Master's Thesis

An Automata-Based Approach for Subsumption w.r.t. General Concept Inclusions in the Description Logic $\mathcal{FL}_0$

Maximilian Pensel

November 30, 2015

Technische Universität Dresden
Faculty of Computer Science
Institute of Theoretical Computer Science
Chair for Automata Theory

Supervised by
Prof. Dr.–Ing. Franz Baader

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig erstellt und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Dresden, den 30. November 2015

Maximilian Pensel

Contents

1 Introduction and Motivation

In modern computer science, the study of knowledge representation has gained a lot of attention over the past decades. Employing logic based systems such as first order logic (FOL) was widely investigated early on. However, due to its expressive power, reasoning in FOL quickly exceeds even the decidability boundary, which resulted in the consideration of less expressive logic systems such as fragments of FOL. Such formalisms have been introduced within the framework of description logics (DL). One of the most important features of DLs is that they can represent knowledge in a formally well founded and *well understood* way. In general, DLs are comprised of unary predicates called concept names and binary predicates called role names. Concept names are used to describe a specific (simple) class of objects from the world, e.g. Cat, Dog, whereas role names describe certain relations between such objects. An assortment of different constructors provide the means for more complex expressions (called concept descriptions). The most common constructors are for instance conjunction, where

$$Dog \sqcap Brown$$

describes the class of objects that are brown and a dog. Disjunction provides the counterpart, where

$$Dog \sqcup Cat$$

describes that something is a dog or a cat. In order to relate complex concept descriptions with role names, there exist two possible constructors, namely existential restriction ($\exists$) and value restriction ($\forall$).

$$Human \sqcap \neg Child \sqcap \exists pet.(Dog \sqcap Brown) \sqcap \forall sibling.Male$$

describes the class of all humans that are not children ($\neg$), having a brown dog for a pet ($\exists$) and only brothers ($\forall$), where *pet* and *sibling* are role names. The collection of basic constructors also includes $\top$ and $\bot$. The concept $\top$ describes everything and the concept $\bot$ describes nothing. Of course there exist many more DL constructors and each distinct set of such constructors describes a new description logic. The way of describing an axiomatic context for concept descriptions is provided by terminologies, also called TBoxes. In a general TBox we can collect general concept inclusion axioms (GCIs) such as

$$Dog \sqsubseteq Animal \sqcap \forall hates.Cat$$

which describes the *fact* that every dog is an animal that only hates cats. Of course, one can hardly speak about knowledge representation without also speaking about the intention of reasoning over the represented knowledge. A common reasoning task is deciding concept subsumption. That is, given two concept descriptions C, D, is one included in the other with respect to a given terminology. $\mathcal{ALC}$ is a very basic DL that is comprised of exactly the constructors we have just introduced. It can be shown that $\mathcal{ALC}$ (as well as most other DLs) is a fragment of FOL, and yet deciding concept subsumption w.r.t. terminologies is already ExpTime complete [7]. Hence, the question for expressivity and computability of different reasoning tasks for different DLs has remained as important as ever. The introduction of logic based knowledge representation

to the world wide web via the web ontology language (OWL) is one of the best indicators that ontology engineers strive for very expressive DLs [8].

First investigations towards tractability of reasoning tasks began during the 1990ies with [6]. From a theoretical point of view, the computational worst–case complexity for standard inference problems such as subsumption quickly reaches the class EXPTIME or worse, even for very restricted DLs [7, 2] upon allowing for general and even acyclic TBoxes. In particular, deciding subsumption for the basic DL $\mathcal{ALC}$ was proven to be EXPTIME complete, which surprisingly holds for the even smaller fragment $\mathcal{FL}_0$ $(\forall, \sqcap, \top)$ as well [2]. Based on this discovery, researchers discarded further investigations towards reasoning in $\mathcal{FL}_0$, because the introduction of tableaux based algorithms for $\mathcal{ALC}$ [7] implicitly provided reasoning mechanisms for all fragments of $\mathcal{ALC}$. The next reaction to such results was reducing the expressivity to a point of computational tractability, as for the light–weight DL $\mathcal{EL}$ $(\exists, \sqcap, \top)$, which has polynomial time complexities for classification and subsumption [1], even in the presence of terminological cycles. Due to results from the late 1990ies, where the computational performance of practical algorithms for worst–case exponential reasoning tasks [9] has been investigated, further research towards more expressive DLs was encouraged, starting with [2]. The results of [2] provide most of our motivation, as they investigated the tractable DL $\mathcal{EL}$ by extending it with several, non–standard DL constructors which quickly results in theoretical EXPTIME complexity. However, in practice, augmenting existing tractable algorithms to handle the $\mathcal{EL}$ extensions directly, yielded a much better performance than expected. This encourages us to re–evaluate the practical computational properties of $\mathcal{FL}_0$, when considering a direct approach to deciding subsumption, rather than employing a non–deterministic tableaux algorithm.

For this thesis we take a look at the restricted, yet still EXPTIME complete (w.r.t. subsumption) DL $\mathcal{FL}_0$, since direct approaches to implement practical reasoners for this language have been overlooked until now. Even though $\mathcal{FL}_0$ remains a fragment of $\mathcal{ALC}$ they both share the same complexity class when allowing for unrestricted terminologies. This persuaded people to simply employ said tableaux based $\mathcal{ALC}$ algorithms to handle $\mathcal{FL}_0$ as well. However, as explained in detail in [3], subsumption in $\mathcal{FL}_0$ can be approached with automata theory and therefore allows for structural algorithmic solutions. We propose an algorithm, providing a direct approach to decide subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes, and we are confident that this algorithm, similar to the results in [2, 9], can behave well in practice despite its theoretical worst–case complexity. As we allow for general terminologies, we need to be aware of the possibility of so–called terminological cycles. The GCI

$$Dog \sqsubseteq Animal \sqcap \forall child.Dog$$

is an example for such a cyclic dependency. A naive traversal through such a terminology may thus result in an infinite propagation.

As an additional result, our automata based approach allows us to show a coherence between $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$, which extends $\mathcal{FL}_0$ with several complex role constructors. This minor excursion paves the way for some advanced questions regarding non–standard inferences like the computation of unifiers or least common subsumers w.r.t. $\mathcal{FL}_0$ in the presence of general TBoxes.

After thoroughly introducing basic notations around $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$ in Section 2, we will proceed with the investigation of our theoretical cornerstone,

called a functional interpretation, in Section 3. As usual for structure based reasoning, in Section 4 we supply a procedure to transform concept descriptions and terminologies into an appropriate normal form as well as a detailed proof of its computational correctness and complexity. On the theoretical side, Section 5 contains the tree automaton that our algorithmic solution is based on. Starting from the given automaton, we are able to show the correlation between $\mathcal{FL}_0$ with general TBoxes and $\mathcal{FL}_{reg}$ regarding subsumption, as well as raise some very interesting questions. Finally, Section 7 contains the algorithm that the thesis builds up to, including formal proofs of its correctness. This is only followed by a conclusion of the entire thesis and a brief outlook on the research to come.

2 Preliminaries

For this thesis the prerequisite knowledge is mostly comprised of syntactical and semantic notations of the description logic $\mathcal{FL}_0$ and — as we are about to show a correlation to it — $\mathcal{FL}_{reg}$. The definitions in this section coincide with the literature, especially [3]. For most description logics $\mathcal{L}$ the basic elements are concept names contained in the set N_C and role names contained in N_R (s.t. $N_C \cap N_R = \emptyset$). $\mathfrak{C}(\mathcal{L}, N_C, N_R)$ denotes the set of all concept descriptions that can be built with the given constructors of $\mathcal{L}$ using only concept names and role names from N_C and N_R respectively. For concept names we usually use the letters A, B, for role names r, s and for complex concept descriptions C, D. The following rules inductively describe the set $\mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$, for all $A \in N_C$, $C, D \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ and $r \in N_R$:

- $\top \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ (top)
- $A \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ (concept name)
- $C \sqcap D \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ (conjunction)
- $\forall r.C \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ (value restriction)

For a value restriction $\forall r.C$, we call the concept description C its *subject*. To be able to handle chains of value restrictions $\forall r_1.\forall r_2.\cdots\forall r_n.C$ more easily, we often denote them as $\forall w.C$ where w is the word $r_1 \cdots r_n$. Note that in some cases it is also easier not having to distinguish between value restrictions and other concept descriptions. For that purpose we simply say that for any concept description C, we may also write $\forall\varepsilon.C$, where ε is the empty word.

The DL $\mathcal{FL}_0$ considers only elementary role names, whereas its extension $\mathcal{FL}_{reg}$ allows for complex roles with the role constructors identity role, empty role, union, composition and reflexive–transitive closure. We use upper case letters R, S to denote complex roles. Let $\mathfrak{R}(N_R)$ denote the set of complex roles that can be inductively built with the rules below, i.e. for $R, S \in \mathfrak{R}(N_R)$ and all $r \in N_R$

- $\varepsilon \in \mathfrak{R}(N_R)$ (identity role)
- $\emptyset \in \mathfrak{R}(N_R)$ (empty role)
- $r \in \mathfrak{R}(N_R)$ (role name)
- $R \cup S \in \mathfrak{R}(N_R)$ (union)
- $R \circ S \in \mathfrak{R}(N_R)$ (composition)
- $R^* \in \mathfrak{R}(N_R)$ (reflexive–transitive closure)

It can be readily seen that these role constructors resemble regular expressions over letters from the alphabet N_R in the obvious way, and we often write RS instead of $R \circ S$. Consider the complex role

$$\varepsilon \cup (r \circ s)^* \cup ((r \circ r) \circ s^*)$$

essentially describing the set of role compositions

$$\{\varepsilon, rs, rsrs, rsrsrs, \ldots, rr, rrs, rrss, rrsss, \ldots\}$$

which is clearly a regular language over N_R, hence the name $\mathcal{FL}_{reg}$. The set $\mathfrak{C}(\mathcal{FL}_{reg}, N_C, N_R)$ extends the set $\mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$, with the construction of concept descriptions containing complex roles. For $R \in \mathfrak{R}(N_R)$ and $C, D \in \mathfrak{C}(\mathcal{FL}_{reg}, N_C, N_R)$

- $\mathfrak{C}(\mathcal{FL}_0, N_C, N_R) \subseteq \mathfrak{C}(\mathcal{FL}_{reg}, N_C, N_R)$,
- $C \sqcap D \in \mathfrak{C}(\mathcal{FL}_{reg}, N_C, N_R)$ and
- $\forall R.C \in \mathfrak{C}(\mathcal{FL}_{reg}, N_C, N_R)$.

For a value restriction $\forall R.C$ in $\mathcal{FL}_{reg}$ we can also write $\forall L.C$ in order to use the words over N_R from the language L, created with the regular expression described by R. Note that, in the following, all definitions regarding concept descriptions in $\mathcal{FL}_{reg}$ implicitly cover concept descriptions in $\mathcal{FL}_0$ unless stated otherwise. Therefore, they can be applied to $\mathcal{FL}_0$ concept descriptions in equal measure.

In terms of knowledge representation, a concept description or a collection of concept descriptions poses as a formal representation describing a class (or classes) of objects from a chosen context. Providing semantic rules, or axioms, for this context can by achieved by defining so–called TBoxes. A general $\mathcal{L}$ TBox $\mathcal{T}$ is a finite set of axioms of the form $C \sqsubseteq D$, where $C, D \in \mathfrak{C}(\mathcal{L}, N_C, N_R)$. These axioms are also called general concept inclusions (GCIs). They are used to express that the class of objects described by C must always be included in the class of objects described by D, essentially making the description C *more specific* than D.

All of the syntactical constructs above deliver a concrete meaning when semantically evaluated with an interpretation. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non–empty domain $\Delta^{\mathcal{I}}$ and a mapping function that assigns

- subsets of $\Delta^{\mathcal{I}}$ to concept names (i.e. $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$), and
- binary relations over $\Delta^{\mathcal{I}}$ to role names (i.e. $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$).

The mapping function is extended to complex roles $R, S \in \mathfrak{R}(N_R)$ for domain elements $d, e, f \in \Delta^{\mathcal{I}}$, in the following way:

$$\varepsilon^{\mathcal{I}} = \{(d, e) \mid d = e\}$$

$$\emptyset^{\mathcal{I}} = \emptyset$$

$$(R \cup S)^{\mathcal{I}} = \{(d, e) \mid (d, e) \in R^{\mathcal{I}} \vee (d, e) \in S^{\mathcal{I}}\}$$

$$(R \circ S)^{\mathcal{I}} = \{(d, f) \mid \exists e \in \Delta^{\mathcal{I}}.(d, e) \in R^{\mathcal{I}} \wedge (e, f) \in S^{\mathcal{I}}\}$$

$$(R^*)^{\mathcal{I}} = \{(d_0, d_n) \mid \exists n \geq 0.\exists d_1, \ldots, d_{n-1} \in \Delta^{\mathcal{I}}.$$
$$\forall i \in \{0, \ldots, n-1\}.(d_i, d_{i+1}) \in R^{\mathcal{I}}\}$$

The interpretation mapping function can be extended to complex concept descriptions in a similar way ($C, D \in \mathfrak{C}(\mathcal{FL}_{reg}, N_C, N_R)$, $R \in \mathfrak{R}(N_R)$):

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(\forall R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall (d,e) \in R^{\mathcal{I}}.e \in C^{\mathcal{I}}\}$$

Given a set of axioms in a TBox $\mathcal{T}$, interpretations are distinguished by the fact that they respect all axioms in $\mathcal{T}$ or that there is an axiom in $\mathcal{T}$ that the interpretation violates. An interpretation $\mathcal{I}$ is called a model of a TBox $\mathcal{T}$ iff it satisfies all axioms of $\mathcal{T}$, i.e.

$$\forall C \sqsubseteq D \in \mathcal{T}.\ C^{\mathcal{I}} \subseteq D^{\mathcal{I}}.$$

Additionally, $\mathcal{I}$ is a model of a concept description C iff $C^{\mathcal{I}} \neq \emptyset$. An interpretation can be a model to several TBoxes and concept descriptions simultaneously. For the current matter, models of a TBox $\mathcal{T}$ and a given concept description C are of particular interest.

The reasoning task called subsumption checking is asking whether $C, D \in \mathfrak{C}(\mathcal{L}, N_C, N_R)$ are in the binary relation $\sqsubseteq_{\mathcal{T}}$ regarding a TBox $\mathcal{T}$. We write $C \sqsubseteq_{\mathcal{T}} D$. An interpretation $\mathcal{I}$ models $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and the TBox $\mathcal{T}$ models the subsumption iff all models of $\mathcal{T}$ do, i.e.

$$C \sqsubseteq_{\mathcal{T}} D \iff \forall \text{ models } \mathcal{I} \text{ of } \mathcal{T}.C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$

As a final remark, for $\mathcal{FL}_0$ concept descriptions there exist equivalence preserving transformations. Therefore, for every $\mathcal{FL}_0$ concept description there exists an equivalent normal form that we call concept–conjunction–normal–form (CCNF), and it has the following structure:

$$\forall w_1.A_1 \sqcap \cdots \sqcap \forall w_n.A_n$$

for $A_1, \ldots, A_n \in N_C$ and $w_1, \ldots, w_n \in N_R^*$. A formal investigation of the existence and equivalence of this normal form is provided in Section 4.

This concludes the syntactic and semantic basics of $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$. In the following we present several additional notations that are useful in terms of brevity and accuracy.

Since concept descriptions are inductively defined, they grow inductively larger in their size. Many computation procedures, propagating into the structure of concept descriptions, rely on the fact that this size is well–defined, supporting termination proofs of such procedures.

Definition 2.1. *The sizes of various elements in $\mathcal{FL}_{reg}$ are defined as follows, where $A \in N_C$, $r \in N_R$, $R, S \in \mathfrak{R}(N_R)$, $C, D \in \mathfrak{C}(\mathcal{FL}_{reg}, N_C, N_R)$ and $\mathcal{T}$ is an $\mathcal{FL}_0$[1] TBox:*

Complex Concepts	*Complex Roles*
$\lvert\top\rvert = 1$	$\lvert\varepsilon\rvert = 1$
$\lvert A\rvert = 1$	$\lvert\emptyset\rvert = 1$
$\lvert\forall R.C\rvert = \lvert C\rvert + \lvert R\rvert$	$\lvert r\rvert = 1$
$\lvert C \sqcap D\rvert = \lvert C\rvert + \lvert D\rvert + 1$	$\lvert R \cup S\rvert = \lvert R\rvert + \lvert S\rvert + 1$
$\lvert C \sqsubseteq D\rvert = \lvert C\rvert + \lvert D\rvert$	$\lvert R \circ S\rvert = \lvert R\rvert + \lvert S\rvert + 1$
$\lvert\mathcal{T}\rvert = \sum_{C \sqsubseteq D \in \mathcal{T}} \lvert C \sqsubseteq D\rvert$	$\lvert R^*\rvert = \lvert R\rvert + 1$

[1]The definition of $|\mathcal{T}|$ is the same for $\mathcal{FL}_{reg}$ TBoxes, however we will not consider those.

In addition to using the size of elements such as concept descriptions or TBoxes, it is often necessary to restrict or at least identify the basic elements that are present in the respective DL constructs. This set of elements is called the signature.

Definition 2.2. *For $\mathcal{FL}_{reg}$ concept descriptions C, D, the set of all occurring concept names and role names is called the* ***signature*** *$sig(C), sig(D)$ respectively. The definition of a signature can be extended to more complex $\mathcal{FL}_{reg}$ elements:*

- $sig(C \sqsubseteq D) := sig(C) \cup sig(D)$, *and*
- $sig(\mathcal{T}) := \bigcup_{C \sqsubseteq D \in \mathcal{T}} sig(C \sqsubseteq D)$.

For convenience, sig can also be extended to accept multiple arguments, i.e. $sig(X_1, \ldots, X_n) := \bigcup_{i=1}^{n} sig(X_i)$, where any X_i is either an $\mathcal{FL}_0$ TBox, an $\mathcal{FL}_{reg}$ concept description or a GCI over two $\mathcal{FL}_{reg}$ concept descriptions.

3 Functional models and $L_{\mathcal{T}}(C)$

It has already been shown in [3] that subsumption between two $\mathcal{FL}_0$ concept descriptions with the empty TBox can be decided with a purely structural approach. That means it suffices to investigate only certain sub-concepts and decide whether they are shared between the concept descriptions or not. Consider the $\mathcal{FL}_0$ concept descriptions

$$C = \forall w_1.A_1 \sqcap \ldots \sqcap \forall w_n.A_n,$$

$$D = \forall v_1.B_1 \sqcap \ldots \sqcap \forall v_m.B_m,$$

where $w_i, v_j \in N_R^*$ and all $A_i, B_j \in N_C$ $(1 \leq i \leq n,\ 1 \leq j \leq m)$. In order to confirm $C \sqsubseteq D$, it must hold that $C \sqsubseteq \forall v_j.B_j$ for all $j \in \{1, \ldots, m\}$, which is only the case if for every j there exists an $i \in \{1, \ldots, n\}$ such that $v_j = w_i$ and $B_j = A_i$.

Example 3.1. *Let*

$$C = \forall rs.A \sqcap \forall ss.A \sqcap \forall r.B \sqcap \forall rrr.B, \text{ and}$$

$$D = \forall rs.A \sqcap \forall rrr.B.$$

$C \sqsubseteq D$ holds iff $C \sqsubseteq \forall rs.A$ and $C \sqsubseteq \forall rrr.B$. In this example that holds true, because there is a value restriction with the subject A and role–word rs as a conjunct in C, likewise for B and rrr. This illustrates, that subsumption between C and D can be characterized by set inclusion. Let $L_A = \{rs, ss\}$, $L_B = \{r, rrr\}$, $M_A = \{rs\}$ and $M_B = \{rrr\}$ describe the sets of words, occurring in a value restriction with the subscripted concept name as a conjunct in C (L) or D (M). It turns out, that $C \sqsubseteq D$ iff $M_A \subseteq L_A \wedge M_B \subseteq L_B$.

Therefore deciding the subsumption of two concept descriptions C and D can be delegated to deciding several set inclusions (in the opposite direction) for the sets of value restrictions occurring in C and D, provided they are in CCNF. This structural approach for subsumption of $\mathcal{FL}_0$ concepts without TBoxes is explained in detail in [3]. It gives rise to the idea that such a structural approach may also be taken for deciding the subsumption of $\mathcal{FL}_0$ concepts w.r.t. general TBoxes. However, within the context of a TBox it does not suffice to only consider the structure of C and D but rather all value restrictions they entail regarding the given TBox. We define a collection of this knowledge in the form of pairs, representing all value restrictions that follow from a given concept and a TBox.

Definition 3.2.

1. $L_{\mathcal{T}}(C) := \{(w, A) \mid C \sqsubseteq_{\mathcal{T}} \forall w.A\}$
2. $L_{\mathcal{T}}(C, A) := \{w \mid (w, A) \in L_{\mathcal{T}}(C)\}$

The set $L_{\mathcal{T}}(C)$ describes the essential knowledge of value restrictions entailed by an $\mathcal{FL}_0$ concept C through the general $\mathcal{FL}_0$ TBox $\mathcal{T}$, whereas the set $L_{\mathcal{T}}(C, A)$ merely extracts certain words from $L_{\mathcal{T}}(C)$ in order to be directly seen as a language over the alphabet N_R. When inspecting these sets for a concept

description C with the empty TBox $\mathcal{T}$, it is obvious that $L_\mathcal{T}(C)$ is finite, because C is finite and (w, A) only occurs in $L_\mathcal{T}(C)$, if $\forall w.A$ is a conjunct in the equivalent CCNF of C. Reasoning over concept descriptions w.r.t. acyclic $\mathcal{FL}_0$ TBoxes can always be reduced to reasoning with only concept descriptions [3]. However when including general TBoxes, the axiom $A \sqsubseteq \forall r.A$ already illustrates the main problem. As soon as $C \sqsubseteq_\mathcal{T} \forall w.A$, it also holds that $C \sqsubseteq_\mathcal{T} \forall wr.A$, $C \sqsubseteq_\mathcal{T} \forall wrr.A$ and so forth, hence $\{(w, A), (wr, A), (wrr, A), \ldots\} \subseteq L_\mathcal{T}(C)$, making $L_\mathcal{T}(C)$ potentially infinite for general TBoxes $\mathcal{T}$. Nevertheless, Lemma 3.3 shows how reasoning with concept descriptions C, D under a general $\mathcal{FL}_0$ TBox can be redirected to the collections of value restrictions they entail (even if they are infinite), at least w.r.t. subsumption.

Lemma 3.3. $C \sqsubseteq_\mathcal{T} D \iff L_\mathcal{T}(D) \subseteq L_\mathcal{T}(C)$

Proof. We prove the each direction separately.

"$\Longrightarrow$" For all $(w, A) \in L_\mathcal{T}(D)$ it holds that $D \sqsubseteq_\mathcal{T} \forall w.A$, by Definition 3.2. From $C \sqsubseteq_\mathcal{T} D$ then follows that $C \sqsubseteq_\mathcal{T} D \sqsubseteq_\mathcal{T} \forall w.A$, which implies $(w, A) \in L_\mathcal{T}(C)$ and thus $L_\mathcal{T}(D) \subseteq L_\mathcal{T}(C)$.

"$\Longleftarrow$" We show the if direction by contraposition:

$$C \not\sqsubseteq_\mathcal{T} D \Longrightarrow L_\mathcal{T}(D) \not\subseteq L_\mathcal{T}(C)$$

As introduced in Section 2 and formally approved in Section 4, we are able to assume w.l.o.g. that C and D are of the following structure:

$$C \equiv \forall w_1.A_1 \sqcap \ldots \sqcap \forall w_n.A_n$$

$$D \equiv \forall u_1.B_1 \sqcap \ldots \sqcap \forall u_m.B_m$$

Then $C \not\sqsubseteq_\mathcal{T} D \Longrightarrow \exists i \in \{1, \ldots, m\}.C \not\sqsubseteq_\mathcal{T} \forall u_i.B_i$, which implies $(u_i, B_i) \in L_\mathcal{T}(D) \setminus L_\mathcal{T}(C)$. With the existence of an element belonging to $L_\mathcal{T}(D)$ but not to $L_\mathcal{T}(C)$, it is clear that $L_\mathcal{T}(D) \not\subseteq L_\mathcal{T}(C)$. □

Lemma 3.3 is actually very powerful, as it allows us to decide subsumption in $\mathcal{FL}_0$ with the structural method of deciding several set inclusions. Even though $L_\mathcal{T}(C)$ and $L_\mathcal{T}(D)$ describe sets of pairs, they can be seen to represent languages of words over N_R thus creating a resemblance to the problem of deciding language inclusion, which is often solved by utilizing appropriate automata. For acyclic TBoxes, hence finite languages, deterministic finite automata are fully sufficient, whereas applying tree automata to decide subsumption w.r.t. general TBoxes is a key interest in the current investigation. In order to create an automaton working on infinite structures, we first need to introduce an appropriate structure to these infinite sets of words. For example, the set

$$L_\mathcal{T}(C) = \{(\varepsilon, A), (rs, A), (rsrs, A), \ldots, (\varepsilon, B), (s, B), (ss, B), \ldots\}$$

can be expressed with the labeled tree structure in Figure 1. In general, the language $L_\mathcal{T}(C)$ can be described by a tree, where every node $w \in N_R^*$ has exactly one successor for every role name in N_R and if $(w, A) \in L_\mathcal{T}(C)$, then A appears in the label of w. This representation of $L_\mathcal{T}(C)$ is formalized with so–called functional models.

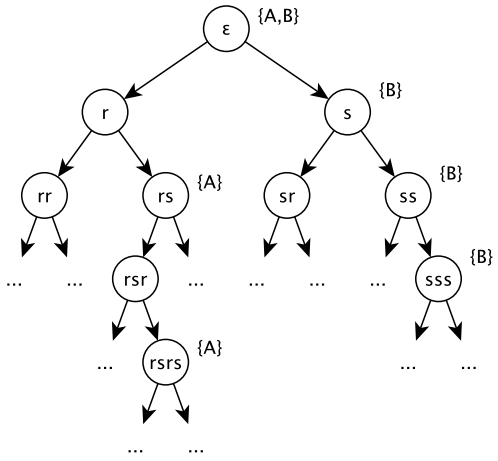


Figure 1: Example tree representation of $L_{\mathcal{T}}(C)$.

Definition 3.4. *$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a **functional model** of C w.r.t. $\mathcal{T}$ iff*

1. *$\Delta^{\mathcal{I}} = N_R^*$ and $\forall r \in N_R.(u, v) \in r^{\mathcal{I}}$ iff $v = ur$ (structure),*
2. *$\forall D \sqsubseteq E \in \mathcal{T}.D^{\mathcal{I}} \subseteq E^{\mathcal{I}}$ (model of $\mathcal{T}$),*
3. *$\varepsilon \in C^{\mathcal{I}}$ (satisfying C at the root).*

We refer to interpretations with the functional model property in graded ways. We call an interpretation $\mathcal{I}$

- a functional interpretation if Property 1 of Definition 3.4 holds,
- a functional model of a TBox $\mathcal{T}$ if Properties 1 and 2 of Definition 3.4 hold, and
- a functional model of a concept description C w.r.t. a TBox $\mathcal{T}$ if all properties of Definition 3.4 hold.

Interpretations are often viewed as labeled graph structures, where domain elements are vertices, labeled with sets of concept names ($A \in N_C$ is in the label set of $d \in \Delta^{\mathcal{I}}$ iff $d \in A^{\mathcal{I}}$), and connected via role–successor relations (directed labeled edges). The graph structure expressed by a functional model is clearly an infinite, labeled and full n–ary tree ($n = |N_R|$). When directly using this tree structure, in order to stay formally correct, we would have to define a function $t_{\mathcal{I}} : N_R^* \to 2^{N_C}$ to represent the tree behind a functional model $\mathcal{I}$. However both structures are essentially equivalent, which is why in later sections, we will use a functional model $\mathcal{I}$ also as the tree $\mathcal{I}$ merely for simplicity. Formalisms like tree–accepting automata will be considered to directly accept functional models $\mathcal{I}$.

It is easy to see that two functional models $\mathcal{I}, \mathcal{J}$ over the same set of roles N_R have the same domain and are structurally identical w.r.t. the interpretation

of roles. Due to this uniform structure, it is easy to define an intersection and a subset relation between functional models over the same domain, based on the set interpretation of concept names.

Definition 3.5. *For functional interpretations $\mathcal{I}$, $\mathcal{J}$ and all $\mathcal{I}_i$ ($i \in \theta$) over the same domain N_R^* with an infinite set of indices θ, we define*

1. *a functional interpretation $\bigcap_{i\in\theta} \mathcal{I}_i$ over the domain N_R^*, such that*

$$\forall A \in N_C . A^{\bigcap_{i\in\theta} \mathcal{I}_i} = \bigcap_{i\in\theta} A^{\mathcal{I}_i}$$

2. *$\mathcal{I} \subseteq \mathcal{J}$ iff $\forall A \in N_C . A^{\mathcal{I}} \subseteq A^{\mathcal{J}}$.*

The relation $\subseteq$ induces an order over functional models and for $\mathcal{I} \subseteq \mathcal{J}$ we say $\mathcal{I}$ is smaller or equal to $\mathcal{J}$.

Proposition 3.6. *$\subseteq$ is a partial order over functional models.*

Proof. We show that for all functional models $\mathcal{I}, \mathcal{J}, \mathcal{K}$ of C w.r.t. $\mathcal{T}$, $\subseteq$ is reflexive, transitive and antisymmetric, making it a partial order.

reflexive
$\mathcal{I} \subseteq \mathcal{I} \iff \forall A \in N_C . A^{\mathcal{I}} \subseteq A^{\mathcal{I}}$,
which holds by reflexivity of $\subseteq$ over sets.

transitive
$\mathcal{I} \subseteq \mathcal{J} \wedge \mathcal{J} \subseteq \mathcal{K} \Longrightarrow \mathcal{I} \subseteq \mathcal{K}$
$\forall A \in N_C . A^{\mathcal{I}} \subseteq A^{\mathcal{J}} \wedge \forall A \in N_C . A^{\mathcal{J}} \subseteq A^{\mathcal{K}} \Longrightarrow \forall A \in N_C . A^{\mathcal{I}} \subseteq A^{\mathcal{K}}$
$\forall A \in N_C . (A^{\mathcal{I}} \subseteq A^{\mathcal{J}} \wedge A^{\mathcal{J}} \subseteq A^{\mathcal{K}} \Longrightarrow A^{\mathcal{I}} \subseteq A^{\mathcal{K}})$,
which holds by transitivity of $\subseteq$ over sets.

antisymmetric
$\mathcal{I} \subseteq \mathcal{J} \wedge \mathcal{J} \subseteq \mathcal{I} \Longrightarrow \mathcal{I} = \mathcal{J}$
$\forall A \in N_C . A^{\mathcal{I}} \subseteq A^{\mathcal{J}} \wedge \forall A \in N_C . A^{\mathcal{J}} \subseteq A^{\mathcal{I}} \Longrightarrow \forall A \in N_C . A^{\mathcal{I}} = A^{\mathcal{J}}$
$\forall A \in N_C . (A^{\mathcal{I}} \subseteq A^{\mathcal{J}} \wedge A^{\mathcal{J}} \subseteq A^{\mathcal{I}} \Longrightarrow A^{\mathcal{I}} = A^{\mathcal{J}})$,
which holds by antisymmetry of $\subseteq$ over sets. $\forall A \in N_C . A^{\mathcal{I}} = A^{\mathcal{J}}$ implies that $\mathcal{I} = \mathcal{J}$ iff $\mathcal{I}$ and $\mathcal{J}$ are functional interpretations over the same domain. □

With the intersection between functional models and a partial ordering, it is already possible to see that an interpretation could be created that shares the essential information of other functional models for the same concept and TBox and is always smaller than all other functional models. The following proposition and Lemma 3.8 help to show the existence and functional model property of the least functional model introduced in Lemma 3.9. Recall that by Definition 3.4, we have for all $i, j \in \theta$:

$$\Delta^{\mathcal{I}_i} = \Delta^{\mathcal{I}_j} = \Delta^{\bigcap_{a\in\theta} \mathcal{I}_a} = N_R^*, \text{ and } r^{\mathcal{I}_i} = r^{\mathcal{I}_j} = r^{\bigcap_{a\in\theta} \mathcal{I}_a}.$$

Proposition 3.7. *For any $\mathcal{FL}_0$ concept C and functional interpretations $\mathcal{I}_i$ ($i \in \theta$) for an infinite set of indices θ, it holds that*

$$C^{\bigcap_{i\in\theta} \mathcal{I}_i} = \bigcap_{i\in\theta} C^{\mathcal{I}_i}.$$

Proof. For convenience, let $\mathcal{I}_\cap := \bigcap_{i\in\theta} \mathcal{I}_i$, for the infinite index set θ.
We show $C^{\mathcal{I}_\cap} = \bigcap_{i\in\theta} C^{\mathcal{I}_i}$ by induction on the structure of C.
Basis: $A^{\mathcal{I}_\cap} = \bigcap_{i\in\theta} A^{\mathcal{I}_i}$ holds for all concept names A by Definition 3.5.
Hypothesis: $C^{\mathcal{I}_\cap} = \bigcap_{i\in\theta} C^{\mathcal{I}_i}$ and $D^{\mathcal{I}_\cap} = \bigcap_{i\in\theta} D^{\mathcal{I}_i}$.
Step:

$\underline{C \sqcap D}$ $(C \sqcap D)^{\mathcal{I}_\cap} = C^{\mathcal{I}_\cap} \cap D^{\mathcal{I}_\cap} = \bigcap_{i\in\theta} C^{\mathcal{I}_i} \cap \bigcap_{i\in\theta} D^{\mathcal{I}_i}$ by induction hypothesis.
Now, $d \in \bigcap_{i\in\theta} C^{\mathcal{I}_i} \cap \bigcap_{i\in\theta} D^{\mathcal{I}_i}$ if and only if $\forall i \in \theta.d \in C^{\mathcal{I}_i} \wedge \forall i \in \theta.d \in D^{\mathcal{I}_i}$, which is equivalent to $\forall i \in \theta.(d \in C^{\mathcal{I}_i} \wedge d \in D^{\mathcal{I}_i})$, because universal quantification distributes over binary conjunction. Therefore, $\bigcap_{i\in\theta} C^{\mathcal{I}_i} \cap \bigcap_{i\in\theta} D^{\mathcal{I}_i} = \bigcap_{i\in\theta} (C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}) = \bigcap_{i\in\theta} (C \sqcap D)^{\mathcal{I}_i}$.

$\underline{\forall r.C}$ $(\forall r.C)^{\mathcal{I}_\cap} = \{v \in N_R^* \mid vr \in C^{\mathcal{I}_\cap}\}$ ($\mathcal{I}_\cap$ is a functional interpretation)
$= \{v \in N_R^* \mid vr \in \bigcap_{i\in\theta} C^{\mathcal{I}_i}\}$ (by induction hypothesis)
$= \bigcap_{i\in\theta} \{v \in N_R^* \mid vr \in C^{\mathcal{I}_i}\}$
$= \bigcap_{i\in\theta} (\forall r.C)^{\mathcal{I}_i}$ □

Lemma 3.8. *Let $\{\mathcal{I}_i \mid i \in \theta\}$ be a set of functional models of a TBox $\mathcal{T}$ over the same domain N_R^*, for an infinite index set θ. The intersection of all $\mathcal{I}_i$ ($i \in \theta$) is again a functional model of $\mathcal{T}$.*

Proof. We show

$$\bigwedge_{i\in\theta} C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i} \Longrightarrow C^{\bigcap_{i\in\theta} \mathcal{I}_i} \subseteq D^{\bigcap_{i\in\theta} \mathcal{I}_i} \tag{3.1}$$

$\bigwedge_{i\in\theta} (C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i})$
$\Longrightarrow \bigwedge_{i\in\theta} (\bigcap_{j\in\theta} C^{\mathcal{I}_j} \subseteq C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i})$
$\Longrightarrow \bigcap_{i\in\theta} C^{\mathcal{I}_i} \subseteq \bigcap_{i\in\theta} D^{\mathcal{I}_i}$
$\Longrightarrow C^{\bigcap_{i\in\theta} \mathcal{I}_i} \subseteq D^{\bigcap_{i\in\theta} \mathcal{I}_i}$ (by Proposition 3.7). If all $\mathcal{I}_i$ ($i \in \theta$) are models of $\mathcal{T}$, i.e. $\forall i \in \theta.\forall C \sqsubseteq D \in \mathcal{T}.C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$, then (3.1) immediately implies that $\bigcap_{i\in\theta} \mathcal{I}_i$ is also a model of $\mathcal{T}$. □

Lemma 3.9. *Let $\mathbb{I}_{C,\mathcal{T}}$ be the set of all functional models of C w.r.t. $\mathcal{T}$. The functional model*

$$\mathcal{I}_{C,\mathcal{T}} := \bigcap_{\mathcal{J}\in\mathbb{I}_{C,\mathcal{T}}} \mathcal{J}$$

is the least functional model w.r.t. $\subseteq$, i.e. $\mathcal{I}_{C,\mathcal{T}} \in \mathbb{I}_{C,\mathcal{T}}$ and $\forall \mathcal{J} \in \mathbb{I}_{C,\mathcal{T}}.\mathcal{I}_{C,\mathcal{T}} \subseteq \mathcal{J}$.

Proof. All $\mathcal{J} \in \mathbb{I}_{C,\mathcal{T}}$ are functional models of $\mathcal{T}$, thus from Lemma 3.8 it follows that $\mathcal{I}_{C,\mathcal{T}}$ is also a functional model of $\mathcal{T}$. Additionally $\varepsilon \in C^{\mathcal{J}}$ for all $\mathcal{J} \in \mathbb{I}_{C,\mathcal{T}}$,

which implies that $\varepsilon \in C^{\bigcap_{\mathcal{J} \in \mathbb{I}_{C,\mathcal{T}}} \mathcal{J}} = C^{\mathcal{I}_{C,\mathcal{T}}}$. Hence, $\mathcal{I}_{C,\mathcal{T}} \in \mathbb{I}_{C,\mathcal{T}}$. It remains to show that $\forall \mathcal{J} \in \mathbb{I}_{C,\mathcal{T}}.\mathcal{I}_{C,\mathcal{T}} \subseteq \mathcal{J}$. It is easy to see that

$$\bigcap_{\mathcal{I} \in \mathbb{I}_{C,\mathcal{T}}} \mathcal{I} = (\bigcap_{\mathcal{I} \in \mathbb{I}_{C,\mathcal{T}}} \mathcal{I}) \cap \mathcal{J}$$

for $\mathcal{J} \in \mathbb{I}_{C,\mathcal{T}}$. Therefore, $\mathcal{I}_{C,\mathcal{T}} = \mathcal{I}_{C,\mathcal{T}} \cap \mathcal{J}$ holds for all $\mathcal{J} \in \mathbb{I}_{C,\mathcal{T}}$, which implies $\mathcal{I}_{C,\mathcal{T}} \subseteq \mathcal{J}$, making $\mathcal{I}_{C,\mathcal{T}}$ the least functional model of C w.r.t. $\mathcal{T}$. □

$\mathcal{I}_{C,\mathcal{T}}$ is the least functional model, containing exactly the minimal information a functional interpretation $\mathcal{I}$ must contain for all concept names in order to be a model for the TBox $\mathcal{T}$ and have $\varepsilon \in C^\mathcal{I}$. In other words, any functional interpretation $\mathcal{J}$ containing less information than $\mathcal{I}_{C,\mathcal{T}}$ (i.e. $J \subset \mathcal{I}_{C,\mathcal{T}}$) cannot be a functional model of $\mathcal{T}$ while having $\varepsilon \in C^\mathcal{J}$. We would ultimately like to restrict our argumentation to this least model with the claim that results for it also hold in every larger functional model and even any general model for that matter. Now we can finally show the correlation between the set $L_\mathcal{T}(C)$ and the least functional model $\mathcal{I}_{C,\mathcal{T}}$, which initially urged us to investigate the tree structure behind $L_\mathcal{T}(C)$.

Lemma 3.10. *For an $\mathcal{FL}_0$ concept description $C \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ and a concept name $A \in N_C$, it holds that*

$$L_\mathcal{T}(C, A) = A^{\mathcal{I}_{C,\mathcal{T}}}.$$

Proof.

"$\subseteq$" If $w \in L_\mathcal{T}(C, A)$, then all functional models $\mathcal{I}$ of C w.r.t. $\mathcal{T}$ satisfy that $C \sqsubseteq_\mathcal{T} \forall w.A$, in particular $\mathcal{I}_{C,\mathcal{T}}$ does. Hence, $\varepsilon \in C^{\mathcal{I}_{C,\mathcal{T}}} \implies \varepsilon \in (\forall w.A)^{\mathcal{I}_{C,\mathcal{T}}} \implies w \in A^{\mathcal{I}_{C,\mathcal{T}}}$ by the structure of functional interpretations.

"$\supseteq$" Show $w \in A^{\mathcal{I}_{C,\mathcal{T}}} \implies w \in L_\mathcal{T}(C, A)$. We can transform $L_\mathcal{T}(C)$ into an interpretation $\mathcal{I} = (N_R^*, \cdot^\mathcal{I})$ with $(u, v) \in r^\mathcal{I}$ iff $v = ur$, and set $A^\mathcal{I} = L_\mathcal{T}(C, A)$. The structural constraint of a functional interpretation is already satisfied by $\mathcal{I}$, and we show now that $\varepsilon \in C^\mathcal{I}$ and that $\mathcal{I}$ is a model of $\mathcal{T}$. W.l.o.g. assume that $C = \prod_{i=1}^{n} \forall u_i.A_i$, hence $\forall i \in \{1, \ldots, n\}.(u_i, A_i) \in L_\mathcal{T}(C)$, which implies $u_i \in L_\mathcal{T}(C, A_i) = A_i^\mathcal{I}$. Given the structure of $\mathcal{I}$ this means $\forall i \in \{1, \ldots, n\}.\varepsilon \in (\forall u_i.A_i)^\mathcal{I}$ which implies $\varepsilon \in \bigcap_{i=1}^{n} (\forall u_i.A_i)^\mathcal{I} = C^\mathcal{I}$.

We show that $\forall E \sqsubseteq F \in \mathcal{T}.E^\mathcal{I} \subseteq F^\mathcal{I}$ by showing that $w \in E^\mathcal{I} \implies w \in F^\mathcal{I}$. W.l.o.g. let $E = \forall u_1.A_1 \sqcap \ldots \sqcap \forall u_l.A_l$ and $F = \forall v_1.B_1 \sqcap \ldots \sqcap \forall v_m.B_m$.
$w \in E^\mathcal{I} \implies \forall i \in \{1, \ldots, l\}.(wu_i, A_i) \in L_\mathcal{T}(C)$
$\implies \forall i \in \{1, \ldots, l\}.C \sqsubseteq_\mathcal{T} \forall wu_i.A_i$
$\implies C \sqsubseteq_\mathcal{T} \forall w.E$,
with the GCI $E \sqsubseteq F \in \mathcal{T}$ we now also know $C \sqsubseteq_\mathcal{T} \forall w.F$, which implies $\forall j \in \{1, \ldots, m\}.C \sqsubseteq_\mathcal{T} \forall wv_j.B_j \implies \forall j \in \{1, \ldots, m\}.(wv_j, B_j) \in L_\mathcal{T}(C) \implies w \in F^\mathcal{I}$.

We have shown that $\mathcal{I}$ satisfies the structural condition of a functional model, it is a model of the TBox and it satisfies C with $\varepsilon \in C^\mathcal{I}$, i.e.

$\mathcal{I}$ is a functional model of C w.r.t. $\mathcal{T}$. Therefore $\mathcal{I}_{C,\mathcal{T}} \subseteq \mathcal{I}$ and thus $A^{\mathcal{I}_{C,\mathcal{T}}} \subseteq A^{\mathcal{I}} = L_{\mathcal{T}}(C, A)$. □

A simple yet very useful consequence from Lemma 3.10 is presented by the following proposition.

Proposition 3.11. $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}} \iff L_{\mathcal{T}}(D) \subseteq L_{\mathcal{T}}(C)$

Proof. We can see that $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}} \iff L_{\mathcal{T}}(D) \subseteq L_{\mathcal{T}}(C)$ is equivalent to $\forall A \in N_C.(A^{\mathcal{I}_{D,\mathcal{T}}} \subseteq A^{\mathcal{I}_{C,\mathcal{T}}} \iff L_{\mathcal{T}}(D, A) \subseteq L_{\mathcal{T}}(C, A))$, which holds by Lemma 3.10. □

Lemma 3.10 shows that the smallest functional model $\mathcal{I}_{C,\mathcal{T}}$ and the "language" $L_{\mathcal{T}}(C)$ essentially describe the same thing.

Proposition 3.12. *For all functional models $\mathcal{J}$ of C w.r.t. $\mathcal{T}$, $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$ implies that $\mathcal{J}$ is also a functional model of D w.r.t. $\mathcal{T}$.*

Proof. If $\mathcal{J}$ is a functional model of C w.r.t. $\mathcal{T}$, then $\varepsilon \in C^{\mathcal{J}}$ and then $C^{\mathcal{J}} \subseteq D^{\mathcal{J}} \implies \varepsilon \in D^{\mathcal{J}}$. Since conditions 1 and 2 of Definition 3.4 are independent of the concept description of a functional model, $\mathcal{J}$ is also a functional model of D w.r.t. $\mathcal{T}$. □

Theorem 3.13. *For an $\mathcal{FL}_0$ TBox $\mathcal{T}$ and two concept descriptions $C, D \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$, $C \sqsubseteq_{\mathcal{T}} D$ if and only if $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$ holds for all functional models $\mathcal{J}$ of $\mathcal{T}$.*

Proof.

"$\implies$" This direction is trivial, if for all models $\mathcal{I}$ of $\mathcal{T}$ $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds, then it holds for all functional models in particular.

"$\impliedby$" If for all functional models $\mathcal{J}$ of $\mathcal{T}$ $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$ holds, then this also holds for $\mathcal{I}_{C,\mathcal{T}}$. $C^{\mathcal{I}_{C,\mathcal{T}}} \subseteq D^{\mathcal{I}_{C,\mathcal{T}}}$ implies that $\mathcal{I}_{C,\mathcal{T}}$ is also a functional model of D w.r.t. $\mathcal{T}$ (Proposition 3.12), hence $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$ (Lemma 3.9).
$\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$
$\implies L_{\mathcal{T}}(D) \subseteq L_{\mathcal{T}}(C)$ (Proposition 3.11)
$\implies C \sqsubseteq_{\mathcal{T}} D$ (Lemma 3.3). □

With Theorem 3.13 we have shown that it is in fact possible to restrict subsumption reasoning not only to functional models, but even to the least functional models of $\mathcal{T}$. As an immediate consequence we are now able to decide subsumption by deciding inclusion of the least models for C and D.

Corollary 3.14. $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}} \iff C \sqsubseteq_{\mathcal{T}} D$. □

Because functional models describe infinite, labeled and full n–ary tree structures ($n = |N_R|$), it seems plausible to utilize tree automata to decide subsumption of $\mathcal{FL}_0$ concepts. In order to build an automaton that accepts trees respecting all GCIs of a given TBox, it is first necessary to transform this TBox into a specific normal form.

1.	NF1.1	$C \sqcap \top$	$\rightsquigarrow$	C
	NF1.2	$\top \sqcap C$	$\rightsquigarrow$	C
	NF1.3	$\forall w.\top$	$\rightsquigarrow$	$\top$
	NF1.4	$\forall w.(C_1 \sqcap \cdots \sqcap C_n)$	$\rightsquigarrow$	$\forall w.C_1 \sqcap \cdots \sqcap \forall w.C_n$
2.	NF2.1	$C_1 \sqcap \forall rw.A \sqcap C_2 \sqsubseteq D$	$\rightsquigarrow$	$C_1 \sqcap \forall r.B \sqcap C_2 \sqsubseteq D$, $\forall w.A \sqsubseteq B$
	NF2.2	$D \sqsubseteq C_1 \sqcap \forall rw.A \sqcap C_2$	$\rightsquigarrow$	$D \sqsubseteq C_1 \sqcap \forall r.B \sqcap C_2$, $B \sqsubseteq \forall w.A$

For $\mathcal{FL}_0$ concept descriptions C, D, concept names A, *fresh* concept names B, $r \in N_R$ and words $w \in N_R^*$ with $|w| > 0$.

Table 1: Normalization phases for $\mathcal{FL}_0$ TBoxes.

4 Normalization in $\mathcal{FL}_0$

Most reasoning systems employing a structural approach rely on two steps. First, transforming the considered concept descriptions or terminologies into an appropriate normal form, and then comparing the resulting structures. The benefit of normalizing concept descriptions and TBoxes is that a specific structure can be assumed, greatly simplifying the structural investigation. We consider two different normal forms of concept descriptions and terminologies in $\mathcal{FL}_0$, one of which can also be considered a normal form for $\mathcal{FL}_{reg}$. An $\mathcal{FL}_0$ concept description is in concept–conjunction–normal–form (CCNF) iff it is of the form

$$\forall w_1.A_1 \sqcap \cdots \sqcap \forall w_n.A_n \tag{4.1}$$

or $\top$, where $A_i \in N_C$ $(1 \leq i \leq n)$ and $w_i \in N_R^*$. Recall that the concept $\forall w.C$ actually represents the chain of value restrictions $\forall r_1. \cdots \forall r_n.C$, if $w = r_1 \dots r_n$ and $r_1, \dots, r_n \in N_R$. Even for $w_i = \varepsilon$ for some $i \in \{1, \dots, n\}$, we already explained that $C = \forall \varepsilon.C$. In words, an $\mathcal{FL}_0$ concept description is in CCNF iff it is a conjunction of only concept names or chains of value restrictions with a concept name as final subject. An $\mathcal{FL}_0$ TBox $\mathcal{T}$ is in CCNF iff the left– and right–hand side of all its GCIs are in CCNF. An $\mathcal{FL}_{reg}$ concept description is also considered to be in CCNF, if it resembles the form of (4.1), i.e. it is of the form $\forall R_1.A_1 \sqcap \cdots \sqcap \forall R_n.A_n$, with $A_i \in N_C$ and $R_i \in \mathfrak{R}(N_R)$ $(1 \leq i \leq n)$. The CCNF for $\mathcal{FL}_0$ is commonly acknowledged and used by the DL community, whereas the second normal form, which is more focused on the structure of TBoxes, mostly benefits the algorithmic part of the current investigation.

An $\mathcal{FL}_0$ TBox $\mathcal{T}$ is considered to be in plane–axiom–normal–form (PANF) iff all left– and right–hand sides of all GCIs in $\mathcal{T}$ are in CCNF and every value restriction $\forall w.A$, occurring in $\mathcal{T}$, has $|w| \leq 1$. Since the transformation to PANF requires a TBox to be in CCNF, we will describe the normalization procedure in two separate phases.

The rules for both normalization procedures are described in Table 1 and referred to as normalization phase 1 ($\rightsquigarrow$ CCNF) and phase 2 ($\rightsquigarrow$ PANF) respectively. The rules of normalization phase 1 can be applied to any concept description. Upon normalizing an entire TBox, phase 1 will be used to normalize every left– and right–hand side of all present GCIs. Phase 2 is applied directly to GCIs exhibiting specific features. First of all note several explanations:

- In both normalization phases, we strictly consider words over role names instead of nested value restrictions, because many transformations on a

chain of value restrictions can also be applied in one step. Also note, that we always consider those words to be of maximal length. We can do that, because for example transforming $\forall u.\forall w.\top \rightsquigarrow \forall u.\top \rightsquigarrow \top$ can also be fully transformed in one step when considering the word uw.

- The rules NF2.1 and NF2.2 from normalization phase 2 are only applied for words w with $|w| > 0$ (i.e. $|rw| > 1$), since value restrictions of depth at most 1 can already be considered as normalized, which is the case for $w = \varepsilon$.
- The expression $C_1 \sqcap \forall rw.A \sqcap C_2$ in phase 2 simply describes an arbitrary concept conjunction (in CCNF), containing the particular value restriction $\forall rw.A$ as a conjunct. This includes the special cases where C_1 and/or C_2 are the top concept, even though after phase 1 they may not be $\top$ anymore. Hence, the cases where $\forall w.A \sqcap C_2$, $C_1 \sqcap \forall w.A$ or just $\forall w.A$ appear on the left– or right–hand side of a GCI are implicitly covered.

Within both normalization phases there is no specific order of rules needed. All rules are applied exhaustively, until no further premise of any rule is satisfied, at which point the concept description or TBox will be in the respective normal form. That being said, it is not excluded that certain heuristics for executing transformation rules may improve the normalization performance. For the proof of Lemma 4.6 we will use exactly such a heuristic for phase 1 in order to simplify interactions between rule applications (e.g. applying NF1.4 may enable an application of NF1.3). Normalization phase 2 can only be applied to TBoxes that are already in CCNF. We denote the TBox obtained from $\mathcal{T}$ through normalization phase 1 as $NF1(\mathcal{T})$ and resulting from phase 2 as $NF2(\mathcal{T})$. As phase 1 is able to transform single concept descriptions (outside of a TBox) into CCNF, we also describe the the concept obtained through normalization phase 1 from C as $NF1(C)$.

Lemma 4.1. *For a concept description $D \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ it holds for any interpretation $\mathcal{I}$ that*

$$D^{\mathcal{I}} = NF1(D)^{\mathcal{I}}.$$

Proof. Due to transitivity of $=$ we only need to show that one application of a rewrite rule does not change the interpretation of D.

NF1.1 $(C \sqcap \top)^{\mathcal{I}} = C^{\mathcal{I}}$ holds, since $C^{\mathcal{I}} \cap \Delta^{\mathcal{I}} = C^{\mathcal{I}}$.

NF1.2 Analogue to NF1.1.

NF1.3 $(\forall w.\top)^{\mathcal{I}} = \top^{\mathcal{I}}$ holds, since $(\forall r.\top)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall (d,e) \in r^{\mathcal{I}}.e \in \top^{\mathcal{I}} = \Delta^{\mathcal{I}}\} = \Delta^{\mathcal{I}} = \top^{\mathcal{I}}$. If $(\forall r.\top)^{\mathcal{I}} = \top^{\mathcal{I}}$, then obviously $(\forall r_1.\cdots\forall r_n.\top)^{\mathcal{I}} = (\forall r_1.\cdots\forall r_{n-1}.\top)^{\mathcal{I}} = \cdots = \top^{\mathcal{I}}$.

NF1.4 We first show for single value restrictions, that $(\forall r.(C_1 \sqcap \cdots \sqcap C_n))^{\mathcal{I}} = (\forall r.C_1 \sqcap \cdots \sqcap \forall r.C_n)^{\mathcal{I}}$ holds, since

$$\{d \in \Delta^{\mathcal{I}} \mid \forall (d,e) \in r^{\mathcal{I}}.e \in C_1^{\mathcal{I}} \cap \cdots \cap C_n^{\mathcal{I}}\}$$

is equivalent to

$$\bigcap_{i=1}^{n} \{d \in \Delta^{\mathcal{I}} \mid \forall (d,e) \in r^{\mathcal{I}}.e \in C_i^{\mathcal{I}}\}$$

which is the same as

$$(\forall r.C_1)^{\mathcal{I}} \cap \cdots \cap (\forall r.C_n)^{\mathcal{I}}.$$

For a chain of value restrictions, we now know that the following lines are equal:

$$(\forall r_1.\cdots\forall r_n.(C_1 \sqcap \cdots \sqcap C_m))^{\mathcal{I}}$$
$$(\forall r_1.\cdots\forall r_{n-1}.(\forall r_n.C_1 \sqcap \cdots \sqcap \forall r_n.C_m))^{\mathcal{I}}$$
$$(\forall r_1.\cdots\forall r_{n-2}.(\forall r_{n-1}.\forall r_n.C_1 \sqcap \cdots \sqcap \forall r_{n-1}.\forall r_n.C_m))^{\mathcal{I}}$$
$$\cdots$$
$$(\forall r_1\cdots r_n.C_1 \sqcap \cdots \sqcap \forall r_1\cdots r_n.C_m)^{\mathcal{I}}$$ □

Lemma 4.2. *The TBoxes $\mathcal{T}$ and $NF1(\mathcal{T})$ are equivalent, that is, an interpretation $\mathcal{I}$ is a model of $\mathcal{T}$ iff it is a model of $NF1(\mathcal{T})$.*

Proof. Since none of the rules NF1.1, NF1.2, NF1.3, NF1.4 introduce new concept or role names, we can see that $sig(\mathcal{T}) = sig(NF1(\mathcal{T}))$ and thus every interpretation $\mathcal{I}$ of $\mathcal{T}$ is also an interpretation for $NF1(\mathcal{T})$ and vice versa. For any GCI $E \sqsubseteq F \in \mathcal{T}$ an application of a rewrite rule merely changes the concept descriptions E, F, it does not introduce or remove (new) GCIs. Hence, after finishing normalization phase 1, it holds that for every GCI $E \sqsubseteq F \in \mathcal{T}$ there is a GCI $NF1(E) \sqsubseteq NF1(F) \in NF1(\mathcal{T})$ (and vice versa) such that $NF1(E), NF1(F)$ is obtained from E, F (respectively) by exhaustively applying the rewrite rules NF1.1 – NF1.4. From Lemma 4.1, we immediately obtain the consequence that every model of $\mathcal{T}$ is also a model of $NF1(\mathcal{T})$. □

For the second normalization phase it is not as simple, since fresh concept names and new GCIs are introduced to the TBox. Equivalence w.r.t. all interpretations does not make sense here because there will potentially exist concept names $B \in sig(NF2(\mathcal{T})) \setminus sig(\mathcal{T})$ that are not mapped to a subset of the domain by the interpretations for $\mathcal{T}$. It rather needs to be shown that models of the original TBox can be extended to provide a model for the normalized one, and since $sig(\mathcal{T}) \subseteq sig(NF2(\mathcal{T}))$, a model of $NF2(\mathcal{T})$ should also be a model of $\mathcal{T}$. This relation from the normalized $NF2(\mathcal{T})$ to the original $\mathcal{T}$ is called a conservative extension.

Definition 4.3. *Given general $\mathcal{FL}_0$ TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, we say that $\mathcal{T}_2$ is a **conservative extension** of $\mathcal{T}_1$ if*

- *$sig(\mathcal{T}_1) \subseteq sig(\mathcal{T}_2)$,*
- *every model of $\mathcal{T}_2$ is a model of $\mathcal{T}_1$, and*
- *for every model $\mathcal{I}_1$ of $\mathcal{T}_1$ there exists a model $\mathcal{I}_2$ of $\mathcal{T}_2$ such that the extensions of the concept and role names from $sig(\mathcal{T}_1)$ coincide in $\mathcal{I}_1$ and $\mathcal{I}_2$, i.e.,*
  - *$A^{\mathcal{I}_1} = A^{\mathcal{I}_2}$ for all concept names $A \in sig(\mathcal{T}_1)$, and*
  - *$r^{\mathcal{I}_1} = r^{\mathcal{I}_2}$ for all role names $r \in sig(\mathcal{T}_1)$.*

Lemma 4.4. *The TBox $NF2(\mathcal{T})$ is a conservative extension of the TBox $\mathcal{T}$ in CCNF.*

Proof. For $NF2(\mathcal{T})$ to be a conservative extension of $\mathcal{T}$, it suffices to show that $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$ when obtained by applying one rewrite rule NF2.1 or NF2.2 to $\mathcal{T}$. By transitivity of rule applications it follows that $NF2(\mathcal{T})$ is also a conservative extension of $\mathcal{T}$. We need to show, that the three properties of Definition 4.3 hold when applying one rewrite rule.

Claim.
$(\forall v.A)^{\mathcal{I}} \subseteq B^{\mathcal{I}} \Longrightarrow (\forall r.\forall v.A)^{\mathcal{I}} \subseteq (\forall r.B)^{\mathcal{I}}$.
$\{d \in \Delta^{\mathcal{I}} \mid \forall (d,e) \in r^{\mathcal{I}}.e \in (\forall v.A)^{\mathcal{I}}\} \subseteq \{d \in \Delta^{\mathcal{I}} \mid \forall (d,e) \in r^{\mathcal{I}}.e \in B^{\mathcal{I}}\}$ obviously holds since $e \in (\forall v.A)^{\mathcal{I}} \Longrightarrow e \in B^{\mathcal{I}}$.

NF2.1. Let $w \in N_R^*$ with $|w| > 0$, $r \in N_R$ and $\mathcal{T}'$ be obtained from $\mathcal{T}$ by rewriting the GCI $C_1 \sqcap \forall rw.A \sqcap C_2 \sqsubseteq D$ to $C_1 \sqcap \forall r.B \sqcap C_2 \sqsubseteq D$ and adding $\forall w.A \sqsubseteq B$ to $\mathcal{T}'$. B is a freshly introduced concept name, i.e. $B \notin sig(\mathcal{T})$.

- $sig(\mathcal{T}) \subseteq sig(\mathcal{T}')$ is obvious, since only the fresh concept name B is introduced. Neither roles nor concept names are being removed.
- Let $\mathcal{I}$ be a model of $\mathcal{T}'$, then $C_1^{\mathcal{I}} \cap (\forall r.B)^{\mathcal{I}} \cap C_2^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $(\forall w.A)^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ hold. By our claim we know that $(\forall r.\forall w.A)^{\mathcal{I}} \subseteq (\forall r.B)^{\mathcal{I}}$ and thus $C_1^{\mathcal{I}} \cap (\forall rw.A)^{\mathcal{I}} \cap C_2^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \cap (\forall r.B)^{\mathcal{I}} \cap C_2^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, which makes $\mathcal{I}$ a model of $\mathcal{T}$, because no other GCIs were changed.
- Let $\mathcal{I}$ be a model of $\mathcal{T}$ and let $\mathcal{I}'$ be the interpretation obtained from $\mathcal{I}$ by setting $\forall A \in sig(\mathcal{T}).A^{\mathcal{I}'} = A^{\mathcal{I}}$ and $\forall r \in sig(\mathcal{T}).r^{\mathcal{I}'} = r^{\mathcal{I}}$. Assume we freshly introduced B in $\mathcal{T}'$, if we set $B^{\mathcal{I}'} = (\forall w.A)^{\mathcal{I}'}$ then $(\forall w.A)^{\mathcal{I}'} \subseteq B^{\mathcal{I}'}$ is satisfied, and $(\forall r.\forall w.A)^{\mathcal{I}'} = (\forall r.B)^{\mathcal{I}'}$ together with

  $$C_1^{\mathcal{I}} = C_1^{\mathcal{I}'}, C_2^{\mathcal{I}} = C_2^{\mathcal{I}'}, D^{\mathcal{I}} = D^{\mathcal{I}'}$$

  yields

  $$\begin{aligned} C_1^{\mathcal{I}'} \cap (\forall r.B)^{\mathcal{I}'} \cap C_2^{\mathcal{I}'} &= C_1^{\mathcal{I}'} \cap (\forall rw.A)^{\mathcal{I}'} \cap C_2^{\mathcal{I}'} \\ &= C_1^{\mathcal{I}} \cap (\forall rw.A)^{\mathcal{I}} \cap C_2^{\mathcal{I}} \subseteq D^{\mathcal{I}} = D^{\mathcal{I}'}, \end{aligned}$$

  thus making $\mathcal{I}'$ a model for $\mathcal{T}'$.

For the normalization rule NF2.2 using the inverted claim (with $\supseteq$), the proof works analogue to NF2.1. □

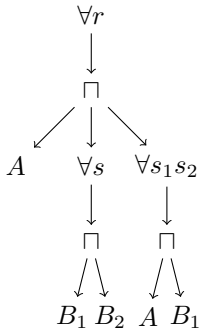
Lemma 4.4 together with the following more general theorem, shows how reasoning w.r.t. the original TBox $\mathcal{T}$ is not affected by our particular normalization procedure.

Theorem 4.5. *For a TBox $\mathcal{T}$ and a conservative extension $\mathcal{T}'$ of $\mathcal{T}$,*
$C \sqsubseteq_{\mathcal{T}} D \Longleftrightarrow C \sqsubseteq_{\mathcal{T}'} D$ holds for any concept descriptions with $sig(C), sig(D) \subseteq sig(\mathcal{T})$.

Proof. Assume that $C \not\sqsubseteq_{\mathcal{T}} D$, then there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$ and since $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$, there exists a model $\mathcal{I}'$ of $\mathcal{T}'$ such that $\forall x \in sig(\mathcal{T}).x^{\mathcal{I}} = x^{\mathcal{I}'}$. Since $sig(C), sig(D) \subseteq sig(\mathcal{T})$, $C^{\mathcal{I}'} = C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}} = D^{\mathcal{I}'}$, hence $C \not\sqsubseteq_{\mathcal{T}'} D$.

For the other direction, assume that $C \not\sqsubseteq_{\mathcal{T}'} D$. Then there exists a model $\mathcal{I}$ of $\mathcal{T}'$ such that $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$. Since $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$, $\mathcal{I}$ is also a model of $\mathcal{T}$ and thus $C \not\sqsubseteq_{\mathcal{T}} D$. □

In the following we will need to argue over the syntactical structure of concept descriptions. For that purpose we make use of the syntax tree formalism from the term rewriting toolbox. Every $\mathcal{FL}_0$ concept description can be expressed with a syntax tree containing $\forall$–nodes, $\sqcap$–nodes, concept–name–nodes and $\top$–nodes. For example, $C = \forall r.(A \sqcap \forall s.(B_1 \sqcap B_2) \sqcap \forall s_1 s_2.(A \sqcap B_1))$ has the following tree–structure:



$\top$– and concept–name–nodes are always leaves, whereas $\forall$–nodes are of arity 1 and $\sqcap$–nodes have a higher arity, depending on the present conjuncts. The depth of a node is defined as the amount of ancestors the node has. In order to handle fewer cases later, we make some demands, so that the syntax trees are tailored to the normalization procedure described in Table 1. The words contained in $\forall$–nodes as well as the arity of $\sqcap$–nodes are maximal. Thus, it is not possible that a $\sqcap$–node has another $\sqcap$–node as a successor, likewise for $\forall$–nodes. As a consequence, for example a concept–name–node (or $\top$–node) with depth 3 has 3 ancestors, either 2 value restrictions, 1 conjunction or 2 conjunctions and 1 value restriction. We will use the expression "X appears under a conjunction" or "X appears under a value restriction", which means that in the current syntax tree, X (as any node in the syntax tree) has an ancestor that is a $\sqcap$–node or a $\forall$–node respectively.

We now show that the entire normalization procedure is linear w.r.t. computation steps in the size of $\mathcal{T}$ as stated by the following lemma.

Lemma 4.6. *A general $\mathcal{FL}_0$ TBox $\mathcal{T}$ can be transformed into PANF, using the rules of Table 1,*

1. *with a linear number of rule applications in the size of $\mathcal{T}$, and*
2. *its normal form $\mathcal{T}'$ is polynomial in the size of $\mathcal{T}$.*

Proof. We can investigate the complexity of phase 1 and 2 separately. For normalization phase 1 we can assume the following heuristic for applying transformation rules without changing the outcome of the normalization procedure.

1. Exhaustively apply NF1.1, NF1.2 and NF1.3 to left– and right–hand sides of all GCIs in $\mathcal{T}$.

2. Exhaustively apply NF1.4 for all concept descriptions on the left– or right–hand side of a GCI in $\mathcal{T}$.

We define a measure $\|\mathcal{T}\|_\top$ of bad $\top$ occurrences within a TBox $\mathcal{T}$:

- $\|\mathcal{T}\|_\top = \sum_{E \sqsubseteq F \in \mathcal{T}} \|E\|_\top^0 + \|F\|_\top^0$
- $\|C \sqcap D\|_\top^d = \|C\|_\top^1 + \|D\|_\top^1$
- $\|\forall w.C\|_\top^d = \|C\|_\top^{d+1}$ (for maximal w, i.e. $C = \top \mid A \mid E \sqcap F$)
- $\|A\|_\top^d = 0$ ($A \in N_C$)
- $\|\top\|_\top^d = d$

for $d \in \mathbb{N}$, describing the distance of $\top$ to the next $\sqcap$–node ancestor in the current syntax tree (or its depth if no $\sqcap$ ancestor exists). Note, that an occurrence of $\top$ can be bad once or twice, but no more. Thus, $\|\mathcal{T}\|_\top \leq 2 \cdot |\mathcal{T}|$. We show that for $\mathcal{T}'$, obtained by applying one of the rules NF1.1, NF1.2 or NF1.3 to $\mathcal{T}$ with $\|\mathcal{T}\|_\top > 0$, that $\|\mathcal{T}'\|_\top < \|\mathcal{T}\|_\top$. For NF1.1,

$$\|C \sqcap \top\|_\top^d = \|C\|_\top^1 + \|\top\|_\top^1 = \|C\|_\top^1 + 1$$

and after the application $\|C\|_\top^1$ remains. Thus, $\|\mathcal{T}'\|_\top = \|\mathcal{T}\|_\top - 1$, which is analogue for NF1.2. For NF1.3, $\|\forall w.\top\|_\top^d = \|\top\|_\top^{d+1}$, which after the application remains as $\|\top\|_\top^d$, provided that w is maximal, which we requested for the normalization procedure anyway. Therefore, the result is the same as for NF1.1 and NF1.2, i.e. $\|\mathcal{T}'\|_\top = \|\mathcal{T}\|_\top - 1$. It is easy to see, that the only occurrence of a $\top$ in $\mathcal{T}$ that is not bad appears at the root of a syntax tree and as long as $\|\mathcal{T}\|_\top > 0$, there is a rule application of NF1.1, NF1.2 or NF1.3 possible. Thus, for $\|\mathcal{T}\|_\top = 0$ it is clear that all concept descriptions on left– or right–hand sides of GCIs in $\mathcal{T}$ are either $\top$, or do not contain $\top$ anymore and none of the first three rules is applicable.

For the exhaustive application of the rule NF1.4, we introduce another measure, describing the badness of $\mathcal{T}$ w.r.t. conjunctions:

- $\|\mathcal{T}\|_\sqcap = \sum_{E \sqsubseteq F \in \mathcal{T}} \|E\|_\sqcap^0 + \|F\|_\sqcap^0$
- $\|C_1 \sqcap \cdots \sqcap C_n\|_\sqcap^d = \|C_1\|_\sqcap^d + \cdots + \|C_n\|_\sqcap^d + d$ (max. conjunctions[2])
- $\|\forall w.C\|_\sqcap^d = \|C\|_\sqcap^1$ (for maximal w, i.e. $C = \top \mid A \mid E \sqcap F$)
- $\|A\|_\sqcap^d = 0$ ($A \in N_C$)
- $\|\top\|_\sqcap^d = 0$

[2]For all $i = 1, \ldots, n$, $C_i = \top \mid A \mid \forall w.D$.

Intuitively, every conjunction appearing in $\mathcal{T}$ is bad, if it has a $\forall$–ancestor in the syntax tree. Because the rules NF1.1, NF1.2 and NF1.3 do not introduce new conjunctions, we know that for the initial input terminology $\mathcal{T}$, $\|\mathcal{T}\|_\sqcap \leq |\mathcal{T}|$. We show that if $\mathcal{T}'$ is obtained from $\mathcal{T}$ by an application of NF1.4, that $\|\mathcal{T}'\|_\sqcap < \|\mathcal{T}\|_\sqcap$. Before the rule application we have

$$\|\forall w.(C_1 \sqcap \cdots \sqcap C_n)\|^d_\sqcap = \|C_1 \sqcap \cdots \sqcap C_n\|^1_\sqcap = \|C_1\|^1_\sqcap + \cdots + \|C_n\|^1_\sqcap + 1,$$

whereas after the transformation,

$$\|\forall w.C_1 \sqcap \cdots \sqcap \forall w.C_n\|^d_\sqcap = \|\forall w.C_1\|^d_\sqcap + \cdots + \|\forall w.C_n\|^d_\sqcap + d$$

and for all $i = 1, \ldots, n$, $\|\forall w.C_i\|^d_\sqcap$ evaluates to $\|C_i\|^1_\sqcap$. We need to distinguish two cases for the position of the concept description $\forall w.(C_1 \sqcap \cdots \sqcap C_n)$. Assume $d = 0$, then the rule application obviously reduces the amount of bad conjunctions by one, since

$$\|C_1\|^1_\sqcap + \cdots + \|C_n\|^1_\sqcap + 1 < \|\forall w.C_1\|^d_\sqcap + \cdots + \|\forall w.C_n\|^d_\sqcap + d.$$

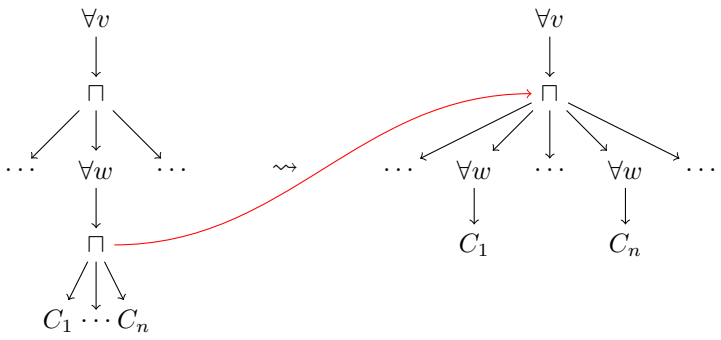
For $d = 1$, another argument is required. Let $D = \forall w.(C_1 \sqcap \cdots \sqcap C_n)$, if $d = 1$, then the value restriction D has another $\forall$–ancestor in the syntax tree, however there must exist a conjunction in between, otherwise it would contradict the maximality of w. Thus, after the rule application s.t. D' is obtained from D by applying the rule NF1.4, D' appears in a conjunction, w.l.o.g. let that be $E \sqcap D'$. In this case,

$$\|\forall v.(E \sqcap \forall w.(C_1 \sqcap \cdots \sqcap C_n))\|^d_\sqcap = \|E\|^1_\sqcap + (\|C_1\|^1_\sqcap + \cdots + \|C_n\|^1_\sqcap + 1) + 1,$$

whereas after the transformation,

$$\|\forall v.(E \sqcap \forall w.C_1 \sqcap \cdots \sqcap \forall w.C_n)\|^d_\sqcap = \|E\|^1_\sqcap + \|\forall w.C_1\|^1_\sqcap + \cdots + \|\forall w.C_n\|^1_\sqcap + 1.$$

Since $\|\forall w.C_i\|^1_\sqcap = \|C_i\|^1_\sqcap$ for all $i = 1, \ldots, n$, it was successfully shown that a rule application of NF1.4 reduces the amount of bad conjunctions in $\mathcal{T}$. To illustrate, consider the following syntax graphs, that depict the merging of subsequent conjunctions during an application of NF1.4:



Hence, every application of NF1.4 reduces the amount of bad conjunctions in $\mathcal{T}$ by one and $\|\mathcal{T}\|_\sqcap$ is bounded by the size of $\mathcal{T}$. Together with the bound of $2 \cdot |\mathcal{T}|$ applications of rules NF1.1, NF1.2 and NF1.3, this shows that normalization phase 1 will always terminate after a linear amount of rule applications in the

size of $\mathcal{T}$. However, we can see that $|\forall w.(C_1 \sqcap \cdots \sqcap C_n)| = |w| + 2n - 1$ increases polynomially in size with an application of NF1.4, since $|\forall w.C_1 \sqcap \cdots \sqcap \forall w.C_n| = n \cdot (|w| + 2) - 1$.

Regarding the second normalization phase, we define a special size of $\mathcal{FL}_0$ elements (in CCNF) in the following way:

- $\|\top\|_\forall = \|A\|_\forall = 0$ $(A \in N_C)$
- $\|C \sqcap D\|_\forall = \|C \sqsubseteq D\|_\forall = \|C\|_\forall + \|D\|_\forall$
- $\|\forall w.A\|_\forall = |w| - 1$ $(A \in N_C)$
- $\|\mathcal{T}\|_\forall = \sum_{C \sqsubseteq D \in \mathcal{T}} \|C \sqsubseteq D\|_\forall$

Intuitively, $\|\mathcal{T}\|_\forall$ is the sum of word lengths of all occurring value restrictions in $\mathcal{T}$ with $|w| \geq 2$ (since we use $|w| - 1$). Naturally, if $\|\mathcal{T}\|_\forall = 0$, then all value restrictions $\forall w.A$ in $\mathcal{T}$ have $|w| = 1$ and thus, $\mathcal{T}$ is in normal form. Furthermore $\|\mathcal{T}\|_\forall$ is bound by the size of $\mathcal{T}$. Showing that each application of NF2.1 or NF2.2 reduces the total word size (of words larger than one letter) is analogue, which is why we focus on NF2.1.

Let $\mathcal{T}'$ be obtained from $\mathcal{T}$ by applying NF2.1 once with the value restriction $\forall rw.A$ $(|w| > 0)$. Then we must show that

$$\|C_1 \sqcap \forall rw.A \sqcap C_2 \sqsubseteq D\|_\forall > \|C_1 \sqcap \forall r.B \sqcap C_2 \sqsubseteq D\|_\forall + \|\forall w.A \sqsubseteq B\|_\forall,$$

which is converted to:

$$\|C_1\|_\forall + |rw| - 1 + \|C_2\|_\forall + \|D\|_\forall >$$
$$\|C_1\|_\forall + |r| - 1 + \|C_2\|_\forall + \|D\|_\forall + |w| - 1 + \|B\|_\forall.$$

Since $r \in N_R$, $|r| - 1 = 0$ and $\|B\|_\forall = 0$ we are left with $|rw| - 1 > |w| - 1$, which obviously holds true for $|w| > 0$. Therefore each application of NF2.1 (or NF2.2) reduces $\|\mathcal{T}\|_\forall$ by at least one. Altogether this implies that after a linear amount of rule applications in phase 2 all value restrictions are of depth at most 1. This shows that if neither rule NF2.1 nor NF2.2 are applicable for $\mathcal{T}$, it must be in PANF. Additionally for both rules NF2.1 and NF2.2, the size of $\mathcal{T}$ increases only by 2, i.e. the size of $NF2(\mathcal{T})$ is also linear in the size of $\mathcal{T}$.

As an immediate consequence of the above, $NF2(NF1(\mathcal{T}))$ can be computed with a total number of rule applications that is linear in the size of $\mathcal{T}$ and furthermore, its size is polynomial in the size of $\mathcal{T}$. □

For every $\mathcal{FL}_0$ TBox $\mathcal{T}$, the TBox $\mathcal{T}' = NF2(NF1(\mathcal{T}))$ is in PANF and a conservative extension of $\mathcal{T}$ by Lemmas 4.2 and 4.4. Together with Theorem 4.5 and Lemma 4.6, this yields the following consequence.

Corollary 4.7. *For every $\mathcal{FL}_0$ TBox $\mathcal{T}$ there exists an $\mathcal{FL}_0$ TBox $\mathcal{T}'$ in PANF such that $C \sqsubseteq_\mathcal{T} D \iff C \sqsubseteq_{\mathcal{T}'} D$ holds for any $\mathcal{FL}_0$ concept descriptions with $sig(C), sig(D) \subseteq sig(\mathcal{T})$.* □

As a final addition to the normalization, we will show that deciding subsumption w.r.t. concept descriptions coincides with deciding subsumption w.r.t. concept names.

Lemma 4.8. *For two $\mathcal{FL}_0$ concept descriptions $C, D \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, let $\mathcal{T}' = \mathcal{T} \cup \{A_C \sqsubseteq C, D \sqsubseteq A_D\}$ with two fresh concept names A_C, A_D not occurring in N_C.*

$$C \sqsubseteq_{\mathcal{T}} D \text{ iff } A_C \sqsubseteq_{\mathcal{T}'} A_D$$

Proof. It is not hard to see that $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$. We use this to prove both directions separately.

"$\Longrightarrow$" Since $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$, it follows from Theorem 4.5 that $C \sqsubseteq_{\mathcal{T}'} D$ holds. The equation $A_C \sqsubseteq C \sqsubseteq_{\mathcal{T}'} D \sqsubseteq A_D$ trivially implies $A_C \sqsubseteq_{\mathcal{T}'} A_D$.

"$\Longleftarrow$" For every model $\mathcal{I}$ of $\mathcal{T}$, there exists a model $\kappa(\mathcal{I})$ of $\mathcal{T}'$ s.t. $\forall A \in N_C . A^{\mathcal{I}} = A^{\kappa(\mathcal{I})}$, $A_C^{\kappa(\mathcal{I})} = C^{\mathcal{I}}$ and $A_D^{\kappa(\mathcal{I})} = D^{\mathcal{I}}$, that is clearly a model of $\mathcal{T}'$. Given that $A_C^{\mathcal{J}} \subseteq A_D^{\mathcal{J}}$ holds for every model $\mathcal{J}$ of $\mathcal{T}'$, it holds for all $\kappa(\mathcal{I})$ in particular. Therefore,

$$C^{\mathcal{I}} = A_C^{\kappa(\mathcal{I})} \subseteq A_D^{\kappa(\mathcal{I})} = D^{\mathcal{I}}$$

implies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. □

5 Looping Tree Automata

The subsumption of two $\mathcal{FL}_0$ concept descriptions w.r.t. a given TBox depends on those interpretations that are models of the given TBox. With Theorem 3.13, we have shown that it suffices to restrict the reasoning to functional models, which bring the advantage of having a specific infinite tree structure. Looping tree automata are capable of accepting such infinite trees. For convenience, we will make no distinction between an infinite labeled tree and a functional interpretation, hence we also say that a looping tree automaton accepts functional interpretations. For the current structural approach, the idea is to build a looping tree automaton that accepts exactly those functional interpretations that are functional models $\mathcal{I}$ of C w.r.t. a given TBox $\mathcal{T}$, that do not satisfy $\varepsilon \in D^{\mathcal{I}}$. This condition represents the counterposition of Proposition 3.12. Therefore, if the automaton does not accept any tree (emptiness test), $C \sqsubseteq_{\mathcal{T}} D$ must hold.

Definition 5.1. *A looping tree automaton on n–ary labeled trees is a 4–tuple $\mathcal{A} = (Q, \Sigma, I, \Delta)$, where*

- *Q is a finite set of states,*
- *Σ is a finite alphabet,*
- *$I \subseteq Q$ is the set of initial states, and*
- *$\Delta = Q \times \Sigma \times Q^n$ is the set of state transitions.*

A looping tree automaton gets infinite tree structures of a specific arity as input, which are either accepted or rejected. For this purpose, such an automaton can also be seen as a labeling mechanism. Given an input tree, the automaton assigns a label to each node of this input. If such a labeling satisfies all criteria formulated in the definition of the respective automaton, the input tree is accepted. Formally these labels are extracted to another tree, representing a so–called run of the automaton on the input t. Note that for a tree t, $dom(t)$ is its domain and contains all nodes occurring in t. Domain nodes are usually identified by the path from the root of the tree to the node. For the tree of a functional interpretation, the tree nodes are domain elements from N_R^*, i.e. $dom(t) = \Delta^{\mathcal{I}} = N_R^*$.

Definition 5.2. *A **run** of a looping tree automaton $\mathcal{A} = (Q, \Sigma, I, \Delta)$ on an n–ary tree $t : dom(t) \to \Sigma$ is a mapping $\rho : dom(t) \to Q$ such that the following condition is fulfilled:*

- $(\rho(w), t(w), \rho(wa_1), \ldots, \rho(wa_n)) \in \Delta$ *(inner node condition)*
  for every node $w \in dom(t)$ s.t. for all $i = 1, \ldots, n$, $wa_i \in dom(t)$ is the i–th successor of w in t.

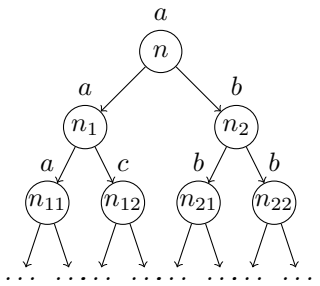
*For a run to be **accepting**, another condition must be fulfilled:*

- $\rho(\varepsilon) \in I$ *(initial state condition)*

The set $\mathcal{L}(\mathcal{A})$ describes all accepted trees of the automaton $\mathcal{A}$. Formally, $t \in \mathcal{L}(\mathcal{A})$, if and only if there exists an accepting run ρ of $\mathcal{A}$ on t.

A tree (even if labeled with elements from Q) that does not fulfill either of the conditions in Definition 5.2 for the automaton $\mathcal{A}$ is not a run of $\mathcal{A}$ on the given tree. Consider the following example, illustrating the capabilities of looping tree automata.

Example 5.3. *Let t be an infinite 2–ary labeled tree with labels from $\Sigma = \{a, b, c\}$ that is "beginning" with the following structure:*



A looping tree automaton $\mathcal{A} = (Q, \Sigma, I, \Delta)$ picks an initial state q_0 from I and picks an applicable state transition reading the symbol a from Σ, i.e. a state transition of the form (q_0, a, q_1, q_2). It continues then to read a in state q_1 and b in state q_2 and so forth. A run of this automaton is a labeling that assigns a label from Q to every tree–node, while structurally staying compatible with the transition relations given in Δ. This non-deterministic automaton will only accept a tree t, if there exists a labeling that does not violate the initial state or inner node condition (Definition 5.2). To emphasize, the automaton supplies a schema of rules that must hold for the labels within a tree in order for it to be accepted. These rules are enforced for instance with state transitions, e.g. let every state q_x only have state transitions of the form $(q_x, x, \_, \_) \in \Delta$ $(x \in \Sigma)$. If there exist only the two transitions (q_a, a, q_a, q_b) and (q_a, a, q_a, q_c) for q_a, every accepted tree must have the label a at the first successor of a node labeled with a and either b or c at the second successor.

The looping tree automaton deciding the subsumption of two concept descriptions C and D will run on functional interpretations. However, due to the nature of state transitions, it is only possible to enforce structural constraints on a local basis (current node and immediate successors). Even though this would suffice for TBoxes in PANF (every value restriction has a nesting depth of at most 1), for a later purpose it is not possible to use the TBox and concept descriptions in PANF here. Therefore, more information has to be encoded into the tree node labels in order to allow enforcement of deeper structural constraints induced by the TBox. For the following definitions we assume the TBox $\mathcal{T}$ and concept descriptions C and D to be in CCNF. We will encode these structural constraints in the states of the automaton. Every state will essentially contain the information that has to hold after following certain paths from the node that this state is assigned to by the automaton. The task for the set of state transitions is in part to ensure that this information is propagated down the tree.

Definition 5.4. *Let C be a concept description of the form*

$$C = \forall w_1.A_1 \sqcap \cdots \sqcap \forall w_n.A_n$$

(1) The set of all value restrictions is defined as

$$val(C) := \{\forall v.A_i \mid \exists i \in \{1, \ldots, n\}.\exists u, v \in N_R^*.uv = w_i\}.$$

The definition of val can be extended to TBoxes such that

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

Similar to the signature, val accepts multiple arguments:

$$val(X_1, \ldots, X_k) = \bigcup_{i=1}^{k} val(X_i)$$

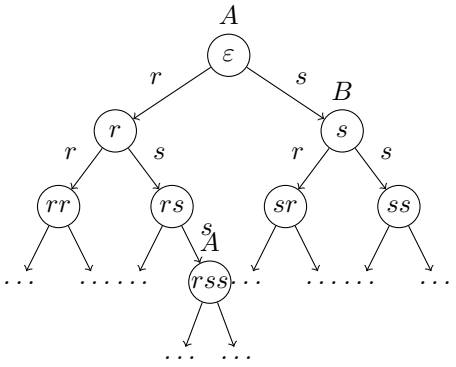
where the X_i are either $\mathcal{FL}_0$ TBoxes or concept descriptions.

(2) The set of all conjuncts in C is $\widehat{C}$, i.e.

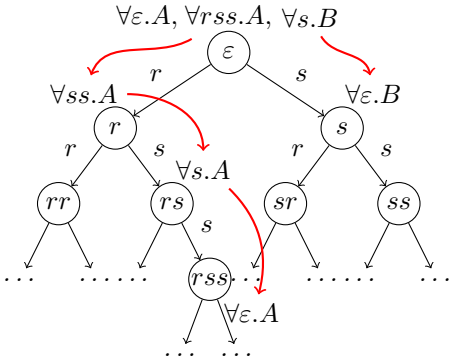
$$\widehat{C} = \{\forall w_i.A_i \mid 1 \leq i \leq n\}.$$

With *val* we have a finite set of all value restrictions that may ever occur in the given context (concepts/TBox), this will help to define an automaton using only finite sets of states and state transitions. The motivation for the looping tree automaton is given by the functional model in Example 5.5.

Example 5.5. *Consider the concept description $C = A \sqcap \forall rss.A \sqcap \forall sB$, whose least functional model $\mathcal{I}$ (w.r.t. $\mathcal{T} = \emptyset$) has the following structure:*



If $\varepsilon \in C^{\mathcal{I}}$ then $\varepsilon \in (\forall rss.A)^{\mathcal{I}}$, thus at node ε lies the information that, after following the path rss, the concept name A must occur in the label of the reached domain element. In order to ensure that A is in the label of rss, an automaton needs to keep track of the information with a local operation. As we have explained before, it is not trivially possible to ensure conditions for nodes that are further away than immediate children. Hence we assign a label (state) to a tree node, that contains the necessary information for all subsequent nodes, as well as respects all GCIs of the given TBox.



Therefore, a run–tree, describing the labeling of the automaton on the given interpretation, is labeled with sets of value restrictions that have to be satisfied for the current domain element. The state transitions can then ensure that longer chains of value restrictions are passed down to the appropriate node. Since labels in a run (states) can contain value restrictions like $\forall\varepsilon.A$, the run is also able to contain the information of the input interpretation (w.r.t. the interpretation of concept names), which is why it is often enough to argue over runs.

In order to be able to process functional interpretations with state transitions in a uniform way, we need to fix the order of children within a functional interpretation. In the following when a TBox and possibly several concept descriptions are given, they are always built over concept descriptions from $\mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$, hence the set of role names N_R is fixed. W.l.o.g. we can assume that for a size $|N_R| = n$, the roles in N_R are called $r_1, \ldots, r_n$. Then, for a functional interpretation, the direct children of a domain element u are $ur_1, \ldots, ur_n$, in that order. This fixes the correlation between a role successor in a functional interpretation and a successor state in a state transition of a looping tree automaton. Explicitly, for a state transition $(q, \sigma, q_1, \ldots, q_n)$, we require q_i to be always associated with the r_i successor of the current domain element $(1 \leq i \leq n)$. Hence, for a run ρ, $(\rho(u), t(u), \rho(ur_1), \ldots, \ldots, \rho(ur_n))$ can correspond to the state transition $(q, \sigma, q_1, \ldots, q_n)$.

Definition 5.6. *Let $\mathcal{T}$ be a general $\mathcal{FL}_0$ TBox in CCNF and C, D two $\mathcal{FL}_0$ concept descriptions in CCNF, built over $\mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$. Then $\mathcal{A}_{C,\mathcal{T}} = (Q, \Sigma, I, \Delta)$ is a looping tree automaton with*

$$
\begin{aligned}
Q &= 2^{val(\mathcal{T},C,D)} \\
\Sigma &= 2^{N_C} \\
I &= \{X \in 2^{val(\mathcal{T},C,D)} \mid \widehat{C} \subseteq X \wedge \widehat{D} \not\subseteq X\} \\
\Delta &= \{(q, \sigma, q_1, \ldots, q_n) \mid (1) \wedge (2) \wedge (3) \wedge (4)\} \text{ with}
\end{aligned}
$$

(1) $\forall E \sqsubseteq F \in \mathcal{T}.\widehat{E} \subseteq q \Longrightarrow \widehat{F} \subseteq q$

(2) $\bigwedge_{i=1}^{n} (\forall r_i w.A \in q \Longrightarrow \forall w.A \in q_i)$

(3) $\bigwedge_{i=1}^{n} (\forall w.A \in q_i \wedge \forall r_i w.A \in val(\mathcal{T}, C, D) \Longrightarrow \forall r_i w.A \in q)$

(4) $A \in \sigma \Longleftrightarrow \forall \varepsilon.A \in q$

We say every state transition $\delta \in \Delta$ is (1) compatible with $\mathcal{T}$, (2) fulfills value restrictions, (3) complements value restrictions and (4) respects the input interpretation.

Note that condition (3) only allows value restrictions to be added to the state q if they already existed as a subterm within the TBox $\mathcal{T}$ or the given concepts C, D. No new value restrictions will be inferred from condition (3), hence keeping the space of all value restrictions confined to $val(\mathcal{T}, C, D)$.

Sometimes the alternative notation

$$\Delta(q, \sigma) = \{(q_1, \ldots, q_n) \mid (q, \sigma, q_1, \ldots, q_n) \in \Delta\}$$

is used to identify state transitions. A run of $\mathcal{A}_{C,\mathcal{T}}$ on a functional interpretation $\mathcal{I}$ is a mapping $\rho : N_R^* \to Q$. In order to simplify the access to the label of a domain element in the tree of a functional interpretation, we define for $w \in N_R^*$

$$\mathcal{I}(w) = \{A \in N_C \mid w \in A^{\mathcal{I}}\}.$$

Lemma 5.7. *Let $\mathcal{I}$ be a functional interpretation. For all accepting runs ρ of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{I}$, it holds for any $u, v \in N_R^*$ and $A \in N_C$ with $\forall v.A \in val(\mathcal{T}, C, D)$ that $\forall v.A \in \rho(u)$ if and only if $\forall \varepsilon.A \in \rho(uv)$.*

Proof. We prove both directions separately for $v = r_1 \cdots r_m$ $(m \geq 1)$. The 4 conditions for state transitions are local structural conditions that have to hold at any node and its immediate successors in a run ρ.

"$\Longrightarrow$" If $\forall v.A \in \rho(u)$ then $\forall r_2 \cdots r_m.A \in \rho(ur_1)$ must hold, otherwise Condition (2) for state transitions in $\mathcal{A}_{C,\mathcal{T}}$ cannot be satisfied. Continuing to argue in that fashion yields that $\forall r_m.A \in \rho(ur_1 \cdots r_{m-1})$, which in turn yields $\forall \varepsilon.A \in \rho(uv)$.

"$\Longleftarrow$" If $\forall \varepsilon.A \in \rho(uv)$ then $\forall r_m.A \in \rho(ur_1 \cdots r_{m-1})$ because otherwise Condition (3) for the state transitions in $\mathcal{A}_{C,\mathcal{T}}$ would be violated, since $\forall v.A \in val(\mathcal{T}, C, D)$, $\forall r_m.A \in val(\mathcal{T}, C, D)$ is satisfied by assumption. Iterating this argument, we obtain $\forall v.A \in \rho(u)$. □

From Lemma 5.7 we obtain the consequence that there cannot exist two runs ρ_1, ρ_2 of $\mathcal{A}_{C,\mathcal{T}}$ accepting $\mathcal{I}$, that are strictly different from each other. Assume that ρ_1 and ρ_2 are accepting runs of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{I}$ and for a node $u \in N_R^*$ and some value restriction $\forall v.A \in val(\mathcal{T}, C, D)$ it holds that $\forall v.A \in \rho_1(u)$ and $\forall v.A \notin \rho_2(u)$. Then Lemma 5.7 implies that $\forall \varepsilon.A \in \rho_1(uv)$ but $\forall \varepsilon.A \notin \rho_2(uv)$. By Condition (4) of the set of state transitions, this yields that $\forall \varepsilon.A \in \rho_1(uv) \iff A \in \mathcal{I}(uv)$, contradicting that ρ_2 is an accepting run. Thus, we have a stronger acceptance condition for functional interpretations. That is, a functional interpretation $\mathcal{I} \in \mathcal{L}(\mathcal{A}_{C,\mathcal{T}})$ iff there exists exactly one accepting run ρ of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{I}$. We say $\mathcal{I}$ is accepted by $\mathcal{A}_{C,\mathcal{T}}$ due to ρ. Most importantly, note that for every $w \in N_R^*$, $A \in N_C$ it holds that $\forall \varepsilon.A \in \rho(w) \iff w \in A^{\mathcal{I}}$, by Condition (4) of Definition 5.6.

Let $\mathcal{J}$ be a functional interpretation such that $uv \in A^{\mathcal{J}}$, then by the structure of $\mathcal{J}$ it is clear that $u \in (\forall v.A)^{\mathcal{J}}$. In the tree of a functional interpretation there are only labels from 2^{N_C}. However, for the accepting run $\rho : N_R^* \to 2^{val(\mathcal{T},C,D)}$ of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{J}$, satisfying the property of Lemma 5.7 that has $\forall \varepsilon.A \in \rho(uv)$, the label of $\rho(u)$ *explicitly* contains the value restriction $\forall v.A$ (if $\forall v.A \in val(\mathcal{T}, C, D)$). Therefore we can see that the accepting run ρ of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{J}$ explicitly contains the value restriction $\forall v.A$ in the labels of all domain nodes contained in the implicitly known set $(\forall v.A)^{\mathcal{J}}$.

Proposition 5.8. *For any concept description $X \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ in CCNF, with $val(X) \subseteq val(\mathcal{T}, C, D)$ and a run ρ of $\mathcal{A}_{C,\mathcal{T}}$ on a functional interpretation $\mathcal{J}$ satisfying Conditions (2), (3) and (4) of Definition 5.6, the following statements are equivalent:*

1. $\widehat{X} \subseteq \rho(u)$
2. *for all* $\forall v.A \in \widehat{X}$ *it holds that* $\forall \varepsilon.A \in \rho(uv)$
3. *for all* $\forall v.A \in \widehat{X}$ *it holds that* $uv \in A^{\mathcal{J}}$
4. $u \in X^{\mathcal{J}}$

Proof. Statements 1 and 2 are equivalent due to Lemma 5.7, which only relies on Conditions (2) and (3) of Definition 5.6. Since Condition (4) of the definition of $\mathcal{A}_{C,\mathcal{T}}$ is satisfied, Statements 2 and 3 are equivalent. Finally, Statement 3 is equivalent to Statement 4 simply by the structure of $\mathcal{J}$ as defined with Property 1 of Definition 3.4. □

Finally, we acquired all the necessary tools to show what kind of functional interpretations are accepted by $\mathcal{A}_{C,\mathcal{T}}$.

Lemma 5.9. *The automaton $\mathcal{A}_{C,\mathcal{T}} = (Q, \Sigma, I, \Delta)$ accepts exactly all functional models $\mathcal{J}$ of C w.r.t. $\mathcal{T}$, such that $\varepsilon \notin D^{\mathcal{J}}$.*

Proof. We need to prove both directions, that is (i) if $\mathcal{J}$ is accepted by $\mathcal{A}_{C,\mathcal{T}}$ due to the run ρ, then $\mathcal{J}$ is a functional model of C w.r.t. $\mathcal{T}$ such that $\varepsilon \notin D^{\mathcal{J}}$ and (ii) if $\mathcal{J}$ is a functional model of C w.r.t. $\mathcal{T}$ such that $\varepsilon \notin D^{\mathcal{J}}$ then there exists an accepting run ρ of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{J}$.

(i) Using Proposition 5.8, it is easy to show the following arguments. Because Condition (1) for the state transitions in Δ is satisfied for ρ, every GCI

in $\mathcal{T}$ is satisfied at every node of ρ, i.e. $\widehat{E} \subseteq \rho(w) \implies \widehat{F} \subseteq \rho(w)$ for all $E \sqsubseteq F \in \mathcal{T}$ and $w \in N_R^*$. Because $\widehat{E} \subseteq \rho(w) \iff w \in E^{\mathcal{J}}$ and $\widehat{F} \subseteq \rho(w) \iff w \in F^{\mathcal{J}}$, we know that $E^{\mathcal{J}} \subseteq F^{\mathcal{J}}$ holds for every GCI $E \sqsubseteq F \in \mathcal{T}$. Hence, $\mathcal{J}$ is a functional model of $\mathcal{T}$. Similarly, we know that $\rho(\varepsilon) \in I$, which is why $\widehat{C} \subseteq \rho(\varepsilon)$, which is equivalent to $\varepsilon \in C^{\mathcal{J}}$, making $\mathcal{J}$ a functional model of C w.r.t. $\mathcal{T}$. Furthermore, $\widehat{D} \not\subseteq \rho(\varepsilon)$ is directly equivalent to $\varepsilon \notin D^{\mathcal{J}}$.

(ii) Let $\mathcal{J}$ be a functional model of C w.r.t. $\mathcal{T}$ such that $\varepsilon \notin D^{\mathcal{J}}$. We can construct a run ρ from $\mathcal{J}$ such that ρ is an accepting run of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{J}$. For all $w \in N_R^*$, let

$$\rho(w) = \{\forall u.A \in val(\mathcal{T}, C, D) \mid w \in (\forall u.A)^{\mathcal{J}}\}.$$

Recall that for $u = rv$, by the structure of $\mathcal{J}$ it holds that $w \in (\forall rv.A)^{\mathcal{J}} \iff wr \in (\forall v.A)^{\mathcal{J}}$. Thus, for every $w \in N_R^*$, $(\rho(w), \mathcal{J}(w), \rho(wr_1), \ldots, \rho(wr_n))$ satisfies the Conditions (2) and (3) of Definition 5.6. Because

$$\forall \varepsilon.A \in \rho(w) \iff w \in (\forall \varepsilon.A)^{\mathcal{J}} \iff w \in A^{\mathcal{J}} \iff A \in \mathcal{J}(w),$$

Condition (4) of Definition 5.6 is satisfied. Since Conditions (2), (3) and (4) are already satisfied for every $(\rho(w), \mathcal{J}(w), \rho(wr_1), \ldots, \rho(wr_n))$, we can apply Proposition 5.8. For every $E \sqsubseteq F \in \mathcal{T}$, $w \in N_R^*$ it holds that $w \in E^{\mathcal{J}} \implies w \in F^{\mathcal{J}}$ because $\mathcal{J}$ is a model of $\mathcal{T}$. Since $w \in E^{\mathcal{J}} \iff \widehat{E} \subseteq \rho(w)$ (analogue for F) holds by Proposition 5.8, it is easy to see that condition (1) is also satisfied for every tuple $(\rho(w), \mathcal{J}(w), \rho(wr_1), \ldots, \rho(wr_n))$. With a similar argument, we know that $\varepsilon \in C^{\mathcal{J}} \iff \widehat{C} \subseteq \rho(\varepsilon)$ and $\varepsilon \notin D^{\mathcal{J}} \iff \widehat{D} \not\subseteq \rho(\varepsilon)$ implies that $\rho(\varepsilon) \in I$, making ρ into an accepted run of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{J}$. □

Theorem 5.10. $\mathcal{L}(\mathcal{A}_{C,\mathcal{T}}) = \emptyset \iff C \sqsubseteq_{\mathcal{T}} D$.

Proof. From Lemma 5.9 and Proposition 5.8 it follows that the existence of the accepting run ρ of $\mathcal{A}_{C,\mathcal{T}}$ on $\mathcal{J}$ implies $\varepsilon \notin D^{\mathcal{J}}$ and thus $\varepsilon \notin D^{\mathcal{I}_{C,\mathcal{T}}}$. However, since $\varepsilon \in D^{\mathcal{I}_{D,\mathcal{T}}}$ holds we have $\mathcal{I}_{D,\mathcal{T}} \not\subseteq \mathcal{I}_{C,\mathcal{T}}$, which implies $C \not\sqsubseteq_{\mathcal{T}} D$ by Corollary 3.14, concluding the contraposition proof of the only–if direction. If $\mathcal{L}(\mathcal{A}_{C,\mathcal{T}}) = \emptyset$, then no run ρ exists that contains a functional model $\mathcal{J}$ of C w.r.t. $\mathcal{T}$ such that $\varepsilon \notin D^{\mathcal{J}}$. Therefore, from Lemma 5.9 follows, that for all functional models $\mathcal{J}$ of C w.r.t. $\mathcal{T}$, $\varepsilon \in D^{\mathcal{J}}$ must hold. Hence, $\varepsilon \in D^{\mathcal{I}_{C,\mathcal{T}}}$ holds, which implies that $\mathcal{I}_{C,\mathcal{T}}$ is also a functional model of D w.r.t. $\mathcal{T}$ and by Lemma 3.9 it holds that $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$. Corollary 3.14 then implies that $C \sqsubseteq_{\mathcal{T}} D$. □

This result shows that deciding subsumption w.r.t. general $\mathcal{FL}_0$ TBoxes can be reduced to the emptiness check for looping tree automata. While this emptiness check can be done in linear time [5], the automaton $\mathcal{A}_{C,\mathcal{T}}$ has exponentially many states in the sizes of $\mathcal{T}, C, D$ ($2^{val(\mathcal{T},C,D)}$ is exponential with $|val(\mathcal{T}, C, D)| \leq |\mathcal{T}| + |C| + |D|$), confirming previous results [2] that deciding subsumption in $\mathcal{FL}_0$ with general TBoxes is of exponential time complexity.

The following investigations are concerned with more questions about the "language" $L_\mathcal{T}(C)$. Motivated by the fact that for acyclic TBoxes, $\mathcal{FL}_0$ concept descriptions can be finitely unfolded to a normal form

$$C = \forall L_1.A_1 \sqcap \ldots \sqcap \forall L_n.A_n,$$

where all $L_i := \{w \in N_R^* \mid C \sqsubseteq_\mathcal{T} \forall w.A_i\}$ (note the resemblance to the definition of $L_\mathcal{T}(C)$) are finite languages, we ask whether there exists a similar normalization for acyclic TBoxes, being aware of the fact that acyclicity of the TBox allows for the L_i to be infinite. The question arises what properties hold for the L_i, in particular, are all $L_i = L_\mathcal{T}(C, A_i)$ regular?

5.1 Minimizing $\mathcal{A}_{C,\mathcal{T}}$

Any language $L \subseteq \Sigma^*$ over some alphabet Σ can be described by a tree of arity $n = |\Sigma|$. The tree $t : \Sigma^* \to \{0, 1\}$ describing L is constructed as follows

$$t(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise.} \end{cases}$$

It is well known that the languages described by finite automata are regular [10]. As a finite automaton accepts the words of only one language (describable by one tree), it is not hard to see that looping tree automata, accepting sets of trees, are a generalization of finite automata. We propose that if a looping tree automaton $\mathcal{A}$ accepts exactly one tree describing a language L, then $\mathcal{A}$ induces a finite automaton describing L (accepting exactly all words in L), showing that L is a regular language. For our purpose, we try to find an automaton accepting exactly the language $L_\mathcal{T}(C, A)$ (for $A \in N_C$) by minimizing the automaton $\mathcal{A}_{C,\mathcal{T}}$ to accept not all functional models of C but exactly the least model $\mathcal{I}_{C,\mathcal{T}}$. It seems natural that if the intersection of all functional models yields the minimal functional model, a similar notion of intersection for state transitions may result in a minimized automaton.

First of all since we do not try to decide subsumption for now, we remove all traces of the second concept description D from the definition of $\mathcal{A}_{C,\mathcal{T}}$. Explicitly the set of initial states will now be defined as

$$I = \{X \subseteq 2^{val(\mathcal{T},C)} \mid \widehat{C} \subseteq X\}$$

and we will only consider value restrictions from $val(\mathcal{T}, C)$. It is clear that the modified $\mathcal{A}_{C,\mathcal{T}}$ accepts all functional models $\mathcal{J}$ of C w.r.t. $\mathcal{T}$ without restrictions. Since for all such $\mathcal{J}$ it holds that $\mathcal{I}_{C,\mathcal{T}} \subseteq \mathcal{J}$, it is obvious that $|\mathcal{I}_{C,\mathcal{T}}(w)| \leq |\mathcal{J}(w)|$ (for all $w \in N_R^*$) which could support the idea to only keep the "least" state transitions for each $q \in Q$ in order to keep acceptable trees minimal. More explicitly, we reduce Δ to

$$\tilde{\Delta} := \{(q, \sigma, q_1, \ldots, q_n) \in \Delta \mid \forall (q, \sigma, q'_1, \ldots, q'_n) \in \Delta, i \in \{1, \ldots, n\}.q_i \subseteq q'_i\}.$$

However, the following example illustrates the error in this reduction.

Example 5.11. *Let $\mathcal{T} = \{A \sqsubseteq \forall r.B, B \sqsubseteq A\}$. $\mathcal{T}$ is already in CCNF and can be viewed uniformly with value restrictions in the following way:*

$$\{\forall \varepsilon.A \sqsubseteq \forall r.B, \forall \varepsilon.B \sqsubseteq \forall \varepsilon.A\}$$

It is easy to see that the least functional model $\mathcal{I}_{A,\mathcal{T}}$ looks like this:

$$\overset{A}{\varepsilon} \xrightarrow{r} \overset{A,B}{r} \xrightarrow{r} \overset{A,B}{rr} \xrightarrow{r} \overset{A,B}{rrr} \xrightarrow{r} \overset{A,B}{rrrr} \xrightarrow{r} \dots$$

For the automaton $\mathcal{A}_{A,\mathcal{T}} = (Q, \Sigma, I, \Delta)$ we have

- $val(\mathcal{T}, A) = \{\forall\varepsilon.A, \forall\varepsilon.B, \forall r.B\}$
- $Q = \{\emptyset, \{\forall\varepsilon.A\}, \{\forall\varepsilon.B\}, \{\forall r.B\}, \{\forall\varepsilon.A, \forall\varepsilon.B\},$
  $\{\forall\varepsilon.A, \forall r.B\}, \{\forall\varepsilon.B, \forall r.B\}, \{\forall\varepsilon.A, \forall\varepsilon.B, \forall r.B\}\}$
- $\Sigma = 2^{N_C} = \{\emptyset, \{A\}, \{B\}, \{A, B\}\}$
- $I = \{\{\forall\varepsilon.A\}, \{\forall\varepsilon.A, \forall\varepsilon.B\}, \{\forall\varepsilon.A, \forall r.B\}, \{\forall\varepsilon.A, \forall\varepsilon.B, \forall r.B\}\}$
- $\Delta = \{(\emptyset, \emptyset, q) \mid q \in Q, \forall\varepsilon.B \notin q\}$
  $\cup\ \{(\{\forall r.B\}, \emptyset, q) \mid q \in Q, \forall\varepsilon.B \in q\}$
  $\cup\ \{(\{\forall\varepsilon.A, \forall r.B\}, \{A\}, q) \mid q \in Q, \forall\varepsilon.B \in q\}$
  $\cup\ \{(\{\forall\varepsilon.A, \forall\varepsilon.B, \forall r.B\}, \{A, B\}, q) \mid q \in Q, \forall\varepsilon.B \in q\}$

Most importantly, note that there are no state transitions for $\{\forall\varepsilon.B\}$, because for transitions of the form $(\{\forall\varepsilon.B\}, \{B\}, X)$, the second GCI would always be violated. Since the automaton cannot continue once state $\{\forall\varepsilon.B\}$ is reached, there does not exist a run containing $\{\forall\varepsilon.B\}$. If we execute the idea to reduce all transitions to the least one right away, we would only keep transitions $(Y, \sigma(Y), \{\forall\varepsilon.B\})$ (with $Y \in \{\{\forall r.B\}, \{\forall\varepsilon.A, \forall r.B\}, \{\forall\varepsilon.A, \forall\varepsilon.B, \forall r.B\}\}$, $\sigma(Y) = \{A \in N_C \mid \forall\varepsilon.A \in Y\}$) and $(\emptyset, \emptyset, \emptyset)$, in which case the resulting automaton has no accepting runs and does not accept $\mathcal{I}_{A,\mathcal{T}}$ anymore. We call $\{\forall\varepsilon.B\}$ a bad state because it may never be used by a run.

The example has shown that the "least" state transition per state q can contain so–called bad states, which is why we must remove bad states and bad state transitions beforehand, so that the minimization cannot run into this problem.

Definition 5.12. *For a looping tree automaton $\mathcal{A} = (Q, \Sigma, I, \Delta)$ a state $q \in Q$ is called **bad** if it does not occur in any run of $\mathcal{A}$. Otherwise it is called **good**.*

In order to identify bad states, we construct a set of bad states inductively.

Definition 5.13.
The set of bad states $Bad_n(\mathcal{A})$ is defined inductively as follows:

$$\begin{aligned} Bad_0(\mathcal{A}) &:= \{q \in Q \mid \neg\exists(q, x, \dots) \in \Delta\} \\ Bad_{i+1}(\mathcal{A}) &:= \{q \in Q \mid \forall(q, x, q_1, \dots, q_n) \in \Delta.\exists i \in \{1, \dots, n\}.q_i \in Bad_i(\mathcal{A})\} \\ &\quad \cup\ Bad_i(\mathcal{A}) \end{aligned}$$

It is not hard to see that $Bad_0(\mathcal{A}) \subseteq Bad_1(\mathcal{A}) \subseteq \dots$, and since $Bad_l(\mathcal{A}) \subseteq Q$ (for any $l \geq 0$) and Q is finite, there exists some $k \in \mathbb{N}$ such that $Bad_k(\mathcal{A}) = Bad_{k+1}(\mathcal{A})$, making the set $Bad_k(\mathcal{A})$ well defined.

Lemma 5.14. *For $n \geq 0$ s.t. $Bad_n(\mathcal{A}) = Bad_{n+1}(\mathcal{A})$, $q \in Bad_n(\mathcal{A})$ iff q is a bad state.*

Proof.

"$\Longrightarrow$" We prove that all $q \in Bad_n(\mathcal{A})$ are bad states by induction on n.
Start: $q \in Bad_0(\mathcal{A})$ implies that for any run ρ and node w such that $\rho(w) = q$ there cannot exist a transition $(\rho(w), t(w), \rho(w\sigma_1), \ldots, \rho(w\sigma_m)) \in \Delta$, which is why ρ is not a run of $\mathcal{A}$, implying that q may not occur in any run and is therefore bad.
Hypothesis: All $q \in Bad_i(\mathcal{A})$ are bad states.
Step: For $Bad_{i+1}(\mathcal{A})$ we only need to show that all $q \in \{q \in Q \mid \forall(q, x, q_1, \ldots, q_m) \in \Delta.\exists j \in \{1, \ldots, m\}.q_j \in Bad_i(\mathcal{A})\}$ are bad, since all states in $Bad_i(\mathcal{A})$ are already known to be bad states. Suppose there exists a run ρ and a node w s.t. $\rho(w) = q$, then all transitions $(\rho(w), t(w), \rho(w\sigma_1), \ldots, \rho(w\sigma_m)) \in \Delta$ lead to a known bad state, contradicting the existence of ρ.
Therefore if $q \in Bad_n(\mathcal{A})$, then q is a bad state.

"$\Longleftarrow$" If $q \notin Bad_n(\mathcal{A})$, then q is good, i.e. there exists a run ρ and a node w s.t. $\rho(w) = q$. We define a tree $t : \{0, \ldots, k\}^* \to \Sigma$ by induction over $w \in \{0, \ldots, k\}^*$ such that the run $\rho : \{0, \ldots, k\}^* \to Q$ of $\mathcal{A}$ on t has $\rho(\varepsilon) = q$. Setting the domain of t to words over natural numbers $\{0, \ldots, k\}^*$ is no restriction. There exists a transition $(q, \sigma, q_0, \ldots, q_k) \in \Delta$, s.t. for all q_i $(0 \leq i \leq k)$ it holds that $q_i \notin Bad_n(\mathcal{A})$, because otherwise q would be in $Bad_n(\mathcal{A}) = Bad_{n+1}(\mathcal{A})$. Therefore we can set $t(\varepsilon) = \sigma$ $(\sigma \in \Sigma)$ and $\rho(\varepsilon) = q$. Assume $\rho(w)$ is already a good state, then there exists a transition $(\rho(w), \sigma, q_0, \ldots, q_k) \in \Delta$ s.t. $q_i \notin Bad_n(\mathcal{A})$ $(0 \leq i \leq k)$ because otherwise $\rho(w)$ would be in $Bad_n(\mathcal{A})$, and thus we can build $t(w) = \sigma$. This inductively describes an infinite tree t such that a run ρ on $\mathcal{A}$ with $\rho(\varepsilon) = q$ exists, which implies that q must be a good state. □

For the first step, before minimizing the automaton $\mathcal{A}_{C,\mathcal{T}}$, we define an intermediary automaton by removing all bad states from $\mathcal{A}_{C,\mathcal{T}}$.

Definition 5.15. *For $Bad_n(\mathcal{A}_{C,\mathcal{T}}) = Bad_{n+1}(\mathcal{A}_{C,\mathcal{T}})$, we obtain the automaton $\mathcal{A}^+_{C,\mathcal{T}} = (Q^+, \Sigma, I^+, \Delta^+)$ from $\mathcal{A}_{C,\mathcal{T}} = (Q, \Sigma, I, \Delta)$ by setting*

- $Q^+ = Q \setminus Bad_n(\mathcal{A}_{C,\mathcal{T}})$,
- $I^+ = I \setminus Bad_n(\mathcal{A}_{C,\mathcal{T}})$,
- $\Delta^+ = \{(q, \sigma, q_1, \ldots, q_m) \in \Delta \mid q \notin Bad_n(\mathcal{A}_{C,\mathcal{T}})\}$

Proposition 5.16. $\mathcal{L}(\mathcal{A}^+_{C,\mathcal{T}}) = \mathcal{L}(\mathcal{A}_{C,\mathcal{T}})$.

Proof. It is not hard to see that the set of accepting runs of $\mathcal{A}_{C,\mathcal{T}}$ is not reduced by removing only bad states, which is why $\mathcal{A}_{C,\mathcal{T}}$ and $\mathcal{A}^+_{C,\mathcal{T}}$ admit the same runs and thus accept the same set of trees. □

Interestingly, $I^+ = \emptyset$ iff $\mathcal{L}(\mathcal{A}_{C,\mathcal{T}}) = \emptyset$ because no run can satisfy the initial state condition, which is why the reduction to $\mathcal{A}^+_{C,\mathcal{T}}$ corresponds to the emptiness check for $\mathcal{A}_{C,\mathcal{T}}$.

Definition 5.17. *The automaton* $\tilde{\mathcal{A}}_{C,\mathcal{T}} = (Q, \Sigma, \tilde{I}, \tilde{\Delta})$ *is obtained from* $\mathcal{A}^+_{C,\mathcal{T}} = (Q, \Sigma, I, \Delta)$*, by setting*

- $\tilde{I} = \{q \in I \mid \forall q' \in I.q \subseteq q'\}$
- $\tilde{\Delta} = \{(q, \sigma, q_1, \ldots, q_n) \in \Delta \mid \forall (q, \sigma, q'_1, \ldots, q'_n) \in \Delta. \; \forall i \in \{1, \ldots, n\}.q_i \subseteq q'_i\}$

Before showing that the minimized automaton $\tilde{\mathcal{A}}_{C,\mathcal{T}}$ accepts only one functional model, we need two more important properties of run–trees that are accepting runs of $\mathcal{A}_{C,\mathcal{T}}$.

Proposition 5.18. *Let* $\mathcal{I}$ *and* $\mathcal{J}$ *be two functional interpretations accepted by* $\mathcal{A}_{C,\mathcal{T}}$ *due to the runs* $\rho_\mathcal{I}$ *and* $\rho_\mathcal{J}$ *respectively.*

(1) $\mathcal{I} \subseteq \mathcal{J} \iff \rho_\mathcal{I} \subseteq \rho_\mathcal{J}$

(2) $\mathcal{I} = \mathcal{J} \iff \rho_\mathcal{I} = \rho_\mathcal{J}$

Proof. For (1) we can prove both directions separately. Let $\rho_\mathcal{I} \subseteq \rho_\mathcal{J}$, due to condition (4) of Definition 5.6, it is obvious that $\forall \varepsilon.A \in \rho_\mathcal{I}(w) \implies \forall \varepsilon.A \in \rho_\mathcal{J}(w)$ implies that $w \in A^\mathcal{I} \implies w \in A^\mathcal{J}$, i.e. $A^\mathcal{I} \subseteq A^\mathcal{J}$, which holds for any $A \in N_C$ and $w \in N_R^*$ and therefore $\mathcal{I} \subseteq \mathcal{J}$. For the other direction, assume that $\rho_\mathcal{I} \not\subseteq \rho_\mathcal{J}$, then there exist some $u, v \in N_R^*$ and $A \in N_C$ such that $\forall v.A \in \rho_\mathcal{I}(u)$ but $\forall v.A \notin \rho_\mathcal{J}(u)$. By Lemma 5.7 we know that $\forall \varepsilon.A \in \rho_\mathcal{I}(uv)$ and also $\forall \varepsilon.A \notin \rho_\mathcal{J}(uv)$, hence by Condition (4) of Definition 5.6 we have $uv \in A^\mathcal{I}$ and $uv \notin A^\mathcal{J}$, which is why $\mathcal{I} \not\subseteq \mathcal{J}$.

Equation (2) is merely a direct consequence of (1), in that

$$\rho_\mathcal{I} = \rho_\mathcal{J} \iff \rho_\mathcal{I} \subseteq \rho_\mathcal{J} \wedge \rho_\mathcal{J} \subseteq \rho_\mathcal{I} \iff \mathcal{I} \subseteq \mathcal{J} \wedge \mathcal{J} \subseteq \mathcal{I} \iff \mathcal{I} = \mathcal{J}. \qquad \square$$

It remains to investigate which functional models are accepted by $\tilde{\mathcal{A}}_{C,\mathcal{T}}$.

Lemma 5.19. $\tilde{\mathcal{A}}_{C,\mathcal{T}} = (Q, \Sigma, \tilde{\mathcal{I}}, \tilde{\Delta})$ *as obtained from* $\mathcal{A}^+_{C,\mathcal{T}} = (Q, \Sigma, I, \Delta)$ *accepts exactly the functional model* $\mathcal{I}_{C,\mathcal{T}}$*.*

Proof. For a functional model $\mathcal{J}$ accepted by $\mathcal{A}^+_{C,\mathcal{T}}$, let $\rho_\mathcal{J}$ always describe the one accepting run of $\mathcal{A}^+_{C,\mathcal{T}}$ on $\mathcal{J}$. From Lemmas 5.9, 3.9 and Proposition 5.18 it follows that $\rho_{\mathcal{I}_{C,\mathcal{T}}} \subseteq \rho_\mathcal{J}$ for all functional models $\mathcal{J}$ accepted by $\mathcal{A}^+_{C,\mathcal{T}}$. In particular, $\rho_{\mathcal{I}_{C,\mathcal{T}}}(\varepsilon) \subseteq \rho_\mathcal{J}(\varepsilon)$ also holds. For all $\mathcal{J}$, accepted by $\mathcal{A}^+_{C,\mathcal{T}}$, it holds that $\rho_\mathcal{J}(\varepsilon) \in I$ and by Lemma 5.9 and the removal of bad initial states for $\mathcal{A}^+_{C,\mathcal{T}}$, $q \in I$ also implies that there exists some $\mathcal{J}$ with $\rho_\mathcal{J}(\varepsilon) = q$. Therefore, with Definition 5.17 it is clear that $\tilde{I} = \{\rho_{\mathcal{I}_{C,\mathcal{T}}}(\varepsilon)\}$.

For every $w \in N_R^*$, Proposition 5.18 yields that $\mathcal{I}_{C,\mathcal{T}}(wr_i) \subseteq \mathcal{J}(wr_i)$ — for all $i = 1, \ldots, n$ and $\mathcal{J}$ accepted by $\mathcal{A}^+_{C,\mathcal{T}}$ — implies that $\rho_{\mathcal{I}_{C,\mathcal{T}}}(wr_i) \subseteq \rho_\mathcal{J}(wr_i)$. Therefore, $(\rho_{\mathcal{I}_{C,\mathcal{T}}}(w), \mathcal{I}_{C,\mathcal{T}}(w), \rho_{\mathcal{I}_{C,\mathcal{T}}}(wr_1), \ldots, \rho_{\mathcal{I}_{C,\mathcal{T}}}(wr_n))$ is the only transition in $\tilde{\Delta}(q, \sigma)$, for every $q = \rho_{\mathcal{I}_{C,\mathcal{T}}}(w)$ and $\sigma = \mathcal{I}_{C,\mathcal{T}}(w)$, which together with $\tilde{I} = \{\rho_{\mathcal{I}_{C,\mathcal{T}}}(\varepsilon)\}$ implies that $\tilde{\mathcal{A}}_{C,\mathcal{T}}$ accepts $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{C,\mathcal{T}}$ is the only model accepted by $\tilde{\mathcal{A}}_{C,\mathcal{T}}$. $\square$

Now that we created a looping tree automaton, accepting exactly the least functional model $\mathcal{I}_{C,\mathcal{T}}$, we can create finite automata accepting words from the language $L_{\mathcal{T}}(C, A)$. A finite automaton accepts a word w, if there exists a path from the initial state to a final state, using state transitions labeled with the letters from w (in their given order). The key idea for our construction is induced by the fact that $w \in L_{\mathcal{T}}(C, A)$ iff $A \in \mathcal{I}_{C,\mathcal{T}}(w)$ (Lemma 3.10). Thus, by following the path w in the finite automaton, it will reach a final state iff following the path w (starting at ε) in the tree $\mathcal{I}_{C,\mathcal{T}}$ yields a node with A in its label.

Definition 5.20. *The deterministic finite automaton $M_A = (Q, \Pi, \delta, q_0, F)$ ($A \in N_C$) is constructed from the looping tree automaton $\tilde{\mathcal{A}}_{C,\mathcal{T}} = (Q, \Sigma, I, \Delta)$ by setting*

- $\Pi = N_R$
- $q_0 = q_\varepsilon$, *if* $I = \{q_\varepsilon\}$
- $F = \{q \in Q \mid \forall \varepsilon.A \in q\}$
- $\delta = \{(q, r_i, q_i) \mid (q, \sigma, q_1, \ldots, q_n) \in \Delta, 1 \leq i \leq n\}$

Theorem 5.21. *$L_{\mathcal{T}}(C, A)$ is regular.*

Proof. Let $M_A = (Q, \Pi, \delta, q_0, F)$ and $\tilde{\mathcal{A}}_{C,\mathcal{T}} = (Q, \Sigma, I, \Delta)$. First of all, the language accepted by a finite automaton is always regular [10]. Hence, $\mathcal{L}(M_A)$ is regular. For $\mathcal{I}_{C,\mathcal{T}}$ being the least functional model, accepted by $\tilde{\mathcal{A}}_{C,\mathcal{T}}$ due to $\rho_{\mathcal{I}_{C,\mathcal{T}}}$ (the only accepting run of $\tilde{\mathcal{A}}_{C,\mathcal{T}}$), it is easy to see from Lemma 3.10 and Proposition 5.18 that

$$\forall \varepsilon.A \in \rho_{\mathcal{I}_{C,\mathcal{T}}}(w) \iff w \in A^{\mathcal{I}_{C,\mathcal{T}}} \iff w \in L_{\mathcal{T}}(C, A).$$

It remains to show that for any word w over N_R, it holds that $w \in \mathcal{L}(M_A) \iff \forall \varepsilon.A \in \rho_{\mathcal{I}_{C,\mathcal{T}}}(w)$. Let $w = r_{i_1} \ldots r_{i_n}$ for $|N_R| = m$, $1 \leq i_j \leq m$ ($1 \leq j \leq n$, hence $|w| = n$).

"$\Longrightarrow$" If $w \in \mathcal{L}(M_A)$, there exists a chain $d_1, \ldots, d_n$ of state transitions ($d_k \in \delta$), such that $d_k = (q_{k-1}, r_{i_k}, q_k)$ ($1 \leq k \leq n$) with q_0 being the initial state of M_A and $q_n \in F$. By construction of M_A there must exist a transition $\overline{d_k} = (q_{k-1}, \sigma_{k-1}, \overline{q_1}, \ldots, q_{i_k}, \ldots, \overline{q_n}) \in \Delta$ with $q_{i_k} = q_k$ for every d_k. By construction q_0 is the same as $q_\varepsilon \in I$ of $\tilde{\mathcal{A}}_{C,\mathcal{T}}$ and since (for d_n) $q_n \in F$, $\forall \varepsilon.A \in q_n$ holds in $\tilde{\mathcal{A}}_{C,\mathcal{T}}$. Since $\tilde{\mathcal{A}}_{C,\mathcal{T}}$ has only the run $\rho_{\mathcal{I}_{C,\mathcal{T}}}$ and $\overline{d_{k-1}}$ is connected to $\overline{d_k}$ for $2 \leq k \leq n$, the chain of state transitions from Δ establishes the chain of states $q_0, \ldots, q_n$ where q_{k-1} reaches q_k via the role r_{i_k} in $\rho_{\mathcal{I}_{C,\mathcal{T}}}$. Finally, since $q_0 \in I$, it follows that $\rho_{\mathcal{I}_{C,\mathcal{T}}}(w) = q_n$ and thus $\forall \varepsilon.A \in \rho_{\mathcal{I}_{C,\mathcal{T}}}(w)$.

"$\Longleftarrow$" Similar to the other direction, $\forall \varepsilon.A \in \rho_{\mathcal{I}_{C,\mathcal{T}}}(w)$ implies that there exists a chain of state transitions $\overline{d_1}, \ldots, \overline{d_n}$ from Δ such that $\overline{d_1}$ starts at the only initial state of $\tilde{\mathcal{A}}_{C,\mathcal{T}}$, $\overline{d_n}$ points (among others) to q_n with $\forall \varepsilon.A \in q_n$ and each $\overline{d_k}$ is connected to its successor via the respective role from the chain given in w. By construction of $\tilde{\mathcal{A}}_{C,\mathcal{T}}$ this implies the existence of a chain of state transitions $d_1 \ldots d_n$ from δ in M_A starting at the initial state of M_A, and ending with a final state q_n (since $\forall \varepsilon.A \in q_n$), thus accepting the word w. □

6 $\mathcal{FL}_0$ with general TBoxes vs. $\mathcal{FL}_{reg}$

Until now we were mainly concerned with the problem of deciding subsumption in $\mathcal{FL}_0$ with general TBoxes. The goal was to find a more elegant approach than simply employing the tableaux algorithm that decides subsumption for the DL $\mathcal{ALC}$ [7], as $\mathcal{FL}_0$ is contained in $\mathcal{ALC}$. The fact that chains of value restrictions entailed by a concept description can be expressed as sets of words, i.e. languages, proved to be advantageous in this effort. It allows the employment of an automata based approach, to tackle the problem of deciding subsumption in $\mathcal{FL}_0$ with general TBoxes. Before we want to bridge the gap between the theoretical mechanism that is a looping tree automaton and deciding subsumption in a practical way, we would like to make a small excursion, building on the most recent result from Section 5.1. Chains of value restrictions implied by a concept description $C \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$ w.r.t. a general $\mathcal{FL}_0$ TBox can be seen as words over the alphabet N_R and it turns out that certain languages composed of such words are in fact regular. This leads to the correlation between $\mathcal{FL}_0$ concept descriptions w.r.t. an unrestricted $\mathcal{FL}_0$ TBox and $\mathcal{FL}_{reg}$ concept descriptions, which we investigate in the following.

Firstly, it is possible to adjust the normalization procedure, mainly normalization phase 1 discussed in Section 4, to work with complex roles from $\mathfrak{R}(N_R)$ instead of just role names. It is not hard to see that this adjustment allows to transform $\mathcal{FL}_{reg}$ concept descriptions into a normal form that can also be called CCNF. Since the description of CCNF needs no adjustment regarding complex roles, we can adapt the results from Section 4 to $\mathcal{FL}_{reg}$. In particular, for every $\mathcal{FL}_{reg}$ concept description, there exists an equivalent $\mathcal{FL}_{reg}$ concept description in CCNF that is of the form $\forall R_1.A_1 \sqcap \ldots \sqcap \forall R_n.A_n$, where $A_i \in N_C$ and R_i describe regular languages L over N_R $(1 \leq i \leq n)$, as proposed in Section 2. Therefore, we can use R and L within $\mathcal{FL}_{reg}$ concept descriptions interchangeably, e.g. $\forall R.X = \forall L.X$. Using this knowledge, together with the result from Theorem 5.21, it is clear that the concept description

$$\forall L_{\mathcal{T}}(C, A_1).A_1 \sqcap \ldots \sqcap \forall L_{\mathcal{T}}(C, A_n).A_n$$

is an $\mathcal{FL}_{reg}$ concept in CCNF that is constructed from the $\mathcal{FL}_0$ concept C and the terminological knowledge given in the TBox $\mathcal{T}$. This supports the assumption that some results for certain inference problems and ontology services regarding $\mathcal{FL}_{reg}$ can be adopted to $\mathcal{FL}_0$ with general TBoxes. In particular, the inference problems subsumption, concept unification and the computation of generalizations such as least–common subsumers (lcs) or most specific concepts (msc) are of a special interest in todays DL community.

Lemma 6.1. *For $\mathcal{FL}_{reg}$ concepts $C = \forall L_1.A_1 \sqcap \ldots \sqcap \forall L_n.A_n$, $D = \forall M_1.A_1 \sqcap \ldots \sqcap \forall M_n.A_n$,*[3] *subsumption can be decided with language inclusion in the following way:*

$$C \sqsubseteq D \Longleftrightarrow \forall i \in \{1, \ldots, n\}.M_i \subseteq L_i$$

Continuing from Lemma 6.1 (proof in [4]) it is not hard to show the correlation between $\mathcal{FL}_{reg}$ concept descriptions and $\mathcal{FL}_0$ concept descriptions given an $\mathcal{FL}_0$ TBox, regarding the task of deciding subsumption.

[3] It is no restriction to assume the same set of concept names A_i because $\forall \emptyset.X \equiv \top$ always allows us to match the set of concept names occurring in both concept descriptions, in order for them to be equal.

Lemma 6.2. *For $\mathcal{FL}_0$ concepts C, D and a general $\mathcal{FL}_0$ TBox $\mathcal{T}$ the concepts*

$$C_{reg} := \prod_{A \in N_C} \forall L_{\mathcal{T}}(C, A).A$$

$$D_{reg} := \prod_{A \in N_C} \forall L_{\mathcal{T}}(D, A).A$$

are $\mathcal{FL}_{reg}$ concept descriptions such that

$$C \sqsubseteq_{\mathcal{T}} D \iff C_{reg} \sqsubseteq D_{reg}.$$

Proof. Due to Lemma 6.1 it is clear that

$$C_{reg} \sqsubseteq D_{reg} \iff \bigwedge_{A \in N_C} L_{\mathcal{T}}(D, A) \subseteq L_{\mathcal{T}}(C, A).$$

It is also not hard to see that

$$\bigwedge_{A \in N_C} L_{\mathcal{T}}(D, A) \subseteq L_{\mathcal{T}}(C, A) \iff L_{\mathcal{T}}(D) \subseteq L_{\mathcal{T}}(C).$$

Finally, by Lemma 3.3 we know $L_{\mathcal{T}}(D) \subseteq L_{\mathcal{T}}(C) \iff C \sqsubseteq_{\mathcal{T}} D$. □

Concerning the construction of the concept descriptions C_{reg} and D_{reg} we can finally elaborate why Section 5 only relies on the CCNF even though the PANF was also introduced earlier. The benefit of using PANF for the automaton in Section 5, would have been that $\mathcal{A}_{C,\mathcal{T}}$ would not require to keep track of value restrictions at every node (using the states in Q). Because every GCI only provides local restrictions, run–trees labeled with sets of concept names would have sufficed to locally enforced conditions for state transitions. However, in the effort to decide the subsumption $C \sqsubseteq_{\mathcal{T}} D$, by Lemma 4.8 we would have required to add the concept names A_C and A_D with the appropriate GCIs to $\mathcal{T}$. Since $\forall L_{\mathcal{T}}(C, A_D).A_D$ would then be a conjunct of C_{reg}, deciding $L_{\mathcal{T}}(C, A_D)$ would already require to decide $C \sqsubseteq_{\mathcal{T}} A_D$.

Investigating the correlation of problems between $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$ other than subsumption requires more care. For instance the computation of the least common subsumer of two $\mathcal{FL}_0$ concept descriptions C, D is everything but trivial. Formally, the lcs of C and D is a concept description E such that $C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$ and for all concept descriptions F such that $C \sqsubseteq_{\mathcal{T}} F$ and $D \sqsubseteq_{\mathcal{T}} F$, it holds that $E \sqsubseteq_{\mathcal{T}} F$. Since the lcs is strongly related to subsumption, one might think to use the result from Lemma 6.2 to construct an lcs w.r.t. the concept descriptions C_{reg}, D_{reg} within $\mathcal{FL}_{reg}$, because it might circumvent the effort to regard the given terminology upon the computation of the lcs within $\mathcal{FL}_0$. Intuitively, the lcs of two $\mathcal{FL}_{reg}$ concept descriptions $C = \prod_{i=1}^{n} \forall L_i.A_i$ and $D = \prod_{i=1}^{n} \forall M_i.A_i$ is a concept description that contains all conjuncts that C and D have in common. From Lemma 6.1 it follows quickly that this concept is $E = \prod_{i=1}^{n} \forall L_i \cap M_i.A_i$ and it is easy to see that $E \in \mathfrak{C}(\mathcal{FL}_{reg}, N_C, N_R)$. The following example illustrates the main problem when creating new $\mathcal{FL}_{reg}$ concept descriptions in the lcs case.

Example 6.3. *Let* $\mathcal{T} = \{B_1 \sqcap A \sqsubseteq \forall r.(A \sqcap B_1), B_2 \sqcap A \sqsubseteq \forall r.(A \sqcap B_2)\}$ *be an obviously cyclic* $\mathcal{FL}_0$ *TBox* $\mathcal{T}$*. The concept descriptions* $C = A \sqcap B_1$ *and* $D = A \sqcap B_2$ *can be transformed into their* $\mathcal{FL}_{reg}$ *correspondents (w.r.t.* $\mathcal{T}$*) as suggested in Lemma 6.2:*

$$C_{reg} = \forall r^*.A \sqcap \forall r^*.B_1$$

$$D_{reg} = \forall r^*.A \sqcap \forall r^*.B_2$$

Then the lcs of C_{reg} *and* D_{reg} *is* $E_{reg} = \forall r^*.A$*. However no* $\mathcal{FL}_0$ *concept description* E *can exist such that* $E_{reg} = \forall L_{\mathcal{T}}(E, A).A$ *w.r.t. the given TBox* $\mathcal{T}$*, because no axiom in* $\mathcal{T}$ *allows to entail infinite value restrictions with concept name* A *without either* B_1 *or* B_2*.*

Even though the constructed lcs is another $\mathcal{FL}_{reg}$ concept description, the languages (complex roles) used in this concept description may not be induced by the given $\mathcal{FL}_0$ TBox $\mathcal{T}$. It appears that the correlation between $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$ for the lcs problem is much more difficult. Example 6.3 immediately gives rise to the following non–trivial questions, including the result that for every $\mathcal{FL}_{reg}$ concept description there exists an equivalent $\mathcal{FL}_{reg}$ concept in CCNF:

1. For any given $\mathcal{FL}_{reg}$ concept description F, does there exist a general $\mathcal{FL}_0$ TBox $\mathcal{T}$ and an $\mathcal{FL}_0$ concept description E such that

$$E_{reg} = \prod_{A \in N_C} \forall L_{\mathcal{T}}(E, A).A \equiv F?$$

2. If not, under what circumstances do $\mathcal{T}$ and E exist?

Since answering these questions very likely requires a lot of effort and research, we have decided to leave this topic as a brief excursion in order to initiate a basis for upcoming research regarding $\mathcal{FL}_0$ with general TBoxes. To tackle these problems as well as the still very interesting problem of concept unification would simply overstep the scope of this thesis. We will now return to discuss the practical task of deciding subsumption in this unrestricted DL environment.

7 Practical Subsumption Algorithm

In the current section we will propose a simple algorithm deciding subsumption between concept names w.r.t. a general $\mathcal{FL}_0$ TBox. Due to Corollary 4.7 and Lemma 4.8, it is enough to consider subsumption between concept names and we can assume w.l.o.g. that the given TBox $\mathcal{T}$ is in PANF. Therefore, for this entire section, we fix $\mathcal{T}$ to be in PANF, consisting of GCIs $E \sqsubseteq F$ s.t. $E, F \in \mathfrak{C}(\mathcal{FL}_0, N_C, N_R)$, thus implicitly fixing the arity of functional interpretations in this section to $n = |N_R|$. In short, our algorithm will start creating the least functional model of one of the given concept names, stop at a certain time and check the root of the created model for containment of the second concept name, thereby deciding subsumption (Corollary 3.14). After presentation of the algorithm, it remains to show its termination, soundness and completeness. We start by introducing a collection of new notations, which overall render the rest of this section easier and more intuitive.

As in Section 5.1, regarding the access of labels of domain elements, let

$$\mathcal{I}(w) = \{A \in N_C \mid w \in A^{\mathcal{J}}\}$$

for any interpretation $\mathcal{I}$. Note, that for $w \notin \Delta^{\mathcal{I}}$, $\mathcal{I}(w) = \emptyset$.

Definition 7.1. *A **head** is an $n+1$ tuple of sets $(L, L_1, \ldots, L_n)$ with $|N_R| = n$, where $L \subseteq N_C$ is the root of the head and $L_i \subseteq N_C$ are the children ($1 \leq i \leq n$). The function hd maps nodes $w \in N_R^* = \Delta^{\mathcal{J}}$ from a given functional interpretation $\mathcal{J}$ to the head $(\mathcal{J}(w), \mathcal{J}(wr_1), \ldots, \mathcal{J}(wr_n))$ within its tree. When considering heads of domain elements within two different functional interpretations $\mathcal{I}$,$\mathcal{J}$ (sharing the same domain), we use $hd_{\mathcal{J}}(w_1)$ and $hd_{\mathcal{I}}(w_2)$ to clarify the distinction.*

The subset relation $\subseteq$ is extended to head structures in the natural way. For two heads $h_1 = (L, L_1, \ldots, L_n)$ and $h_2 = (M, M_1, \ldots, M_n)$,

$$h_1 \subseteq h_2 \iff L \subseteq M \wedge \bigwedge_{i=1}^{n} L_i \subseteq M_i.$$

In a similar fashion, the union is extended to head structures by constructing the set union element wise:

$$h_1 \cup h_2 = (L \cup M, L_1 \cup M_1, \ldots, L_n \cup M_n)$$

Additionally, for GCIs $E \sqsubseteq F$ in a TBox $\mathcal{T}$ that is in PANF, we also say that the concept descriptions E, F are in PANF. Recall that for a concept description C in CCNF (implied by PANF), $\widehat{C}$ describes the set of its conjuncts. Since concept descriptions in PANF are a conjunction of either concept names or value restrictions of depth 1, we can see how those concept descriptions can be displayed as head structures. For a concept X in PANF, $hd(X)$ is defined as follows:

$$hd(X) = (\widehat{X} \cap N_C, \{A \in N_C \mid \forall r_1.A \in \widehat{X}\}, \ldots, \{A \in N_C \mid \forall r_n.A \in \widehat{X}\})$$

For an algorithm that needs to procedurally alter the heads of domain elements, it is very cumbersome to only be able to alter the sets of domain elements

that concept names are mapped to. Therefore, we will use much simpler notations like

$$\mathcal{J}(w) := \{A_1, A_2, \ldots, A_k\}$$

or

$$hd(w) := (L, L_1, \ldots, L_n)$$

for $A_1, \ldots, A_k \in N_C$ and $L, L_1, \ldots, L_n \subseteq N_C$. Implicitly, the assignment $\mathcal{J}(w) := \{A_1, A_2, \ldots, A_k\}$ adds w to (exactly[4]) all $A_i^{\mathcal{J}}$ $(1 \leq i \leq k)$ and using this, $hd(w) := (L, L_1, \ldots, L_n)$ assigns $\mathcal{J}(w) := L$ and for all $j = 1, \ldots, n$ $\mathcal{J}(wr_j) := L_j$. In case for some j, $wr_j \notin \Delta^{\mathcal{J}}$, the assignment shall add wr_j to the current domain. The algorithm will iteratively create an interpretation that structurally resembles a functional interpretation (w.r.t. the interpretation of role relations, Definition 3.4, Property 1), starting at the root. Therefore, at every time during the computation, the domain of the created interpretation is finite. Consequently, the interpretations created by our algorithm will be called functional interpretation stubs and we distinctly use variables like $\mathcal{I}^{\triangle}$, $\mathcal{J}^{\triangle}$ to refer to them. Note that for a functional interpretation stub $\mathcal{J}^{\triangle}$ with the finite domain $\Delta^{\mathcal{J}^{\triangle}}$, not all domain elements w need to have n children. The function hd can yet be applied to such w (i.e. $hd_{\mathcal{J}^{\triangle}}(w)$), because for a missing child wr, the function $\mathcal{J}^{\triangle}(wr)$ will evaluate to the empty set.

Because a general $\mathcal{FL}_0$ TBox may contain terminological cycles, there may exist infinitely many domain elements in a functional interpretation that have a non–empty label. We propose that these labels recur in a cyclic fashion. In order to allow the algorithm to terminate, we need to recognize those recurring labels and *block* the respective nodes appropriately, to circumvent multiple computations of the same sub–tree. First, consider the order of domain elements in functional interpretations and functional interpretation stubs.

Definition 7.2. *Let $u = r_{i_1} \cdots r_{i_k}$ and $v = r_{j_1} \cdots r_{j_l}$ be two domain elements with $i_1, \ldots, i_k, j_1, \ldots, j_l \in \{1, \ldots, n\}$ for $N_R = \{r_1, \ldots, r_n\}$ in a domain $\Delta^{\mathcal{I}} \subseteq N_R^*$. The relation $\prec \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ is a strict total order over domain elements, s.t. $(u, v) \in \prec$ iff $|u| < |v|$ or for $k = l$, $(i_1, \ldots, i_k) \prec_{\mathbb{N}} (j_1, \ldots, j_l)$, for $\prec_{\mathbb{N}}$, the lexicographic order over tuples of natural numbers.*

It is easy to see that $\prec$ is a strict total order, because $<$ and $\prec_{\mathbb{N}}$ over (tuples of) natural numbers are both strict and total. Furthermore, $\prec$ describes a breadth–first–type order over elements of a tree.

Definition 7.3 (Anywhere Blocking)**.**
*A domain element $w \in \Delta^{\mathcal{J}^{\triangle}}$ is **blocked** iff*

$$\exists u, u' \in \Delta^{\mathcal{J}^{\triangle}}, v \in N_R^*. u \prec u' \wedge w = u'v \wedge \mathcal{J}^{\triangle}(u) = \mathcal{J}^{\triangle}(u')$$

At this point we introduced every notation that is necessary to present the practical subsumption algorithm in a simple and intuitive way.

[4]Essentially, the interpretation $\mathcal{J}$ will be altered so that by definition of $\mathcal{J}(w)$, the element w will be mapped to $\{A_1, \ldots, A_k\}$. That also implies that w will not be contained in any $B \in N_C \setminus \{A_1, \ldots, A_k\}$.

Definition 7.4 (Subs)**.**
The algorithm Subs$(A, B, \mathcal{T})$*, given two concept names* $A, B \in N_C$ *and an* $\mathcal{FL}_0$ *TBox* $\mathcal{T}$ *in PANF executes the following steps in their given order, unless stated otherwise within a step.*

1. $\Delta^{\mathcal{J}_0^\triangle} := \{\varepsilon\} =: A^{\mathcal{J}_0^\triangle}$, $i := 0$
2. $i := i + 1$, $\mathcal{J}_i^\triangle := \mathcal{J}_{i-1}^\triangle$
3. *Select the least* $w \in \Delta^{\mathcal{J}_i^\triangle}$ *w.r.t.* $\prec$*, that is* <u>*not blocked*</u>*, s.t.* $\exists E \sqsubseteq F \in \mathcal{T}.hd(E) \subseteq hd_{\mathcal{J}_{i-1}^\triangle}(w) \wedge hd(F) \not\subseteq hd_{\mathcal{J}_{i-1}^\triangle}(w)$
4. *If such a* w *exists, assign* $hd_{\mathcal{J}_i^\triangle}(w) := hd_{\mathcal{J}_{i-1}^\triangle}(w) \cup hd(F)$ *and continue with Step 2.*
5. *Else, terminate and answer* $A \sqsubseteq_\mathcal{T} B$ *iff* $B \in \mathcal{J}_i^\triangle(\varepsilon)$.

Subs starts by **initializing** (Step 1) the domain and the concept name A under the interpretation $\mathcal{J}_0^\triangle$, only to contain the root element ε. During the **incrementation** step (Step 2), the functional interpretation stub is copied to the next iteration and the iteration counter i is incremented by 1. The **selection** step (Step 3) attempts to find the first (according to $\prec$), not blocked domain element that violates some GCI in $\mathcal{T}$. If such an element exists, the head of the selected node will be altered in order to satisfy the respective GCI in what we call the **update** step (Step 4). The increment, select and update steps are looped until the select step cannot find a suitable domain element to update anymore. This happens exactly when all elements that are not blocked satisfy every GCI in $\mathcal{T}$. At this point, the **termination** step (Step 5) will produce the answer of Subs according to the existence of concept name B in the label of the root of the current interpretation and the computation comes to an end.

First of all, whenever referencing the functional interpretation stub $\mathcal{J}_i^\triangle$ at a specific iteration i, we refer to $\mathcal{J}_i^\triangle$ in its state after the iteration i is completed, i.e. at the end of Step 4 or 5. When a node w is selected by Step 3 and its head is updated in iteration i, we say that w *changes* at time/iteration i.

Theorem 7.5 (Termination)**.** Subs *terminates on every finite input.*

Proof. Every update step is strictly monotone. That is, in every iteration where Subs is not terminating, some label will increase in size. More precisely, any single node w cannot change more than $|\mathcal{T}|$ times, because every time some w is selected with some GCI $E \sqsubseteq F \in \mathcal{T}$ at iteration i, we have $hd(F) \subseteq hd_{\mathcal{J}_{i+1}^\triangle}(w)$ at the next iteration. Thus w cannot be updated with the same GCI more than once and therefore w cannot change infinitely many times for a finite $\mathcal{T}$.

For any iteration $i \geq 0$, the amount of nodes that are not blocked in $\Delta^{\mathcal{J}_i^\triangle}$ is upper bounded by $2^{|N_C|}$ nodes. From Definition 7.3, it follows that every node w, s.t. $|\{v \in \Delta^{\mathcal{J}_i^\triangle} \mid v \prec w\}| \geq 2^{|N_C|}$ must be blocked, because there are only $2^{|N_C|}$ distinct labels and if w has $2^{|N_C|}$ predecessors w.r.t. $\prec$, its label must already occur at one of these predecessors. Because of the finite amount of nodes that are not blocked and the argument that every node can only change finitely many times, there must always exist an iteration for which no node can be selected by Step 3 of Definition 7.4, leading to the termination of Subs. □

For the soundness of SUBS, we will show that the algorithm never adds a concept name to the label of a node w, that would not also be in $\mathcal{I}_{A,\mathcal{T}}(w)$. To keep the soundness proof as simple as possible, some intermediate results are required first. We extend the subset relation between functional interpretations, to allow for a functional interpretation stub on the left hand side:

$$\mathcal{J}^\triangle \subseteq \mathcal{I} \iff \forall A \in N_C . A^{\mathcal{J}^\triangle} \subseteq A^{\mathcal{I}}.$$

Proposition 7.6. *The following statements are equivalent*

1. $\mathcal{J}^\triangle \subseteq \mathcal{I}$
2. $\forall w \in \Delta^{\mathcal{J}^\triangle} . \mathcal{J}^\triangle(w) \subseteq \mathcal{I}(w)$
3. $\forall w \in \Delta^{\mathcal{J}^\triangle} . hd_{\mathcal{J}^\triangle}(w) \subseteq hd_{\mathcal{I}}(w)$

Proof. The equivalence of Statements 2 and 3 is rather trivial, by the definition of $\subseteq$ for head structures (Definition 7.1).

Regarding the Statements 1 and 2, by the definition of $\mathcal{J}^\triangle(w)$ (and $\mathcal{I}(w)$), it holds for any $w \in \Delta^{\mathcal{J}^\triangle}$, that $w \in A^{\mathcal{J}^\triangle}$ iff $A \in \mathcal{J}^\triangle(w)$ and analogous for $\mathcal{I}(w)$. Therefore, $\forall A \in N_C, w \in \Delta^{\mathcal{J}^\triangle} . w \in A^{\mathcal{J}^\triangle} \implies w \in A^{\mathcal{I}}$ is equivalent to $\forall A \in N_C, w \in \Delta^{\mathcal{J}^\triangle} . A \in \mathcal{J}^\triangle(w) \implies A \in \mathcal{I}(w)$. □

Proposition 7.7. *For an $\mathcal{FL}_0$ concept description X in PANF, a functional interpretation $\mathcal{J}$ and a domain element $w \in \Delta^{\mathcal{J}}$, it holds that*

$$hd(X) \subseteq hd_{\mathcal{J}}(w) \iff w \in X^{\mathcal{J}}.$$

Proof. Let $X^\varepsilon = \widehat{X} \cap N_C$ and $X^{r_j} = \{A \in N_C \mid \forall r_j . A \in \widehat{X}\}$ $(1 \leq j \leq n)$, then by definition, we have

$$hd(X) = (X^\varepsilon, X^{r_1}, \ldots, X^{r_n}), \text{ and}$$

$$hd_{\mathcal{J}}(w) = (\mathcal{J}(w), \mathcal{J}(wr_1), \ldots, \mathcal{J}(wr_n)),$$

which is why we need to show that

$$X^\varepsilon \subseteq \mathcal{J}(w) \wedge \bigwedge_{j=1}^{n} X^{r_j} \subseteq \mathcal{J}(wr_j) \iff \bigwedge_{Y \in \widehat{X}} w \in Y^{\mathcal{J}} \iff w \in X^{\mathcal{J}}$$

holds. Because $A \in \mathcal{J}(w)$ iff $w \in A^{\mathcal{J}}$ and due to the structure of functional interpretations, it holds for all $j = 1, \ldots, n$ that $A \in \mathcal{J}(wr_j)$ iff $w \in (\forall r_j . A)^{\mathcal{J}}$. Therefore $A \in X^\varepsilon$ implies $w \in A^{\mathcal{J}}$ and $A \in X^{r_j}$ implies $w \in (\forall r_j . A)^{\mathcal{J}}$ $(1 \leq j \leq n)$. Since $\widehat{X} = X^\varepsilon \cup \bigcup_{j=1}^{n} X^{r_j}$, w is contained in all conjuncts Y of X under $\mathcal{J}$, making it also contained in $X^{\mathcal{J}}$.

For the other direction, assume $w \in Y^{\mathcal{J}}$ for all conjuncts $Y \in \widehat{X}$. For $Y \in X^\varepsilon \subseteq N_C$, $w \in Y^{\mathcal{J}} \iff Y \in \mathcal{J}(w)$ holds by definition of $\mathcal{J}(w)$. If $Y = \forall r . A$, then $A \in X^r$ and $w \in Y^{\mathcal{J}} \iff A \in \mathcal{J}(wr)$ also holds, thus satisfying every subset relation on the left–hand side of the equation. □

Note, that the result of Proposition 7.7 only applies to functional interpretations, because it relies on the fact that every domain element has exactly one successor per role. The argument that $A \in \mathcal{J}(wr)$ iff $w \in (\forall r.A)^{\mathcal{J}}$ would not hold if w has no r–successor. In this case, $w \in (\forall r.A)^{\mathcal{J}}$ would be trivially satisfied but does not necessarily imply $A \in \mathcal{J}(wr)$. Actually, this singular argument holds for all domain elements that have exactly one successor per role in N_R, which we shall use at a later time.

The previous statement — that the algorithm will never add a concept name to the label of a node, that would not also be contained in the label of that node within the least functional model of A w.r.t. $\mathcal{T}$— will be expressed by stating that the functional model stub $\mathcal{J}_i^{\triangle}$ will be a subset of $\mathcal{I}_{A,\mathcal{T}}$ at every iteration $i \geq 0$.

Lemma 7.8. *For $\mathcal{J}_i^{\triangle}$, created with* Subs *at any iteration $i \geq 0$, it holds that $\mathcal{J}_i^{\triangle} \subseteq \mathcal{I}_{A,\mathcal{T}}$.*

Proof. We can prove this by induction on the iteration counter i. The induction start with $i = 0$ is trivial by Definition 3.4, Property 3. For the hypothesis, assume that $\mathcal{J}_i^{\triangle} \subseteq \mathcal{I}_{A,\mathcal{T}}$ already holds. We need to show that $\mathcal{J}_{i+1}^{\triangle} \subseteq \mathcal{I}_{A,\mathcal{T}}$ also holds. If Subs terminates at iteration $i+1$, then no node was selected in Step 3 and thus $\mathcal{J}_i^{\triangle} = \mathcal{J}_{i+1}^{\triangle}$. In this case, the claimed subset relation is trivial by the induction hypothesis.

Otherwise, there exists a w that is the least domain element w.r.t. $\prec$, that is not blocked and there exists a GCI $E \sqsubseteq F \in \mathcal{T}$ s.t. $hd(E) \subseteq hd_{\mathcal{J}_i^{\triangle}}(w)$ and $hd(F) \not\subseteq hd_{\mathcal{J}_i^{\triangle}}(w)$. By Proposition 7.6 and the induction hypothesis, $hd(E) \subseteq hd_{\mathcal{J}_i^{\triangle}}(w)$ implies $hd(E) \subseteq hd_{\mathcal{I}_{A,\mathcal{T}}}(w)$ and by Definition 3.4, Property 2 and Proposition 7.7, this implies $hd(F) \subseteq hd_{\mathcal{I}_{A,\mathcal{T}}}(w)$. Then, $hd_{\mathcal{J}_i^{\triangle}}(w) \subseteq hd_{\mathcal{I}_{A,\mathcal{T}}}(w) \wedge hd(F) \subseteq hd_{\mathcal{I}_{A,\mathcal{T}}}(w) \Longrightarrow hd_{\mathcal{J}_i^{\triangle}}(w) \cup hd(F) \subseteq hd_{\mathcal{I}_{A,\mathcal{T}}}(w)$. Since only labels of domain elements associated with the head of the selected w change during one iteration, the last result, the hypothesis and Proposition 7.6 imply $\mathcal{J}_{i+1}^{\triangle} \subseteq \mathcal{I}_{A,\mathcal{T}}$. □

Theorem 7.9 (Soundness)**.** *The algorithm* Subs *is sound.*

Proof. At any iteration i (especially the iteration m where Subs terminates) it holds by Lemma 7.8 that $\mathcal{J}_i^{\triangle}(\varepsilon) \subseteq \mathcal{I}_{A,\mathcal{T}}(\varepsilon)$. Thus, if Subs answers that $A \sqsubseteq_{\mathcal{T}} B$ holds, then B is at the root in $\mathcal{J}_m^{\triangle}$ and thereby at the root of $\mathcal{I}_{A,\mathcal{T}}$. This implies that $\mathcal{I}_{A,\mathcal{T}}$ is also a functional model of B w.r.t. $\mathcal{T}$. By Lemma 3.9 and Corollary 3.14 this implies that $A \sqsubseteq_{\mathcal{T}} B$ must hold, making Subs sound. □

The main idea behind the completeness proof for Subs is to create a functional model of A w.r.t. $\mathcal{T}$, that "starts" with the functional interpretation stub created by Subs. To accomplish this, we introduce the following helpful notation.

Definition 7.10. *For a functional interpretation (stub) $\mathcal{J}$ and a node $w \in \Delta^{\mathcal{J}}$, the least node $v \in \Delta^{\mathcal{J}}$ w.r.t. $\prec$ with $\mathcal{J}(v) = \mathcal{J}(w)$ is called the* ***first occurrence*** *of the label of w.*

When considering a functional interpretation stub $\mathcal{J}_i^{\triangle}$ created by Subs at iteration i, we can make several observations regarding Definition 7.10. Note

that a node is either blocked because its parent is blocked, or because its label already exists for a smaller node w.r.t. $\prec$. If w is a blocked node whose parent is not blocked, then there exists a node $v \neq w$, that is the least node with $\mathcal{J}(v) = \mathcal{J}(w)$ (w.r.t. $\prec$), therefore, v cannot be blocked. If w is not blocked, then w itself must be the first occurrence of the head of w.

Theorem 7.11 (Completeness)**.** *The algorithm* SUBS *is complete.*

Proof. We show that it is possible to construct a functional model of A w.r.t. $\mathcal{T}$, that "starts" with the functional interpretation stub $\mathcal{J}_m^\triangle$ that was created by SUBS terminating at iteration m. Let $\mathcal{I} = \mathcal{J}_m^\triangle$ and apply a few initial adjustments to $\mathcal{I}$. First, remove every element from $\Delta^\mathcal{I}$ that has a blocked parent (in $\mathcal{J}_m^\triangle$). It is easy to see, that this will remove only blocked domain elements and all remaining blocked nodes will be leaves in the tree described by $\mathcal{I}$. Also, for every node $w \in \Delta^\mathcal{I}$ that is not blocked, for all $j \in \{1, \ldots, n\}$ s.t. $wr_j \notin \Delta^\mathcal{I}$, add wr_j to $\Delta^\mathcal{I}$. Note that when these wr_j are not added to the interpretation of any concept name under $\mathcal{I}$, $\mathcal{I}(wr_j) = \emptyset$ holds. Now it is easy to see that all leaves in the tree of $\mathcal{I}$ are either nodes that where blocked in $\mathcal{J}_m^\triangle$ or nodes with an empty label. After these adjustments, we say that $\mathcal{I}$ is in its initial state.

For the iterative construction of $\mathcal{I}$, repeat the following step infinitely many times:

> Pick the least leaf $w \in \Delta^\mathcal{I}$ w.r.t. $\prec$ and add wr_j to $\Delta^\mathcal{I}$ s.t. $\mathcal{I}(wr_j) = \emptyset$ for all $j = 1, \ldots, n$. If $\mathcal{I}(w) \neq \emptyset$, let u be the first occurrence of the label of w and adjust the labels of wr_j in $\mathcal{I}$ s.t. $\mathcal{I}(wr_j) = \mathcal{I}(ur_j)$, for all $j = 1, \ldots, n$.

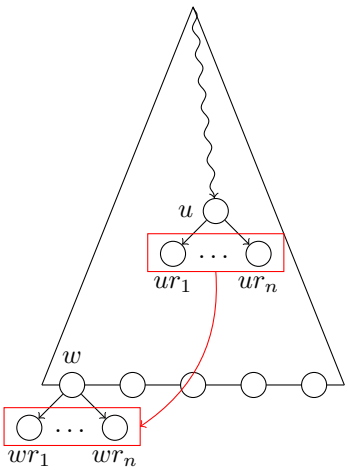


Figure 2: Illustration of one iteration for the construction of $\mathcal{I}$.

Figure 2 shows how one iteration as described above can be seen as a copying procedure of successor elements. If $\mathcal{I}(w) \neq \emptyset$ for a leaf w, then w must be a blocked node and therefore the first occurrence of its label must exist and cannot be blocked. Additionally, for $\mathcal{I}$ in its initial state, all u that are not blocked have exactly one successor per role in N_R.

We show that at every time, all inner nodes (not leaf nodes) in $\mathcal{I}$ satisfy all GCIs in $\mathcal{T}$. Since the construction of $\mathcal{I}$ is defined iteratively, we can prove this by induction. For the induction start we observe the initial state of $\mathcal{I}$. It is easy to see that every inner node is a node that is not blocked in $\mathcal{J}_m^{\triangle}$. Because of the initial adjustment to add empty nodes, all inner nodes of $\mathcal{I}$ assume the structure required by a functional interpretation. Therefore, the argument of Proposition 7.7 applies to all inner nodes of $\mathcal{I}$. By termination of SUBS we know that no GCI $E \sqsubseteq F \in \mathcal{T}$ exists such that $hd(E) \subseteq hd(w)$ and $hd(F) \not\subseteq hd(w)$ for every inner node w, thus $w \in E^{\mathcal{I}} \Longrightarrow w \in F^{\mathcal{I}}$ holds for every GCI $E \sqsubseteq F \in \mathcal{T}$.

When applying one iteration of constructing $\mathcal{I}$, the least leaf w is picked and all (structurally required) successors are added and given specific labels. No other labels within $\mathcal{I}$ are affected by such a step and the set of inner nodes increases only by w. Therefore, we only need to show that the head of w satisfies all GCIs after the iteration. If $\mathcal{I}(w) = \emptyset$ then all successors will remain with an empty label as well. Thus, w trivially satisfies every GCI in $\mathcal{T}$. In case $\mathcal{I}(w) \neq \emptyset$, then let u be the first occurrence of the label of w. Because the labels of the successors of u are copied into the labels of the successors of w, we have $hd(w) = hd(u)$. By our induction hypothesis, we know that the head of u satisfies all GCIs in $\mathcal{T}$ and therefore so does the head of w.

We have shown that in the infinity, every inner node in the interpretation $\mathcal{I}$ will satisfy every GCI in $\mathcal{T}$. It remains to show that every possible node $w \in N_R^*$ is an inner node in $\mathcal{I}$ in the infinity, i.e. the construction of $\mathcal{I}$ is fair because every possible node in N_R^* is created at some time. For any leaf w in $\Delta^{\mathcal{I}}$ at any time, there only exist finitely many nodes in N_R^*, that are smaller than w w.r.t. $\prec$. Since every iteration turns a leaf into an inner node, there can only be finitely many nodes that are processed before w. Hence every element $w \in N_R^*$ exists in $\mathcal{I}$ in the infinity.

Since $\Delta^{\mathcal{I}} = N_R^*$, all nodes in $\Delta^{\mathcal{I}}$ are inner nodes that satisfy all GCIs. Furthermore, because $A \in \mathcal{I}(\varepsilon)$ by initialization of SUBS and $\mathcal{I}$, $\mathcal{I}$ is a functional model of A w.r.t. $\mathcal{T}$. From Lemma 3.9 we know that $\mathcal{I}_{A,\mathcal{T}} \subseteq \mathcal{I}$ and since $\mathcal{I}(\varepsilon) = \mathcal{J}_m^{\triangle}(\varepsilon)$, $B \in \mathcal{I}_{A,\mathcal{T}}(\varepsilon)$ implies $B \in \mathcal{J}_m^{\triangle}(\varepsilon)$, concluding the completion proof of SUBS. □

Altogether, the three theorems of this section ensure that SUBS can be used in a practical scenario and that it will always give the right answer. We want to highlight at this point that the algorithm employs anywhere blocking instead of subtree blocking, resulting in a lower complexity class.[5] With anywhere blocking, we can use an argument from Theorem 7.5 to show that, because every node with more than $2^{|N_C|}$ predecessors (according to $\prec$) has to be blocked, the algorithm is of exponential complexity. Due to already known results towards the complexity of $\mathcal{FL}_0$ [2], we cut this discussion short. Most importantly, it remains to investigate the realistic computational performance of an implementation of SUBS, in order to reasonably discuss the difficulty of deciding subsumption w.r.t. general $\mathcal{FL}_0$ TBoxes.

[5] With subtree blocking, it would only be ensured that every node with $2^{|N_C|}$ ancestors must be blocked. Therefore, in the worst case, SUBS would create an n–ary tree of exponential depth, leading to a domain with $n^{(2^{|N_C|})}$ elements

8 Conclusion and Future Work

To conclude the present thesis, we begin with a thorough recap. Starting from the motivation to gain tractable reasoning mechanisms for large–scale yet realistic scenarios, we set out to introduce a practical procedure, specifically tailored to handle the problem of concept subsumption w.r.t. general TBoxes in the description language $\mathcal{FL}_0$. We started by describing the potentially infinite set of value restrictions $L_{\mathcal{T}}(C)$, that are entailed by a given $\mathcal{FL}_0$ concept description C. $L_{\mathcal{T}}(C)$ can be described by a so–called least functional model, providing semantics in a specific tree structure. The result that reasoning w.r.t. functional models suffices to make general conclusions regarding subsumption allowed for the employment of looping tree automata, to solve exactly the task of deciding subsumption. Recall that the automaton $\mathcal{A}_{C,\mathcal{T}}$ as described in Section 5 accepts exactly those functional models of C, that do not support the subsumption between the given concepts C and D. Since functional models of the same concept description are closed under intersection, we were able to find an elegant minimization of $\mathcal{A}_{C,\mathcal{T}}$, so that it may only accept the least functional model of C. This minimization was used to show regularity of the languages $L_{\mathcal{T}}(C, A)$ (comprised in $L_{\mathcal{T}}(C)$), which allowed for an interesting excursion towards the correlation between $\mathcal{FL}_0$ with general TBoxes and $\mathcal{FL}_{reg}$, with regard to concept subsumption. It turned out that at least subsumption coincides for these two paradigms, yet some questions remained that are worth a thorough investigation. Finally, we turned towards the task proposed by our initial motivation, namely creating the algorithm Subs, that is able to decide subsumption in $\mathcal{FL}_0$ with general TBoxes for practical applications. The scope of this thesis was concluded with the formal investigation of Subs and its properties regarding correctness.

We have seen that general $\mathcal{FL}_0$ TBoxes describe a set of regular languages of words over the alphabet N_R, starting from a given concept description C. In this direction, such a C w.r.t. a background terminology can be expressed as an $\mathcal{FL}_{reg}$ concept description such that at least subsumption coincides in $\mathcal{FL}_0$ with GCIs and $\mathcal{FL}_{reg}$. However, as we have pointed out, the other direction seems not as obvious. Given an $\mathcal{FL}_{reg}$ concept description, does there exist an $\mathcal{FL}_0$ terminology such that we can create an $\mathcal{FL}_0$ concept, essentially mimicking the initial $\mathcal{FL}_{reg}$ concept. This can be especially helpful for problems that are already solved in $\mathcal{FL}_{reg}$, including the computation of generalizations such as least common subsumers or even concept unification [4] and matching.

As motivated by [2, 9], we aimed for a direct approach to decide concept subsumption with general TBoxes. Of course at this point, we can only make assumptions about the improved performance of such an approach. Since Subs will compete with tableaux based algorithms for $\mathcal{ALC}$, the basis for our assumption lies in the fact that those tableaux algorithms are non–deterministic and therefore include expensive backtracking mechanisms in order to find appropriate solutions. Subs is deterministic (aside from the don't–care non–deterministic choice of GCI in Step 3), and even though it is of exponential worst–case complexity, we propose that the realistic performance will largely depend on the length of terminological cycles instead of the general size of the given TBox. Therefore, we conclude that an experimental verification of this assumption is of a high priority within affiliated future research.

References

[1] Franz Baader. Terminological cycles in a description logic with existential restrictions. IJCAI'03, pages 325–330. Morgan Kaufmann, 2003.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. IJCAI'05, pages 364–369, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

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

[4] Franz Baader and Ralf Küsters. Unification in a description logic with transitive closure of roles. In Robert Nieuwenhuis and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2250 of *Lecture Notes in Computer Science*, pages 217–232. Springer Berlin Heidelberg, 2001.

[5] Franz Baader and Stephan Tobies. The inverse method implements the automata approach for modal satisfiability. *CoRR*, abs/cs/0412101, 2004.

[6] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics, 1997.

[7] Francesco M. Donini and Fabio Massacci. Exptime tableaux for $\mathcal{ALC}$. *ARTIFICIAL INTELLIGENCE*, 124(1):87–138, 2000.

[8] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[9] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning*, LPAR '99, pages 161–180, London, UK, UK, 1999. Springer-Verlag.

[10] Grzegorz Rozenberg and Arto Salomaa. *Handbook of Formal Languages: Volume 1. Word, Language, Grammar*. Handbook of Formal Languages. Springer, 1997.