



Subsumption in $\mathcal{FL}_{\perp reg}$ with TBoxes Is in EXPTIME*

Michał Henne¹, Barbara Morawska¹, and Paweł Parys²

¹ Institute of Computer Science, University of Opole, Poland
{michal.henne, barbara.morawska}@uni.opole.pl

² Institute of Informatics, University of Warsaw, Poland.
parys@mimuw.edu.pl

Abstract

Description Logics (DLs) are a family of formal languages used for representing and reasoning about structured knowledge in terms of concepts and their relationships. The expressive power of a DL depends on the constructors available for building complex concepts.

In this work, we investigate subsumption in the restricted description logic $\mathcal{FL}_{\perp reg}$ and the related fragments $\mathcal{FL}_{reg}$, $\mathcal{FL}_{\perp}$, and $\mathcal{FL}_0$. These formalisms support value restrictions over role names, where the subscript *reg* indicates the use of regular expressions over roles.

Subsumption between two concept descriptions in $\mathcal{FL}_{\perp reg}$ and $\mathcal{FL}_{reg}$ is PSPACE-complete. When subsumption is considered with respect to a TBox (i.e., a set of axioms), the complexity increases to EXPTIME-complete. These results can be derived either from complexity bounds established for more expressive logics or from algorithms designed for harder reasoning problems.

We reprove the PSPACE-completeness result and provide a new proof of EXPTIME-completeness for $\mathcal{FL}_{reg}$ and $\mathcal{FL}_{\perp reg}$ with TBoxes via a novel reduction to parity pushdown games. Our algorithm relies only on the constructs available in these logics and may therefore be implemented more easily.

1 Introduction

Description logics constitute a family of formal languages designed to represent structured knowledge in terms of concept hierarchies and the relationships between them. They provide a rigorous framework for describing classes of objects, their properties, and the relations that hold among them. A central aim of description logics is to support precise concept definitions and to enable automated reasoning about class membership and inter-concept relationships.

A fundamental reasoning task in this context is the decision problem of whether a subsumption relation holds between two given concept descriptions. If subsumption is decidable in a description logic, then the concept hierarchy (taxonomy) induced by the subsumption relation can be computed automatically, facilitating ontology construction and maintenance [7].

*This work was supported by the National Science Centre, Poland, under the OPUS 29 project no. 2025/57/B/ST6/04535, “From Games to Algorithms: Studying Reasoning in Description Logics”.

In this paper, we present a novel algorithm for the subsumption problem in certain restricted description logics, with potential applications in ontology maintenance.

Furthermore, it is often desirable to restrict the class of admissible interpretations of an ontology to those that satisfy a set of general constraints, while preserving the decidability of key reasoning tasks such as subsumption. These constraints can be formalized as a finite set of axioms describing relationships between concepts, known as a TBox.

The subsumption problem, both with and without TBoxes, has been extensively studied across a broad spectrum of description logics.

A particular description logic is characterized by the set of constructors allowed for concept definitions. Individual logics therefore differ in their expressive power, depending on the constructors provided.

In this paper, we focus on description logics that admit only conjunction ($\sqcap$), the top concept ($\top$), value restrictions ($\forall r.C$), and, optionally, the bottom concept ($\perp$), interpreted as the empty set. These constructors are used to build complex concept descriptions from a given set of atomic concept and role names.

The logic comprising the first three constructors is denoted by $\mathcal{FL}_0$, while the extension including $\perp$ is denoted by $\mathcal{FL}_{\perp}$. If value restrictions are generalized to regular expressions over the alphabet of role names, the resulting logic is denoted by $\mathcal{FL}_{reg}$, and its extension with the bottom concept $\perp$ is denoted by $\mathcal{FL}_{\perp reg}$.

For instance, in $\mathcal{FL}_0$, one can describe the concept of an undergraduate student as a student who attends only undergraduate courses by the equivalence

$$\text{UndergraduateStudent} \equiv \text{Student} \sqcap \forall \text{attends}.\text{UndergraduateCourse},$$

where **attends** is a role name relating a person and a course they attend.

In $\mathcal{FL}_{\perp}$, one can express the fact that professors do not attend any courses by the subsumption axiom

$$\text{Professor} \sqsubseteq \forall \text{attends}.\perp.$$

In $\mathcal{FL}_{reg}$, one can describe the more complex concept of a computer science course whose prerequisites are computer science courses, all prerequisites of those prerequisites are computer science courses, and so on, for any number of steps through the prerequisite relation:

$$\forall \text{prerequisite}^*.\text{ComputerScienceCourse}.$$

The computational complexity of the subsumption problem between two concept descriptions depends on the set of constructors available in the underlying logic. [Table 1](#) summarizes the current state of the art. The entries in bold font are those in which we are particularly interested in this paper.

The first two entries in the first column indicate the existence of polynomial-time algorithms for deciding subsumption in $\mathcal{FL}_0$ and $\mathcal{FL}_{\perp}$, results that have been known for a long time (see, e.g., [7]).

The third and fourth entries in the first column were not stated anywhere explicitly up to our knowledge, although these complexities are consequences of other results. For example, PSPACE complexity for $\mathcal{FL}_{reg}$ follows from PSPACE-completeness of subsumption in $\mathcal{FL}_0$ with cyclic concept definitions interpreted in gfp-semantics [2], which is equivalent to $\mathcal{FL}_{reg}$ [3]. The complexity of $\mathcal{FL}_{\perp reg}$ follows from the characterization of concept equivalence in Baader and Küsters [10]. In this paper we use a similar characterization of subsumption as in Baader and Küsters [10] to prove the complexity of the subsumption problem in $\mathcal{FL}_{reg}$ and $\mathcal{FL}_{\perp reg}$ (Section 4). Another way to infer this complexity upper bound would be to refer to the result

DL	without TBox	with TBox
$\mathcal{FL}_0$	PTime	EXPTIME-complete
$\mathcal{FL}_{\perp}$	PTime	EXPTIME-complete
$\mathcal{FL}_{reg}$	PSPACE-complete	EXPTIME-complete
$\mathcal{FL}_{\perp reg}$	PSPACE-complete	EXPTIME-complete

Table 1: Complexity of the subsumption problem in various description logics.

of Baader and Küsters [9] which states that the matching problem is in PSPACE for $\mathcal{FL}_{reg}$, and matching subsumes the subsumption problem.

The first entry in the second column is established by Baader, Brandt, and Lutz [5] by a reduction to and from the subsumption problem in $\mathcal{FL}_0^{tf}$ with TBox [21]. The description logic $\mathcal{FL}_0^{tf}$ is a variant of $\mathcal{FL}_0$ in which all roles are interpreted as total functions. The EXPTIME-completeness of the problem was also established independently by Hofmann [16].

The result mentioned in the second entry in the second column follows from Theorem 2.2 in Baader et al. [8].

The third and fourth entries in the second column are consequences of the fact that satisfiability in Propositional Dynamic Logic (PDL) is decidable in EXPTIME [13]. The description logics $\mathcal{FL}_{reg}$ and $\mathcal{FL}_{\perp reg}$ are contained in PDL and the TBoxes can be *internalized* in a satisfiability problem as described in De Giacomo and Lenzerini [11] (and observed in [20, 6]).

Clearly the EXPTIME-hardness result from the first entry in the second column ($\mathcal{FL}_0$ with TBoxes) implies EXPTIME-hardness of the subsumption problem in the more general logics in the last three entries.

Hence, the results presented in this paper are not new and belong to the common body of knowledge. Nevertheless, the aim of this paper is to provide a novel method for solving the subsumption problem in all the logics considered—a method that does not rely on stronger description logics or on algorithms designed to solve more difficult problems. In this way, we aim to solve the subsumption problem using only the means provided by a given logic. Since the logics considered here are very restricted, the proposed procedure may be implemented relatively easily.

The approach to constructing new algorithms is to establish an equivalence between an instance of the subsumption problem with TBoxes and a suitably defined parity pushdown game associated with the instance, and then to apply existing techniques from game theory to solve it.

Parity pushdown games arise from the combination of pushdown systems, that is, transition systems equipped with a stack, and parity games—two-player infinite-duration games played on finite graphs, in which the winner is determined by the parity of the priorities visited infinitely often. This framework has found widespread application in the verification of recursive programs and pushdown models [1, 22], as well as in interprocedural static analysis frameworks based on weighted pushdown systems [19].

Solving parity pushdown games, that is, deciding the existence of a winning strategy for one of the players, is EXPTIME-complete [22]. A variant of pushdown games has been used to establish EXPTIME-completeness of the subsumption problem in the description logic $\mathcal{FL}_0$ with TBoxes [16]. Motivated by this result, we present a new method for solving subsumption problems in several description logics extended with TBoxes.

The remainder of this paper is structured as follows. [Section 2](#) recalls the necessary pre-

liminaries. In [Section 3](#), we present syntactic laws and normal forms used throughout the paper. [Section 4](#) establishes PSPACE-completeness of subsumption in $\mathcal{FL}_{reg}$ and $\mathcal{FL}_{\perp reg}$ without TBoxes. In [Section 5](#), we present a reduction from $\mathcal{FL}_{\perp reg}$ with TBoxes to $\mathcal{FL}_{reg}$ with TBoxes, allowing us to focus on subsumption in $\mathcal{FL}_{reg}$ with TBoxes. [Section 6](#) provides the necessary background on parity pushdown games. In [Section 7](#), we prove our main result—EXPTIME-completeness of subsumption in $\mathcal{FL}_{reg}$ with TBoxes—by reduction to parity pushdown games. Finally, [Section 8](#) concludes the paper.

2 Preliminaries

We introduce the description logics considered in this paper, starting with the most complex one, $\mathcal{FL}_{\perp reg}$; the other logics are obtained as its syntactic fragments.

2.1 The logic $\mathcal{FL}_{\perp reg}$

Let $\mathbf{N}_C$ and $\mathbf{N}_R$ be disjoint countable sets of *concept names* and *role names*, respectively.

Regular role expressions are just usual regular expressions built from role names using union, concatenation, and Kleene star. Thus, they are defined by the grammar

$$E ::= \emptyset \mid \varepsilon \mid r \mid (E + E) \mid (EE) \mid E^* \quad (r \in \mathbf{N}_R).$$

Then, *concept descriptions* of $\mathcal{FL}_{\perp reg}$ are generated by the following grammar:

$$C ::= A \mid \top \mid \perp \mid (C \sqcap C) \mid \forall E.C \quad (A \in \mathbf{N}_C).$$

The semantics of $\mathcal{FL}_{\perp reg}$ is given by *interpretations* of the form $\mathcal{I} = (\mathcal{U}^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\mathcal{U}^{\mathcal{I}}$ is a nonempty domain of discourse (*universe*), and $\cdot^{\mathcal{I}}$ is an interpretation function mapping each concept name $A \in \mathbf{N}_C$ to a set $A^{\mathcal{I}} \subseteq \mathcal{U}^{\mathcal{I}}$, and each role name $r \in \mathbf{N}_R$ to a binary relation $r^{\mathcal{I}} \subseteq \mathcal{U}^{\mathcal{I}} \times \mathcal{U}^{\mathcal{I}}$. For a regular role expression, we can define its language $\mathcal{L}(E) \subseteq \mathbf{N}_R^*$ in the usual way:

$$\begin{aligned} \mathcal{L}(\emptyset) &= \emptyset, & \mathcal{L}(\varepsilon) &= \{\varepsilon\}, & \mathcal{L}(r) &= \{r\}, & \mathcal{L}(E_1 E_2) &= \{w_1 w_2 \mid w_1 \in \mathcal{L}(E_1), w_2 \in \mathcal{L}(E_2)\}, \\ \mathcal{L}(E_1 + E_2) &= \mathcal{L}(E_1) \cup \mathcal{L}(E_2), & \mathcal{L}(E^*) &= \{w_1 w_2 \dots w_n \mid n \in \mathbb{N}, w_1, w_2, \dots, w_n \in \mathcal{L}(E)\}, \end{aligned}$$

where ε is the empty word, while $w_1 w_2$ and $w_1 w_2 \dots w_n$ denote concatenations of words. Regular role expressions and complex concept descriptions are interpreted as follows, where “ $\circ$ ” denotes the composition of binary relations:

$$\begin{aligned} E^{\mathcal{I}} &= \{r_1^{\mathcal{I}} \circ r_2^{\mathcal{I}} \circ \dots \circ r_n^{\mathcal{I}} \mid r_1 r_2 \dots r_n \in \mathcal{L}(E)\}, \\ \top^{\mathcal{I}} &= \mathcal{U}^{\mathcal{I}}, & \perp^{\mathcal{I}} &= \emptyset, & (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \\ (\forall E.C)^{\mathcal{I}} &= \{x \in \mathcal{U}^{\mathcal{I}} \mid y \in C^{\mathcal{I}} \text{ for all } y \text{ such that } (x, y) \in E^{\mathcal{I}}\}. \end{aligned}$$

2.2 Weaker logics

The logic $\mathcal{FL}_{\perp}$ is a syntactically limited fragment of $\mathcal{FL}_{\perp reg}$, where we are not allowed to use value restrictions $\forall E.C$ for arbitrary regular expressions E , but only $\forall r.C$ for single role names r .

The logic $\mathcal{FL}_{reg}$ is obtained by syntactically limiting $\mathcal{FL}_{\perp reg}$ in another way: we forbid the use of $\perp$.

In order to obtain the logic $\mathcal{FL}_0$, one needs to apply both limitations: value restrictions are limited to $\forall r.C$, and $\perp$ cannot be used.

2.3 Subsumption problem

The notions of subsumption and a TBox are defined uniformly for all description logics introduced above. In each case, concept descriptions are formed according to the given logic, and interpretations assign meaning to the constructors as specified.

Definition 2.1. A *subsumption* between the concept descriptions C and D is written $C \sqsubseteq D$, and we also say that C is *subsumed* by D . Such a subsumption holds in an interpretation $\mathcal{I}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. When no interpretation is specified, $C \sqsubseteq D$ means that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$. The concept descriptions C and D are *equivalent* (written $C \equiv D$) if $C \sqsubseteq D$ and $D \sqsubseteq C$.

A *TBox* $\mathcal{T}$ is a finite set of axioms of the form $C \sqsubseteq D$, where C and D are concept descriptions. Such axioms express inclusion (implication) between concepts, stating that every instance of C is also an instance of D . For example, $\text{Human} \sqsubseteq \text{Animal}$ states that every human is an animal. Sometimes one also allows axioms of the form $A \equiv B$, expressing equivalence of two concepts. Such axioms can be split into two subsumptions, $A \sqsubseteq B$ and $B \sqsubseteq A$, hence we assume that they are not present.

Subsumption can be relativized to a TBox as follows.

Definition 2.2. Let $\mathcal{T}$ be a TBox and let C, D be concept descriptions.

- An interpretation $\mathcal{I}$ is a *model* of $\mathcal{T}$ if all axioms $(E \sqsubseteq F) \in \mathcal{T}$ hold in $\mathcal{I}$.
- C is *subsumed* by D with respect to $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if $C \sqsubseteq D$ holds in every model $\mathcal{I}$ of $\mathcal{T}$.

The *subsumption problem* in the considered logic with TBoxes is defined as follows:

- **Input:** a TBox $\mathcal{T}$ and concept descriptions C, D .
- **Output:** **Yes**, if the subsumption $C \sqsubseteq_{\mathcal{T}} D$ holds, and **No** otherwise.

An instance of this problem is denoted $C \sqsubseteq_{\mathcal{T}}^? D$, with the question mark indicating that the subsumption is the *goal* to be determined. We refer to such expressions as *goal subsumptions*.

If the TBox is absent (empty), we talk about the subsumption problem without TBoxes, with instances of the form $C \sqsubseteq^? D$.

Notice that a *set of goal subsumptions* is simply a set of independent subsumption problems, because each goal subsumption is considered independently of the others.

3 Syntactic laws and normal forms

Before addressing the subsumption problem, in this section we present some simple syntactic laws that can be used to transform concept descriptions or to reason about subsumptions in our logics.

We start by observing that the equivalence relation is a congruence: if a concept description C syntactically contains a concept description D (as a subterm), and $D \equiv D'$, then after replacing D by D' in the definition of C we obtain a concept description C' that is equivalent to C (i.e., $C \equiv C'$).

We now present several equivalences, which are immediate consequences of the semantics. First, we note that value restrictions distribute over conjunction:

$$\forall E.(C_1 \sqcap C_2) \equiv \forall E.C_1 \sqcap \forall E.C_2.$$

The following two equivalences, used exhaustively as rewrite rules from left to right, allow us to eliminate the $\top$ symbol (except in the case where the entire concept description is $\top$):

$$\forall E.\top \equiv \top, \quad C \sqcap \top \equiv C.$$

We can also merge nested value restrictions into a single one, or introduce a value restriction when it is not present:

$$\forall E_1.\forall E_2.C \equiv \forall E_1 E_2.C, \quad C \equiv \forall \varepsilon.C.$$

Value restrictions involving the same concept name, as well as those involving $\perp$, may be merged:

$$\forall E_1.C \sqcap \forall E_2.C \equiv \forall (E_1 + E_2).C.$$

The above transformations allow us to convert every concept description into an equivalent concept description in a normal form defined as follows. A concept description is in *normal form* if it is of the form either $\top$, or $\forall E_1.A_1 \sqcap \dots \sqcap \forall E_n.A_n$, or $\forall E_1.A_1 \sqcap \dots \sqcap \forall E_n.A_n \sqcap \forall E_{n+1}.\perp$, where all $A_1, \dots, A_n$ are pairwise different concept names. A concept description of the form $\forall E.C$, where C is either $\perp$ or a concept name from $\mathbf{N}_C$, is called a *particle*.

We also observe that for any concept name of interest, $A \in \mathbf{N}_C$ (as well as for $\perp$), every concept description D is equivalent to a concept description in normal form containing a particle that involves A (or $\perp$, respectively). Indeed, if such a particle is not present in D , we can introduce one with the empty regular expression thanks to the equivalence

$$\top \equiv \forall \emptyset.C.$$

The next equivalence is used in [Section 4](#):

$$C \sqcap \perp \equiv \perp. \tag{1}$$

We now turn to properties of subsumption. Clearly subsumption is transitive: for every interpretation $\mathcal{I}$, if $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_3$ hold in $\mathcal{I}$, then $C_1 \sqsubseteq C_3$ holds as well. Another property is monotonicity of value restrictions: if $C \sqsubseteq D$ holds in an interpretation $\mathcal{I}$, then $\forall E.C \sqsubseteq \forall E.D$ as well, for any regular role expression E . Moreover, the following subsumption holds in every interpretation $\mathcal{I}$:

$$C \sqcap D \sqsubseteq C. \tag{2}$$

Next, observe that for every interpretation $\mathcal{I}$ we have

$$C^{\mathcal{I}} \subseteq (D_1 \sqcap D_2)^{\mathcal{I}} \iff C^{\mathcal{I}} \subseteq D_1^{\mathcal{I}} \wedge C^{\mathcal{I}} \subseteq D_2^{\mathcal{I}}. \tag{3}$$

Finally, for every interpretation $\mathcal{I}$ and every element x of its domain we have the following equivalence (where a word $w \in \mathbf{N}_R^*$ is treated as a regular role expression):

$$x \in (\forall E.C)^{\mathcal{I}} \iff (x \in (\forall w.C)^{\mathcal{I}} \text{ for all } w \in \mathcal{L}(E)). \tag{4}$$

4 Subsumption in $\mathcal{FL}_{reg}$ and $\mathcal{FL}_{\perp reg}$ without TBoxes

In this section we solve the subsumption problem in $\mathcal{FL}_{reg}$ and in $\mathcal{FL}_{\perp reg}$ without TBoxes.

Theorem 4.1. *The following two problems are PSPACE-complete:*

- the subsumption problem in $\mathcal{FL}_{reg}$ without TBoxes, and

- the subsumption problem in $\mathcal{FL}_{\perp reg}$ without TBoxes.

Consider an instance $C \sqsubseteq^? D$ of the subsumption problem. As already mentioned, we may assume that C and D are in normal form; in particular these are conjunctions of particles.

We first note that a subsumption $C \sqsubseteq D_1 \sqcap \dots \sqcap D_k$ holds if and only if the subsumptions $C \sqsubseteq D_i$ hold for all $i \in \{1, \dots, k\}$ (as follows from [Formula \(3\)](#)). By considering each of the latter subsumptions separately, we may assume that the right side of our goal subsumption consists of a single particle.

After the above step, the considered instance of the subsumption problem in $\mathcal{FL}_{\perp reg}$ is of the form either $C \sqsubseteq^? \forall G.A$, where A is a concept name, or $C \sqsubseteq^? \forall G.\perp$; the left side C is a conjunction of particles involving different concept names. Moreover, as explained in [Section 3](#), we may assume that C contains a particle involving $\perp$, as well as a particle involving A (in the case when the right side is $\forall G.A$ for $A \in \mathbf{N}_C$). In other words, the instance of the problem, after preparatory syntactic transformations, is of the form either

- $\forall E.A \sqcap \forall F.\perp \sqcap C_0 \sqsubseteq^? \forall G.A$, where C_0 does not use $\perp$ nor the concept name A , or
- $\forall F.\perp \sqcap C_0 \sqsubseteq^? \forall G.\perp$, where C_0 does not use $\perp$.

As a next simplification, we observe that the part C_0 can be removed from these subsumptions, as stated in the following lemma, whose proof can be found in the extended version of this paper [\[15\]](#).

Lemma 4.2. *For subsumptions as above we have equivalences:*

- $\forall E.A \sqcap \forall F.\perp \sqcap C_0 \sqsubseteq \forall G.A$ holds if and only if $\forall E.A \sqcap \forall F.\perp \sqsubseteq \forall G.A$ holds;
- $\forall F.\perp \sqcap C_0 \sqsubseteq \forall G.\perp$ holds if and only if $\forall F.\perp \sqsubseteq \forall G.\perp$ holds.

Let $\Sigma_R \subseteq \mathbf{N}_R$ contain all role names occurring in the considered concept descriptions. It is a finite set, hence disjunction of its elements, written simply Σ_R , can be used as a regular role expression.¹ The next lemma (with a proof in the extended version of this paper [\[15\]](#)) reduces the subsumption problem to the containment problem for languages of regular expressions.

Lemma 4.3. *We have the following equivalences:*

- the subsumption $\forall E.A \sqcap \forall F.\perp \sqsubseteq \forall G.A$ holds if and only if $\mathcal{L}(G) \subseteq \mathcal{L}(E + F\Sigma_R^*)$;
- the subsumption $\forall F.\perp \sqsubseteq \forall G.\perp$ holds if and only if $\mathcal{L}(G) \subseteq \mathcal{L}(F\Sigma_R^*)$.

We have thus reduced the subsumption problem in $\mathcal{FL}_{\perp reg}$ without TBoxes (and hence also in $\mathcal{FL}_{reg}$ without TBoxes, which is a special case) to the problem of deciding inclusion between two languages given by regular expressions. It is known that the latter problem is PSPACE-complete [\[17\]](#). Thus, our subsumption problems are in PSPACE.

In order to show PSPACE-hardness, we reduce in the opposite direction. Given two arbitrary regular expressions G, E for which we want to check whether language inclusion holds, we treat their letters as elements of $\mathbf{N}_R$. We fix some concept name A , and we define Σ_R to contain all letters from E and G . Taking $F = \emptyset$ we have $\mathcal{L}(E) = \mathcal{L}(E + F\Sigma_R^*)$ as well as $\forall E.A \equiv \forall E.A \sqcap \forall F.\perp$. From [Lemma 4.3](#) it thus follows that the inclusion $\mathcal{L}(G) \subseteq \mathcal{L}(E)$ (equivalently: $\mathcal{L}(G) \subseteq \mathcal{L}(E + F\Sigma_R^*)$) is equivalent to the subsumption $\forall E.A \sqsubseteq \forall G.A$ (equivalently: $\forall E.A \sqcap \forall F.\perp \sqsubseteq \forall G.A$). Note that the resulting subsumption $\forall E.A \sqsubseteq \forall G.A$ does not use $\perp$, hence it belongs to both logics $\mathcal{FL}_{reg}$ and $\mathcal{FL}_{\perp reg}$. This concludes our proof of [Theorem 4.1](#).

We note that the correspondence between problems concerning $\mathcal{FL}_{reg}$ and language inclusion is well known and has been used, for example, in the context of the matching problem [\[4, Lemma 6\]](#) and the unification problem [\[10, Lemma 3\]](#); some of the arguments from the cited papers could be adapted to the present setting and reused.

¹The only reason for using Σ_R instead of $\mathbf{N}_R$ is that $\mathbf{N}_R$ is infinite, hence it cannot be legally used in a regular expression.

5 Reduction from $\mathcal{FL}_{\perp reg}$ with TBoxes to $\mathcal{FL}_{reg}$ with TBoxes

In this section, we present a reduction from the subsumption problem in $\mathcal{FL}_{\perp reg}$ with TBoxes to the subsumption problem in $\mathcal{FL}_{reg}$ with TBoxes. Our construction is inspired by a reduction in Kost and Morawska [18]; however, their result is obtained in a different setting, namely unification, and considers a transformation from $\mathcal{FL}_{\perp}$ *without TBoxes* to $\mathcal{FL}_0$ with TBoxes. Thus, in contrast to their work, we deal with subsumption (rather than unification), include TBoxes in the source problem, and target $\mathcal{FL}_{reg}$ instead of $\mathcal{FL}_0$.

A further related approach is presented in Baader et al. [8], where a different reduction is used to transform a subsumption problem in $\mathcal{FL}_{\perp}$ with TBoxes into one in $\mathcal{FL}_0$ with TBoxes (their approach requires so-called flattening of the TBox, which does not seem to work in the presence of regular expressions). It follows that the subsumption problem in $\mathcal{FL}_{\perp}$ with TBoxes belongs to EXPTIME. Our result in this paper is more general, since it concerns the more expressive logics $\mathcal{FL}_{\perp reg}$ and $\mathcal{FL}_{reg}$.

The idea of the reduction presented below (as well as of the earlier reductions) is to simulate the constructor $\perp$ by a fresh concept name B , axiomatized in a TBox.

Throughout this section we consider an instance $C \sqsubseteq_{\mathcal{T}}^? D$ of the subsumption problem in $\mathcal{FL}_{\perp reg}$ with TBoxes. All C , D , and $\mathcal{T}$ may contain occurrences of $\perp$. Let $\Sigma_C \subseteq \mathbf{N}_C$ and $\Sigma_R \subseteq \mathbf{N}_R$ be the sets of, respectively, all concept names and all role names occurring in C , D , or in the TBox $\mathcal{T}$. We fix a fresh concept name B , which does not occur anywhere in C , D , or $\mathcal{T}$ (i.e., $B \notin \Sigma_C$). Then, we define:

- K^B is K with all occurrences of $\perp$ replaced by B , for any concept description K ;
- $\mathcal{T}^B$ is the TBox $\mathcal{T}$ with all occurrences of $\perp$ replaced by B .

Next, we define a TBox $\mathcal{R}$, which simulates properties of $\perp$. It contains the following axioms:

- $B \sqsubseteq A$, for every concept name $A \in \Sigma_C$;
- $B \sqsubseteq \forall r.B$, for every role name $r \in \Sigma_R$.

Note that C^B , D^B , and concept descriptions in $\mathcal{T}^B \cup \mathcal{R}$ do not use the $\perp$ symbol, hence belong to the $\mathcal{FL}_{reg}$ logic. Thus, the following theorem gives us a reduction for the subsumption problem from $\mathcal{FL}_{\perp reg}$ with TBoxes to $\mathcal{FL}_{reg}$ with TBoxes.

Theorem 5.1. *The subsumption $C \sqsubseteq_{\mathcal{T}} D$ from $\mathcal{FL}_{\perp reg}$ holds if and only if the subsumption $C^B \sqsubseteq_{\mathcal{T}^B \cup \mathcal{R}} D^B$ from $\mathcal{FL}_{reg}$ holds.*

We now sketch main ideas of a proof of [Theorem 5.1](#); missing details are contained in the extended version of this paper [15]. Throughout this proof, letter $\mathcal{I}$ denotes interpretations used for original concept descriptions in $\mathcal{FL}_{\perp reg}$, and letter $\mathcal{J}$ interpretations used for concept descriptions in $\mathcal{FL}_{reg}$, obtained from those in $\mathcal{FL}_{\perp reg}$ by replacing $\perp$ with the concept name B .

For the right-to-left implication, consider a model $\mathcal{I}$ of $\mathcal{T}$ in which we want to check whether $C \sqsubseteq D$ holds. As $\mathcal{J}$ we take a modification of $\mathcal{I}$ in which B is interpreted as $\emptyset$, so in the same way as $\perp$. It is then easy to see that $(K^B)^{\mathcal{J}} = K^{\mathcal{I}}$ for every concept description K (possibly using $\perp$, but not the concept name B), as described by the following lemma.

Lemma 5.2. *Let $\mathcal{I}$ and $\mathcal{J}$ be interpretations such that*

- $\mathcal{U}^{\mathcal{J}} = \mathcal{U}^{\mathcal{I}}$,
- $A^{\mathcal{J}} = A^{\mathcal{I}}$ for each $A \in \mathbf{N}_C \setminus \{B\}$,
- $r^{\mathcal{J}} = r^{\mathcal{I}}$ for each $r \in \mathbf{N}_R$, and
- $B^{\mathcal{J}} = \emptyset$.

Then, for every concept description K in $\mathcal{FL}_{\perp reg}$ not using the concept name B we have $(K^B)^{\mathcal{J}} = K^{\mathcal{I}}$.

It follows that $\mathcal{J}$ is a model of $\mathcal{T}^B \cup \mathcal{R}$, so the subsumption $C^B \sqsubseteq D^B$ holds there by assumption, which implies that $C \sqsubseteq D$ holds in $\mathcal{I}$.

The left-to-right implication is a bit more involved. This time we start with a model $\mathcal{J}$ of $\mathcal{T}^B \cup \mathcal{R}$ in which we want to check whether $C^B \sqsubseteq D^B$ holds. Note that $\mathcal{J}$ may interpret B as a nonempty set. First, we observe that the axioms from $\mathcal{R}$ imply the following lemma.

Lemma 5.3. *For every $\mathcal{FL}_{reg}$ concept description K involving only concept names from $\Sigma_C \cup \{B\}$ and role names from Σ_R , the subsumption $B \sqsubseteq K$ holds in every model $\mathcal{J}$ of $\mathcal{R}$.*

If $B^{\mathcal{J}} = \mathcal{U}^{\mathcal{J}}$, then the subsumption $C^B \sqsubseteq D^B$ holds in $\mathcal{J}$ trivially, since Lemma 5.3 yields $B^{\mathcal{J}} \subseteq (D^B)^{\mathcal{J}}$, and thus $(D^B)^{\mathcal{J}} = \mathcal{U}^{\mathcal{J}}$. Otherwise, we define $\mathcal{I}$ by removing $B^{\mathcal{J}}$ from the universe of $\mathcal{J}$, hence also from the interpretation of every concept name and role name. One then has to check that $K^{\mathcal{I}} = (K^B)^{\mathcal{J}} \setminus B^{\mathcal{J}}$ for every concept description K using only concept names from Σ_C and role names from Σ_R . This is described by the following lemma, whose proof is by induction on the structure of K , and uses Lemma 5.3 while handling the case of value restrictions.

Lemma 5.4. *Let $\mathcal{J}$ be a model of $\mathcal{R}$ such that $B^{\mathcal{J}} \neq \mathcal{U}^{\mathcal{J}}$, and let $\mathcal{I}$ be an interpretation such that*

- $\mathcal{U}^{\mathcal{I}} = \mathcal{U}^{\mathcal{J}} \setminus B^{\mathcal{J}}$,
- $A^{\mathcal{I}} = A^{\mathcal{J}} \setminus B^{\mathcal{J}}$ for each $A \in \Sigma_C$, and
- $r^{\mathcal{I}} = \{(x, y) \in r^{\mathcal{J}} \mid x \notin B^{\mathcal{J}} \wedge y \notin B^{\mathcal{J}}\}$ for each $r \in \Sigma_R$.

Then, for every concept description K in $\mathcal{FL}_{\perp reg}$ involving only concept names from Σ_C and role names from Σ_R , we have

$$K^{\mathcal{I}} = (K^B)^{\mathcal{J}} \setminus B^{\mathcal{J}}.$$

Using Lemma 5.4 one can deduce that $\mathcal{I}$ is a model of $\mathcal{T}$. Since, by assumption, the subsumption $C \sqsubseteq_{\mathcal{T}} D$ holds, we obtain in particular $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Using Lemma 5.4 again, for $K = C$ and for $K = D$, one can then deduce that $(C^B)^{\mathcal{J}} \setminus B^{\mathcal{J}} \subseteq (D^B)^{\mathcal{J}} \setminus B^{\mathcal{J}}$. Taking the union with the inclusion $B^{\mathcal{J}} \subseteq (D^B)^{\mathcal{J}}$ resulting from Lemma 5.3, this yields $(C^B)^{\mathcal{J}} \subseteq (D^B)^{\mathcal{J}}$. Hence $C^B \sqsubseteq D^B$ holds in $\mathcal{J}$.

We have shown that the subsumption problem in $\mathcal{FL}_{\perp reg}$ with TBoxes reduces to that in $\mathcal{FL}_{reg}$ with TBoxes. In the next part, we show how to decide the latter by reduction to the problem of solving parity pushdown games.

6 Parity pushdown games

A parity pushdown game is an infinite-duration two-player game played on the configuration graph of a pushdown system. Positions of the game consist of a control state together with a stack content, and are partitioned between the two players, called Adam and Eve. Starting from a chosen configuration, the players move a token along transitions induced by the pushdown system, thereby generating an infinite play. The winner is determined by a parity condition on the sequence of control states visited during the play.

Formally, a *parity pushdown game* is a tuple $\mathcal{G} = (P_A, P_E, \Gamma, \Delta, \Omega)$, where

- $P_A \uplus P_E$, which we denote P , is a finite set of control states, partitioned into states of Adam and Eve;
- Γ is a finite stack alphabet;
- $\Delta \subseteq (P \times \Gamma^*) \times (P \times \Gamma^*)$ is a finite set of transition rules;
- $\Omega: P \rightarrow \mathbb{N}$ is a priority function.

A *configuration* is a pair $(p, w) \in P \times \Gamma^*$. The *transition relation*, denoted $\rightarrow$, is a relation between configurations defined as follows: for every $(p, \alpha, q, \beta) \in \Delta$ and $w \in \Gamma^*$ there is a transition

$$(p, w\alpha) \rightarrow (q, w\beta).$$

Observe that the stack grows to the right (i.e., the top of the stack is the rightmost symbol of w). Note also that transitions in Δ may remove (pop) an arbitrary number of stack symbols in a single step. This more general model can be effectively reduced to the standard one in which each transition removes precisely one stack symbol.

A *path* is a sequence of configurations, $(p_0, w_0)(p_1, w_1)(p_2, w_2) \dots$, either finite or infinite, such that each consecutive pair follows the transition relation. A path is called a *play* if it is maximal: either infinite, or ending in a configuration (p_n, w_n) from which there is no transition. In a finite play, the winner is given by the state of the last configuration: the player owning this state loses, as no move is available. In an infinite play, the winner is determined by the parity condition applied to the sequence of control states $(p_i)_{i \geq 0}$: Eve wins if the greatest priority occurring infinitely often in this sequence is even; otherwise, Adam wins.

A (positional) *strategy* for Eve is a function defined on configurations $(p, w) \in P_E \times \Gamma^*$ with at least one available move, assigning to each such configuration a valid successor. Strategies for Adam are defined analogously. A play $(p_0, w_0)(p_1, w_1)(p_2, w_2) \dots$ (finite or infinite) is consistent with a strategy σ for Eve if, for every i such that $p_i \in P_E$ and (p_i, w_i) has at least one available move, we have

$$(p_{i+1}, w_{i+1}) = \sigma(p_i, w_i).$$

We say that Eve *wins* the game $\mathcal{G}$ from a *configuration* c if she has a strategy σ such that every play starting from c and consistent with σ is winning for her (and analogously for Adam). It is well known that parity games are positionally determined, meaning that from every configuration exactly one of the players has a winning positional strategy [12].

In the next section we use the following result regarding parity pushdown games.

Theorem 6.1 ([22]). *Given a parity pushdown game $\mathcal{G}$ together with an initial configuration c_0 , one can decide in EXPTIME whether Eve wins $\mathcal{G}$ from c_0 .*

7 Reduction of subsumption problems to parity pushdown games

In this section, we present a reduction of the subsumption problem in $\mathcal{FL}_{reg}$ with TBoxes to the problem of finding a winner in parity pushdown games.

We shall use nondeterministic finite automata to represent regular expressions occurring in the TBox axioms. A *nondeterministic finite automaton* (NFA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite input alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states. Instead of $(p, a, q) \in \delta$, where $a \in \Sigma$, we write $p \xrightarrow{a} q$. A run of $\mathcal{A}$ on a word $w \in \Sigma^*$ is a sequence of transitions from q_0 reading the symbols of w . The automaton $\mathcal{A}$ accepts w if there exists a run on w ending in a state from F . The language recognized by $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of all words accepted by $\mathcal{A}$.

Let $C \sqsubseteq_{\mathcal{T}}^? D$ be an instance of the subsumption problem in $\mathcal{FL}_{reg}$ with TBoxes. First, we introduce two fresh concept names L and R , and define

$$\mathcal{T}_{C,D} := \{L \sqsubseteq C, C \sqsubseteq L, R \sqsubseteq D, D \sqsubseteq R\},$$

making sure that L and R define the same concepts as C and D , respectively. It is easy to see that then²

$$C \sqsubseteq_{\mathcal{T}} D \quad \text{if and only if} \quad L \sqsubseteq_{\mathcal{T} \cup \mathcal{T}_{C,D}} R.$$

Hence, from now on, it suffices to consider a subsumption problem of the form $L \sqsubseteq_{\mathcal{T}} R$ with L, R being concept names, where, by abuse of notation, $\mathcal{T}$ already includes the additional axioms from $\mathcal{T}_{C,D}$.

Moreover, we assume that every concept occurring in the TBox $\mathcal{T}$ is in normal form, that is, it is a conjunction of particles of the form $\forall E.A$, where E is a regular expression over the alphabet $\mathbf{N}_R$, and $A \in \mathbf{N}_C$ is a concept name.

Going further, we transform every axiom of the form $\forall G_1.A_1 \sqcap \dots \sqcap \forall G_n.A_n \sqsubseteq \forall H_1.B_1 \sqcap \dots \sqcap \forall H_k.B_k$ from the TBox $\mathcal{T}$ into an equivalent set of axioms with a single particle on the right side:

$$\forall G_1.A_1 \sqcap \dots \sqcap \forall G_n.A_n \sqsubseteq \forall H_j.B_j \quad \text{for } j \in \{1, \dots, k\}.$$

Let Σ_C and Σ_R be the finite sets of, respectively, all concept names and all role names occurring in $\mathcal{T}$; in particular Σ_C contains the two special concept names L and R . Based on the TBox $\mathcal{T}$ we construct a pushdown game $\mathcal{G}_{\mathcal{T}} = (P_A, P_E, \Gamma, \Delta, \Omega)$. As the stack alphabet, we take $\Gamma = \Sigma_R \cup \{\#\}$, where $\#$ is a special symbol, which will mark the bottom of the stack. For each concept name $A \in \mathbf{N}_C$, we introduce a state owned by Eve, $p_A \in P_E$. We designate $c_0 = (p_R, \#)$ as the initial configuration of the game.

For every TBox axiom τ , we proceed as follows. The axiom τ is of the form $\forall G_1.A_1 \sqcap \dots \sqcap \forall G_n.A_n \sqsubseteq \forall H.B$. We construct a nondeterministic finite automaton $\mathcal{A}_H = (Q_H, \Sigma_R, \delta_H, q_{H,0}, F_H)$ such that $\mathcal{L}(\mathcal{A}_H) = \mathcal{L}(H)$. For each state $q \in Q_H$, we introduce a game state owned by Eve, $q \in P_E$. We then add transitions that allow to enter the automaton in any accepting state:

$$(p_B, \varepsilon) \rightarrow (q, \varepsilon) \quad \text{if } q \in F_H.$$

For every transition $p \xrightarrow{r} q$ of the automaton $\mathcal{A}_H$, we add the reverse of this transition to the game:

$$(q, r) \rightarrow (p, \varepsilon).$$

We also introduce a new state $p_{\tau} \in P_A$, and add a transition that allows to reach it from the initial state:

$$(q_{H,0}, \varepsilon) \rightarrow (p_{\tau}, \varepsilon).$$

This way, Eve can go from p_B to p_{τ} while removing from the stack any word that matches H (the word is read left-to-right, that is, bottom-to-top).

Next, for every particle $\forall G_i.A_i$ on the left side of τ , we allow Adam to go from p_{τ} to p_{A_i} while inserting on the stack any word that matches G_i . To this end, for every $i \in \{1, \dots, n\}$ we construct a nondeterministic finite automaton $\mathcal{A}_{G_i} = (Q_{G_i}, \Sigma_R, \delta_{G_i}, q_{G_i,0}, F_{G_i})$ such that $\mathcal{L}(\mathcal{A}_{G_i}) = \mathcal{L}(G_i)$, and we continue as follows. First, for each state $q \in Q_{G_i}$, we introduce a game state owned by Adam, $q \in P_A$. Then, we add a transition that allows Adam to enter the initial state of $\mathcal{A}_{G_i}$:

$$(p_{\tau}, \varepsilon) \rightarrow (q_{G_i,0}, \varepsilon).$$

Next, for every transition $p \xrightarrow{r} q$ of $\mathcal{A}_{G_i}$, we add an analogous transition to the game:

$$(p, \varepsilon) \rightarrow (q, r).$$

²A careful reader may notice that the middle two subsumptions are not needed; it is enough to take $L \sqsubseteq C$ and $D \sqsubseteq R$.

Finally, for each accepting state $q \in F_{G_j}$ we add the transition

$$(q, \varepsilon) \rightarrow (p_{A_j}, \varepsilon).$$

Of course we assume that state sets of all the constructed automata are disjoint, so that the automata do not interfere with each other, after adding them to the game.

Success is reached when the current obligation becomes L and no role symbol remains on the stack, that is, at the configuration $(p_L, \#)$. To realize this, we introduce an artificial winning state $p_{win} \in P_A$, and we add the transition

$$(p_L, \#) \rightarrow (p_{win}, \#).$$

It is a state of Adam without any successors, so reaching it causes that Eve wins.

The priority function Ω is defined as follows:

$$\Omega(p) = \begin{cases} 0, & \text{if } p \in P_A, \\ 1, & \text{if } p \in P_E. \end{cases}$$

Such a definition causes that if Adam stays in his states (i.e., in some automaton $\mathcal{A}_{G_i}$) forever, then Eve wins; otherwise, Eve has to reach p_{win} in a finite time, as otherwise infinitely many states of priority 1 would be produced.

Theorem 7.1. *Let $L \sqsubseteq_{\mathcal{T}}^? R$ be an instance of the subsumption problem in $\mathcal{FL}_{reg}$ with TBoxes (after the preliminary transformation described above), and let $\mathcal{G}_{\mathcal{T}}$ be the parity pushdown game associated with this instance by the above construction. Then Eve has a winning strategy from the configuration $c_0 = (p_R, \#)$ if and only if $L \sqsubseteq_{\mathcal{T}} R$.*

We prove separately two implications.

For the left-to-right implication assume that σ is a winning strategy for Eve from $(p_R, \#)$, and suppose towards a contradiction that $L \not\sqsubseteq_{\mathcal{T}} R$. Then there exists a model $\mathcal{I}$ of $\mathcal{T}$ and an element $x \in L^{\mathcal{I}}$ such that $x \notin R^{\mathcal{I}}$. Fix one such element.

We construct a play starting from $(p_R, \#)$ and consistent with σ , but won by Adam. This yields a contradiction. We construct the play so that the following invariant is preserved: whenever the current configuration is of the form $(p_B, \#u)$, we have $x \notin (\forall u.B)^{\mathcal{I}}$.

Recall that $x \notin R^{\mathcal{I}} = (\forall \varepsilon.R)^{\mathcal{I}}$, meaning that the initial configuration $(p_R, \#)$ satisfies the invariant.

Assume therefore that the game is currently in a configuration $(p_B, \#u)$ satisfying the invariant. First, if no move is available to Eve from that configuration, then she loses and we are done. Second, observe that $(p_B, \#u) \neq (p_L, \#)$. Indeed, for $B = L$ and $u = \varepsilon$ the invariant would say that $x \notin (\forall \varepsilon.L)^{\mathcal{I}} = L^{\mathcal{I}}$, contradicting the assumption that $x \in L^{\mathcal{I}}$. Thus, the transition to the state p_{win} is not available.

It follows that from the configuration $(p_B, \#u)$, Eve chooses some axiom

$$\tau = (\forall G_1.A_1 \sqcap \dots \sqcap \forall G_n.A_n \sqsubseteq \forall H.B) \in \mathcal{T}$$

and moves to the part of the game corresponding to this axiom. Every transition in this part removes a symbol from the stack, so the play cannot stay in this part for an infinite time. Moreover, all states in this part belong to Eve, so if the play gets stuck there, then Adam wins, as we wanted.

The remaining case is that the play reaches the state p_{τ} after some time. By construction of the game, this could happen only if some word $v \in \mathcal{L}(\mathcal{A}_H) = \mathcal{L}(H)$ was removed from the

stack. Consequently, the original word u was necessarily of the form $u = u'v$ for some $u' \in \Sigma_R^*$, and after this phase of the play the configuration reached is $(p_\tau, \sharp u')$. Recall that τ , as an axiom from $\mathcal{T}$, holds in $\mathcal{I}$. Since $v \in \mathcal{L}(H)$, all elements of $(\forall H.B)^\mathcal{I}$ are elements of $(\forall v.B)^\mathcal{I}$ (as follows from [Formula \(4\)](#)), so the subsumption $\forall G_1.A_1 \sqcap \dots \sqcap \forall G_n.A_n \sqsubseteq \forall v.B$ holds in $\mathcal{I}$ as well. By monotonicity of value restriction, also the subsumption $\forall u'.(\forall G_1.A_1 \sqcap \dots \sqcap \forall G_n.A_n) \sqsubseteq \forall u'.\forall v.B$, which can be syntactically rewritten as $\forall u'.G_1.A_1 \sqcap \dots \sqcap \forall u'.G_n.A_n \sqsubseteq \forall u'.B$, holds in $\mathcal{I}$. Recalling the invariant $x \notin (\forall u.B)^\mathcal{I}$, this subsumption implies that $x \notin (\forall u'.G_1.A_1 \sqcap \dots \sqcap \forall u'.G_n.A_n)^\mathcal{I}$, hence $x \notin (\forall u'.G_j.A_j)^\mathcal{I}$ for some $j \in \{1, \dots, n\}$. Using [Formula \(4\)](#) we observe that then $x \notin (\forall u'w.A_j)^\mathcal{I}$ for some word $w \in \mathcal{L}(G_j)$ (where we note that all words in $\mathcal{L}(u'G_j)$ are of the form $u'w$ for some $w \in \mathcal{L}(G_j)$).

Adam now chooses the transition from p_τ leading to the initial state $q_{G_j,0}$ of the automaton $\mathcal{A}_{G_j}$, and then follows an accepting run of $\mathcal{A}_{G_j}$ on the word w . Since all states in this part of the game belong to Adam, he can enforce this finite fragment of the play. By construction of the game, the reached configuration is $(p_{A_j}, \sharp u'w)$, and the invariant holds again: $x \notin (\forall u'w.A_j)^\mathcal{I}$. From this point on, we continue the construction of the play in the same way, preserving the invariant.

Note that if we continue this construction forever, we obtain a play that visits infinitely many configurations of the form $(p_B, \sharp u)$, having priority 1; such a play is winning for Adam. Concluding, assuming $L \not\sqsubseteq_{\mathcal{T}} R$ we were able to construct a play won by Adam and consistent with the winning strategy of Eve, leading to a contradiction. Therefore $L \sqsubseteq_{\mathcal{T}} R$.

We now come to the right-to-left implication from the theorem: assuming $L \sqsubseteq_{\mathcal{T}} R$, we show that Eve wins from the configuration $(p_R, \sharp)$.

We define an interpretation $\mathcal{I}$ as follows: the universe is $\mathcal{U}^\mathcal{I} = \Sigma_R^*$, for each role $r \in \Sigma_R$ we take $r^\mathcal{I} = \{(u, ur) \mid u \in \Sigma_R^*\}$, and for each concept name $A \in \Sigma_C$ we take

$$A^\mathcal{I} = \{u \in \Sigma_R^* \mid \text{Eve wins from } (p_A, \sharp u)\}.$$

Interpretation of role names from $\mathbf{N}_R \setminus \Sigma_R$ and concept names from $\mathbf{N}_C \setminus \Sigma_C$ is irrelevant.

We show that $\mathcal{I}$ is a model of $\mathcal{T}$. Take an arbitrary axiom

$$\tau = (\forall G_1.A_1 \sqcap \dots \sqcap \forall G_n.A_n \sqsubseteq \forall H.B) \in \mathcal{T},$$

and an arbitrary word $u \in \Sigma_R^*$ such that $u \in (\forall G_1.A_1 \sqcap \dots \sqcap \forall G_n.A_n)^\mathcal{I}$. This means that for every $j \in \{1, \dots, n\}$ and every $w \in \mathcal{L}(G_j)$ we have $uw \in A_j^\mathcal{I}$, that is, Eve wins from configuration $(p_{A_j}, \sharp uw)$. We must show that $u \in (\forall H.B)^\mathcal{I}$, that is, that for every $v \in \mathcal{L}(H)$, Eve wins from the configuration $(p_B, \sharp uv)$.

Fix an arbitrary $v \in \mathcal{L}(H)$, and consider the configuration $(p_B, \sharp uv)$. Since parity games are positionally determined, it is enough to show that Adam does not win from $(p_B, \sharp uv)$. Suppose towards a contradiction that Adam wins from $(p_B, \sharp uv)$, and fix some his winning strategy σ_A . We construct a play consistent with σ_A and won by Eve, contradicting the fact that σ_A is winning for Adam.

First, Eve chooses the axiom τ , enters the part of the game corresponding to $\mathcal{A}_H$, and follows backwards an accepting run of $\mathcal{A}_H$ on the word v . By construction, this removes exactly the suffix v , and the play reaches $(p_\tau, \sharp u)$. At this point Adam chooses some index $j \in \{1, \dots, n\}$, and the play enters the part corresponding to $\mathcal{A}_{G_j}$. There are now three possibilities.

If Adam stays forever in states belonging to P_A , then the resulting infinite play is winning for Eve, since only priority 0 occurs infinitely often.

If Adam gets stuck in a state belonging to P_A , then the resulting finite play is winning for Eve, because a finite play ending in a position of Adam is losing for Adam.

The remaining possibility is that Adam follows an accepting run of $\mathcal{A}_{G_j}$ on some word $w \in \mathcal{L}(\mathcal{A}_{G_j}) = \mathcal{L}(G_j)$. In this case the play reaches the configuration $(p_{A_j}, \#uw)$. As explained above, our assumption on u implies that $uw \in A_j^{\mathcal{I}}$, meaning that Eve wins from $(p_{A_j}, \#uw)$. Let σ_E be her winning strategy from this configuration. We define the remaining part of the play following both σ_A and σ_E . Because σ_E is winning for Eve, the suffix of the play starting from $(p_{A_j}, \#uw)$ is won by Eve, hence the whole play as well.

We have thus shown that Adam cannot win from $(p_B, \#uv)$; Eve wins there. Since $v \in \mathcal{L}(H)$ was arbitrary, we conclude that $u \in (\forall H.B)^{\mathcal{I}}$. Thus τ holds in $\mathcal{I}$. Since τ was an arbitrary axiom of $\mathcal{T}$, we obtain that $\mathcal{I}$ is a model of $\mathcal{T}$.

Finally, Eve trivially wins from $(p_L, \#)$, because there is a transition $(p_L, \#) \rightarrow (p_{win}, \#)$, and p_{win} is a dead-end state owned by Adam. Hence $\varepsilon \in L^{\mathcal{I}}$. By the assumed subsumption $L \sqsubseteq_{\mathcal{T}} R$, this implies that $\varepsilon \in R^{\mathcal{I}}$. Therefore, by the definition of $R^{\mathcal{I}}$, Eve wins from $(p_R, \#)$.

Theorem 7.2. *The subsumption problem in each of the logics $\mathcal{FL}_0$, $\mathcal{FL}_{\perp}$, $\mathcal{FL}_{reg}$, and $\mathcal{FL}_{\perp reg}$, with TBoxes, is EXPTIME-complete.*

Proof. For the lower bound, recall that the subsumption problem in $\mathcal{FL}_0$ with TBoxes is EXPTIME-hard [16]. Since $\mathcal{FL}_0$ is a fragment of each of the logics $\mathcal{FL}_{\perp}$, $\mathcal{FL}_{reg}$, and $\mathcal{FL}_{\perp reg}$, it follows immediately that the subsumption problem with TBoxes is EXPTIME-hard for all of them.

It remains to prove membership in EXPTIME.

For $\mathcal{FL}_{reg}$, this follows from the previous theorem: every instance of the subsumption problem in $\mathcal{FL}_{reg}$ with TBoxes can be reduced in polynomial time to a parity pushdown game, and parity pushdown games are solvable in EXPTIME, as stated in Theorem 6.1.

For $\mathcal{FL}_{\perp reg}$, we use Theorem 5.1, which yields a polynomial reduction from an instance of the subsumption problem in $\mathcal{FL}_{\perp reg}$ with TBoxes to an equivalent instance of the subsumption problem in $\mathcal{FL}_{reg}$ with TBoxes. Hence the subsumption problem in $\mathcal{FL}_{\perp reg}$ with TBoxes also belongs to EXPTIME.

Finally, $\mathcal{FL}_0$ and $\mathcal{FL}_{\perp}$ are fragments of $\mathcal{FL}_{\perp reg}$. Therefore subsumption in $\mathcal{FL}_0$ with TBoxes and in $\mathcal{FL}_{\perp}$ with TBoxes are special cases of subsumption in $\mathcal{FL}_{\perp reg}$ with TBoxes, and thus also belongs to EXPTIME (this also follows from earlier work [5, 16, 8]).

Consequently, in each of the logics $\mathcal{FL}_0$, $\mathcal{FL}_{\perp}$, $\mathcal{FL}_{reg}$, and $\mathcal{FL}_{\perp reg}$, the subsumption problem with TBoxes is EXPTIME-complete. $\square$

Example 7.3. We illustrate the construction of the parity pushdown game associated with the subsumption problem

$$\forall rrr^*.A \sqcap B \sqsubseteq \forall rr^*.A$$

with the TBox containing the axiom $B \sqsubseteq \forall r.A$. Since this instance already belongs to $\mathcal{FL}_{reg}$, no preliminary reduction from $\mathcal{FL}_{\perp reg}$ is needed.

Following the general construction, we introduce two fresh concept names L and R , and replace the original problem by the equivalent one $L \sqsubseteq_{\mathcal{T}} R$, where $\mathcal{T}$ consists of the following axioms, displayed here already after switching to normal forms, and splitting subsumptions with multiple particles on the right side:

$$\begin{aligned} \forall \varepsilon.B \sqsubseteq \forall r.A, \quad \forall \varepsilon.L \sqsubseteq \forall rrr^*.A, \quad \forall \varepsilon.L \sqsubseteq \forall \varepsilon.B, \\ \forall rrr^*.A \sqcap \forall \varepsilon.B \sqsubseteq \forall \varepsilon.L, \quad \forall \varepsilon.R \sqsubseteq \forall rr^*.A, \quad \forall rr^*.A \sqsubseteq \forall \varepsilon.R. \end{aligned}$$

We now describe the corresponding parity pushdown game. The concept-name states are p_L, p_R, p_A, p_B , all owned by Eve, and the initial configuration is $(p_R, \#)$.

The nondeterministic finite automata used in the construction are the following:

- an automaton $\mathcal{A}_{rr^*}$ recognizing $\mathcal{L}(rr^*) = \{r^n \mid n \geq 1\}$,
- an automaton $\mathcal{A}_r$ recognizing $\mathcal{L}(r) = \{r\}$,
- an automaton $\mathcal{A}_{rrr^*}$ recognizing $\mathcal{L}(rrr^*) = \{r^n \mid n \geq 2\}$,
- and the trivial automaton for ε , used for particles of the form $\forall \varepsilon.L$.

The game starts in the configuration $(p_R, \#)$. Eve first chooses the axiom

$$\tau_1 = (\forall rr^*.A \sqsubseteq R).$$

Since the right side is just $\forall \varepsilon.R$, this corresponds to the regular expression ε , so no symbol is removed from the stack. The play reaches the state p_{τ_1} . From p_{τ_1} , Adam must choose the unique particle on the left side, namely $\forall rr^*.A$. To do so, he enters the part of the game corresponding to $\mathcal{A}_{rr^*}$ and generates some word from the language $\mathcal{L}(rr^*)$. Hence, he chooses an arbitrary word r^n , where $n \geq 1$, and the play reaches the configuration $(p_A, \#r^n)$. Now Eve must reduce the current obligation to L . At this point she distinguishes two cases.

Case 1: $n = 1$. Then the current configuration is $(p_A, \#r)$. Eve chooses the axiom

$$\tau_2 = (\forall \varepsilon.B \sqsubseteq \forall r.A).$$

Using the automaton $\mathcal{A}_r$, she removes the symbol r from the top of the stack and reaches the state p_{τ_2} with stack $\#$. The only particle on the left side is $\forall \varepsilon.B$, so Adam inserts the empty word and the play reaches $(p_B, \#)$. Here, Eve chooses the axiom $\tau_3 = (\forall \varepsilon.L \sqsubseteq \forall \varepsilon.B)$, reaches the state p_{τ_3} while removing the empty word from the stack, from which Adam can only insert the empty word and go to $(p_L, \#)$.

Case 2: $n \geq 2$. Then the current configuration is $(p_A, \#r^n)$. Eve chooses the axiom

$$\tau_4 = (\forall \varepsilon.L \sqsubseteq \forall rrr^*.A).$$

Since $r^n \in \mathcal{L}(rrr^*)$ whenever $n \geq 2$, she can use the automaton $\mathcal{A}_{rrr^*}$ to remove the whole suffix r^n from the stack and reach a state p_{τ_4} with stack $\#$. As above, Adam is then forced to move to $(p_L, \#)$.

In both cases Eve reaches the success configuration $(p_L, \#)$, from which she moves to the designated winning state. Therefore Eve has a winning strategy from the initial configuration $(p_R, \#)$.

This example reflects exactly why the original subsumption is valid. The formula $\forall rr^*.A$ requires that all r -paths lead to elements satisfying A . Adam may therefore choose any word r^n with $n \geq 1$. If $n = 1$, Eve answers by the conjunct $\forall r.A$; if $n \geq 2$, she answers by the conjunct $\forall rrr^*.A$. Thus the two conjuncts on the left side cover all words in $\mathcal{L}(rr^*)$, and the subsumption holds.

8 Conclusions

We have shown that the subsumption problem in $\mathcal{FL}_{reg}$ extended with TBoxes is EXPTIME-complete. This result is obtained via a reduction from subsumption to parity pushdown games, which are known to be solvable in exponential time.

Based on this reduction and existing algorithms for solving such games, a decision procedure for subsumption in $\mathcal{FL}_{reg}$ with TBoxes can be derived in a straightforward manner. An implementation of this procedure is planned. In particular, we intend to employ existing solvers for parity pushdown games, such as PDSolver [14].

The approach also applies to $\mathcal{FL}_{\perp reg}$ due to the reduction presented in [Section 5](#). Moreover, since $\mathcal{FL}_0$ is a sublogic of $\mathcal{FL}_{reg}$, the procedure is also applicable to subsumption in $\mathcal{FL}_0$ with TBoxes.

An implementation, *FL₀wer*, for subsumption in $\mathcal{FL}_0$ with TBoxes already exists and is described in Baader et al. [8]. *FL₀wer* is based on a different algorithmic approach. It is therefore of interest to investigate experimentally whether the proposed implementation performs better or worse than *FL₀wer* on subsumption problems in $\mathcal{FL}_0$ with TBoxes. Nevertheless, the presented method provides a general framework for deciding the corresponding reasoning problems in the other logics considered in this work.

A further direction for future work is the extension of the presented techniques to other reasoning tasks in $\mathcal{FL}_0$, $\mathcal{FL}_{\perp}$, $\mathcal{FL}_{reg}$, and $\mathcal{FL}_{\perp reg}$ with TBoxes, such as concept matching and unification.

References

- [1] Rajeev Alur, Michael Benedikt, Kousha Etessami, Patrice Godefroid, Thomas W. Reps, and Mihalis Yannakakis. Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.*, 27(4):786–818, 2005.
- [2] Franz Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *AAAI*, pages 621–626. AAAI Press / The MIT Press, 1990.
- [3] Franz Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *IJCAI*, pages 446–451. Morgan Kaufmann, 1991.
- [4] Franz Baader, Alex Borgida, and Deborah L. McGuinness. Matching in description logics: Preliminary results. In *ICCS*, Lecture Notes in Computer Science, pages 15–34. Springer, 1998.
- [5] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *IJCAI*, pages 364–369. Professional Book Center, 2005.
- [6] Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *J. Log. Lang. Inf.*, 2(1):1–18, 1993.
- [7] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2nd edition, 2007.
- [8] Franz Baader, Patrick Koopmann, Frank Michel, Anni-Yasmin Turhan, and Benjamin Zarriess. Efficient TBox reasoning with value restrictions using the *FL₀wer* reasoner. *Theory Pract. Log. Program.*, 22(2):162–192, 2022.
- [9] Franz Baader and Ralf Küsters. Unification in a description logic with transitive closure of roles. In *LPAR*, Lecture Notes in Computer Science, pages 217–232. Springer, 2001.
- [10] Franz Baader and Ralf Küsters. Unification in a description logic with inconsistency and transitive closure of roles. In *Description Logics*, CEUR Workshop Proceedings. CEUR-WS.org, 2002.
- [11] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *AAAI*, pages 205–212. AAAI Press / The MIT Press, 1994.
- [12] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS*, pages 368–377. IEEE Computer Society, 1991.
- [13] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.
- [14] Matthew Hague and C.-H. Luke Ong. Analysing μ -calculus properties of pushdown systems. In *SPIN*, Lecture Notes in Computer Science, pages 187–192. Springer, 2010.

- [15] Michał Henne, Barbara Morawska, and Paweł Parys. Subsumption in $\mathcal{FL}_{\perp reg}$ with TBoxes is in ExpTime. *CoRR*, abs/2605.13553, 2026.
- [16] Martin Hofmann. Proof-theoretic approach to description logic. In *LICS*, pages 229–237. IEEE Computer Society Press, 2005.
- [17] Harry B. Hunt III, Daniel J. Rosenkrantz, and Thomas G. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *J. Comput. Syst. Sci.*, 12(2):222–268, 1976.
- [18] Sławomir Kost and Barbara Morawska. Around unification in $\mathcal{FL}_{\perp}$ – three related problems (extended abstract). In *DL*, CEUR Workshop Proceedings. CEUR-WS.org, 2025.
- [19] Thomas Reps, Stefan Schwoon, Somesh Jha, and David Melski. Weighted pushdown systems and their application to interprocedural dataflow analysis. *Sci. Comput. Program.*, 58(1-2):206–263, 2005.
- [20] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *IJCAI*, pages 466–471. Morgan Kaufmann, 1991.
- [21] David Toman and Grant E. Weddell. On reasoning about structural equality in XML: a description logic approach. *Theor. Comput. Sci.*, 336(1):181–203, 2005.
- [22] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001.