EPiC Series in Computing

Volume 55, 2018, Pages 1–14

GCAI-2018. 4th Global Conference on Artificial Intelligence

Standard and Non-Standard Inferences in the Description Logic $\mathcal{FL}_0$ Using Tree Automata*

Franz Baader, Oliver Fernández Gil, and Maximilian Pensel

Theoretical Computer Science, TU Dresden
01062 Dresden, Germany
firstname.lastname@tu-dresden.de

Abstract

Although being quite inexpressive, the description logic (DL) $\mathcal{FL}_0$, which provides only conjunction, value restriction and the top concept as concept constructors, has an intractable subsumption problem in the presence of terminologies (TBoxes): subsumption reasoning w.r.t. acyclic $\mathcal{FL}_0$ TBoxes is coNP-complete, and becomes even ExpTime-complete in case general TBoxes are used. In the present paper, we use automata working on infinite trees to solve both standard and non-standard inferences in $\mathcal{FL}_0$ w.r.t. general TBoxes. First, we give an alternative proof of the ExpTime upper bound for subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes based on the use of looping tree automata. Second, we employ parity tree automata to tackle non-standard inference problems such as computing the least common subsumer and the difference of $\mathcal{FL}_0$ concepts w.r.t. general TBoxes.

1 Introduction

In the early days of DL research, the inexpressive DL $\mathcal{FL}_0$, which has only conjunction, value restriction and the top concept as concept constructors, was considered to be the smallest possible DL. In fact, when providing a formal semantics for so-called property edges of semantic networks in the first DL system KL-ONE [12], value restrictions were used. For this reason, the language for constructing concepts in KL-ONE and all of the other early DL systems [11, 21, 19, 27] contained $\mathcal{FL}_0$. It came as a surprise when it was shown that subsumption reasoning w.r.t. acyclic $\mathcal{FL}_0$ terminologies (TBoxes) is coNP-hard [20]. The complexity increases when more expressive forms of TBoxes are used: for cyclic TBoxes to PSpace [1, 15] and for general TBoxes consisting of general concept inclusions (GCIs) even to ExpTime [2]. Thus, w.r.t. general TBoxes, subsumption reasoning in $\mathcal{FL}_0$ is as hard as in $\mathcal{ALC}$, its closure under negation. These negative complexity results for $\mathcal{FL}_0$ were one of the reasons why the attention in DL research shifted from $\mathcal{FL}_0$ to $\mathcal{EL}$, which is obtained from $\mathcal{FL}_0$ by replacing value restriction with existential restriction as a constructor. In fact, subsumption reasoning in $\mathcal{EL}$ stays polynomial even in the presence of general TBoxes [13, 2]. In addition to standard inferences such as the subsumption and the instance problem, also various non-standard inferences have been

*Supported by the DFG Research Training Group 1763 (QuantLA) and the DFG grant BA 1122/20-1.

investigated in detail for $\mathcal{EL}$ [28, 16, 18]. For $\mathcal{FL}_0$, there are very few results that consider general TBoxes. For the ExpTime-completeness result for subsumption in [2], the ExpTime upper bound is actually inherited from the more expressive DL $\mathcal{ALC}$. Dedicated subsumption algorithms and nonstandard inferences have been considered mostly for the case without TBoxes or w.r.t. a restricted form of TBoxes in $\mathcal{FL}_0$ and its extension by number restrictions [1, 7, 5, 6, 9].

In the present paper, we use automata working on infinite trees to solve both standard and non-standard inferences in $\mathcal{FL}_0$ w.r.t. general TBoxes. First, we introduce the so-called *least functional model* of an $\mathcal{FL}_0$ concept w.r.t. an $\mathcal{FL}_0$ TBox, and prove that subsumption corresponds to inclusion of least functional models. Then we show that such least functional models can be represented using so-called looping tree automata (LTAs), and use this result to reduce the subsumption problem to the emptiness problem for LTAs. Since the constructed automata are of exponential size and the emptiness problem for LTAs can be decided in linear time, this yields an alternative proof of the ExpTime upper bound for subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes. Note that, in contrast to the case of acyclic or cyclic TBoxes, where automata on finite or infinite *words* can be employed to decide subsumption [1], GCIs require the use of automata working on *trees*.

In order to deal also with non-standard inferences such as computing the least common subsumer (lcs) [8, 17, 28] or the difference [24, 14] of two $\mathcal{FL}_0$ concepts w.r.t. a general $\mathcal{FL}_0$ TBox, the automata constructions need to be extended considerably. Instead of constructing an automaton that represents the least functional model of a fixed $\mathcal{FL}_0$ concept C, we are now interested in building an automaton accepting all trees representing least functional models of $\mathcal{FL}_0$ concepts. For this task, simple LTAs (which do not have an acceptance condition) are not sufficient; instead we use so-called parity tree automata (PTAs). We show that the constructed automaton can be used to decide whether the lcs or the difference of two given $\mathcal{FL}_0$ concepts w.r.t. a general $\mathcal{FL}_0$ TBox exists, and to actually compute the lcs or difference concept in case the answer is affirmative. For the DL $\mathcal{EL}$, the problem of deciding the existence of the lcs w.r.t. a general TBox was investigated in [28].

Due to the space constraints, we cannot provide all technical details and proofs in this paper. They can be found in the technical report [4].

2 Least functional models for $\mathcal{FL}_0$ TBoxes

In Description Logics, *concept constructors* are used to build complex *concepts* out of *concept names* (unary predicates) and *role names* (binary predicates). A particular DL is determined by the available constructors. Starting with fixed finite sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of concept and role names, respectively, the set of $\mathcal{FL}_0$ concepts is inductively defined as follows:

- $\top$ (top concept) and every concept name $A \in \mathsf{N_C}$ is an $\mathcal{FL}_0$ concept,
- if C, D are $\mathcal{FL}_0$ concepts and $r \in \mathsf{N_R}$ is a role name, then $C \sqcap D$ (conjunction) and $\forall r.C$ (value restriction) are $\mathcal{FL}_0$ concepts.

The *semantics* of $\mathcal{FL}_0$ is defined using first-order 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 a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each concept name A and a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role name r. This function is extended to $\mathcal{FL}_0$ concepts as follows:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \text{ and } (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},$$
$$(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}.$$

An $\mathcal{FL}_0$ *TBox* $\mathcal{T}$ is a finite set of *general concept inclusions (GCIs)*, which are expressions of the form $C \sqsubseteq D$ for $\mathcal{FL}_0$ concepts C, D. The interpretation $\mathcal{I}$ is a *model* of $\mathcal{T}$ if it satisfies all the GCIs in $\mathcal{T}$, i.e., $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all GCIs $C \sqsubseteq D$ in $\mathcal{T}$.

Given an $\mathcal{FL}_0$ TBox $\mathcal{T}$ and two $\mathcal{FL}_0$ concepts C, D, we say that C *is subsumed by* D (denoted as $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. These two concepts are *equivalent* (denoted as $C \equiv_{\mathcal{T}} D$) if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. If the TBox is empty, we write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\emptyset} D$ and $C \equiv_{\emptyset} D$. For $\mathcal{FL}_0$, the subsumption problem is ExpTime-complete: the complexity upper bound is inherited from the more expressive DL $\mathcal{ALC}$ [23], and the lower bound was shown in [2]. In the next section, we will actually describe an alternative way for showing the upper bound. It is based on the characterization of subsumption introduced in the remainder of this section.

In $\mathcal{FL}_0$, subsumption and equivalence can be nicely characterized using language inclusion. This characterization relies on transforming $\mathcal{FL}_0$ concepts into an appropriate normal form as follows. First, the semantics given to concept constructors in $\mathcal{FL}_0$ implies that value restrictions distribute over conjunction, i.e., for all $\mathcal{FL}_0$ concepts C, D and roles r it holds that $\forall r.(C \sqcap D) \equiv \forall r.C \sqcap \forall r.D$. Using this equivalence as a rewrite rule from left to right, each $\mathcal{FL}_0$ concept can be translated into an equivalent one that is either $\top$ or a conjunction of concepts of the form $\forall r_1 \dots \forall r_n.A$, where $\{r_1, \dots, r_n\} \subseteq \mathsf{N_R}$ and $A \in \mathsf{N_C}$. Such concepts can be abbreviated as $\forall w.A$, where w represents the word $r_1 \dots r_n$. Note that $n = 0$ means that w is the empty word ε, and thus $\forall \varepsilon.A$ corresponds to A. Furthermore, a conjunction of the form $\forall w_1.A \sqcap \dots \sqcap \forall w_m.A$ can be written as $\forall L.A$ where $L \subseteq \mathsf{N_R}^*$ is the finite language $\{w_1, \dots, w_m\}$. We use the convention that $\forall \emptyset.A$ corresponds to the top concept $\top$. Thus, if $\mathsf{N_C} = \{A_1, \dots, A_\ell\}$, then any two $\mathcal{FL}_0$ concepts C, D can be represented as

$$\begin{aligned} C &\equiv \forall K_1.A_1 \sqcap \dots \sqcap \forall K_\ell.A_\ell, \\ D &\equiv \forall L_1.A_1 \sqcap \dots \sqcap \forall L_\ell.A_\ell, \end{aligned} \tag{1}$$

where $K_1, L_1, \dots, K_\ell, L_\ell$ are finite languages over the alphabet of role names $\mathsf{N_R}$, i.e., finite subsets of $\mathsf{N_R}^*$. We call this representation the *language normal form (LNF)* of C, D. Using the LNF, it was shown in [9] that $C \sqsubseteq D$ iff $L_i \subseteq K_i$ for all $i, 1 \leq i \leq \ell$.

In the presence of a non-empty TBox $\mathcal{T}$, a similar characterization of subsumption and equivalence can be obtained [22], but now the definition of the languages needs to take the GCIs in $\mathcal{T}$ into account. Given an $\mathcal{FL}_0$ concept C and a TBox $\mathcal{T}$, we define

$$\mathcal{L}_{\mathcal{T}}(C) := \{(w, A) \in \mathsf{N_R}^* \times \mathsf{N_C} \mid C \sqsubseteq_{\mathcal{T}} \forall w.A\},$$

and call this set the *value restriction set* of C with respect to $\mathcal{T}$.

Proposition 1 ([22]). *Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox and C, D $\mathcal{FL}_0$ concepts. Then $C \sqsubseteq_{\mathcal{T}} D$ iff $\mathcal{L}_{\mathcal{T}}(D) \subseteq \mathcal{L}_{\mathcal{T}}(C)$.*

Characterizing an $\mathcal{FL}_0$ concept C whose LNF is $C \equiv \forall K_1.A_1 \sqcap \dots \sqcap \forall K_\ell.A_\ell$ by its value restriction set $\mathcal{L}_{\mathcal{T}}(C)$ generalizes determining the finite languages K_i $(1 \leq i \leq \ell)$ in the normalization step. Indeed, it is easy to see that, for the empty TBox, we have $\mathcal{L}_{\emptyset}(C) = \{(w, A_i) \mid w \in K_i, 1 \leq i \leq \ell\}$. For a general TBox $\mathcal{T}$, $\mathcal{L}_{\mathcal{T}}(C)$ may be infinite, as illustrated by the TBox $\mathcal{T} = \{A \sqsubseteq \forall r.A\}$, where e.g. $\mathcal{L}_{\mathcal{T}}(A) = \{(\varepsilon, A), (r, A), (rr, A), \dots\}$. Therefore, to determine subsumption of two concepts by comparing the respective value restriction sets, inclusion between infinite sets must be considered. In the next section, we will show how such infinite sets can be represented using infinite trees, and how tree automata can be used to test inclusion. But first, we introduce a semantic equivalent of the value restriction set, called least functional model.

Definition 2. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is called a *functional interpretation* if $\Delta^{\mathcal{I}} = \mathsf{N}_\mathsf{R}^*$ and $r^{\mathcal{I}} = \{(u, ur) \mid u \in \mathsf{N}_\mathsf{R}^*\}$ for all $r \in \mathsf{N}_\mathsf{R}$. We call such a functional interpretation a *functional model* of the $\mathcal{FL}_0$ concept C w.r.t. the $\mathcal{FL}_0$ TBox $\mathcal{T}$ if it is a model of $\mathcal{T}$ such that $\varepsilon \in C^{\mathcal{I}}$.

Calling such interpretations and models $\mathcal{I}$ *functional* is justified by the fact that roles are indeed interpreted as (total) functions: for every $u \in \mathsf{N}_\mathsf{R}^*$ and every $r \in \mathsf{N}_\mathsf{R}$, the word ur is the unique r-successor of u. As an immediate consequence of this functional interpretation of roles we have

$$w \in (\forall u.A)^{\mathcal{I}} \text{ iff } wu \in A^{\mathcal{I}}. \tag{2}$$

We define inclusion and intersection of functional interpretations as follows:

- $\mathcal{I} \subseteq \mathcal{J}$ if $A^{\mathcal{I}} \subseteq A^{\mathcal{J}}$ for all $A \in \mathsf{N}_\mathsf{C}$;
- $\mathcal{I} \cap \mathcal{J}$ is the unique functional interpretation that satisfies $A^{\mathcal{I}\cap\mathcal{J}} = A^{\mathcal{I}} \cap A^{\mathcal{J}}$ for all $A \in \mathsf{N}_\mathsf{C}$.

It is easy to see that functional models are closed under intersection, i.e., if $\mathcal{I}$ and $\mathcal{J}$ are both functional models of C w.r.t. $\mathcal{T}$, then so is their intersection $\mathcal{I} \cap \mathcal{J}$. This actually not only holds for binary intersection, but also for arbitrary intersection of functional models [22], which implies that there must exist a *least functional model*, i.e., a functional model $\mathcal{J}$ of C w.r.t. $\mathcal{T}$ such that $\mathcal{J} \subseteq \mathcal{I}$ holds for all functional models $\mathcal{I}$ of C w.r.t. $\mathcal{T}$. Here we describe a different and more constructive way of showing the existence of a least functional model, which is based on the use of the value restriction set.

Definition 3. Given an $\mathcal{FL}_0$ concept C and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, we define $\mathcal{I}_{C,\mathcal{T}} = (\mathsf{N}_\mathsf{R}^*, \cdot^{\mathcal{I}_{C,\mathcal{T}}})$ to be the functional interpretation satisfying

$$A^{\mathcal{I}_{C,\mathcal{T}}} = \{w \in \mathsf{N}_\mathsf{R}^* \mid (w, A) \in \mathcal{L}_{\mathcal{T}}(C)\} \text{ for all } A \in \mathsf{N}_\mathsf{C}.$$

In [4] we show that $\mathcal{I}_{C,\mathcal{T}}$ is indeed the least functional model of C w.r.t. $\mathcal{T}$.

Theorem 4. *Let C be an $\mathcal{FL}_0$ concept and $\mathcal{T}$ an $\mathcal{FL}_0$ TBox. Then the functional interpretation $\mathcal{I}_{C,\mathcal{T}}$ is the least functional model of C w.r.t. $\mathcal{T}$.*

As an immediate consequence of Definition 3 we obtain that

$$\mathcal{L}_{\mathcal{T}}(D) \subseteq \mathcal{L}_{\mathcal{T}}(C) \text{ iff } \mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}.$$

Thus, the characterization of subsumption formulated in Proposition 1 can be reformulated in terms of least functional models as follows.

Corollary 5. *Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox and C, D $\mathcal{FL}_0$ concepts. Then $C \sqsubseteq_{\mathcal{T}} D$ iff $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$.*

In the next section we show how least functional models can be represented using tree automata. In particular, this will allow us to reduce the subsumption problem in $\mathcal{FL}_0$ to a well-know decision problem for tree automata.

3 Least functional models as trees

Given a non-empty, finite set of symbols $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ and a finite set of labels L, an *L-labeled Σ-tree* is a mapping $t : \Sigma^* \to L$ that assigns a label $t(w) \in L$ to every node $w \in \Sigma^*$. Intuitively, the nodes of a Σ-tree t correspond to finite words in Σ^*, where the empty word ε represents the root of t and every node w has k children corresponding to the words

$w\sigma_1, \ldots, w\sigma_k$. Since for a non-empty alphabet Σ the set Σ^* of all words over Σ is infinite, Σ-trees are by definition infinite. The set of all L-labeled Σ-trees is denoted by $\mathfrak{T}^{\omega}_{\Sigma,L}$.

Functional interpretations can be represented as $2^{\mathsf{N_C}}$-labeled $\mathsf{N_R}$-trees and vice versa. In fact, let $\mathcal{I} = (\mathsf{N}_\mathsf{R}^*, \cdot^{\mathcal{I}})$ be a functional interpretation. Then we define $t_{\mathcal{I}} : \mathsf{N}_\mathsf{R}^* \to 2^{N_C}$ as $t_{\mathcal{I}}(w) := \{A \in \mathsf{N_C} \mid w \in A^{\mathcal{I}}\}$ for all $w \in \mathsf{N}_\mathsf{R}^*$. Conversely, let t be a $2^{\mathsf{N_C}}$-labeled $\mathsf{N_R}$-tree. Then we define the functional interpretation $\mathcal{I}_t = (\mathsf{N}_\mathsf{R}^*, \cdot^{\mathcal{I}_t})$ as $A^{\mathcal{I}_t} := \{w \in \mathsf{N}_\mathsf{R}^* \mid A \in t(w)\}$ for all $A \in \mathsf{N_C}$. Obviously, these two transformations are bijections that are inverse to each other. If $\mathcal{I} = \mathcal{I}_{C,\mathcal{T}}$ is the least functional model of the $\mathcal{FL}_0$ concept C w.r.t. the $\mathcal{FL}_0$ TBox $\mathcal{T}$, then we use $T_{C,\mathcal{T}}$ to denote the corresponding tree $t_{\mathcal{I}}$.

In case the set of role names is non-empty (which we will always assume in the following), the tree $T_{C,\mathcal{T}}$ is infinite. Our goal is now to give a finite representation of such trees using tree automata [26].

Definition 6 (Looping tree automaton (LTA))**.** A *looping tree automaton* is a tuple $\mathcal{A} = (\Sigma, Q, L, \Theta, I)$ where $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ is a finite set of symbols, Q is a finite set of states, L is a finite set of labels, $\Theta \subseteq Q \times L \times Q^k$ is the transition relation and $I \subseteq Q$ is a set of initial states. A run of this automaton on an L-labeled Σ-tree t is a Q-labeled Σ-tree $\rho : \Sigma^* \to Q$ such that $\rho(\varepsilon) \in I$ and $(\rho(w), t(w), \rho(w\sigma_1), \ldots, \rho(w\sigma_k)) \in \Theta$ for all $w \in \Sigma^*$. The tree language $\mathcal{L}(\mathcal{A})$ *recognized* by $\mathcal{A}$ is the set of all L-labeled Σ-trees t such that $\mathcal{A}$ *accepts* t, i.e., such that $\mathcal{A}$ has a run on t.

In general, LTAs recognize *sets* of trees. In order to uniquely represent the tree $T_{C,\mathcal{T}}$, we consider LTAs recognizing singleton sets.

Definition 7. Let $\mathcal{A} = (\Sigma, Q, L, \Theta, I)$ be a looping tree automaton. We say that $\mathcal{A}$ *represents the L-labeled Σ-tree t* if $\mathcal{L}(\mathcal{A}) = \{t\}$.

As shown in [3], the trees that can be represented in this way are exactly the *regular trees*, where an infinite tree is regular if (up to isomorphism) it contains only finitely many distinct subtrees [25].

To construct an automaton that represents the tree $T_{C,\mathcal{T}}$, we first construct an LTA that accepts exactly the tree representations of functional models of C w.r.t. $\mathcal{T}$. We assume without loss of generality that C and all concepts occurring in $\mathcal{T}$ are in *word normal form (WNF)*, i.e., they are conjunctions of value restrictions $\forall w.A$ where $w \in \mathsf{N}_\mathsf{R}^*$ and $A \in \mathsf{N_C}$. Given an $\mathcal{FL}_0$ concept $E = \forall w_1.A_1 \sqcap \ldots \sqcap \forall w_n.A_n$ in WNF, we define $\widehat{E} := \{\forall w_1.A_1, \ldots, \forall w_n.A_n\}$ and

$$val(E) := \{\forall u_i.A_i \mid 1 \leq i \leq n \text{ and } u_i \text{ is a suffix of } w_i\}.$$

The latter definition is extended to TBoxes by setting

$$val(\mathcal{T}) := \bigcup_{E \sqsubseteq F \in \mathcal{T}} val(E) \cup val(F).$$

Definition 8. Let C be an $\mathcal{FL}_0$ concept and $\mathcal{T}$ an $\mathcal{FL}_0$ TBox such that C and all concepts occurring in $\mathcal{T}$ are in WNF. We set $V_{C,\mathcal{T}} := val(C) \cup val(\mathcal{T})$ and define the LTA $\mathcal{A}_{C,\mathcal{T}} = (\Sigma, Q, L, \Theta, I)$ as follows:

- $\Sigma := \mathsf{N_R} = \{r_1, \ldots, r_k\}$, $Q := 2^{V_{C,\mathcal{T}}}$, and $L := 2^{\mathsf{N_C}}$;
- $I := \{X \in Q \mid \widehat{C} \subseteq X\}$;
- $\Theta := \{(q, \sigma, q_1, \ldots, q_k) \mid$ the properties (1),…,(4) hold$\}$, where

1. $\widehat{E} \subseteq q$ implies $\widehat{F} \subseteq q$ for all $E \sqsubseteq F \in \mathcal{T}$,
2. $\forall r_i u.A \in q$ implies $\forall u.A \in q_i$ for all $i, 1 \leq i \leq k$,
3. $\forall u.A \in q_i$ and $\forall r_i u.A \in V_{C,\mathcal{T}}$ implies $\forall r_i u.A \in q$ for all $i, 1 \leq i \leq k$,
4. $A \in \sigma$ iff $\forall \varepsilon.A \in q$ for all $A \in \mathsf{N_C}$.

Intuitively, the set Q consists of all possible sets of relevant value restrictions that may be satisfied by a node of a functional model of C w.r.t. $\mathcal{T}$. In particular, the definition of I ensures that only trees of functional interpretations for which ε belongs to C are accepted. Property (1) in the definition of Θ ensures that the GCIs of $\mathcal{T}$ are satisfied by the functional interpretation corresponding to the tree. Properties (2) and (3) basically realize the fact that, in a functional interpretation, an element w belongs to the value restriction $\forall r_i u.A$ iff its unique r_i successor wr_i belongs to $\forall u.A$. Finally, property (4) expresses that A and $\forall \varepsilon.A$ are equivalent. With this intuition, it is not hard to show that $\mathcal{A}_{C,\mathcal{T}}$ is the automaton we are looking for (see [22] for details).

Lemma 9. *$\mathcal{L}(\mathcal{A}_{C,\mathcal{T}}) = \{t_{\mathcal{I}} \mid \mathcal{I}$ is a functional model of C w.r.t. $\mathcal{T}\}$.*

In order to obtain an automaton that represents $T_{C,\mathcal{T}}$, we restrict the automaton $\mathcal{A}_{C,\mathcal{T}}$ to the transitions with minimal states, where states are ordered using set inclusion. However, before we can do this, we need to remove states that cannot occur in a run. Intuitively, this is necessary to avoid using a minimal state that leads into a dead-end. We say that a state q of an LTA $\mathcal{A}$ is *inactive* if there is no run of $\mathcal{A}$ that contains q. As shown in [10], the set of inactive states of an LTA can be computed in linear time.

Definition 10. Let $\mathcal{A}_{C,\mathcal{T}} = (\Sigma, Q, L, \Theta, I)$ be the LTA defined in Definition 8, $Q^+ \subseteq Q$ be the states of $\mathcal{A}_{C,\mathcal{T}}$ that are not inactive, and $\Theta^+ := \Theta \cap (Q^+ \times L \times (Q^+)^k)$ the transitions of $\mathcal{A}_{C,\mathcal{T}}$ that do not use inactive states. We define the automaton $\widehat{\mathcal{A}}_{C,\mathcal{T}} = (\Sigma, Q^+, L, \widehat{\Theta}, \widehat{I})$ as follows:

$$\begin{aligned} \widehat{I} &:= \{q \in I \cap Q^+ \mid q \subseteq q' \text{ for all } q' \in I \cap Q^+\}; \\ \widehat{\Theta} &:= \{(q, \sigma, q_1, \ldots, q_k) \in \Theta^+ \mid \; q_1 \subseteq q'_1, \ldots, q_k \subseteq q'_k \\ &\qquad\qquad\qquad\qquad \text{for all } (q, \sigma, q'_1, \ldots, q'_k) \in \Theta^+\}. \end{aligned}$$

The reason why the least states w.r.t. set inclusion required by the definitions of $\widehat{I}$ and $\widehat{\Theta}$ exist is basically that runs are closed under intersection, i.e., if ρ_1, ρ_2 are runs of $\mathcal{A}_{C,\mathcal{T}}$ on some trees, then $\rho_1 \cap \rho_2$ with $(\rho_1 \cap \rho_2)(w) = \rho_1(w) \cap \rho_2(w)$ is also a run of $\mathcal{A}_{C,\mathcal{T}}$. The automaton $\widehat{\mathcal{A}}_{C,\mathcal{T}}$ is in fact deterministic: it has exactly one initial state and for every pair $(q, \sigma) \in Q^+ \times L$ exactly one transition with these two first components. Together with property (4) in Definition 8 this implies that $\widehat{\mathcal{A}}_{C,\mathcal{T}}$ accepts exactly one tree, and it is not hard to show that this tree is $T_{C,\mathcal{T}}$ (see [22] for details).

Theorem 11. *The automaton $\widehat{\mathcal{A}}_{C,\mathcal{T}}$ represents the tree $T_{C,\mathcal{T}}$ that corresponds to the least functional model of C w.r.t. $\mathcal{T}$. In particular, this implies that $T_{C,\mathcal{T}}$ is a regular tree.*

By Corollary 5, subsumption corresponds to inclusion between least functional models. The latter can be checked using a product construction on the corresponding automata. Assume that $\widehat{\mathcal{A}}_{C,\mathcal{T}} = (\Sigma, Q_C, L, \Theta_C, I_C)$ and $\widehat{\mathcal{A}}_{D,\mathcal{T}} = (\Sigma, Q_D, L, \Theta_D, I_D)$ are the automata respectively representing the trees $T_{C,\mathcal{T}}$ and $T_{D,\mathcal{T}}$, as constructed in Definition 10. We define a new automaton $\mathcal{P}_{C,D,\mathcal{T}} = (\Sigma, Q, \{\emptyset\}, \Theta, I)$ that accepts the infinite $\{\emptyset\}$-labeled Σ-tree $t_\emptyset$ iff $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$:

- $Q := Q_C \times Q_D$ and $I := I_C \times I_D$;

- $((q^{(1)}, q^{(2)}), \emptyset, (q_1^{(1)}, q_1^{(2)}), \ldots, (q_k^{(1)}, q_k^{(2)})) \in \Theta$ iff there are $\sigma^{(1)}, \sigma^{(2)} \in \Sigma$ such that
  - $\sigma^{(1)} \supseteq \sigma^{(2)}$,
  - $(q^{(1)}, \sigma^{(1)}, q_1^{(1)}, \ldots, q_k^{(1)}) \in \Theta_C$ and $(q^{(2)}, \sigma^{(2)}, q_1^{(2)}, \ldots, q_k^{(2)}) \in \Theta_D$.

We have $\mathcal{L}(\mathcal{P}_{C,D,\mathcal{T}}) \neq \emptyset$ iff $\mathcal{L}(\mathcal{P}_{C,D,\mathcal{T}}) = \{t_\emptyset\}$ iff $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$. Since the emptiness problem for looping tree automata is decidable in linear time [10] and the automata $\widehat{\mathcal{A}}_{C,\mathcal{T}}$ and $\widehat{\mathcal{A}}_{D,\mathcal{T}}$, and thus also $\mathcal{P}_{C,D,\mathcal{T}}$, have a size that is exponential in the size of $C, D, \mathcal{T}$, this yields an exponential-time algorithm for checking subsumption.

Corollary 12. *Subsumption in $\mathcal{FL}_0$ w.r.t. TBoxes is in ExpTime.*

A product construction similar to the one employed above can also be used to obtain an automaton representing the intersection of two least functional models. Given least functional models $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{D,\mathcal{T}}$, their *intersection* is the functional interpretation $\mathcal{I} = (\mathsf{N}_\mathsf{R}^*, \cdot^\mathcal{I})$ that satisfies $A^\mathcal{I} = A^{\mathcal{I}_{C,\mathcal{T}}} \cap A^{\mathcal{I}_{D,\mathcal{T}}}$ for all $A \in \mathsf{N}_\mathsf{C}$. For the corresponding trees this means that $t_\mathcal{I}(w) = T_{C,\mathcal{T}}(w) \cap T_{D,\mathcal{T}}(w)$ for all $w \in \mathsf{N}_\mathsf{R}^*$. It is easy to see that $\mathcal{I}$ is again a model of $\mathcal{T}$. However, it need not be the least functional model of some $\mathcal{FL}_0$-concept (see Section 5 for an example). As a first step towards deciding whether it is or not, we show in [4] that $t_\mathcal{I}$ can be represented by a looping automaton.

Lemma 13. *There is an LTA $\mathcal{P}^\cap_{C,D,\mathcal{T}}$ of size exponential in the size of $\mathcal{T}, C, D$ such that $\mathcal{L}(\mathcal{P}^\cap_{C,D,\mathcal{T}}) = \{t_\mathcal{I}\}$ where $\mathcal{I}$ is the intersection of $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{D,\mathcal{T}}$.*

The functional interpretation $\mathcal{I}$ obtained as the intersection of $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{D,\mathcal{T}}$ may or may not be the least functional model of some $\mathcal{FL}_0$ concept E w.r.t. $\mathcal{T}$. In the next section, we show how this can be decided by constructing an automaton that accepts exactly the least functional models w.r.t. $\mathcal{T}$, i.e., the tree language $\mathcal{LF}(\mathcal{T}) := \{T_{C,\mathcal{T}} \mid C \text{ is an } \mathcal{FL}_0 \text{ concept}\}$.

In Section 5 we use this result to show that the existence of the least common subsumer of two $\mathcal{FL}_0$ concepts w.r.t. an $\mathcal{FL}_0$ TBox is decidable.

4 An automaton accepting all least functional models

Given an $\mathcal{FL}_0$ TBox $\mathcal{T}$, we want to construct a tree automaton $\widehat{\mathcal{A}}_\mathcal{T}$ recognizing the tree language $\mathcal{LF}(\mathcal{T})$ of all least functional models w.r.t. $\mathcal{T}$. In the following, we assume that $\mathcal{T}$ is an arbitrary, but fixed $\mathcal{FL}_0$ TBox.

The looping automata employed in the previous section are not sufficient to construct $\widehat{\mathcal{A}}_\mathcal{T}$ since we need to express things like finiteness, which requires an appropriate acceptance condition. It turns out that *parity* tree automata [26] are well-suited for our purpose since they have the required closure properties (intersection, complement, projection) and there exists a fine-grained analysis of the complexity of these operations.

Definition 14 (Parity tree automaton (PTA))**.** A *parity tree automaton* is a tuple $\mathcal{A} = (\Sigma, Q, L, \Theta, I, c)$, where $(\Sigma, Q, L, \Theta, I)$ is an LTA and $c : Q \to \mathbb{N}$ is a mapping that assigns to each state in Q a number, called its *priority*. A run ρ of $\mathcal{A}$ on a tree t is called *accepting* if for all paths in the tree ρ the maximum of the priorities appearing infinitely often in the path is an even number. The tree language $\mathcal{L}(\mathcal{A})$ *recognized* by $\mathcal{A}$ is the set of all L-labeled Σ-trees t such that $\mathcal{A}$ has an accepting run on t.

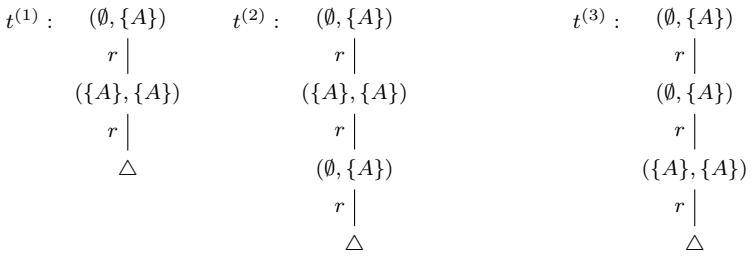

Figure 1: Trees belonging to $\mathcal{L}_\mathcal{T}$

In addition to the usual set operations *complementation* and *intersection*, we will need *projection*. A projection is a mapping h from an alphabet Σ to an alphabet Σ'. It is applied to a tree t by applying it to the label of each node in t. Given a set $S \subseteq \mathfrak{T}^\omega_{\Sigma,L}$, the application of h to S yields the set $h(S) = \{h(t) \mid t \in S\}$, where $h(t)$ stands for the composition $h \circ t$ of the functions h and t. We say that $h(S)$ is obtained from S by projection.

Proposition 15 ([26])**.** *The class of languages recognized by parity tree automata is closed under complement, intersection and projection.*

Our first step towards constructing $\widehat{\mathcal{A}}_\mathcal{T}$ is to construct a PTA $\mathcal{A}_\mathcal{T}$ that recognizes $\mathcal{FL}_0$ concepts C together with (not necessarily least) functional models of C w.r.t. $\mathcal{T}$. Here "together with" means that we represent C and its functional model within a single tree, which has tuples as labels. To be more precise, we consider infinite L'-labeled Σ-trees t where the label set is $L' = 2^{\mathsf{N_C}} \times 2^{\mathsf{N_C}}$, i.e., the nodes of t are labeled with tuples of the form $(\sigma_\mathsf{c}, \sigma_\mathsf{i})$ where each component is a set of concept names. By applying the projections $h_\mathsf{c}(\sigma_\mathsf{c}, \sigma_\mathsf{i}) = \sigma_\mathsf{c}$ and $h_\mathsf{i}(\sigma_\mathsf{c}, \sigma_\mathsf{i}) = \sigma_\mathsf{i}$ to such a tree t, we obtain the L-labeled trees t_c and t_i, where $L = 2^{\mathsf{N_C}}$. In addition, if $t_\mathsf{c}(w) \neq \emptyset$ for only finitely many $w \in \mathsf{N}_\mathsf{R}^*$, then t_c induces an $\mathcal{FL}_0$ concept $C(t_\mathsf{c})$ as follows:

$$C(t_\mathsf{c}) := \bigsqcap_{w \in \mathsf{N}_\mathsf{R}^*} \bigsqcap_{A \in t_\mathsf{c}(w)} \forall w.A. \tag{3}$$

Let $\mathfrak{T}^{fin}_{\mathsf{N_R},L'}$ denote the subset of $\mathfrak{T}^\omega_{\mathsf{N_R},L'}$ consisting of the trees satisfying this finiteness restriction. Our goal is to construct a PTA $\mathcal{A}_\mathcal{T}$ that recognizes $\mathcal{L}_\mathcal{T} := \{t \in \mathfrak{T}^{fin}_{\mathsf{N_R},L'} \mid t_\mathsf{i}$ represents a functional model of $C(t_\mathsf{c})$ w.r.t. $\mathcal{T}\}$.

Example 16. Let $\mathsf{N_C} = \{A\}$ and $\mathsf{N_R} = \{r\}$, and consider the $\mathcal{FL}_0$ TBox $\mathcal{T} = \{\forall r.A \sqsubseteq A\}$. In addition, consider the three trees depicted in Fig. 1, where the symbol $\triangle$ stands for the infinite $\mathsf{N_R}$-tree whose nodes are labeled with $(\emptyset, \emptyset)$. Obviously, these trees belong to $\mathfrak{T}^{fin}_{\mathsf{N_R},L'}$, and thus the concepts induced by the first components of the labels are well-defined. Indeed, we have $C(t^{(1)}_\mathsf{c}) = \forall r.A = C(t^{(2)}_\mathsf{c})$ and $C(t^{(3)}_\mathsf{c}) = \forall rr.A$. The functional interpretations represented by $t^{(1)}_\mathsf{i}$, $t^{(2)}_\mathsf{i}$, and $t^{(3)}_\mathsf{i}$ satisfy $A^{\mathcal{I}_{t^{(1)}_\mathsf{i}}} = \{\varepsilon, r\}, A^{\mathcal{I}_{t^{(2)}_\mathsf{i}}} = \{\varepsilon, r, rr\} = A^{\mathcal{I}_{t^{(3)}_\mathsf{i}}}$. It is easy to see that $\mathcal{I}_{t^{(1)}_\mathsf{i}}$ is the least functional model of $\forall r.A$ w.r.t. $\mathcal{T}$. The functional interpretations $\mathcal{I}_{t^{(2)}_\mathsf{i}}$ and $\mathcal{I}_{t^{(3)}_\mathsf{i}}$ are identical. This interpretation is a functional model of $\forall r.A$ w.r.t. $\mathcal{T}$, but not the least one, whereas it is the least functional model of $\forall rr.A$ w.r.t. $\mathcal{T}$.

Our construction of $\mathcal{A}_\mathcal{T}$ is similar to the one of $\mathcal{A}_{C,\mathcal{T}}$, but now the concept C is not given, but needs to be guessed. This is done in the first component of the states of $\mathcal{A}_\mathcal{T}$. The second

component together with the parity acceptance condition ensures that only *finite* concepts are guessed. The third component of the states basically corresponds to a state of $\mathcal{A}_{C,\mathcal{T}}$, and thus ensures the functional model condition.

Definition 17. Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox such that all concepts occurring in $\mathcal{T}$ are in WNF. We set $V_{\mathsf{N_C},\mathcal{T}} := \mathsf{N_C} \cup \mathit{val}(\mathcal{T})$ and define the PTA $\mathcal{A}_\mathcal{T} := (\Sigma, Q_\mathcal{T}, L', \Theta_\mathcal{T}, I_\mathcal{T}, c)$ as follows:

- $Q_\mathcal{T} := \{(q_\mathsf{c}, \ell, q) \in 2^{\mathsf{N_C}} \times \{0,1\} \times 2^{V_{\mathsf{N_C},\mathcal{T}}} \mid \ q_\mathsf{c} \subseteq q \wedge (\ell = 0 \implies q_\mathsf{c} = \emptyset)\}$;
- $\Sigma := \mathsf{N_R} = \{r_1, \dots, r_k\}$ and $\mathcal{I}_\mathcal{T} := Q_\mathcal{T}$;
- $\Theta_\mathcal{T} := \{((q_\mathsf{c}, \ell, q), (\sigma_\mathsf{c}, \sigma_\mathsf{i}), (q_{\mathsf{c}_1}, \ell_1, q_1), \dots, (q_{\mathsf{c}_k}, \ell_k, q_k)) \mid$ the properties (1), (2), (3) hold$\}$, where
  1. $(q, \sigma_\mathsf{i}, q_1, \dots, q_k) \in \Theta$ (see Definition 8 with $V_{C,\mathcal{T}}$ replaced by $V_{\mathsf{N_C},\mathcal{T}}$),
  2. $\ell = 0$ implies $\ell_1 = 0, \dots, \ell_k = 0$,
  3. $q_\mathsf{c} = \sigma_\mathsf{c}$;
- $c(q_\mathsf{c}, \ell, q) = \ell$, for all $(q_\mathsf{c}, \ell, q) \in Q_\mathcal{T}$.

Assume that ρ is an accepting run of $\mathcal{A}_\mathcal{T}$ on a tree t. Then, by König's lemma, (2) together with the parity acceptance condition ensures that only finitely many nodes in the run are labeled with a state whose middle component is 1. In combination with the condition $\ell = 0 \Rightarrow q_\mathsf{c} = \emptyset$ in the definition of $Q_\mathcal{T}$ this ensures that only finitely many nodes in ρ are labeled with a state whose first component is non-empty.[1] Consequently, (3) implies that $t \in \mathfrak{T}^{\mathit{fin}}_{\mathsf{N_R},L'}$, and thus $C(t_\mathsf{c})$ is a well-defined $\mathcal{FL}_0$ concept.

It remains to see why t_i is a functional model of $C(t_\mathsf{c})$ w.r.t. $\mathcal{T}$. Basically, this is a consequence of Lemma 9 since, in the third component, $\mathcal{A}_\mathcal{T}$ behaves like $\mathcal{A}_{C,\mathcal{T}}$ for $C = C(t_\mathsf{c})$. Note that the condition $q_\mathsf{c} \subseteq q$ in the definition of $Q_\mathcal{T}$ together with (3) realizes the condition $\widehat{C} \subseteq X$ in the definition of the set of initial states of $\mathcal{A}_{C,\mathcal{T}}$. Given this intuition, it is not hard to see that $\mathcal{A}_\mathcal{T}$ recognizes the tree language $\mathcal{L}_\mathcal{T}$ (see [4] for a formal proof).

Proposition 18. *Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox. Then, $\mathcal{L}(\mathcal{A}_\mathcal{T}) = \mathcal{L}_\mathcal{T}$.*

At first sight one might think that going from $\mathcal{A}_\mathcal{T}$ to $\widehat{\mathcal{A}}_\mathcal{T}$ can be achieved in a way similar to our construction of $\widehat{\mathcal{A}}_{C,\mathcal{T}}$ out of $\mathcal{A}_{C,\mathcal{T}}$ in the previous section. However, the minimization approach employed in Section 3 does not work if the concept C is not fixed from the outset, but is guessed during the run of the automaton. In fact, assume that ρ is a run of $\mathcal{A}_\mathcal{T}$. Whether a concept name appearing in the third component of a state $\rho(w)$ violates minimality or not also depends on what is guessed in the first component of states $\rho(wv)$ labeling successors wv of w.

Instead of trying to minimize $\mathcal{A}_\mathcal{T}$, we follow a different approach that uses the closure properties of PTAs. Basically, given $\mathcal{A}_\mathcal{T}$, it is not hard to construct an automaton $\mathcal{A}_{\mathsf{nl}}$ that accepts all $\mathcal{FL}_0$ concepts C together with their *non-least* functional models. By applying complementation and the other closure properties, we can then use $\mathcal{A}_{\mathsf{nl}}$ and $\mathcal{A}_\mathcal{T}$ to construct the desired automaton $\widehat{\mathcal{A}}_\mathcal{T}$. To be more precise, our construction of $\widehat{\mathcal{A}}_\mathcal{T}$ proceeds as follows:

[1] Note that dispensing with the middle component of the states and defining $c(q) = 0$ iff the first component of q is empty would not work. In fact, a state with empty first component may have a successor node where this component is non-empty. Thus, though in every path there would be only finitely many states with non-empty first component, one could still have infinitely many such states in the whole tree, e.g., at all nodes $r^n s$ for $n \geq 0$.

1. The automaton $\mathcal{A}_{\mathsf{nl}}$ needs to accept exactly those L'-labeled $\mathsf{N_R}$-trees t such that t_{i} represents a non-least functional model of $C(t_{\mathsf{c}})$ w.r.t. $\mathcal{T}$. To achieve this, we first construct an automaton that accepts all $\mathcal{FL}_0$ concepts C together with two functional models of C w.r.t. $\mathcal{T}$, where one is a witness for the fact that the other is not the least one. From this, we obtain $\mathcal{A}_{\mathsf{nl}}$ by projecting away the component corresponding to the witness. Details of this construction can be found in [4].

2. Once we have $\mathcal{A}_{\mathsf{nl}}$, we can obtain an automaton $\overline{\mathcal{A}_{\mathsf{nl}}}$ recognizing the complement language $\overline{\mathcal{L}(\mathcal{A}_{\mathsf{nl}})}$. Notice that, although $\mathcal{L}(\overline{\mathcal{A}_{\mathsf{nl}}})$ contains all trees t such that t_{i} represents the least functional model of $C(t_{\mathsf{i}})$, it also contains many other trees that do not encode a concept together with a functional model (e.g., trees that violate the finiteness restriction for the first component or where the second component does not encode a functional model of the concept represented by the first component). However, to filter out such "rough" trees, we can simply intersect the language accepted by $\overline{\mathcal{A}_{\mathsf{nl}}}$ with the one accepted by $\mathcal{A}_{\mathcal{T}}$.

3. Closure under intersection of the class of languages accepted by PTAs thus yields a PTA for the language $\mathcal{L}(\overline{\mathcal{A}_{\mathsf{nl}}}) \cap \mathcal{L}(\mathcal{A}_{\mathcal{T}})$, which consists exactly of those trees $t \in \mathfrak{T}^{fin}_{\mathsf{N_R},L'}$ such that t_{i} represents the least functional model of $C(t_{\mathsf{c}})$. Thus, the desired automaton $\widehat{\mathcal{A}}_{\mathcal{T}}$ is obtained by applying closure under the projection $h_{\mathsf{i}}(\sigma_{\mathsf{c}}, \sigma_{\mathsf{i}}) = \sigma_{\mathsf{i}}$ to the intersection automaton.

Summing up, we thus have proved the following main result of this paper.

Theorem 19. *Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox. Then we can effectively construct a PTA $\widehat{\mathcal{A}}_{\mathcal{T}}$ such that $\mathcal{L}(\widehat{\mathcal{A}}_{\mathcal{T}}) = \{T_{C,\mathcal{T}} \mid C$ is an $\mathcal{FL}_0$ concept$\}$.*

The complexity of closure operations and decision procedures for parity tree automata is determined not only by the number of states, but also by the number of different priorities used. Using known results on the complexity of closure operations on PTAs, we can show that $\widehat{\mathcal{A}}_{\mathcal{T}}$ has the following size.

Corollary 20. *The number of states of $\widehat{\mathcal{A}}_{\mathcal{T}}$ is double exponential and the number of priorities single exponential in the size of $\mathcal{T}$.*

Basically, this is due to the fact that the automaton $\mathcal{A}_{\mathcal{T}}$ has an exponential number of states in the size of $\mathcal{T}$ and a constant number of priorities. The second exponential blowup for the number of states comes from the complement operation as does the exponential blowup for the number of priorities (see [4] for details).

5 Non-standard inferences

As an application of the automaton $\widehat{\mathcal{A}}_{\mathcal{T}}$ constructed above, we consider two non-standard inferences for $\mathcal{FL}_0$ w.r.t. general TBoxes, the least common subsumer and the difference.

Least common subsumers

Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox and C, D $\mathcal{FL}_0$ concepts. The $\mathcal{FL}_0$ concept E is a *least common subsumer (lcs) of C and D w.r.t. $\mathcal{T}$* if

- $C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$, and
- for all $\mathcal{FL}_0$ concepts F such that $C \sqsubseteq_{\mathcal{T}} F$ and $D \sqsubseteq_{\mathcal{T}} F$ we have $E \sqsubseteq_{\mathcal{T}} F$.

As an immediate consequence of this definition we obtain that least common subsumers of C, D w.r.t. $\mathcal{T}$ are unique up to equivalence $\equiv_{\mathcal{T}}$, if they exist. This justifies talking about *the* lcs of C, D w.r.t. $\mathcal{T}$ and to denote it (i.e., some element of the equivalence class) as $E = \mathsf{lcs}_{\mathcal{T}}(C, D)$. We will see below that the lcs need not always exist. The following lemma, whose proof can be found in [4], characterizes the cases where it does.

Lemma 21. *Let C, D, E be $\mathcal{FL}_0$ concepts and $\mathcal{T}$ a general $\mathcal{FL}_0$ TBox. Then, E is the lcs of C and D w.r.t. $\mathcal{T}$ iff $\mathcal{I}_{E,\mathcal{T}} = \mathcal{I}_{C,\mathcal{T}} \cap \mathcal{I}_{D,\mathcal{T}}$. In particular, the lcs of C and D w.r.t. $\mathcal{T}$ exists iff there is an $\mathcal{FL}_0$ concept E such that $\mathcal{I}_{E,\mathcal{T}} = \mathcal{I}_{C,\mathcal{T}} \cap \mathcal{I}_{D,\mathcal{T}}$.*

To see that in general the lcs w.r.t. general $\mathcal{FL}_0$ TBoxes need not exist, consider the TBox $\mathcal{T} := \{A \sqsubseteq \forall r.(A \sqcap C), B \sqsubseteq \forall r.(B \sqcap C)\}$ where $A, B, C \in \mathsf{N_C}$, and assume that we are interested in the lcs of A and B w.r.t. $\mathcal{T}$. It is easy to see that $\mathcal{I} := \mathcal{I}_{A,\mathcal{T}} \cap \mathcal{I}_{B,\mathcal{T}}$ satisfies $A^{\mathcal{I}} = \emptyset = B^{\mathcal{I}}$ and $C^{\mathcal{I}} = \{r^n \mid n \geq 1\}$. To show that $\mathcal{I}$ is not the least functional model of some $\mathcal{FL}_0$ concept, assume that E is an $\mathcal{FL}_0$ concept such that $\mathcal{I}$ is a functional model of E. Then there is a finite subset $W \subseteq \{r^n \mid n \geq 1\}$ such that $E = \bigsqcap_{w \in W} \forall w.C$. But then the least functional model $\mathcal{I}_{E,\mathcal{T}}$ of E w.r.t. $\mathcal{T}$ actually satisfies $C^{\mathcal{I}_{E,\mathcal{T}}} = W$, and thus $\mathcal{I} \neq \mathcal{I}_{E,\mathcal{T}}$. Thus, there is no $\mathcal{FL}_0$ concept E such that $\mathcal{I}_{E,\mathcal{T}} = \mathcal{I}_{A,\mathcal{T}} \cap \mathcal{I}_{B,\mathcal{T}}$, which shows that A, B do not have an lcs w.r.t. $\mathcal{T}$.

Theorem 22. *Let C, D be $\mathcal{FL}_0$ concepts and $\mathcal{T}$ a general $\mathcal{FL}_0$ TBox. Then we can effectively decide whether C, D have an lcs w.r.t. $\mathcal{T}$ or not. In case the answer is yes, we can effectively compute this lcs.*

Proof. By Lemma 13 we can effectively construct an LTA $\mathcal{P}^{\cap}_{C,D,\mathcal{T}}$ that represents $\mathcal{I}_{C,\mathcal{T}} \cap \mathcal{I}_{D,\mathcal{T}}$. By applying the intersection construction to $\mathcal{P}^{\cap}_{C,D,\mathcal{T}}$ and $\widehat{\mathcal{A}}_{\mathcal{T}}$ and then testing emptiness of the resulting automaton, we can check whether there is an $\mathcal{FL}_0$ concept E that has $\mathcal{I}_{C,\mathcal{T}} \cap \mathcal{I}_{D,\mathcal{T}}$ as its least functional model w.r.t. $\mathcal{T}$.

In order to actually compute the lcs, we need to use a modified version $\widehat{\mathcal{A}}^{c}_{\mathcal{T}}$ of $\widehat{\mathcal{A}}_{\mathcal{T}}$ where the concept component is not projected away. Instead of applying the intersection construction to $\widehat{\mathcal{A}}^{c}_{\mathcal{T}}$ and $\mathcal{P}^{\cap}_{C,D,\mathcal{T}}$, one then needs to use a similar product construction that checks whether the second component of the tree accepted by $\widehat{\mathcal{A}}^{c}_{\mathcal{T}}$ coincides with the tree accepted by $\mathcal{P}^{\cap}_{C,D,\mathcal{T}}$. When applying the emptiness test to the resulting automaton one then extracts a regular tree that witnesses non-emptiness. From the first components in this tree, the lcs E can easily be read off. □

Regarding the complexity of the decision problem, note that the number of states of the LTA $\mathcal{P}^{\cap}_{C,D,\mathcal{T}}$ is exponential in the size of C, D, and $\mathcal{T}$, and that the number of states of $\widehat{\mathcal{A}}_{\mathcal{T}}$ is double exponential in the size of $\mathcal{T}$. The product construction thus again yields an automaton of double exponential size. Since the emptiness problem for parity tree automata is in NP ∩ co-NP, this shows that we can decide the existence of the lcs in 2NExpTime ∩ co-2NExpTime (see [4] for details).

Concept difference

In DLs, the notion of difference has recently mostly been considered for TBoxes rather than concepts [16]. Given two TBoxes $\mathcal{T}_1, \mathcal{T}_2$, the difference between $\mathcal{T}_1$ and $\mathcal{T}_2$ is a TBox $\mathcal{T}_1 - \mathcal{T}_2$ that has, as its consequences, exactly the consequences of $\mathcal{T}_1$ that are not consequences of $\mathcal{T}_2$. In general, such a TBox need not exist, and thus it is interesting to decide whether it does. Here, we modify this approach by considering a fixed TBox $\mathcal{T}$ and two concepts C, D. The consequences of C and D w.r.t. $\mathcal{T}$ are then the value restrictions that respectively follow from these two concepts.

Definition 23. Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox and C, D $\mathcal{FL}_0$ concepts. The concept difference of C and D w.r.t. $\mathcal{T}$ is an $\mathcal{FL}_0$ concept E such that $\mathcal{L}_\mathcal{T}(E) = \mathcal{L}_\mathcal{T}(C) \setminus \mathcal{L}_\mathcal{T}(D)$.

By Proposition 1, the concept difference is unique up to equivalence w.r.t. $\mathcal{T}$, if it exists. In this case, we denote it as $C -_\mathcal{T} D$. Deciding whether the concept difference exists or not can be done in the same way as for the lcs. The only change is that, instead of an LTA representing the intersection of two functional models, we now need to construct an LTA representing their difference. Given least functional models $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{D,\mathcal{T}}$, their *difference* is the functional interpretation $\mathcal{I} = (\mathsf{N}_\mathsf{R}^*, \cdot^\mathcal{I})$ that satisfies $A^\mathcal{I} = A^{\mathcal{I}_{C,\mathcal{T}}} \setminus A^{\mathcal{I}_{D,\mathcal{T}}}$ for all $A \in \mathsf{N}_\mathsf{C}$. Given automata representing the trees $T_{C,\mathcal{T}}$ and $T_{D,\mathcal{T}}$, an automaton representing the tree $t_\mathcal{I}$ where $\mathcal{I}$ is the difference of $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{D,\mathcal{T}}$ can be constructed similarly to the automaton for the intersection.

Theorem 24. *Let C, D be $\mathcal{FL}_0$ concepts and $\mathcal{T}$ a general $\mathcal{FL}_0$ TBox. Then we can effectively decide whether the difference of C and D w.r.t. $\mathcal{T}$ exists or not. In case the answer is yes, we can effectively compute $C -_\mathcal{T} D$.*

The complexity of the decision procedure is obviously the same as for the lcs, i.e., in 2NExpTime ∩ co-2NExpTime.

The difference of concepts has been considered before in the literature, but for DLs other than $\mathcal{FL}_0$ and without a TBox [24, 14]. In [24] this is restricted to the case where $C \sqsubseteq D$, whereas in [14] no subsumption relationship between C and D is required. In the more general setting in [14], the set of *difference candidates* $C \ominus D := \{E \in \mathcal{C}_\mathcal{L} \mid D \sqcap E \equiv C \sqcap D\}$ is considered, where $\mathcal{C}_\mathcal{L}$ is the set of concepts definable in the DL $\mathcal{L}$ under consideration. In [24] it is argued that, semantically, the difference should be as large as possible, which motivates considering the elements of $C \ominus D$ that are maximal w.r.t. subsumption as difference concepts. The following proposition (whose proof can be found in [4]) shows that, in case of an empty TBox, our definition of the difference of $\mathcal{FL}_0$ concepts (Definition 23) coincides with these earlier definitions.

Proposition 25. *Let C, D be $\mathcal{FL}_0$ concepts. Then $C -_\emptyset D$ always exists, and it is the unique subsumption maximal $\mathcal{FL}_0$ concept in $C \ominus D$.*

6 Conclusion

We have shown that least functional models of $\mathcal{FL}_0$ concepts w.r.t. general $\mathcal{FL}_0$ TBoxes can be used to characterize subsumption, and that automata $\widehat{\mathcal{A}}_{C,\mathcal{T}}$ and $\widehat{\mathcal{A}}_\mathcal{T}$ can be constructed that respectively accept (i) exactly the least functional model of a fixed concept C w.r.t. a TBox $\mathcal{T}$; (ii) all least functional models w.r.t. a TBox $\mathcal{T}$. We have used these automata to show that subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes is in ExpTime and that the existence of the lcs and the difference of two $\mathcal{FL}_0$ concepts is decidable. The complexity of the latter decision procedures is quite high, in part because of the use of complementation to construct $\widehat{\mathcal{A}}_\mathcal{T}$. One topic for future research will be to lower this complexity or to show matching lower bounds. One might think that the use of alternating automata could avoid the complementation in the construction of $\widehat{\mathcal{A}}_\mathcal{T}$, and thus allow us to guess a concept C and verify minimality of the interpretation using the same automaton. However, it is not clear how one could enforce that the parts of the automaton that check minimality at different places in the tree are all based on the same guessed concept C.

Another topic for future research is to investigate what other non-standard inferences for $\mathcal{FL}_0$ can be tackled with the approach introduced in this paper. In particular, the unification

problem for $\mathcal{FL}_0$ w.r.t. the empty TBox has been solved by using tree automata that basically guess a unifier [9]. However, ensuring that different occurrences of the same variable are replaced with the same concept is only possible with tree automata if the words occurring in the value restrictions are reversed. Unfortunately, in this reversed representation of value restrictions, checking the satisfaction of a GCI is no longer a local property, and thus cannot be done with a tree automaton. Consequently, it is not clear how these two approaches could be combined.

References

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

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh (UK), 2005. Morgan Kaufmann, Los Altos.

[3] Franz Baader, Oliver Fernández Gil, and Pavlos Marantidis. Approximation in description logics: How weighted tree automata can help to define the required concept comparison measures in $\mathcal{FL}_0$. In *Proc. of the 11th Int. Conf. on Language and Automata Theory and Applications (LATA 2017)*, volume 10168 of *Lecture Notes in Computer Science*, pages 3–26. Springer-Verlag, 2017.

[4] Franz Baader, Oliver Fernández Gil, and Maximilian Pensel. Standard and non-standard inferences in the description logic $\mathcal{FL}_0$ using tree automata. LTCS-Report LTCS-18-04, Chair for Automata Theory, Institute for Theoretical Computer Science, TU Dresden, Germany, 2018. See http://lat.inf.tu-dresden.de/research/reports.html.

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

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

[7] Franz Baader, Ralf Küsters, and Ralf Molitor. Structural subsumption considered from an automata theoretic point of view. In *Proc. of the 1998 Description Logic Workshop (DL'98)*. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-11/, 1998.

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

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

[10] Franz Baader and Stephan Tobies. The inverse method implements the automata approach for modal satisfiability. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 92–106. Springer-Verlag, 2001.

[11] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori Alperin Resnick, and Alexander Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, Los Altos, 1991.

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

[13] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, pages 298–302, 2004.

[14] Sebastian Brandt, Ralf Küsters, and Anni-Yasmin Turhan. Approximation and difference in description logics. In *Proc. of the Eights Int. Conf. on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 203–214, 2002.

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

[16] Boris Konev, Michel Ludwig, Dirk Walther, and Frank Wolter. The logical difference for the lightweight description logic EL. *J. of Artificial Intelligence Research*, 44:633–708, 2012.

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

[18] Carsten Lutz, Inanç Seylan, and Frank Wolter. An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 286–296. AAAI Press, 2012.

[19] E. Mays, R. Dionne, and R. Weida. K-REP system overview. *SIGART Bull.*, 2(3), 1991.

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

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

[22] Maximilian Pensel. An automata based approach for subsumption w.r.t. general concept inclusions in the description logic $\mathcal{FL}_0$. Master's thesis, Chair for Automata Theory, TU Dresden, Germany. See http://lat.inf.tu-dresden.de/research/mas., 2015.

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

[24] Gunnar Teege. Making the difference: A subtraction operation for description logics. In *Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994.*, pages 540–550, 1994.

[25] Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Vol. B*, pages 134–189. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[26] Moshe Y. Vardi and Thomas Wilke. Automata: from logics to algorithms. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas].*, volume 2 of *Texts in Logic and Games*, pages 629–736. Amsterdam University Press, 2008.

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

[28] Benjamin Zarrieß and Anni-Yasmin Turhan. Most specific generalizations w.r.t. general $\mathcal{EL}$-TBoxes. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI 2013)*, pages 1191–1197, Beijing, China, 2013. AAAI Press.