similarity measure if and only if two concepts are totally similar if and only if they are equivalent. This corresponds with the idea that indiscernible things are identical. Equivalence closure is considered to be a basic property for concept similarity measures [8, 12] especially since it is inherited from metrics.

One asset of DLs is their reasoning services. An intuitive idea is to characterize similarity of concepts in terms of these services. The subsumption relation yields a total partial order on concepts. Consider the case where $C, D, E \in \mathcal{C}(\mathcal{ELH})$ and $C \sqsubseteq D \sqsubseteq E$. A natural requirement of similarity measures is to reflect this constellation.

Subsumption preservation expresses that the similarity of C and D is higher than the one of C and E because C is 'closer' to D than to E.

Reverse subsumption preservation states likewise that the similarity of D and E is higher than the similarity of C and E, since E is 'closer' to D than to C.

In [15] we also employ the reasoning service least common subsumer to capture the characteristics of total dissimilarity of concept similarity.

Tversky [22] presents the *feature model*, where an object is described by a set of features. The similarity of two objects is measured by a relation between the number of common features of both objects and the number of unique features of each object. The basic rule is that if

1. the number of common features increases and
2. the number of uncommon features is constant

then the similarity must increase.

Structural dependence reflects this basic rule. Concepts are our objects to compare and the atoms of a conjunction represent the features of the object. The intuition is that the more features (atoms) two complex concepts share, the higher their similarity should be.

For a more detailed explanation of the last property and for a presentation of examples illustrating the above properties see [15].

Table 1. Overview of similarity measures and their properties

	symm.	triang.	eq. inv.	eq. cl.	subs.	rev. subs.	struc. dep.	DL
simi	✓	-	✓	✓	✓	-	✓	$\mathcal{ELH}$
Jacc [11]	✓	✓	✓	✓	✓	✓	✓	$\mathcal{L}_0$
[13]	✓	-	-	-	-	-	✓	$\mathcal{SHI}$
[12]	✓	-	-	-	-	-	✓	$\mathcal{ALCHQ}$
[7]	-	-	-	-	-	-	-	$\mathcal{ALC}$
[10]	✓	-	✓	-	✓	✓	-	$\mathcal{ALN}$
[6]	✓	-	✓	-	✓	✓	-	$\mathcal{ALC}$
[8]	✓	-	✓	-	✓	✓	-	$\mathcal{ALE}$

3.1 Inspecting Existing Concept Similarity Measures

We distinguish two kinds of concept similarity measures: structural measures and interpretation based measures. *Structural measures* are defined using the syntax of the concepts to measure. Since conjunction and disjunction are commutative and associative, these measures are invariant to the order of the atoms in a conjunction or disjunction. The measures differ regarding the similarity of primitive concepts: [12] uses the TBox whereas [7] and [10] use the canonical interpretation which takes the set of ABox individuals as the interpretation domain (for an introduction to ABoxes see [1]).

Interpretation based measures are defined using interpretations and cardinality, instead of the syntax of the (possibly) complex concepts to measure. Therefore, they are trivially equivalence invariant. The two interpretation based measures we investigated [6, 8] are using the canonical interpretation $\mathcal{I}_\mathcal{A}$. These measures need a populated and representative ABox as a significant domain.

Table 1 presents an overview of similarity measures for concepts written in different DLs (including our measure *simi* to be defined in Section 4) and whether or not they fulfill the properties from Definition 2. The proofs can be found in [15]. The first four measures are purely structural measures. The next two are structural measures which use the canonical interpretations to measure primitives. The last two are purely interpretation based measures.

We included the Jaccard index [11], which is originally a set measure, here adapted to $\mathcal{L}_0$. Interestingly, this is the only measure of those investigated that fulfills the triangle inequality.

Our thorough investigation of the similarity measures defined in the literature showed that defining a similarity measure that fulfills most of the properties from Definition 2 is by no means a trivial task—in particular if the DL allows the use of roles, as the lightweight DL $\mathcal{ELH}$ already does.

4 Developing Concept Similarity Measures for $\mathcal{ELH}$

We present *simi*, a framework for similarity measures for concepts written in the DL $\mathcal{ELH}$ based on the semantics of the logic. It operates on (complex) concepts

and an RBox $\mathcal{R}$, which contains role inclusion axioms. If concepts to be processed contain concepts defined in an unfoldable TBox $\mathcal{T}$, we assume that these concepts are expanded w.r.t. $\mathcal{T}$, i.e., all concept names occurring in them are primitive names.

Another preprocessing step is to transform the concepts into $\mathcal{ELH}$-normal form (defined in Section 2). Concepts in this normal form are unique (modulo associativity and commutativity), which ensures that *simi* (and any other measure processing concepts in this normal form) is equivalence invariant. We assume for the remainder of the paper that the concepts are in $\mathcal{ELH}$-normal form.

The framework *simi* constructs similarity measures from several free parameters, i.e., it allows functions to be combined in such a way that, if these functions fulfill certain properties, then the resulting similarity measure can be shown to fulfill all properties from Definition 2 except reverse subsumption preserving and the triangle inequality. Furthermore, it can be computed efficiently.

Simi is inspired by the Jaccard index and it is a conservative extension of the Jaccard index, in the sense that $\forall C, D \in \mathcal{C}(\mathcal{L}_0) : simi(C, D) = Jacc(C, D)$ (proven in [15]). Another inspiration is the equivalence of concepts, which can be regarded as a trivial similarity measure: the similarity of two concepts is 1 if they are equivalent and 0 otherwise. To determine if $C \equiv D$ is true, one can use the subsumption test to find out whether or not $C \sqsubseteq D$ and $D \sqsubseteq C$ are true. We generalize this approach in *simi* by introducing a generalization of the subsumption operator. Since such an operator is in general an asymmetric function, we call it *directed simi* and denote it with $simi_d$ (to be introduced in Section 4.1). Now, once the values $simi_d(C, D)$ and $simi_d(D, C)$ are computed, we have to combine them with an operator to obtain a value for *simi*. Instead of fixing a specific operator, we identify the properties such an operator needs to provide such that *simi* fulfills as many of the properties as possible. We call such an operator a *fuzzy connector* (denoted with $\otimes$). A fuzzy connector $\otimes$ is an operator on the interval $[0, 1]$, $\otimes : [0, 1]^2 \longrightarrow [0, 1]$ such that for all $x, y \in [0, 1]$ the following properties are true.

- Commutativity: $x \otimes y = y \otimes x$,
- Equivalence closed: $x \otimes y = 1 \iff x = y = 1$,
- Weak monotonicity: $x \leq y \implies 1 \otimes x \leq 1 \otimes y$,
- Bounded: $x \otimes y = 0 \implies x = 0$ or $y = 0$ and
- Grounded: $0 \otimes 0 = 0$.

Using a fuzzy connector, *simi* is simply defined as

$$simi(C, D) := simi_d(C, D) \otimes simi_d(D, C)$$

where C and D are arbitrary $\mathcal{ELH}$-concepts.

The commutativity of a fuzzy connector ensures that *simi* is symmetric, the property equivalence closed provides the same property for the resulting similarity measure and weak monotonicity is sufficient to prove that *simi* fulfills subsumption preserving. Examples for fuzzy connectors are the average and triangular norms (t-norms, $\otimes$) [14] which fulfill the property that for all $x, y \in [0, 1] : x \otimes y = 0 \implies x = 0$ or $y = 0$ as shown in [15].

4.1 A Directed Similarity Measure: $simi_d$

To formulate $simi_d$, we need a bit of notation. If convenient, we treat concepts as sets of atoms. Let $C \in \mathcal{C}(\mathcal{ELH})$, then it can be written as $C = \bigsqcap_{i \leq n} C_i$ where $\forall i \leq n : \ C_i \in N_A$. The function $(\widehat{\cdot})$ maps concepts to sets of atoms, so for C, $\widehat{C} := \{C_1, C_2, \ldots, C_n\}$. Now, the starting point for the derivation of $simi_d$ is the function

$$d(C, D) := \frac{|\widehat{C} \cap \widehat{D}|}{|\widehat{C}|}$$

which is inspired by the Jaccard Index. This function can be used to measure the similarity of sets of concept names. In order to be able to incorporate existential restrictions, we rewrite the numerator of d to

$$|\widehat{C} \cap \widehat{D}| = \sum_{C' \in \widehat{C}} \max_{D' \in \widehat{D}} f(C', D'), \tag{1}$$

where the function $f : N_C \longrightarrow \{0, 1\}$ is defined as $f(C', D') := 1$ if $C' = D'$ and 0 otherwise.

The simplifying assumption for f is that two different concept names denote always totally dissimilar concepts. However, this assumption may not be correct in all cases. Therefore, we generalize f by introducing a measure for concept names. In order to work for existential restrictions, this measure has to be able to deal with role names, too. In addition, we have to ensure some properties for this measure to guarantee properties for $simi$. We call this measure for (concept or role) names a *primitive measure* and denote it with pm. More formally, it is a function of type $pm : N_C^2 \cup N_r^2 \longrightarrow [0, 1]$ with the property that for all $A, B \in N_C$ and $r, s, t \in N_r$ the following holds:

- $pm(A, B) = 1 \iff A = B$,
- $pm(r, s) = 1 \iff s \sqsubseteq r$,
- $s \sqsubseteq_{\mathcal{R}} r \implies pm(s, r) > 0$, and
- $t \sqsubseteq_{\mathcal{R}} s \implies pm(r, s) \leq pm(r, t)$.

The first two properties are sufficient to ensure that $simi$ fulfills equivalence closed and the last one is needed to prove that $simi$ fulfills subsumption preserving. Note that pm does not need to be symmetric.

To incorporate existential restrictions into d we have three cases to consider. Namely, we need to be able to compute the similarity of two concept names, of a concept name and an existential restriction and of two existential restrictions. The first case is handled directly by the primitive measure pm. In the second case, we assert that a concept name and an existential restriction are always totally dissimilar and thus their similarity is 0. For the third case, let $\exists r.C^*$ and $\exists s.D^*$ be the two existential restrictions. To compute the similarity of both atoms, we proceed component-wise. The similarity of the role names is computed using the primitive measure pm and the similarity of the concepts C^* and D^*

is computed by a recursive call to d. Then, to combine both values we use a number $w \in (0,1)$ and the formula

$$d(\exists r.C^*, \exists s.D^*) := pm(r,s) \cdot [w + (1-w) \cdot d(C^*, D^*)].$$

Forcing $w > 0$, enables us for $d(C^*, D^*) = 0$ to distinguish between the cases where the roles are similar and where they are not. In the first case, the similarity is w, whereas in the second one, the similarity is 0.

As a suitable w, we suggest the value n where one would say that the concepts

$$C := \underbrace{\exists r. \cdots \exists r.}_{n} A \text{ and } D := \underbrace{\exists r. \cdots \exists r.}_{n} B$$

with $pm(A,B) = 0$ are regarded (almost) totally similar.

In Equation 1, we search for each atom of C for that atom of D with the highest similarity value. This method does not always yield satisfactory results. Consider the case, where $pm(A,B_1) = 0.5$ and $pm(A,B_2) = 0.5$ and we want to measure A towards $B_1 \sqcap B_2$, then the current version of function d does not take into account that A is 'known to be similar' to each of B_1 and B_2 alone and thus should even be more similar to their combination. The function chooses *one* 'best matching partner' instead of combining the two sources of similarity.

To deal with this effect, we propose to replace the maximum operator with a triangular conorm (t-conorm, $\oplus$) [14] which is *bounded*, meaning that for all $x, y \in [0,1] : x \oplus y = 1 \implies x = 1$ or $y = 1$. There are several reasons for the use of a t-conorm. First, the operator max is an instance of a bounded t-conorm. Second, all t-conorms yield values greater or equal to those of max which is consistent with our expectation that the value should be higher or equal to the maximum. Also, 0 acts as neutral element for t-conorms. Therefore, all atoms from D that are totally dissimilar do not influence the value. If we use the probabilistic sum ($x \oplus_{sum} y = x + y - xy$) instead of the maximum for our example above, then we obtain the value 0.75 instead of 0.5, since the measure takes both similarity values (towards B_1 and B_2) into account.

Another parameter of $simi_d$ is the *weighting function* (denoted g). It weights the atoms by assigning each of them a value greater than 0, so $g : N_A \longrightarrow \mathbb{R}_{>0}$. The effect is that some atoms can 'contribute more' to the similarity than others, thus a part of the vocabulary can be picked by g to supply a context under which the concepts from the KB are assessed. Let's assume we are interested in similarity regarding Anatomy and our KB, say SNOMED, contains atoms from two different subject areas like Anatomy and medical procedures. Now, weighting the atoms related to Anatomy higher would result in their similarity having a greater influence on the overall similarity value between concepts.

Note, that the KB does not need to be changed or adapted to achieve this. Several different such weighting functions can easily be employed for the same KB. To incorporate the weighting function we generalize the cardinality of a set of atoms to the sum of the weights of its elements. To obtain a well-defined measure, the weight needs to be added to the numerator of d as well.

By combining the above presented parts, we can already obtain a definition of $simi_d$ except for some corner cases involving $\top$. If we want to be formally correct, then the type of the function $simi_d$ depends on the used parameters as well as on the concepts to be measured. However, for better readability, we omit writing these parameters.

Definition 3 ($simi_d$)**.** *Let $C, D \in \mathcal{C}(\mathcal{ELH}) \setminus \{\top\}$, $E, F \in \mathcal{C}(\mathcal{ELH})$, $A, B \in N_C$ and $r, s \in N_R$.* Directed simi *is the function $simi_d : \mathcal{C}(\mathcal{ELH})^2 \longrightarrow [0,1]$ defined (w.r.t. a bounded t-conorm $\oplus$, a primitive measure pm, a weighting function g and $w \in (0,1)$) by*

$$\begin{aligned} simi_d(\top, \top) &:= simi_d(\top, D) \;:=\; 1, \\ simi_d(C, \top) &:= 0, \end{aligned}$$

$$simi_d(C, D) := \frac{\sum\limits_{C' \in \widehat{C}} [g(C') \cdot \bigoplus_{D' \in \widehat{D}} simi_a(C', D')]}{\sum\limits_{C' \in \widehat{C}} g(C')},$$

where $simi_a$ measures the similarity of two atoms and is defined as

$$\begin{aligned} simi_a(A, B) &:= pm(A, B), \\ simi_a(\exists r.E, A) &:= simi_a(A, \exists r.E) \;:=\; 0, \\ simi_a(\exists r.E, \exists s.F) &:= pm(r, s) \cdot [w + (1 - w) simi_d(E, F)]. \end{aligned}$$

4.2 Properties of $simi_d$ and $simi$

We present the lemma needed to prove various properties of $simi$. The proofs can be found in [15] (p. 67 ff). In the following we assume that the primitive measure is pm, the weighting function is g, the t-conorm is $\oplus$ and the fuzzy connector is $\otimes$.

Lemma 1. *Let $C, D, E \in \mathcal{C}(\mathcal{ELH})$. Then*

1. $simi_d(C, D) = 1 \iff D \sqsubseteq C$.
2. $D \sqsubseteq E \implies simi_d(C, E) \leq simi_d(C, D)$.

Proof. We present a proof sketch for the left-to-right implication of the first statement. Let $simi_d(C, D) = 1$. If $C = \top$ then $D \sqsubseteq C = \top$ is true. Let $C \neq \top$. To prove $D \sqsubseteq C$ we have to show that $\forall C' \in \widehat{C}\ \exists D' \in \widehat{D} :\ D' \sqsubseteq C'$. Let C' be an arbitrary atom of C. $simi_d(C, D) = 1$ implies that

$$\sum_{C' \in \widehat{C}} g(C') = \sum_{C' \in \widehat{C}} [g(C') \cdot \bigoplus_{D' \in \widehat{D}} simi_a(C', D')].$$

Because of $g(C') \cdot \bigoplus_{D' \in \widehat{D}} simi_a(C', D') \leq g(C')$ we derive that for all $C' \in \widehat{C} :\ \bigoplus_{D' \in D} simi_a(C', D') = 1$. Since the t-conorm is bounded, $\exists D' \in D$ such

that $simi_a(C', D') = 1$. The rest of the proof uses structural induction and case distinction.

If $C' = A$ then $simi_a(C', D') = 1$ leads to $D' = A$ which implies $D' \sqsubseteq C'$. Next, let $C' = \exists r.C^*$. $simi_a(C', D') = 1$ implies that D' is of the form $D' = \exists s.D^*$ and $1 = pm(r, s) \cdot [w + (1 - w)simi_d(C^*, D^*)]$. This leads to $pm(r, s) = 1$ which according to the definition of the primitive measure implies $s \sqsubseteq r$. Since $w < 1$, $simi_d(C^*, D^*) = 1$. Using the induction hypothesis we can derive $D^* \sqsubseteq C^*$, therefore $D' \sqsubseteq C'$.

Recall, $simi(C, D) := simi_d(C, D) \otimes simi_d(D, C)$. The resulting function has the following properties.

Theorem 1. *The function simi fulfills*

1. *symmetry,*
2. *equivalence invariance,*
3. *equivalence closed,*
4. *subsumption preserving.*

Let g' be a weighting function with $inf\{g(C') \mid C' \in \mathcal{C}(\mathcal{ELH})\} > 0$. Furthermore, let $\otimes'$ be a fuzzy connector s.t. for all sequences $(x_n)_n$ and $(y_n)_n$ ($x_i, y_i \in [0, 1]$) with $\lim_{n\to\infty} x_n = \lim_{n\to\infty} y_n = 1$ and $\lim_{n\to\infty} x_n \otimes' y_n = 1$. Then simi together with $\otimes'$ and g' fulfills structural dependence.

The main reason why *simi* neither fulfills the triangle inequality nor reverse subsumption preserving is that the computation of $simi_d(C, D)$ does not use the similarity values between the atoms of C (and between the atoms of D). Consider $C := A \sqcap \bigsqcap_{i \leq n} B_i$, where the B_i are very similar to each other, $D := A \sqcap B_0$ and $E := A$ then the similarity of D and E is approximately 0.5, the similarity of C and D is close to 1 (since each B_i is very similar to B_0) but the similarity of C and E converges to 0 with increasing n. For the proofs of other properties of *simi* and further details see [15].

An important property of *simi* is that it can be computed efficiently, provided that the involved parameter functions can be computed efficiently as well.

Lemma 2. *If the specific fuzzy connector, the bounded t-conorm, the primitive measure and the weighting function can be computed in polynomial time, then simi can be computed in time polynomial in the size of the concepts to measure.*

5 Conclusions

Similarity measures are important procedures for central ontology management tasks such as alignment of ontologies. Often these measures are built in an ad-hoc way by simply tuning them to test data.

In this paper we have proposed a different approach to construct a whole range of such measures for $\mathcal{ELH}$-concepts. Our starting point was a set of formally defined properties for concept similarity measures, which make use of the

semantics of DL concepts and of DL reasoning services. We devised a framework that, if instantiated with appropriate functions and operators as discussed in this paper, allows to generate similarity measures that have 5 of the proposed 7 properties (reverse subsumption preservation and triangle inequality are missing). In that sense one could claim that our framework for similarity measures is not only semantics-based, but also provides the measures with semantics. Moreover, our approach does not restrict users to a single similarity measure, but allows them to design their own measures by selecting the functions and operators appropriate to yield the needed individual similarity measure. If the selected functions conform to the framework described in this paper, the resulting similarity measure is equipped with the properties.

Similarity is often perceived as a context-dependent characteristic. Even in this case our framework can offer support, in the sense that the directed measure $simi_d$ allows atoms appearing in the concept to be weighted differently using the weighting function g. Different instantiations of g allow different thematic subdomains of the domain of discourse to be highlighted.

To test our framework empirically is a non-trivial task, since each application may require a different instantiation of *simi* with functions and operators. To aquire such instantiations suitable for each application requires profound knowledge of the application in question. Thus for now it remains future work to compare the outcome of *simi* instantiations with other well-accepted similarity measures.

On the theoretical side it would be interesting to investigate such frameworks for more expressive DLs and for the concepts defined w.r.t. general TBoxes. Since a unique normal form is the main means to achieve an equivalence invariant similarity measure, it is not obvious how to extend *simi* to these more expressive scenarios.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann, Los Altos.

[3] A. Borgida, T. J. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In *Proceedings of the International Workshop on Description Logics (DL2005)*, 2005.

[4] B. Bowdle and D. Gentner. Informativity and asymmetry in comparisons. *Cognitive Psychology*, 34(3):244–286, 1997. PMID: 9466832.

[5] T. G. O. Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[6] C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In *Convegno Italiano di Logica Computazionale (CILC 2005)*, 2005.

[7] C. d'Amato, N. Fanizzi, and F. Esposito. A dissimilarity measure for $\mathcal{ALC}$ concept descriptions. In *Proceedings of the ACM symposium on Applied computing*, SAC '06, pages 1695–1699, 2006.

[8] C. d'Amato, S. Staab, and N. Fanizzi. On the influence of description logics ontologies on conceptual similarity. In *Proceedings of the 16th Knowledge Engineering Conference (EKAW2008)*, volume 5268, pages 48–63, 2008.

[9] J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*, pages 333–337. IOS Press, 2004.

[10] N. Fanizzi and C. d'Amato. A similarity measure for the $\mathcal{ALN}$ description logic. In *Convegno Italiano di Logica Computazionale (CILC 2006)*, 2006.

[11] P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[12] K. Janowicz. SIM-DL: Towards a semantic similarity measurement theory for the description logic $\mathcal{ALCNR}$ in geographic information retrieval. *SeBGIS 2006, OTM Workshops 2006*, pages 1681–1692, 2006.

[13] K. Janowicz and M. Wilkes. SIM-DLA: a novel semantic similarity measure for description logics reducing Inter-Concept to Inter-Instance similarity. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web Research and Applications*, pages 353–367, 2009.

[14] E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. SV, 2000.

[15] K. Lehmann. A framework for semantic invariant similarity measures for $\mathcal{ELH}$ concept descriptions. Master's thesis, TU Dresden, 2012. Available from: http://lat.inf.tu-dresden.de/research/mas.

[16] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 863—872, 2003.

[17] M. Li and M. R. Sleep. Melody classification using a similarity metric based on Kolmogorov complexity. In *Proceedings of the Sound and Music Computing Conference (SMC'04)*, 2004.

[18] D. Lin. An Information-Theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, 1998.

[19] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 web ontology language profiles. W3C Recommendation, 27 October 2009. http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/.

[20] N. Nikolova and J. Jaworska. Approaches to measure chemical similarity - a review. *QSAR & Combinatorial Science*, 22:1006–1026, 2003.

[21] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. *Journal of the American Medical Informatics Assoc.*, 2000. Fall Symposium Special Issue.

[22] A. Tversky. Features of similarity. In *Psychological Review*, volume 84, pages 327–352, 1977.

[23] W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation, 27th October 2009. http://www.w3.org/TR/2009/REC-owl2-overview-20091027/.

Towards Instance Query Answering for Concepts Relaxed by Similarity Measures

Andreas Ecke[*] Theoretical Computer Science, TU Dresden, Germany

Rafael Peñaloza[†] Theoretical Computer Science, TU Dresden, Germany

Anni-Yasmin Turhan[‡] Theoretical Computer Science, TU Dresden, Germany

Center for Advancing Electronics Dresden

Abstract

In Description Logics (DL) knowledge bases (KBs) information is typically captured by crisp concept descriptions. However, for many practical applications querying the KB by crisp concepts is too restrictive. A controlled way of gradually relaxing a query concept can be achieved by the use of similarity measures.

To this end we formalize the task of instance query answering for crisp DL KBs using concepts relaxed by similarity measures. We identify relevant properties for the similarity measure and give first results on a computation algorithm.

1 Introduction

Description Logics (DLs) are a family of knowledge representation formalisms that have unambiguous semantics. A particular DL is characterized by a set of concept constructors, which allow to build complex concept descriptions. Intuitively, *concept descriptions* characterize categories from an application domain. In addition, binary relations on the domain of interest can be captured by *roles*. These in turn can be used in concept descriptions. The terminological knowledge of an application domain is stored in the *TBox*, where complex concept descriptions can be assigned to concept names. Facts from the application domain and relations between them are represented by *individuals* in the *ABox*. TBox and ABox together form the DL *knowledge base* (KB).

The formal semantics of DLs allow the definition of a variety of reasoning services. The most prominent ones are *subsumption*, i.e. to compute whether a sub-concept relationship holds between two concept descriptions and *instance query answering*, where for a given concept description all individuals from an ABox that are instances of the concept are computed. These reasoning services are implemented in highly optimized reasoning systems, see for example [Tsarkov and Horrocks, 2006; Kazakov *et al.*, 2012; Haarslev *et al.*, 2012].

DLs of varying expressivity are the underlying logics for the W3C standardized ontology language OWL 2 and its profiles [Motik *et al.*, 2009]. This has led to an increased use of DLs and DL reasoning systems in the recent years in many application areas. By now there is a large collection of KBs written in these languages. However, many applications need to query the knowledge base in a less strict fashion.

In the application area of service matching OWL TBoxes are employed to describe types of services. Here, a user request for a service specifies several conditions for the desired service. These conditions are represented by a concept description. For such a concept description the OWL ABox that contains the individual services is searched for a service matching the specified request by employing instance query answering. In cases where an exact match with the provided requirements is not possible, a "feasible" alternative needs to be retrieved from the ABox containing the services. This means that those individuals from the ABox should be retrieved for the given query concept that fulfill the main conditions, while for some conditions only a relaxed variant is fulfilled.

A natural idea on how to relax the notion of instance query answering is to simply employ fuzzy DLs and perform query answering on a fuzzy variant of the initial query concept. However, on the one hand reasoning in fuzzy DLs easily becomes undecidable [Borgwardt *et al.*, 2012; Borgwardt and Peñaloza, 2012; Cerami and Straccia, 2013] and on the other hand depending on the user and on the request, different ways of relaxing the query concept are needed. For instance, for a request to a car rental company to rent a particular car model in Beijing, it might be acceptable to get an offer for a similar car model to be rented in Beijing, instead of getting the offer to rent the requested car model in London. Whereas for a handicapped user in a wheelchair it might not be acceptable to relax the requested car model from a two-door one to a four-door one. Here fuzzy concepts would relax the initial concept in an unspecific and uniform way. Ideally, relaxed instance query answering should allow to

1. choose *which aspects* of the query concept can be relaxed and

[*]Supported by the German Research Foundation (DFG) Graduiertenkolleg 1763 (QuantLA).

[†]Partially supported by DFG within the Cluster of Excellence 'cfAED'

[‡]Partially supported by DFG in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing".

2. choose the *degree* to how much these aspects can be relaxed.

The reasoning service addressed in this paper is a relaxed notion of instance querying, such that it allows for a given query concept the selective and gradual extension of the answer set of individuals. We develop a formal definition of this reasoning service in Section 3.

Our approach for achieving selective and gradual extension of the answer sets is to employ concept similarity measures to relax the query concept. A *concept similarity measure* yields, for a pair of concept descriptions, a value from the interval $[0, 1]$—indicating how similar the concepts are. The goal is to compute for a given concept C, a concept similarity measure $\sim$ and a degree t ($t \in [0, 1]$), a set of concept descriptions such that each of these concepts is similar to C by a degree of at least t, if measured by $\sim$, and finding all their instances.

For DLs there is whole range of similarity measures defined (see for example [Borgida *et al.*, 2005; d'Amato *et al.*, 2005; Lehmann and Turhan, 2012]), which could be employed for this task. In particular the similarity measures generated by the framework described in [Lehmann and Turhan, 2012] allow users to specify which part of the vocabulary used in their knowledge base is to be regarded more important when it comes to the assessment of similarity of concepts. Thus, these measures naturally allow to select which aspect of the query concept to relax.

The core reasoning problem encountered in our algorithm for relaxed instance query answering is to compute for an individual a and the query concept description C a concept description C' that *mimics* C, i.e. a concept description that is 'sufficiently similar' to C w.r.t. the used similarity measure $\sim$ and the degree t.

We propose in this paper an algorithm to compute the above mentioned reasoning service of relaxed instance query answering in the lightweight DL $\mathcal{EL}$. For instance, for the Gene ontology [Gene Ontology Consortium, 2000], which is written in $\mathcal{EL}$ and is used (among other things) to solve the task of finding genes that realize similar functionality [Lord *et al.*, 2003], a proliferation of different similarity measures has been defined [Lord *et al.*, 2003; Schlicker *et al.*, 2006; Mistry and Pavlidis, 2008; Alvarez and Yan, 2011]. In principle these measures could be used in our approach to query ABoxes. We identify properties of concept similarity measures that allow to compute relaxed instances of concepts.

The paper is organized as follows: after introducing basic notions on DLs and concept similarity measures in Section 2, we develop a formal notion of relaxed instances in Section 3. In order to compute relaxed instances it is necessary, as we shall see, to compute mimics of a concept and an individual. An way of finding a mimic and its application to construct an algorithm that computes all relaxed instances of a query concept is provided in Section 4. As customary, the paper ends with conclusions and future work.

2 Preliminaries

In this section we introduce the basic notions of Description Logics and similarity measures between concepts. For a thorough introduction to Description Logics, see [Baader *et al.*, 2003]. While we try to formalize the notion of relaxed instances of a concept w.r.t. a similarity measure independently from a specific DL, Section 4 will show how instance querying for relaxed concepts can be computed in the restricted DL $\mathcal{EL}$.

	Syntax	Semantics
top concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
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: Concept constructors, TBox axioms and ABox assertions for $\mathcal{EL}$.

Let N_C, N_R, and N_I be non-empty, disjoint sets of *concept names*, *role names*, and *individual names*. A *concept description* (or short concept) is constructed from concept names by applying *concept constructors* such as conjunction, negation, quantification, or the top concept $\top$. In particular, $\mathcal{EL}$ only admits the concept constructors conjunctions, existential restrictions and the top concept, as seen in Table 1. We denote the set of all $\mathcal{L}$-concept descriptions constructed is such a way by $\mathcal{C}(\mathcal{L})$.

For example, using the following $\mathcal{EL}$-concept description, one can describe a service which currently waits for requests, but runs on an overloaded server:

$$\mathsf{Service} \sqcap \exists \mathsf{has\text{-}state}.\mathsf{WaitingForRequest} \sqcap \exists \mathsf{runs\text{-}on}(\mathsf{Server} \sqcap \exists \mathsf{has\text{-}condition}.\mathsf{Overloaded})$$

The semantics of concept descriptions is defined by means of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names, subsets of $\Delta^{\mathcal{I}}$ to concept names, and elements of $\Delta^{\mathcal{I}}$ to individual names. The interpretation function can be recursively extended to $\mathcal{EL}$-concept descriptions as shown in Table 1.

An *$\mathcal{EL}$-knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of an *$\mathcal{EL}$-TBox* $\mathcal{T}$, which captures the terminological knowledge, and an *$\mathcal{EL}$-ABox* $\mathcal{A}$, which contains the assertions about specific individual. In this paper we only consider *unfoldable* TBoxes, i.e., sets of concept definitions such that each concept name occurs at most once on the left-hand side of a concept definition and there are no cyclic dependencies between defined concepts. An ABox is a set of concept and role assertions. The semantics of interpretations is extended to concept definitions and assertions as shown in Table 1. We say that an interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ (ABox $\mathcal{A}$), if it satisfies all concept definition in $\mathcal{T}$ (assertions in $\mathcal{A}$). $\mathcal{I}$ is a model of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it is a model for both $\mathcal{T}$ and $\mathcal{A}$.

There exists a number of inferences for DLs. Three com-

monly used inferences are *concept subsumption*, *concept equivalence* and *instance checking*. Concept subsumption tests if a concept C is subsumed by a concept D w.r.t. a TBox $\mathcal{T}$ (denoted $C \sqsubseteq_{\mathcal{T}} D$), i.e. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. Similarly, two concepts C and D are equivalent w.r.t. $\mathcal{T}$ (denoted $C \equiv_{\mathcal{T}} D$), if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. Finally, an individual a is an instance of a query concept description C w.r.t. a KB $\mathcal{K}$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$.

Besides these standard reasoning tasks, other inferences have been developed for certain applications. The most specific concept, first introduced in [Nebel, 1990], is such a non-standard inference. This inference computes a concept description that describes an individual a from the knowledge base as exact as it is possible in the used DL.

Definition 1. Let $\mathcal{L}$ be a DL and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{L}$-KB. The concept description C is the *most specific concept* of an individual a w.r.t. $\mathcal{K}$ (denoted $\text{msc}(a)$) iff

- a is an instance of C, and
- for all concept descriptions $D \in \mathcal{C}(\mathcal{L})$, if a is an instance of D, then $C \sqsubseteq_{\mathcal{T}} D$.

Similarity measures. For a DL $\mathcal{L}$, a *concept similarity measure* $\sim$: $\mathcal{C}(\mathcal{L}) \times \mathcal{C}(\mathcal{L}) \to [0, 1]$ is a function that assigns a similarity value $C \sim D$ to each pair C, D of $\mathcal{L}$-concept descriptions. A value $C \sim D = 0$ means that C and D are totally dissimilar, while a value $C \sim D = 1$ means that C and D are totally similar.

A collection of properties for concept similarity measures is given in [Lehmann and Turhan, 2012]. In particular, a similarity measure $\sim$ for $\mathcal{L}$-concept descriptions is:

1. *symmetric* iff $C \sim D = D \sim C$ for all $C, D \in \mathcal{C}(\mathcal{L})$;
2. fulfilling the *triangle inequality* iff
$$1 \; + \; D \sim E \; \geq \; D \sim C \; + \; C \sim E$$
for all $C, D, E \in \mathcal{C}(\mathcal{L})$;
3. *equivalence invariant* iff for all $C, D, E \in \mathcal{C}(\mathcal{L})$ with $C \equiv D$ it holds that $C \sim E = D \sim E$;
4. *equivalence closed* iff $C \sim D = 1 \iff C \equiv D$.

In this paper, we only consider symmetric similarity measures, since they better capture our intuitive understanding of similarity. However, all definitions and results can easily be extended to asymmetric similarity measures. Furthermore, the triangle inequality was found to be hard to achieve for similarity measures for even restricted DLs like $\mathcal{EL}$, and thus will not be discussed here.

Observe that the property 'equivalence closed' interacts with relaxed instances of a query concept C in the following way: clearly, if we want only relaxed instances with a similarity of exactly 1, then equivalence closed similarity measures should result in exactly the instances of C, while similarity measures that are not equivalence closed might result in additional individuals.

Most previously proposed concept similarity measures can be divided into two groups: *structural measures*, which are defined using the syntax of the concepts, and *interpretation based measures*, which are defined using interpretations and cardinality instead of the syntax. We later describe a result for structural similarity measures, therefore we will describe these in more detail: Basically, a similarity measure $\sim$ on $\mathcal{L}$-concepts descriptions is called structural, if it computes the similarity of two concepts C and D recursively by computing the similarity of concept names in C and D and the similarity of the existential restrictions occurring in C and D and combining these values monotonically to the overall similarity. For structural similarity measures to be equivalence invariant, the concepts often need to be transformed into a normal form before comparing them [Lehmann and Turhan, 2012]. For a similarity measure $\sim$, we call the normal form used for the computation of the similarity the *$\sim$-normal form*.

3 Relaxed Instances

In this section we introduce the main reasoning problems that we want to solve, as well as a first approach for obtaining a solution.

Our main goal is to generalize query answering to allow for more relaxed solutions. Intuitively, given a concept C, we are interested in finding all the certain instances of C, but also in finding those individuals that are *close* to being instances of C; we call these individuals the *relaxed instances* of C. To emphasize the contrast, we some times call the instances of C *certain instances* of C.

Before we can try to compute these relaxed instances, we need to formalize the notion of relaxed instances of a query concept. In principle there are are many ways to do so and we discuss next some of these options.

One natural approach would be to try to decide which individuals are *similar* to any of the certain instances of C. However, this method would require the definition of a similarity measure on the *elements* of the domain, rather than on the concepts. Such a DL with a similarity measure on the domain elements was introduced in [Lutz *et al.*, 2003]. However, for this DL the similarity measure (or more precisely, a distance metric) is part of the interpretation and cannot be adjusted to different user needs.

A different idea that has been proposed is to simply generalize the concept C by considering named concepts that subsume C. Thus for a named concept C, consider its direct subsumers in the concept hierarchy. This idea is easy to implement and understand, but provides only very rough approximations to the concept C determined by the set of concept names only. Moreover, users have no control on the quality of the approximation provided; in fact even the direct subsumers might describe a concept that is already very dissimilar to C.

We follow a different approach, in which we ask for the instances of those concepts that are similar to C. We can then control how inclusive the relaxed instance solutions should be, by adjusting the degree t of similarity allowed.

Definition 2 (relaxed instance)**.** Let $\mathcal{L}$ be some DL, C be an $\mathcal{L}$-concept, $\sim$ a similarity measure over $\mathcal{L}$-concepts, and $t \in (0, 1]$. The individual $a \in N_I$ is a *relaxed instance* of C w.r.t. the $\mathcal{L}$-knowledge base $\mathcal{K}$, $\sim$ and the threshold t, denoted $a \in_t^{\sim} C$, iff there exists a concept description $X \in \mathcal{C}(\mathcal{L})$ such that $C \sim X \geq t$ and $a \in X^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$.

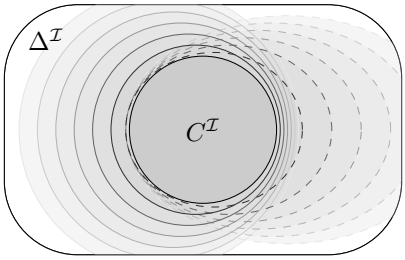

Figure 1: Relaxed instances w.r.t. two different similarity measures. Darker colors represent the relaxed instances of C w.r.t. higher degrees t.

For brevity, we will denote as $\text{Relax}_t^{\sim}(C)$ the set of all relaxed instances of C w.r.t. $\mathcal{K}$, $\sim$ and t. Clearly, the elements of $\text{Relax}_t^{\sim}(C)$ depend strongly on the value of t, but also on the similarity measure $\sim$ chosen, as shown in Figure 1. For a fixed similarity measure $\sim$, if $t \leq t'$, then it holds that $\text{Relax}_{t'}^{\sim}(C) \subseteq \text{Relax}_t^{\sim}(C)$. In the figure, the central circle represents the interpretation of the concept C. The other lines show the interpretation of $\text{Relax}_t^{\sim}(C)$ with darker lines gradually representing large values t. We use two different kinds of lines (continuous vs. dashed) to represent two different similarity measures, that relax the concepts based on different features. As can be seen, the sets obtained can greatly differ from each other.

As mentioned before, our goal is to find all the instances in $\text{Relax}_t^{\sim}(C)$. Following Definition 2, this task could be performed by first computing all concepts X that are similar to C with degree at least t, and then obtaining all the instances of these concepts X; in symbols,

$$\text{Relax}_t^{\sim}(C) = \bigcup_{C \sim X \geq t} \{a \mid a \text{ is an instance of } X\}.$$

However, this approach suffers from two main drawbacks. First, the set of all concepts that are similar to C with degree at least t might be infinite, thus requiring an infinite number of queries to obtain $\text{Relax}_t^{\sim}(C)$, even though this set contains only finitely many individuals. Second, it is not known how to compute the similar concepts X. Similarity measures tell us only how similar two given concepts are, but not how to build a concept that is similar to another with at least some given degree.

To avoid these issues, we consider a different reasoning problem, that considers the computation of a concept that has a given individual a as an instance and resembles C most. We call this the *mimic* of C w.r.t. a.

Definition 3 (mimic)**.** Let $\mathcal{L}$ be a DL, $\mathcal{K}$ be an $\mathcal{L}$-knowledge base, $a \in N_I$ be an individual name, C be an $\mathcal{L}$-concept description, and $\sim$ be a similarity measure. An $\mathcal{L}$-concept D is called a *mimic* of C w.r.t. a, denoted $\mathfrak{M}(C, a)$, iff the following two conditions hold:

- a is an instance of D, i.e., $a^{\mathcal{I}} \in D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$, and
- for all $\mathcal{L}$-concept descriptions E holds, if a is an instance of E, then $C \sim D \geq C \sim E$.

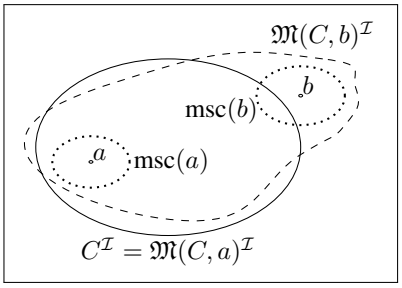

Figure 2: Two individuals, their most specific concepts (dotted), and the mimics of a concept C w.r.t. the individuals (dashed).

Intuitively, a mimic of C w.r.t. a is a concept that is as similar to C as possible, while still having a as an instance. As for relaxed instances, the mimic strongly depends on the similarity measure chosen. Figure 2 depicts the idea of mimics. In the figure, a and b are two named individuals. The former is an instance of C while the latter is not. The dotted lines depict their most specific concepts. Since a is an instance of C, C is also a mimic of C w.r.t. a: $C \sim C = 1$. The dashed line depicts a mimic of C w.r.t. b. Notice that this mimic must contain the msc of b, but need not be a subsumer of C.

We must point out that the mimic of C w.r.t. an individual a need not be unique, even modulo concept equivalence. For example, let $\mathcal{K}$ be a knowledge base consisting of the empty TBox $\mathcal{T}$ and the ABox $\mathcal{A} = \{A \sqcap B(a)\}$, and $\sim$ be a similarity measure with $A \sim C = 0.5$, $B \sim C = 0.5$ and $(A \sqcap B) \sim C = \max\{A \sim C, B \sim C\} = 0.5$. Then A, B, and $A \sqcap B$, are all mimics of C w.r.t. a, as they all have a similarity value of 0.5 to C. In fact, there can be infinitely many such mimics for a given concept C and individual a. As we will see, it suffices to compute one of them.

Using mimics, we can compute the relaxed instances of a concept. The idea is to compute, for each individual a appearing in the knowledge base $\mathcal{K}$, the mimic of C w.r.t. a. If this mimic has similarity at least t with C, then a is a relaxed instance of C; otherwise, it cannot be a relaxed instance, as no concept can have a greater similarity degree with C while still containing a. This is formalized in the following proposition. The proof is a simple consequence of the arguments given above.

Proposition 4. *Let $\mathcal{K}$ be a knowledge base, a be an individual occurring in $\mathcal{K}$, C be a concept description, $\sim$ be a similarity measure and $t \in [0, 1]$. Then $a \in \text{Relax}_t^{\sim}(C)$ iff there is a mimic D of C w.r.t. individual a such that $C \sim D \geq t$.*

In the next section we will study the problem of computing a mimic for a given concept C w.r.t. an individual a. Since all mimics must have the same degree of similarity w.r.t. C, a simple similarity computation provides us with a decision whether a is a relaxed instance of C or not, up to degree t. As computing a mimic may be an expensive task, we also provide an optimization criterion: if a mimic D of C w.r.t. a is similar to C to degree at least t, then all certain instances of D must also be relaxed instances of C, and hence there is no need of computing their corresponding mimics.

4 Computing Mimics in $\mathcal{EL}$

In general there are infinitely many concepts, for which an individual a is an instance of, and thus enumerating them and computing the similarity to C to find the mimic is not a feasible option. However, under some circumstances we can limit the number of concepts that need to be tested in order to find a mimic.

Recall that the notion of a mimic combines a property that is based on the semantics (it must have a as an instance) and a syntactic property (it must be similar to C). The semantic property gives us a starting point on how to find a mimic. A mimic D of C w.r.t. a must always have a as an instance, and hence, by definition of the msc, $\text{msc}(a) \sqsubseteq_{\mathcal{T}} D$ holds. For equivalence invariant similarity measures the idea is to use the $\text{msc}(a)$ as a lower bound for the mimic guaranteeing the semantic property, and to only consider concept descriptions that can be obtained from syntactic manipulations of $\text{msc}(a)$ that result in a generalized concept, i.e., by removing some concept names or existential restrictions.

Definition 5 (generalized concept)**.** Let C be a concept description of the form

$$C = \bigsqcap_{i \in I} A_i \sqcap \bigsqcap_{j \in J} \exists r_j.E_j,$$

with $A_i \in N_C$ for all $i \in I$, and $r_j \in N_R$, E_j is a concept description for all $j \in J$. Then a concept description D is a *generalized concept* of C iff it has the form

$$D = \bigsqcap_{i \in I'} A_i \sqcap \bigsqcap_{j \in J'} \exists r_j.E'_j$$

with $I' \subseteq I$, $J' \subseteq J$ and E'_j is a generalized concept of E_j for $j \in J'$.

This idea, however, only works if the msc is given in a particular syntactic form. It needs to be fully expanded.

Definition 6 (fully expanded concept)**.** Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox. A concept description C is *fully expanded* w.r.t. $\mathcal{T}$ iff for all concept definitions $D = E \in \mathcal{T}$ with $C \sqsubseteq_{\mathcal{T}} D$ we have that E is a generalized concept of C.

The idea is that C contains all its subsumers explicitly as sub-concept descriptions. Now, we can show that the mimic of C w.r.t. a must be a generalized concept of the fully expanded most specific concept of a.

Lemma 7. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-knowledge base, a be an individual from $\mathcal{A}$, C be an $\mathcal{EL}$-concept description, and $\sim$ be an equivalence invariant similarity measure. Let further $E = \text{msc}(a)$ be the fully expanded most specific concept of a. Then there is a mimic $D = \mathfrak{M}(C, a)$ of C w.r.t. a and $\mathcal{K}$ that is a generalized concept of E.*

Proof. We show that any concept F which has a as an instance must be equivalent to a generalized concept of the fully expanded msc. Since the mimic of C w.r.t. a has a as an instance and $\sim$ is equivalence invariant, the lemma follows.

Let F be a concept description with $a^{\mathcal{I}} \in F^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$. Then $E \sqsubseteq_{\mathcal{K}} F$ by definition of the msc. Since E is fully expanded and contains all its subsumers explicitly, any part of the concept description F must also be part of the concept description E. Thus F is a generalized concept of E. □

In general, the msc may contain a chain of infinitely nested existential restrictions for cyclic ABoxes, and hence describing it as a concept would require infinite size. Then there are still infinitely many generalized concepts (of finite size) that need to be checked to find a mimic. This means that Lemma 7 does not always provide a solution to the problem. However, the query concept C (in $\sim$-normal form) has always a finite role-depth, and most structural similarity measures used in practice compute the similarity recursively between concepts at the same role-depth. Therefore, for these similarity measures, it is possible to limit the role-depth of the most specific concept and still get the same result.

Definition 8. Let $\mathcal{K}$ be an $\mathcal{EL}$-KB. By $\text{rd}(C)$ we denote the *role-depth* of a concept C, i.e. the maximal number of nested quantifiers.

The $\mathcal{EL}$-concept description C is the *role-depth bounded most specific concept* (denoted k-$\text{msc}(a)$) of an individual a w.r.t. $\mathcal{K}$ and the role-depth bound k iff

- $\text{rd}(C) \leq k$,
- $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$, and
- for all $\mathcal{EL}$-concepts $D \in \mathcal{C}(\mathcal{L})$ with $\text{rd}(D) \leq k$ and all $a^{\mathcal{I}} \in D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$ it holds that $C \sqsubseteq_{\mathcal{T}} D$.

The role-depth bounded msc is a commonly used approximation of the msc, since it always exists and is unique. An algorithm to compute the k-msc in the $\mathcal{EL}$-family, even w.r.t. general TBoxes, has been introduced in [Peñaloza and Turhan, 2011] and [Ecke *et al.*, 2013]. Using this, we can now show that for structural similarity measures we can find the mimic always as a generalized concept of the role-depth bounded msc.

Lemma 9. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-knowledge base, a be an individual from $\mathcal{A}$, C be an $\mathcal{EL}$-concept description in $\sim$-normal form, and $\sim$ be a structural, equivalence invariant similarity measure with the following property:*

$$X \sim \bigsqcap_{i \in I} A_i \;\geq\; X \sqcap \exists r.B \sim \bigsqcap_{i \in I} A_i. \qquad (1)$$

Let further $k = \text{rd}(C)$ and $E = k$-$\text{msc}(a)$ be the fully expanded role-depth bounded most specific concept of a. Then there is a mimic $D = \mathfrak{M}(C, a)$ of C w.r.t. a that is a generalized concept of E.

Proof. By Lemma 7 we know that there exists a mimic F of C w.r.t. a that is a generalized concept of the (possibly infinite) $\text{msc}(a)$. Since E is the fully expanded k-msc of a, F must also be a generalized concept of E up to role-depth k (but of course, it may contain additional existential restrictions which increase the role-depth of F). We show by induction on k, that there is a generalized concept F' of E with $F' \sim C \geq F \sim C$. This will imply that F' is a mimic of C w.r.t. a, which proves the lemma.

For the case $k = 0$, $C = \bigsqcap_{i \in I} A_i$ and $E = \bigsqcap_{j \in J} B_j$ are conjunctions of concept names and since F a generalized concept of E up to role-depth $k = 0$, we know that F is of the form $F = \bigsqcap_{j \in J'} B_j \sqcap \bigsqcap_{h \in H} \exists r_h.F_h$ with $J' \subseteq J$. But then property (1) yields for $F' = \bigsqcap_{j \in J'} B_j$:

$$F' \sim C \;\geq\; F' \sqcap \bigsqcap_{h \in H} \exists r_h.F_h \sim C \;=\; F \sim C.$$

Procedure relaxed-instance?$(a, C, \mathcal{K}, \sim, t)$
Input: a: individual in $\mathcal{K}$; C: $\mathcal{EL}$-concept description;
$\mathcal{K}$: $\mathcal{EL}$-knowledge base; $\sim$: similarity measure;
t: similarity degree;
Output: whether $a \in_t^\sim C$ w.r.t. $\mathcal{K}$

1: $k := \text{rd}(C)$
2: $E := k\text{-msc}(a)$ w.r.t. $\mathcal{K}$
3: guess a generalized concept F of E
4: **if** $F \sim C \geq t$ **then**
5: **return** true
6: **else**
7: **return** false

Figure 3: Computation algorithm for relaxed instances in $\mathcal{EL}$.

For the case $k > 0$, $C = \bigsqcap_{i \in I} A_i \sqcap \bigsqcap_{h \in H} \exists s_h.C_h$ and $E = \bigsqcap_{j \in J} B_j \sqcap \bigsqcap_{l \in L} \exists r_l.E_l$ are conjunctions of concept names and existential restrictions with $\text{rd}(C_h), \text{rd}(E_l) \leq k-1$ for $h \in H$, $l \in L$. Once again, since F is a generalized concept of E up to role-depth k, it must be of the form $F = \bigsqcap_{j \in J'} B_j \sqcap \bigsqcap_{l \in L'} \exists r_l.F_l$ with $J' \subseteq J$, $L' \subseteq L$ and each F_l is a generalized concept of E_l up to role-depth $k-1$. But then, the induction hypothesis yields for each $h \in H$ and $l \in L'$ that $F_l' \sim C_h \geq F_l \sim C_h$ for generalized concepts F_l' of E_l. Then also $F' = \bigsqcap_{j \in J'} B_j \sqcap \bigsqcap_{l \in L'} \exists r_l.F_l'$ is a generalized concept of E and since the similarity measure $\sim$ is structural, this yields: $F' \sim C \geq F \sim C$. □

We have now identified some constraints on the similarity measure such that we can always find the mimic of C w.r.t. a from a finite set of concept descriptions: the generalized concepts of the fully expanded role-depth bounded msc of the individual a.

Instead of computing the mimic $D = \mathfrak{M}(C, a)$ of C w.r.t. a and testing whether the similarity between the C and D is at least t, it is enough to find *any* concept D' with a as an instance and $C \sim D' \geq t$ to show that a is a relaxed instance of C; Such a non-deterministic algorithm that, given an $\mathcal{EL}$-KB $\mathcal{K}$, an individual a, an $\mathcal{EL}$-concept description C, a similarity measure $\sim$, and a similarity degree t, computes whether a is a relaxed instance of C w.r.t. $\sim$ and t, is given in Figure 3. The algorithm works by computing the k-msc of a with $k = \text{rd}(C)$ and then guessing a generalized concept F of E with similarity $F \sim C \geq t$, if such a concept exists.

Corollary 10. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-knowledge base, C be an $\mathcal{EL}$ concept in $\sim$-normal form, a be an individual in $\mathcal{K}$, $\sim$ be a structural equivalence invariant similarity measure fulfilling Property 1 from Lemma 9 and $t \in [0, 1]$. Then* relaxed-instance?$(a, C, \mathcal{K}, \sim, t)$ *computes whether $a \in_t^\sim C$ w.r.t. $\mathcal{K}$.*

Proof. Lemma 9 shows that a mimic of C w.r.t. a is a generalized concept of $E = k\text{-msc}(a)$ for $k = \text{rd}(C)$. Thus, if the algorithm returns false, we know that no generalized concept F exists with $C \sim F \geq t$, and in particular also the mimic of C w.r.t. a must have a similarity of less than t to C. Thus no concept that has a as an instance is similar enough to C and thus $a \notin_t^\sim C$. If the algorithm returns true, the guessed concept F shows $a \in_t^\sim C$, since a is an instance of F and $F \sim C \geq t$. □

Guessing a generalized concept F of a concept description E can be done in time linear to size $\|E\|$ of E by recursively guessing for each concept name and each existential restriction in E whether they should occur in F or not. However, the size of $E = k\text{-msc}(a)$ can be exponential in k and polynomial in $\|\mathcal{K}\|$ [Peñaloza and Turhan, 2011]. Since $k = \text{rd}(C)$ is bounded linearly by $\|C\|$, the algorithm runs in NEXP-time (provided that $\sim$ can be computed in NEXP-time). However, the algorithm runs in NP-time in $\|\mathcal{K}\|$ (provided that $\sim$ can be computed in NP), and since C is an input concept, its role-depth can be assumed to be rather low. Hence, we conjecture that the exponential blow-up of the msc usually plays only a minor role in practical applications.

To obtain a deterministic algorithm, the mimic of C w.r.t. a can be computed by enumerating all generalized concepts of $k\text{-msc}(a)$ and taking one with the maximal similarity to C. Of course, there are a few optimizations possible: if the individual a belongs to C, we can directly return true, since the mimic will always be C itself. If we find a generalized concept F with $C \sim F \geq t$, we can stop to search for even more similar concepts and return true. And finally, if we find a mimic D for an individual a with $C \sim D \geq t$, we know that all other instances of D besides a will be relaxed instances of C as well, without needing to compute *their* mimics.

5 Conclusions

In this paper we have studied a new inference service for description logics, which consists in computing the relaxed instances of a given query concept C w.r.t. a similarity measure $\sim$ and a similarity degree t. This problem is relevant to the field of artificial intelligence in general, and to knowledge representation and reasoning in particular, as it provides a formal and unambiguous method for computing answers for a relaxed notion of instance query. Thus it is useful for ontology-based applications that need to obtain answers that fit the query criteria only to a certain degree.

The inference has two main degrees of freedom: in the choice of the similarity measure, and in the degree of relaxation of the concept. The similarity degree t allows the user to tune how strict or relaxed the answers provided are: a degree closer to 1 will yield only a few additional individuals that do not belong to C, while relaxing to a level closer to 0 yields almost all individuals in the ontology as relaxed instances. The similarity measure provides also criteria on how the relaxed instances are obtained. Intuitively, different similarity measures yield different weights on specific criteria. For example, one could require that small changes inside existential restrictions produce a high level of dissimilarity.

As a step for computing the relaxed instances of a concept C, we introduced the problem of finding a mimic of the query concept C w.r.t. a given individual a. Such a mimic is a concept D that contains a as instance, and has the highest similarity possible to C; i.e., it is a concept that tries to imitate C while containing a. Computing mimics w.r.t. all individuals appearing in an ontology provides a method for finding the relaxed instances of C.

The problem of finding a mimic is non-trivial. We have provided an algorithm capable of finding such a mimic, based on the msc of an individual a for certain structural similarity measures. While this computation is expensive, some obvious optimizations can be used to reduce the number of times these mimics are constructed.

As future work, we plan to expand on the two main inference problems described in this paper. First, we intend to improve the algorithms that compute the mimics. On the one hand, we will try to find one such mimic efficiently. On the other, it would also be beneficial to compute the most general mimic, if it exists; this concept would have the most possible instances, and hence would be useful as an optimization approach. Second, we will try to find tight complexity bounds on the problems of computing relaxed instances and finding mimics for a given concept. Third, we plan to obtain a better understanding on the properties of similarity measures that can impact (positively or negatively) on the complexity and run-time of solving these problems. As we have mentioned before, both inferences depend strongly on the similarity measure chosen. However, we do not know precisely which measures would allow for better results, be it in terms of execution time, or in terms of precision and fine-grained tuning.

References

[Alvarez and Yan, 2011] M. A. Alvarez and C. Yan. A graph-based semantic similarity measure for the gene ontology. *J. Bioinformatics and Computational Biology*, 9(6):681–695, 2011.

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Borgida *et al.*, 2005] A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*, volume 147 of *CEUR Workshop Proceedings*, 2005.

[Borgwardt and Peñaloza, 2012] S. Borgwardt and R. Peñaloza. Undecidability of fuzzy description logics. In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-12)*, pages 232–242. AAAI Press, 2012.

[Borgwardt *et al.*, 2012] S. Borgwardt, F. Distel, and R. Peñaloza. How fuzzy is my fuzzy description logic? volume 7364 of *Lecture Notes In Artificial Intelligence*, pages 82–96. Springer-Verlag, 2012.

[Cerami and Straccia, 2013] M. Cerami and U. Straccia. On the (un)decidability of fuzzy description logics under lukasiewicz t-norm. *Inf. Sci.*, 227:1–21, 2013.

[d'Amato *et al.*, 2005] C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In *Proc. of Convegno Italiano di Logica Computazionale, CILC05*, 2005.

[Ecke *et al.*, 2013] A. Ecke, R. Peñaloza, and A.-Y. Turhan. Computing role-depth bounded generalizations in the description logic $\mathcal{ELOR}$. In *Proceedings of the 36th German Conference on Artificial Intelligence (KI 2013)*, volume 8077 of *Lecture Notes in Artificial Intelligence*, Koblenz, Germany, 2013. To appear.

[Gene Ontology Consortium, 2000] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[Haarslev *et al.*, 2012] V. Haarslev, K. Hidde, R. Möller, and M. Wessel. The RacerPro knowledge representation and reasoning system. *Semantic Web Journal*, 3(3):267–277, 2012.

[Kazakov *et al.*, 2012] Y. Kazakov, M. Krötzsch, and F. Simančík. ELK reasoner: Architecture and evaluation. In *Proceedings of the OWL Reasoner Evaluation Workshop (ORE'12)*, volume 858 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[Lehmann and Turhan, 2012] K. Lehmann and A.-Y. Turhan. A framework for semantic-based similarity measures for $\mathcal{EL}\mathcal{H}$-concepts. In *Proceedings of the 13th European Conference on Logics in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 307–319. Springer Verlag, 2012.

[Lord *et al.*, 2003] P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble. Investigating semantic similarity measures across the gene ontology: The relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–1283, 2003.

[Lutz *et al.*, 2003] C. Lutz, F. Wolter, and M. Zakharyaschev. Reasoning about concepts and similarity. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, 2003.

[Mistry and Pavlidis, 2008] M. Mistry and P. Pavlidis. Gene ontology term overlap as a measure of gene functional similarity. *BMC Bioinformatics*, 9, 2008.

[Motik *et al.*, 2009] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 web ontology language profiles. W3C Recommendation, 27 October 2009. `http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/`.

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

[Peñaloza and Turhan, 2011] R. Peñaloza and A.-Y. Turhan. A practical approach for computing generalization inferences in $\mathcal{EL}$. In *Proceedings of the 8th European Semantic Web Conference (ESWC'11)*, Lecture Notes in Computer Science. Springer-Verlag, 2011.

[Schlicker *et al.*, 2006] A. Schlicker, F. S. Domingues, J. Rahnenführer, and T. Lengauer. A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinformatics*, 7:302, 2006.

[Tsarkov and Horrocks, 2006] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR-06)*, 2006. FaCT++ download page: `http://owl.man.ac.uk/factplusplus/`.