TPLP **22** (2): 162–192, 2022. © The Author(s), 2021. Published by Cambridge University Press.

doi:10.1017/S1471068421000466 First published online 15 October 2021

Efficient TBox Reasoning with Value Restrictions using the $\mathcal{FL}_o$wer Reasoner[*]

FRANZ BAADER, PATRICK KOOPMANN, FRIEDRICH MICHEL,
ANNI-YASMIN TURHAN and BENJAMIN ZARRIESS
Technische Universitat Dresden, Dresden, Germany
(*e-mails:* franz.baader@tu-dresden.de, patrick.koopmann@tu-dresden.de,
friedrich.michel@tu-dresden.de, anni-yasmin.turhan@tu-dresden.de,
benjamin.zarriess@tu-dresden.de)

submitted 5 May 2020; revised 6 September 2021; accepted 13 September 2021

Abstract

The inexpressive Description Logic (DL) $\mathcal{FL}_0$, which has conjunction and value restriction as its only concept constructors, had fallen into disrepute when it turned out that reasoning in $\mathcal{FL}_0$ w.r.t. general TBoxes is ExpTime-complete, that is, as hard as in the considerably more expressive logic $\mathcal{ALC}$. In this paper, we rehabilitate $\mathcal{FL}_0$ by presenting a dedicated subsumption algorithm for $\mathcal{FL}_0$, which is much simpler than the tableau-based algorithms employed by highly optimized DL reasoners. Our experiments show that the performance of our novel algorithm, as prototypically implemented in our $\mathcal{FL}_o$*wer* reasoner, compares very well with that of the highly optimized reasoners. $\mathcal{FL}_o$*wer* can also deal with ontologies written in the extension $\mathcal{FL}_\bot$ of $\mathcal{FL}_0$ with the top and the bottom concept by employing a polynomial-time reduction, shown in this paper, which eliminates top and bottom. We also investigate the complexity of reasoning in DLs related to the Horn-fragments of $\mathcal{FL}_0$ and $\mathcal{FL}_\bot$.

KEYWORDS: description logics, reasoning, subsumption

1 Introduction

Description Logics (DLs) (Baader *et al.* 2003; 2017) are a well-investigated family of logic-based knowledge representation languages, which are frequently used to formalize ontologies for application domains such as the Semantic Web (Horrocks *et al.* 2003) or biology and medicine (Hoehndorf *et al.* 2015). To define the important notions of such an application domain as formal concepts, DLs state necessary and sufficient conditions for an individual to belong to a concept. These conditions can be Boolean combinations of atomic properties required for the individual (expressed by concept names) or properties that refer to relationships with other individuals and their properties (expressed as role restrictions). For example, the concept of a parent that has only daughters can be formalized by the concept description $C := \exists child.Human \sqcap \forall child.Female$, which uses the concept names *Female* and *Human* and the role name *child* as well as the concept

[*] This paper is under consideration in Theory and Practice of Logic Programming (TPLP).

constructors conjunction ($\sqcap$), existential restriction ($\exists r.D$), and value restriction ($\forall r.D$). Constraints on the interpretation of concept and role names can be formulated as general concept inclusions (GCIs). For example, the GCIs $\mathit{Human} \sqsubseteq \forall \mathit{child}.\mathit{Human}$ and $\exists \mathit{child}.\mathit{Human} \sqsubseteq \mathit{Human}$ say that humans have only human children, and they are the only ones that can have human children. DL systems provide their users with reasoning services that allow them to derive implicit knowledge from the explicitly represented one. In our example, the above GCIs imply that elements of our concept C also belong to the concept $D := \mathit{Human} \sqcap \forall \mathit{child}.\mathit{Human}$, i.e., C is subsumed by D w.r.t. these GCIs. A specific DL is determined by which kind of concept constructors are available.

In the early days of DL research, the inexpressive DL $\mathcal{FL}_0$, which has only conjunction and value restriction as concept constructors, was considered to be the smallest possible DL. In fact, when providing a formal semantics for so-called property edges of semantic networks in the first DL system KL-ONE (Brachman and Schmolze 1985), value restrictions were used. For this reason, the language for constructing concepts in KL-ONE and all of the other early DL systems (Brachman *et al.* 1991; Peltason 1991; Mays *et al.* 1991; Woods and Schmolze 1992) contained $\mathcal{FL}_0$. It came as a surprise when it was shown that subsumption reasoning w.r.t. acyclic $\mathcal{FL}_0$ TBoxes (a restricted form of GCIs) is co-NP-hard (Nebel 1990). The complexity increases when more expressive forms of TBoxes are used: for cyclic TBoxes to PSpace (Baader 1990; Kazakov and de Nivelle 2003) and for general TBoxes consisting of GCIs even to ExpTime (Baader *et al.* 2005; Hofmann 2005). Thus, w.r.t. general TBoxes, subsumption reasoning in $\mathcal{FL}_0$ is as hard as subsumption reasoning in $\mathcal{ALC}$, its closure under negation (Schild 1991).

These negative complexity results for $\mathcal{FL}_0$ were one of the reasons why the attention in the research of inexpressive DLs shifted from $\mathcal{FL}_0$ to $\mathcal{EL}$, which is obtained from $\mathcal{FL}_0$ by replacing value restriction with existential restriction as a concept constructor. In fact, subsumption reasoning in $\mathcal{EL}$ stays polynomial even in the presence of general TBoxes (Brandt 2004). The reasoning method employed in Brandt (2004), which is nowadays called consequence-based reasoning, can be used to establish the PTime complexity upper bounds also for reasoning in the extension $\mathcal{EL}^+$ of $\mathcal{EL}$ (Baader *et al.* 2005). This approach also applies to Horn fragments of expressive DLs such as $\mathcal{SHIQ}$, for which reasoning is ExpTime-complete, but consequence-based reasoning approaches behave considerably better in practice than the usual tableau-based approaches for expressive DLs (Kazakov 2009). The DL $\mathcal{FL}_0$ is not Horn,[1] but it shares with $\mathcal{EL}^+$ and Horn-$\mathcal{SHIQ}$ that (general) TBoxes have canonical models, that is, models such that a subsumption relationship between concept names follows from the TBox if and only if it holds in the canonical model. Consequence-based reasoning basically generates these models. However, whereas the canonical models for $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ are respectively of polynomial and exponential size, the canonical models for $\mathcal{FL}_0$, called least functional models (Baader *et al.* 2018a), may be infinite.

In this paper we build on and extend the results from Michel *et al.* (2019). We devise a novel algorithm for deciding subsumption w.r.t. general $\mathcal{FL}_0$ TBoxes, describe a first implementation of it in the new $\mathcal{FL}_o\mathit{wer}$ reasoner,[2] and report on an evaluation of $\mathcal{FL}_o\mathit{wer}$ on a large collection of ontologies, which shows that $\mathcal{FL}_o\mathit{wer}$ competes well with existing highly optimized DL reasoners. Basically, our new algorithm generates "large enough"

[1] Actually, reasoning in its Horn fragment is PTime (Krötzsch *et al.* 2007; 2013).
[2] `https://github.com/attalos/fl0wer`.

parts of the least functional model and achieves termination using a blocking mechanism similar to the ones employed by tableau-based reasoners. The key idea of the implementation is to apply the TBox statements like rules and to use a variant of the well-known Rete algorithm for rule application (Forgy 1982), adapted to the case without negation. To create a large set of challenging $\mathcal{FL}_0$ ontologies we have used, on the one hand, the OWL 2 EL ontologies of the OWL reasoner competition (Parsia *et al.* 2017) transformed into $\mathcal{FL}_0$ by exchanging the quantifier and omitting too small ontologies as too easy. On the other hand, we have extracted $\mathcal{FL}_0$ sub-ontologies of decent size from the ontologies of the Manchester OWL Corpus (MOWLCorp).[3]

In the next section, we introduce $\mathcal{FL}_0$ and its extension $\mathcal{FL}_\bot$ with the top ($\top$) and the bottom ($\bot$) concepts. We recall the characterization of subsumption based on least functional models from Baader *et al.* (2018a), introduce a normal form for $\mathcal{FL}_0$ TBoxes, and show that the bottom concept $\bot$ and the top concept $\top$ can be simulated by such TBoxes. In Section 3, we introduce our new algorithm, and prove that it is sound, complete, and terminating. Section 4 considers the Horn fragments of $\mathcal{FL}_0$ and $\mathcal{FL}_\bot$. First, we show that, for Horn-$\mathcal{FL}_0$, our algorithm can be restricted such that it runs in polynomial time. A polynomial upper bound for subsumption in Horn-$\mathcal{FL}_0$ has already been shown in Krötzsch *et al.* (2007; 2013) for an extension of Horn-$\mathcal{FL}_0$ that contains $\bot$. However, this extension is weaker than Horn-$\mathcal{FL}_\bot$. In fact, we also show in Section 4 that subsumption in Horn-$\mathcal{FL}_\bot$ is PSpace-complete, and that it becomes ExpTime-complete in a small extension of Horn-$\mathcal{FL}_\bot$. Section 5 describes how to realize our novel algorithm based on Rete, and Section 6 presents our experimental results, which evaluate several optimizations of the algorithm, and compare its performance with that of existing highly optimized DL reasoners.

2 Preliminaries on $\mathcal{FL}_0$ and extensions

We introduce the DL $\mathcal{FL}_0$, recall the characterization of subsumption based on least functional models from Baader *et al.* (2018a), introduce a normal form for $\mathcal{FL}_0$ TBoxes, and show that the bottom concept $\bot$ and the top concept $\top$ can be simulated by such TBoxes.

2.1 Syntax, semantics, and functional interpretations

Syntax. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be disjoint, at most countably infinite sets of *concept names* and *role names*, respectively. An $\mathcal{FL}_0$ *concept description* (*concept* for short) C is built according to the following syntax rule

$$C ::= A \mid C \sqcap C \mid \forall r.C, \quad \text{where } A \in \mathsf{N_C}, r \in \mathsf{N_R}.$$

Additionally allowing the use of the top concept $\top$ and the bottom concept $\bot$ in the above rule yields the DL $\mathcal{FL}_\bot$. A *general concept inclusion* (GCI) for any of these DLs is of the form $C \sqsubseteq D$, where C and D are concepts of the respective DL. A *TBox* is a finite set of GCIs. The *signature* $\mathsf{sig}(C)$ ($\mathsf{sig}(\mathcal{T})$) of a concept C (TBox $\mathcal{T}$) is the set of concept and role names occurring in C ($\mathcal{T}$). For convenience, we use further functions

[3] https://zenodo.org/record/16708.

to refer only to the concept names and only to the role names in an expression. For a concept or TBox E, we set $\mathsf{sig}_\mathsf{C}(E) = \mathsf{sig}(E) \cap \mathsf{N_C}$ and $\mathsf{sig}_\mathsf{R}(E) = \mathsf{sig}(E) \cap \mathsf{N_R}$.

The expression $\forall r.C$ is called a *value restriction*. For nested value restrictions we use the following notation: given a word $\sigma = r_1 \cdots r_m \in \mathsf{N_R}^*$, $m \geq 0$, over the alphabet $\mathsf{N_R}$ of role names, and a concept C, we write $\forall\sigma.C$ as an abbreviation of $\forall r_1. \cdots \forall r_m.C$. For the empty word ϵ, we have $\forall\epsilon.C = C$.

Semantics. An interpretation $\mathcal{I}$ is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, consisting of a non-empty set $\Delta^\mathcal{I}$ (the *domain* of $\mathcal{I}$) and an *interpretation function* $\cdot^\mathcal{I}$ that maps every concept name $A \in \mathsf{N_C}$ to a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ of the domain, and every role name $r \in \mathsf{N_R}$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation function is extended to (complex) concepts as follows:

$$(C \sqcap D)^\mathcal{I} := C^\mathcal{I} \sqcap D^\mathcal{I}, \qquad \top^\mathcal{I} := \Delta^\mathcal{I}, \qquad \bot^\mathcal{I} := \emptyset, \text{ and}$$
$$(\forall r.C)^\mathcal{I} := \{d \in \Delta^\mathcal{I} \mid \forall e \in \Delta^\mathcal{I}.(d,e) \in r^\mathcal{I} \ \rightarrow \ e \in C^\mathcal{I}\}.$$

The GCI $C \sqsubseteq D$ is *satisfied* in $\mathcal{I}$, denoted as $\mathcal{I} \models C \sqsubseteq D$, if $C^\mathcal{I} \subseteq D^\mathcal{I}$. The interpretation $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$, denoted as $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I}$ satisfies all GCIs in $\mathcal{T}$. The concept C is *subsumed* by the concept D *w.r.t.* $\mathcal{T}$, denoted as $C \sqsubseteq_\mathcal{T} D$, if $C^\mathcal{I} \subseteq D^\mathcal{I}$ is satisfied in all models $\mathcal{I}$ of $\mathcal{T}$.

To decide subsumption in $\mathcal{FL}_0$, it is sufficient to consider so-called functional interpretations, which are tree-shaped interpretations in which every element has exactly one child for each role name. In such interpretations, domain elements are identified by sequences of role names.

Definition 2.1
An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is called a *functional interpretation* if $\Delta^\mathcal{I} = \mathsf{N_R}^*$ and for all $r \in \mathsf{N_R}$, $r^\mathcal{I} = \{(\sigma, \sigma r) \mid \sigma \in \mathsf{N_R}^*\}$. It is called a *functional model* of the $\mathcal{FL}_0$ concept C w.r.t. the $\mathcal{FL}_0$ TBox $\mathcal{T}$ if $\mathcal{I} \models \mathcal{T}$ and $\epsilon \in C^\mathcal{I}$. For two functional interpretations $\mathcal{I}$ and $\mathcal{J}$ we write

$$\mathcal{I} \subseteq \mathcal{J} \quad \text{if} \quad A^\mathcal{I} \subseteq A^\mathcal{J} \text{ for all } A \in \mathsf{N_C}.$$

The notion of a functional interpretation fixes the domain and the interpretation of role names. Thus, a functional interpretation is uniquely determined by the interpretation of the concept names. Given a family $(\mathcal{I}_i)_{i\geq 0}$ of functional interpretations, their *intersection* $\mathcal{J} := \bigcap_{i\geq 0} \mathcal{I}_i$ is the functional interpretations that satisfies $A^\mathcal{J} = \bigcap_{i\geq 0} A^{\mathcal{I}_i}$.

Lemma 2.1 (*see* Baader *et al.* 2018a)
Given an $\mathcal{FL}_0$ concept C and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, the functional models of C w.r.t. $\mathcal{T}$ are closed under intersection. In particular, this implies that there exists a *least functional model* $\mathcal{I}_{C,\mathcal{T}}$ of C w.r.t. $\mathcal{T}$, i.e., a functional model of C w.r.t. $\mathcal{T}$ such that $\mathcal{I}_{C,\mathcal{T}} \subseteq \mathcal{J}$ holds for all functional models $\mathcal{J}$ of C w.r.t. $\mathcal{T}$.

In Baader *et al.* (2018a), subsumption in $\mathcal{FL}_0$ was characterized as inclusion of least functional models as follows: given $\mathcal{FL}_0$ concepts C, D and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, we have

$$C \sqsubseteq_\mathcal{T} D \ \text{ iff } \ \mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}. \tag{1}$$

For our purposes, the following characterization of subsumption turns out to be more useful.

Theorem 2.1
Given $\mathcal{FL}_0$ concepts C, D and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, we have $C \sqsubseteq_{\mathcal{T}} D$ iff $\varepsilon \in D^{\mathcal{I}_{C,\mathcal{T}}}$.

Proof
Assume that $C \sqsubseteq_{\mathcal{T}} D$. Then $\varepsilon \in C^{\mathcal{I}_{C,\mathcal{T}}}$ (which we know since $\mathcal{I}_{C,\mathcal{T}}$ is a functional model of C w.r.t. $\mathcal{T}$) implies $\varepsilon \in D^{\mathcal{I}_{C,\mathcal{T}}}$ since $\mathcal{I}_{C,\mathcal{T}}$ is a model of $\mathcal{T}$. Conversely, $\varepsilon \in D^{\mathcal{I}_{C,\mathcal{T}}}$ implies that $\mathcal{I}_{C,\mathcal{T}}$ is a functional model of D w.r.t. $\mathcal{T}$, and thus $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$, which yields $C \sqsubseteq_{\mathcal{T}} D$ by (1). □

2.2 Normal forms for $\mathcal{FL}_\bot$ and $\mathcal{FL}_0$ concepts and TBoxes

An $\mathcal{FL}_\bot$ *concept* is in *normal form* if it is of the form

- $\top$ or $\bot$, or
- a non-empty conjunction of concepts of the form $\forall r.\bot$, A, $\forall r.A$, where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$.

An $\mathcal{FL}_\bot$ *TBox* is in *normal form* if it contains only GCIs of the form $C \sqsubseteq D$, where C, D are in normal form, and C is not $\bot$ and D is not $\top$. In addition, $\mathcal{FL}_0$ concepts (TBoxes) in normal form are $\mathcal{FL}_\bot$ concepts (TBoxes) in normal form that contain neither $\top$ nor $\bot$.

It is easy to see that every ($\mathcal{FL}_\bot$ or $\mathcal{FL}_0$) TBox $\mathcal{T}$ can be transformed in linear time into a TBox in normal form such that all subsumption relationships in the signature of $\mathcal{T}$ are preserved. For this, one removes tautological GCIs with $\bot$ on the left-hand side or $\top$ on the right-hand side, and flattens value-restrictions $\forall r.E$ with $E \notin \mathsf{N_C} \cup \{\bot\}$. To *flatten* an occurrence of $\forall r.E$ in a GCI $C \sqsubseteq D$ means that E is replaced by a fresh concept name A_E. If the occurrence is within C, then the GCI $E \sqsubseteq A_E$ is added to the TBox, and otherwise $A_E \sqsubseteq E$.

It is well-known that subsumption between complex concepts can be reduced in linear time to subsumption between concept names. In fact, we have $C \sqsubseteq_{\mathcal{T}} D$ iff $A \sqsubseteq_{\mathcal{T}'} B$, where A, B are concept names not occurring in C, D, or $\mathcal{T}$, and $\mathcal{T}'$ is obtained from $\mathcal{T}$ by adding the GCIs $A \sqsubseteq C$ and $D \sqsubseteq B$.

Proposition 2.1
Subsumption in $\mathcal{FL}_\bot$ ($\mathcal{FL}_0$) w.r.t. TBoxes can be reduced in linear time to subsumption of concept names w.r.t. $\mathcal{FL}_\bot$ ($\mathcal{FL}_0$) TBoxes in normal form.

For subsumption between concept names A, B in the DL $\mathcal{FL}_0$, the characterization of subsumption given in Theorem 2.1 means that, to decide whether $A \sqsubseteq_{\mathcal{T}} B$ holds, it is sufficient to check whether the root of $\mathcal{I}_{A,\mathcal{T}}$ is contained in $B^{\mathcal{I}_{A,\mathcal{T}}}$, that is, whether the label of this root contains the concept name B.

2.3 Reducing subsumption in $\mathcal{FL}_\bot$ to subsumption in $\mathcal{FL}_0$

Subsumption between concept names in $\mathcal{FL}_\bot$ can be reduced to subsumption in $\mathcal{FL}_0$ using the following transformation rules on *normalized* $\mathcal{FL}_\bot$ TBoxes $\mathcal{T}$:

T1 Replace $\bot$ and $\top$ everywhere by the fresh concept names $A_\bot$ and $A_\top$, respectively;
T2 add the axioms $A_\bot \sqsubseteq B$ for all $B \in \mathsf{sig_C}(\mathcal{T})$;
T3 add the axioms $B \sqsubseteq A_\top$ and $A_\top \sqsubseteq \forall r.A_\top$ for all $B \in \mathsf{sig_C}(\mathcal{T})$ and all $r \in \mathsf{sig_R}(\mathcal{T})$.

We denote the TBox resulting from this transformation as $\mathcal{FL}_0(\mathcal{T})$.

Lemma 2.2
For all $\mathcal{FL}_\bot$ TBoxes $\mathcal{T}$ in normal form and all concept names A, B occurring in $\mathcal{T}$, we have $A \sqsubseteq_\mathcal{T} B$ iff $A \sqsubseteq_{\mathcal{FL}_0(\mathcal{T})} B$.

Proof
"$\Leftarrow$": Assume that $A \not\sqsubseteq_\mathcal{T} B$. Then there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $A^\mathcal{I} \not\subseteq B^\mathcal{I}$. We modify $\mathcal{I}$ to an interpretation $\mathcal{J}$ by setting ${A_\bot}^\mathcal{J} := \emptyset$ and ${A_\top}^\mathcal{J} := \Delta^\mathcal{I}$, and leave the domain as well as the interpretation of the other concept names and the role names as in $\mathcal{I}$. It is easy to see that $\mathcal{J}$ is a model of $\mathcal{FL}_0(\mathcal{T})$ that satisfies $A^\mathcal{J} = A^\mathcal{I} \not\subseteq B^\mathcal{I} = B^\mathcal{J}$.

"$\Rightarrow$": Assume that $A \not\sqsubseteq_{\mathcal{FL}_0(\mathcal{T})} B$, and let $\mathcal{I}$ be a model of $\mathcal{FL}_0(\mathcal{T})$ that contains an element d_0 with $d_0 \in A^\mathcal{I} \setminus B^\mathcal{I}$. We may assume without loss of generality that all elements of $\Delta^\mathcal{I}$ are reachable from d_0 via a path of roles in $\mathsf{sig}_\mathsf{R}(\mathcal{T})$. Due to the GCIs introduced by **T3**, $d_0 \in A^\mathcal{I}$ yields $d_0 \in {A_\top}^\mathcal{I}$, and thus $d \in {A_\top}^\mathcal{I}$ holds for all $d \in \Delta^\mathcal{I}$. We also know that $d_0 \notin {A_\bot}^\mathcal{I}$ since otherwise the GCI $A_\bot \sqsubseteq B$ added by **T2** would yield $d_0 \in B^\mathcal{I}$, contradicting our assumption that d_0 is a counterexample to the subsumption. The interpretation $\mathcal{J}$ is obtained from $\mathcal{I}$ by removing all elements of ${A_\bot}^\mathcal{I}$. Then d_0 is an element of $\Delta^\mathcal{J}$ and it satisfies $d_0 \in A^\mathcal{J} \setminus B^\mathcal{J}$. Thus, it remains to show that $\mathcal{J}$ is a model of $\mathcal{T}$.

First, note that ${A_\top}^\mathcal{J} = \Delta^\mathcal{J} = \top^\mathcal{J}$ and ${A_\bot}^\mathcal{J} = \emptyset = \bot^\mathcal{J}$. This implies that it is enough to prove that the GCIs from $\mathcal{T}$ transformed by **T1**, which are satisfied by $\mathcal{I}$ since it is a model of $\mathcal{FL}_0(\mathcal{T})$, are also satisfied by $\mathcal{J}$. For this, it is in turn sufficient to show that, for all concepts C in normal form occurring in $\mathcal{FL}_0(\mathcal{T})$ and all $d \in \Delta^\mathcal{J}$ we have $d \in C^\mathcal{I}$ iff $d \in C^\mathcal{J}$. For concept names this is trivial by the definition of $\mathcal{J}$. Thus, consider a value restriction of the form $\forall r.A_1$.

First, assume that $d \in (\forall r.A_1)^\mathcal{I}$, but $d \notin (\forall r.A_1)^\mathcal{J}$. Then there is an element $e \in \Delta^\mathcal{J}$ with $(d,e) \in r^\mathcal{J}$, but $e \notin {A_1}^\mathcal{J}$. However, since $e \in \Delta^\mathcal{J}$, we already know that $e \notin {A_1}^\mathcal{J}$ implies $e \notin {A_1}^\mathcal{I}$. Since we also have $(d,e) \in r^\mathcal{I}$, this contradicts our assumption that $d \in (\forall r.A_1)^\mathcal{I}$.

Second, assume that $d \in (\forall r.A_1)^\mathcal{J}$, but $d \notin (\forall r.A_1)^\mathcal{I}$. Then there is an element $e \in \Delta^\mathcal{I}$ with $(d,e) \in r^\mathcal{I}$, but $e \notin {A_1}^\mathcal{I}$. If $e \in \Delta^\mathcal{J}$, then we also have $(d,e) \in r^\mathcal{J}$ and $e \notin {A_1}^\mathcal{J}$, which contradicts our assumption that $d \in (\forall r.A_1)^\mathcal{J}$. Otherwise, we must have $e \in {A_\bot}^\mathcal{I}$ since e was removed. But then the GCIs introduced by **T2** yield $e \in {A_1}^\mathcal{I}$, contradicting our assumption on e.[4] □

Since normalization of an $\mathcal{FL}_\bot$ TBox and the transformation into an $\mathcal{FL}_0$ TBox described in this subsection are polynomial, we obtain the following result.

Theorem 2.2
Subsumption in $\mathcal{FL}_\bot$ can be reduced in polynomial time to subsumption in $\mathcal{FL}_0$.

3 Subsumption algorithm for $\mathcal{FL}_0$ with general TBoxes

We define a decision procedure for subsumption of two concepts w.r.t. a TBox based on a finite representation of the least functional model obtained by "applying" GCIs

[4] Note that A_1 cannot be $A_\top$ since a value restriction of the form $\forall r.\top$ is not normalized.

like rules. By Proposition 2.1, it is sufficient to focus on $\mathcal{FL}_0$ TBoxes in normal form and subsumption between concept names. We can then use Lemma 2.2 to extend the applicability of our algorithm to $\mathcal{FL}_\bot$.

In the remainder of this section, $\mathcal{T}$ denotes a $\mathcal{FL}_0$ TBox in normal form, and we focus on the task of deciding $A \sqsubseteq_\mathcal{T} B$ for two concept names A, B occurring in $\mathcal{T}$. For the sake of simplicity, we assume in this section that $\mathsf{N_C}$ and $\mathsf{N_R}$ consist exactly of the concept and role names occurring in $\mathcal{T}$. In particular, this means that $\mathsf{N_C}$ and $\mathsf{N_R}$ are finite and their cardinalities are bounded by the size of $\mathcal{T}$.

The algorithm computes a finite subtree of the tree $\mathcal{I}_{A,\mathcal{T}}$ such that one can read off the named subsumers (concept names) of A w.r.t. $\mathcal{T}$ at the root. The finite structure that the algorithm operates on is called *partial functional interpretation*. This is similar to a functional interpretation, except that the domain is a *finite* prefix-closed subset of $\mathsf{N_R}^*$, that is, a finite tree.

Definition 3.1
An interpretation $\mathcal{Y} = (\Delta^\mathcal{Y}, \cdot^\mathcal{Y})$ is a *partial functional interpretation* iff $\Delta^\mathcal{Y} \subseteq \mathsf{N_R}^*$ is a *finite* prefix-closed set and $r^\mathcal{Y} = \{(\sigma, \sigma r) \mid \sigma r \in \Delta^\mathcal{Y}\}$ for all $r \in \mathsf{N_R}$.

Note that, as with functional interpretations, the interpretation of the role names is already determined by the domain. Thus, it suffices to give the domain and the interpretation of concept names to fix a partial interpretation.

Informally, the algorithm for deciding $A \sqsubseteq_\mathcal{T} B$ proceeds as follows: it starts with a partial functional interpretation $\mathcal{Y}$ that has ϵ as only domain element, and for which $A^\mathcal{Y} = \{\epsilon\}$. In each iteration, a domain element d of the current tree $\mathcal{Y}$ and a single GCI $C \sqsubseteq D$ from $\mathcal{T}$ is chosen such that d matches C and does not match D. The tree is then extended so that d matches D. The extension can affect both the domain and the interpretation of concept names. The method proceeds in such a way that, for every generated tree $\mathcal{Y}$, the invariant $\mathcal{Y} \subseteq \mathcal{I}_{A,\mathcal{T}}$ is satisfied. Termination is established by blocking further extensions for duplicate elements. The algorithm terminates if the following holds for every non-blocked element d and every GCI $C \sqsubseteq D$ in $\mathcal{T}$: if d matches C, then d also matches D. Soundness and completeness is shown by establishing a correspondence between the nodes in the final tree and nodes in the least function model of A w.r.t. $\mathcal{T}$. To describe the procedure more formally, we must define the following notions:

1. the condition under which a domain element of a partial interpretation matches a concept,
2. the extension of the tree to achieve a match of an element with the right-hand side of a GCI, and
3. the conditions that distinguish blocked from non-blocked elements.

To address the first point, we introduce the following auxiliary notions.

Definition 3.2
Let $\mathcal{Y} = (\Delta^\mathcal{Y}, \cdot^\mathcal{Y})$ be a partial functional interpretation and D a concept in normal form. The set of elements in $\Delta^\mathcal{Y}$ that *match* D, denoted by $\mathsf{match}(D, \mathcal{Y})$, is defined inductively as follows:

$$\begin{aligned} \mathsf{match}(A, \mathcal{Y}) &:= A^\mathcal{Y} \text{ for all } A \in \mathsf{N_C}; \\ \mathsf{match}(\forall r.A, \mathcal{Y}) &:= \{\sigma \in \Delta^\mathcal{Y} \mid \sigma r \in A^\mathcal{Y}\} \text{ for all } r \in \mathsf{N_R} \text{ and } A \in \mathsf{N_C}; \\ \mathsf{match}(C_1 \sqcap C_2, \mathcal{Y}) &:= \mathsf{match}(C_1, \mathcal{Y}) \cap \mathsf{match}(C_2, \mathcal{Y}). \end{aligned}$$

Since $\mathcal{Y}$ is partial functional (i.e. has *at most* one child per node for each role name), it is easy to see that $\sigma \in \mathsf{match}(C, \mathcal{Y})$ implies $\sigma \in C^{\mathcal{Y}}$. The converse need not be true, as σ may have no r-child in $\Delta^{\mathcal{Y}}$. We say that $\sigma \in \Delta^{\mathcal{Y}}$ *violates* the GCI $C \sqsubseteq D$ iff $\sigma \in \mathsf{match}(C, \mathcal{Y})$ and $\sigma \notin \mathsf{match}(D, \mathcal{Y})$. In this case, σ is called an *incomplete element.* Given a TBox $\mathcal{T}$ in normal form and a partial functional interpretation $\mathcal{Y}$, we define the *set of all incomplete elements* as follows:

$$\mathsf{ic}(\mathcal{Y}, \mathcal{T}) := \{\sigma \in \Delta^{\mathcal{Y}} \mid \text{ there is } C \sqsubseteq D \in \mathcal{T} \text{ such that } \sigma \text{ violates } C \sqsubseteq D\}.$$

Intuitively, the elements in $\mathsf{ic}(\mathcal{Y}, \mathcal{T})$ are those eligible for an extension of $\mathcal{Y}$ toward building a representation of the least functional model, while those in $\Delta^{\mathcal{Y}} \setminus \mathsf{ic}(\mathcal{Y}, \mathcal{T})$ are not. As an additional filter for extensions, we define a blocking condition. First, we introduce auxiliary notions for the blocking mechanism consisting of the standard notions of prefix, proper prefix, and a strict total order on $(\mathsf{N_R})^*$.

Let $\sigma, \rho \in \mathsf{N_R}^*$. The *length of an element* $\sigma \in \mathsf{N_R}^*$ is denoted by $|\sigma|$. We write $\rho \in \mathsf{prefix}(\sigma)$ if $\sigma = \rho\widehat{\sigma}$ for some $\widehat{\sigma} \in \mathsf{N_R}^*$, and $\rho \in \mathsf{pprefix}(\sigma)$ if $\rho \in \mathsf{prefix}(\sigma)$ and $\rho \neq \sigma$. In the latter case, ρ is called a *proper prefix* of σ. Let $\prec$ be any total order on $\mathsf{N_R}^*$ such that $|\sigma| < |\rho|$ implies $\sigma \prec \rho$ for all $\sigma, \rho \in \mathsf{N_R}^*$. Since $\mathsf{N_R}$ is finite, this implies that, for any element of $\sigma \in \mathsf{N_R}^*$, there are only finitely many elements ρ such that $\rho \prec \sigma$. In particular, the order $\prec$ is well-founded.

For a (partial) functional interpretation $\mathcal{Y} = (\Delta^{\mathcal{Y}}, \cdot^{\mathcal{Y}})$ and $\sigma \in \Delta^{\mathcal{Y}}$, we define the *label* of σ in $\mathcal{Y}$ as $\mathcal{Y}(\sigma) := \{A \in \mathsf{N_C} \mid \sigma \in A^{\mathcal{Y}}\}$. The cardinality of $\mathcal{Y}(\sigma)$ is bounded by the size of $\mathcal{T}$, and thus there can be only exponentially many different such labels.

Definition 3.3
Let $\mathcal{Y} = (\Delta^{\mathcal{Y}}, \cdot^{\mathcal{Y}})$ be a partial functional interpretation. The *set of all blocked elements* in $\Delta^{\mathcal{Y}}$ is defined by induction over the well-founded order $\prec$:

B1 The least element ϵ is not blocked.

B2 The element $\sigma \in \Delta^{\mathcal{Y}}$ is blocked if there exists $\omega \in \Delta^{\mathcal{Y}}$ with $\omega \prec \sigma$ such that $\mathcal{Y}(\sigma) = \mathcal{Y}(\omega)$ and ω is not blocked.

B3 Furthermore, the element $\sigma \in \Delta^{\mathcal{Y}}$ is blocked if there exists $\rho \in \mathsf{pprefix}(\sigma)$ such that ρ is blocked.

Only elements of $\Delta^{\mathcal{Y}}$ for which **B1** or **B2** holds can be blocked. All other elements are *non-blocked* elements, which are collected in the set $\mathsf{nb}(\mathcal{Y})$.

Condition **B2** corresponds to *anywhere blocking* in classical tableau algorithms: intuitively, if there are two nodes with the same label, it suffices to reason only on one of them, and the ordering decides which one is used. Condition **B3** corresponds to *ancestor blocking*: if it is already decided that a node can be ignored, it is not necessary to consider its descendants either. Nodes blocked due Condition **B2** are called *directly blocked*, while nodes blocked due Condition **B3** are called *indirectly blocked.*

Next, we define what an extension step is. Such a step expands a single non-blocked and incomplete element in a partial functional interpretation.

Definition 3.4
Let $\mathcal{Y}$ be a partial functional interpretation, $\mathcal{T}$ a TBox in normal form, $m, n \geq 0$ and

$$\alpha \text{ a GCI in } \mathcal{T} \text{ of the form } \ \alpha = C \sqsubseteq (A_1 \sqcap \cdots \sqcap A_m \sqcap \forall r_1.B_1 \sqcap \cdots \forall r_n.B_n)\,.$$

In addition, let $\sigma \in \mathsf{nb}(\mathcal{Y}) \cap \mathsf{ic}(\mathcal{Y}, \mathcal{T})$ be a non-blocked, incomplete element in $\mathcal{Y}$ violating α. Then, the *expansion of α at σ in $\mathcal{Y}$* is the partial interpretation $\mathcal{Z}$ defined by

- $\Delta^{\mathcal{Z}} = \Delta^{\mathcal{Y}} \cup \{\sigma r_1, \dots, \sigma r_n\}$;
- $A_i^{\mathcal{Z}} = A_i^{\mathcal{Y}} \cup \{\sigma\}$ for all $i = 1, \dots, m$;
- $B_i^{\mathcal{Z}} = B_i^{\mathcal{Y}} \cup \{\sigma r_j \mid 1 \le j \le n, B_j = B_i\}$ for all $i = 1, \dots, n$; and
- $Q^{\mathcal{Z}} = Q^{\mathcal{Y}}$ for all $Q \in \mathsf{N_C} \setminus \{A_1, \dots A_m, B_1, \dots, B_n\}$.

A partial functional interpretation $\mathcal{Z}$ is a *$\mathcal{T}$-completion* of $\mathcal{Y}$, written as $\mathcal{Y} \vdash_{\mathcal{T}} \mathcal{Z}$, iff $\mathcal{Z}$ is an expansion of some $\alpha \in \mathcal{T}$ at some $\sigma' \in \mathsf{nb}(\mathcal{Y}) \cap \mathsf{ic}(\mathcal{Y}, \mathcal{T})$. We denote by ${\vdash_{\mathcal{T}}}^*$ the reflexive transitive closure of $\vdash_{\mathcal{T}}$ and call $\mathcal{Z}$ with $\mathcal{Y} {\vdash_{\mathcal{T}}}^* \mathcal{Z}$ *complete* if every incomplete element is blocked, that is, $\mathsf{nb}(\mathcal{Y}_n) \cap \mathsf{ic}(\mathcal{Y}_n, \mathcal{T}) = \emptyset$.

Depending on the choice of σ and the GCI, there can be several $\mathcal{T}$-completions of $\mathcal{Y}$. Also note that it is guaranteed that either $\mathsf{nb}(\mathcal{Y}) \cap \mathsf{ic}(\mathcal{Y}, \mathcal{T}) = \emptyset$ or there exists a $\mathcal{T}$-completion of $\mathcal{Y}$. Thus, in case a given $\mathcal{Z}$ with $\mathcal{Y} {\vdash_{\mathcal{T}}}^* \mathcal{Z}$ is not complete, it can be further completed.

Given the input $A_0, B_0 \in \mathsf{N_C}$ and $\mathcal{T}$, the algorithm $\mathit{Subs}(A_0, B_0, \mathcal{T})$ for deciding $A_0 \sqsubseteq_{\mathcal{T}} B_0$ computes a sequence of $\mathcal{T}$-completions until it reaches a complete partial functional interpretation, i.e., one where no non-blocked element violates any GCI from $\mathcal{T}$. The algorithm starts with the following partial functional interpretation:

$$\Delta^{\mathcal{Y}_0} := \{\epsilon\}; \quad A_0^{\mathcal{Y}_0} := \{\epsilon\} \quad \text{and} \quad B^{\mathcal{Y}_0} := \emptyset \text{ for all } B \in \mathsf{N_C} \setminus \{A_0\}, \tag{2}$$

and computes a sequence

$$\mathcal{Y}_0 \vdash_{\mathcal{T}} \mathcal{Y}_1 \vdash_{\mathcal{T}} \cdots \mathcal{Y}_{(n-1)} \vdash_{\mathcal{T}} \mathcal{Y}_n$$

such that $\mathcal{Y}_n$ is complete in the sense introduced above. It answers "yes" if $B_0 \in \mathcal{Y}_n(\epsilon)$ (or equivalently $\epsilon \in B_0^{\mathcal{Y}_n}$) and "no" otherwise.

Example 3.1
In this example, we illustrate the completion steps and how the blocking conditions are applied. Let $\mathsf{N_C} = \{A, B, K, L, M\}$ and $\mathsf{N_R} = \{r, s\}$. The TBox $\mathcal{T}$ is defined as follows:

$$\begin{array}{lrlrl} \mathcal{T} := \{ & A \sqsubseteq \forall r.A, & & A \sqsubseteq & B, \\ & A \sqsubseteq \forall s.K, & & K \sqsubseteq & \forall s.A, \\ & \forall s.B \sqsubseteq L, & & \forall s.L \sqsubseteq & M \quad \}. \end{array}$$

One can verify that

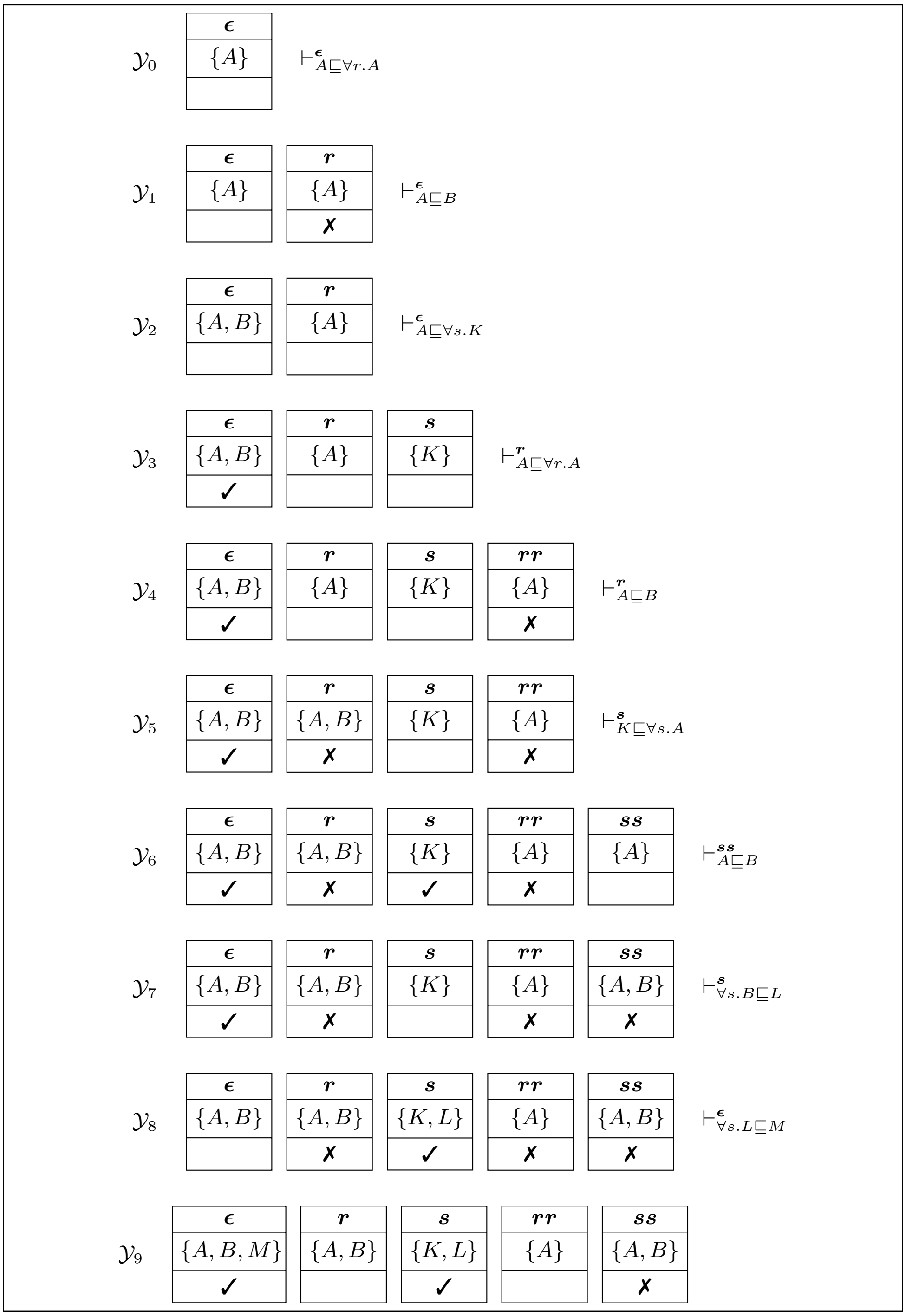
$$A \sqsubseteq_{\mathcal{T}} M.$$

In fact, the GCIs $A \sqsubseteq \forall s.K$, $K \sqsubseteq \forall s.A$ and $A \sqsubseteq B$ yield $A \sqsubseteq_{\mathcal{T}} \forall s.\forall s.B$. Using $\forall s.B \sqsubseteq L$ and $\forall s.L \sqsubseteq M$, we then obtain $A \sqsubseteq_{\mathcal{T}} M$. We use a total order on ${\mathsf{N_R}}^*$ that satisfies

$$\epsilon \prec r \prec s \prec rr \prec rs \prec sr \prec ss \prec rrr \prec \cdots$$

and compute a sequence of completion steps for $\mathit{Subs}(A, M, \mathcal{T})$ sketched in Figure 1, where

✗ marks blocked elements, and
✓ marks non-blocked elements not violating any GCI in $\mathcal{T}$.



$\mathcal{Y}_0$ $\vdash^{\epsilon}_{A\sqsubseteq\forall r.A}$

ϵ
$\{A\}$

$\mathcal{Y}_1$ $\vdash^{\epsilon}_{A\sqsubseteq B}$

ϵ	r
$\{A\}$	$\{A\}$
	✗

$\mathcal{Y}_2$ $\vdash^{\epsilon}_{A\sqsubseteq\forall s.K}$

ϵ	r
$\{A,B\}$	$\{A\}$

$\mathcal{Y}_3$ $\vdash^{r}_{A\sqsubseteq\forall r.A}$

ϵ	r	s
$\{A,B\}$	$\{A\}$	$\{K\}$
✓		

$\mathcal{Y}_4$ $\vdash^{r}_{A\sqsubseteq B}$

ϵ	r	s	rr
$\{A,B\}$	$\{A\}$	$\{K\}$	$\{A\}$
✓			✗

$\mathcal{Y}_5$ $\vdash^{s}_{K\sqsubseteq\forall s.A}$

ϵ	r	s	rr
$\{A,B\}$	$\{A,B\}$	$\{K\}$	$\{A\}$
✓	✗		✗

$\mathcal{Y}_6$ $\vdash^{ss}_{A\sqsubseteq B}$

ϵ	r	s	rr	ss
$\{A,B\}$	$\{A,B\}$	$\{K\}$	$\{A\}$	$\{A\}$
✓	✗	✓	✗	

$\mathcal{Y}_7$ $\vdash^{s}_{\forall s.B\sqsubseteq L}$

ϵ	r	s	rr	ss
$\{A,B\}$	$\{A,B\}$	$\{K\}$	$\{A\}$	$\{A,B\}$
✓	✗		✗	✗

$\mathcal{Y}_8$ $\vdash^{\epsilon}_{\forall s.L\sqsubseteq M}$

ϵ	r	s	rr	ss
$\{A,B\}$	$\{A,B\}$	$\{K,L\}$	$\{A\}$	$\{A,B\}$
	✗	✓	✗	✗

$\mathcal{Y}_9$

ϵ	r	s	rr	ss
$\{A,B,M\}$	$\{A,B\}$	$\{K,L\}$	$\{A\}$	$\{A,B\}$
✓		✓		✗

Fig. 1. Example run.

We write $\vdash^{\boldsymbol{\epsilon}}_{A \sqsubseteq \forall r.A}$ to denote the completion step that takes $\boldsymbol{\epsilon}$ as a non-blocked element violating $A \sqsubseteq \forall r.A$ and applies the expansion. Figure 1 shows the first completion steps needed to obtain $M \in \mathcal{Y}_9(\epsilon)$, which yields $A \sqsubseteq_{\mathcal{T}} M$. For example, in $\mathcal{Y}_1$ the blocking condition **B2** is used to block the node $\boldsymbol{r}$. In $\mathcal{Y}_2$, $\boldsymbol{r}$ is no longer blocked since the label of $\boldsymbol{\epsilon}$ has been expanded. In $\mathcal{Y}_5$, $\boldsymbol{r}$ gets again blocked since its label is expanded, and thus $\boldsymbol{rr}$ is indirectly blocked due to **B3**. Also note that in $\mathcal{Y}_6$ we have $\boldsymbol{rr} \prec \boldsymbol{ss}$ and both have the same label, but since $\boldsymbol{rr}$ is already blocked, **B2** does not apply to $\boldsymbol{ss}$, which allows us to do further completion steps needed to derive $A \sqsubseteq_{\mathcal{T}} M$.

Before we prove that the algorithm is sound and complete, we first show that the computed sequence is always finite, thus ensuring termination of the algorithm. The *depth* of a partial functional interpretation $\mathcal{Y} = (\Delta^{\mathcal{Y}}, \cdot^{\mathcal{Y}})$, denoted by $\mathsf{depth}(\mathcal{Y})$, is the maximum length of role words in $\Delta^{\mathcal{Y}}$, i.e., $\mathsf{depth}(\mathcal{Y}) := \mathsf{max}(\{|\sigma| \mid \sigma \in \Delta^{\mathcal{Y}}\})$.

Lemma 3.1
If $\mathcal{Z}$ is a partial functional interpretation such that $\mathcal{Y}_0 \vdash_{\mathcal{T}}{}^* \mathcal{Z}$, then $\mathsf{depth}(\mathcal{Z}) \leq 2^{|\mathsf{N_C}|} + 1$.

Proof
Let $\mathcal{Y}_0 \vdash_{\mathcal{T}} \mathcal{Y}_1 \vdash_{\mathcal{T}} \ldots \vdash_{\mathcal{T}} \mathcal{Y}_n = \mathcal{Z}$ be a sequence of expansions. We show for each i, $1 \leq i \leq n$, that the length of words in $\Delta^{\mathcal{Y}_i}$ is bounded by $2^{|\mathsf{N_C}|} + 1$. A new element $\sigma \in \Delta^{\mathcal{Y}_i} \setminus \Delta^{\mathcal{Y}_{i-1}}$ is only added by the expansion at σ of $\mathcal{Y}_{i-1}$ if $\sigma = \omega r$ and $\omega \in \mathsf{nb}(\mathcal{Y}_{i-1})$. Now, $\omega \in \mathsf{nb}(\mathcal{Y}_{i-1})$ is only possible if there exist no two distinct $\sigma_1, \sigma_2 \in \mathsf{prefix}(\omega)$ such that $\mathcal{Y}_{i-1}(\sigma_1) = \mathcal{Y}_{i-1}(\sigma_2)$. Otherwise, since either $\sigma_1 \prec \sigma_2$ or $\sigma_2 \prec \sigma_1$, one of these two nodes would be blocked by blocking condition **B2**, and ω would be blocked by condition **B3**. It follows that $\mathcal{Y}_{i-1}(\sigma_1) \neq \mathcal{Y}_{i-1}(\sigma_2)$ for every two distinct σ_1, $\sigma_2 \in \mathsf{prefix}(\omega)$, and consequently $|\omega| \leq 2^{|\mathsf{N_C}|}$ and $|\sigma| = |\omega| + 1 \leq 2^{|\mathsf{N_C}|} + 1$. Hence, $|\sigma| \leq 2^{|\mathsf{N_C}|} + 1$ for every $\sigma \in \Delta^{\mathcal{Z}}$, which yields $\mathsf{depth}(\mathcal{Z}) \leq 2^{|\mathsf{N_C}|} + 1$. □

The upper bound on the depth of the tree in a $\mathcal{T}$-completion sequence also yields an upper bound on its overall size, since the outdegree of the tree is limited by $|\mathsf{N_R}|$. Furthermore, we observe that $\mathcal{Y} \vdash_{\mathcal{T}} \mathcal{Y}'$ implies that $\mathcal{Y} \subsetneq \mathcal{Y}'$, that is a $\mathcal{T}$-completion always adds something and never removes anything. At the same time, each label set can contain at most $|\mathsf{N_C}|$ many names. Thus, due to the depth bound, the bound on the outdegree, and the upper bound on the label size, there cannot be an infinite sequence of $\mathcal{T}$-completions. Hence, $\mathit{Subs}(A_0, B_0, \mathcal{T})$ always terminates. Note that we have used both blocking conditions, **B2** and **B3**, in the proof.

Lemma 3.2
$\mathit{Subs}(A_0, B_0, \mathcal{T})$ always terminates.

Note, however, that our termination argument only yields a double-exponential bound on the run time of the algorithm. The reason is that Lemma 3.1 only shows an exponential bound on the *depth* of the generated trees, and thus only a double-exponential bound on the *size* of these trees. At the moment, it is not clear whether one can construct examples where the algorithm only terminates after an double-exponential number of steps, but we also do not have a proof that it always terminates in exponential time. Thus, we currently do not know whether the algorithm is worst-case optimal or not. However, our experimental evaluation shows that it works reasonably well in practice.

It remains to show that $\mathit{Subs}(A_0, B_0, \mathcal{T})$ always computes the correct result, that is, that it is sound and complete. The following lemma is crucial for proving this.

Lemma 3.3
Let $\mathcal{Y}_0$ be as in (2) and $\mathcal{Y}$ be a partial functional interpretation that is reachable from $\mathcal{Y}_0$ and complete, that is, $\mathcal{Y}_0 \vdash_{\mathcal{T}}{}^* \mathcal{Y}$ and $\mathsf{ic}(\mathcal{Y}, \mathcal{T}) \cap \mathsf{nb}(\mathcal{Y}) = \emptyset$. Then there is a functional model $\mathcal{I}$ of $\mathcal{T}$ such that $\mathcal{Y}(\epsilon) = \mathcal{I}(\epsilon)$.

Proof
We extend $\mathcal{Y}$ to a functional interpretation $\mathcal{I}$ such that $\mathcal{Y}(\epsilon) = \mathcal{I}(\epsilon)$. Note that, in $\mathcal{Y}$, even non-blocked nodes σ need not have r-successors for all $r \in \mathsf{N_R}$. This is the case if there is no GCI that requires generating an r-successor for σ. In the least functional model, the successor σr exists, but it has label $\emptyset$. We will represent such successors by a dummy node $d_\top$ with an empty label in our construction.

To construct $\mathcal{I}$, we first define a mapping $m : \mathsf{N_R}^* \to \mathsf{nb}(\mathcal{Y}) \cup \{d_\top\}$ by induction on the length of $\sigma \in \mathsf{N_R}^*$ as follows:

- By definition, ϵ is not blocked, and thus we can set $m(\epsilon) = \epsilon$.
- Now, consider a node σr of length > 0, and assume that $m(\sigma)$ is already defined. We distinguish two cases:
  - Assume that $m(\sigma)r \in \Delta^{\mathcal{Y}}$. Note that this node cannot be indirectly blocked since $m(\sigma)$ is then a node in $\Delta^{\mathcal{Y}}$ that is not blocked. Thus, there exists $\sigma' \in \mathsf{nb}(\mathcal{Y})$ such that $\mathcal{Y}(\sigma') = \mathcal{Y}(m(\sigma)r)$. We set $m(\sigma r) = \sigma'$.
  - If $m(\sigma)r \notin \Delta^{\mathcal{Y}}$, then we set $m(\sigma r) = d_\top$.

Based on m and $\mathcal{Y}$, we define the functional interpretation $\mathcal{I}$ by setting

$$A^{\mathcal{I}} = \{\sigma \mid m(\sigma) \in A^{\mathcal{Y}}\} \quad \text{for all } A \in \mathsf{N_C}.$$

It follows from Definition 3.2 that, for every $\sigma \in \mathsf{N_R}^*$ and every $\mathcal{FL}_0$ concept C in normal form, if $m(\sigma)$ matches C in $\mathcal{Y}$, then $\sigma \in C^{\mathcal{I}}$. In fact, assume that $m(\sigma)$ matches C. If A is a conjunct in C, then $m(\sigma) \in A^{\mathcal{Y}}$, and thus $\sigma \in A^{\mathcal{I}}$. If $\forall r.A$ is a conjunct in C, then $m(\sigma)r \in A^{\mathcal{Y}}$. This implies $m(\sigma)r \in \Delta^{\mathcal{Y}}$, and thus $m(\sigma r)$ satisfies $\mathcal{Y}(m(\sigma r)) = \mathcal{Y}(m(\sigma)r)$, which yields $m(\sigma r) \in A^{\mathcal{Y}}$, and thus $\sigma r \in A^{\mathcal{I}}$. This shows $\sigma \in (\forall r.A)^{\mathcal{I}}$.

The other direction also holds. Assume that $\sigma \in C^{\mathcal{I}}$. If A is a conjunct in C, then $\sigma \in A^{\mathcal{I}}$ implies $m(\sigma) \in A^{\mathcal{Y}}$. If $\forall r.A$ is a conjunct in C, then $\sigma \in (\forall r.A)^{\mathcal{I}}$ implies $\sigma r \in A^{\mathcal{I}}$, and thus $m(\sigma r) \in A^{\mathcal{Y}}$. Consequently, $A \in \mathcal{Y}(m(\sigma r)) = \mathcal{Y}(m(\sigma)r)$ yields $m(\sigma)r \in A^{\mathcal{Y}}$, which completes the proof that $m(\sigma)$ matches C

We are now ready to show that $\mathcal{I}$ is a model of $\mathcal{T}$, that is, for every $C \sqsubseteq D \in \mathcal{T}$ and $\sigma \in C^{\mathcal{I}}$, also $\sigma \in D^{\mathcal{I}}$ holds. Thus, assume $C \sqsubseteq D \in \mathcal{T}$ and $\sigma \in C^{\mathcal{I}}$. The latter implies that $m(\sigma)$ matches C. This is only possible if $m(\sigma) \neq d_\top$. Thus, $m(\sigma) \in \mathsf{nb}(\mathcal{Y})$ and since $\mathcal{Y}$ is complete, $m(\sigma) \notin \mathsf{ic}(\mathcal{Y})$. Consequently, $m(\sigma)$ matches D, which yields $\sigma \in D^{\mathcal{I}}$. □

Theorem 3.1
$\mathit{Subs}(A_0, B_0, \mathcal{T})$ is sound and complete, that is, it outputs "yes" iff $A_0 \sqsubseteq_{\mathcal{T}} B_0$.

Proof
Assume that the algorithm has generated a complete partial functional interpretation $\mathcal{Y}$ such that $\mathcal{Y}_0 \vdash_{\mathcal{T}}{}^* \mathcal{Y}$. Lemma 3.3 yields a model $\mathcal{I}$ of $\mathcal{T}$ such that $\mathcal{I}(\epsilon) = \mathcal{Y}(\epsilon)$.

If $\mathit{Subs}(A_0, B_0, \mathcal{T})$ outputs "no", then $B_0 \notin \mathcal{Y}(\epsilon)$. Since $A_0 \in \mathcal{Y}(\epsilon) = \mathcal{I}(\epsilon)$ and $B_0 \notin \mathcal{Y}(\epsilon) = \mathcal{I}(\epsilon)$, the model $\mathcal{I}$ of $\mathcal{T}$ yields a counterexample to the subsumption relation $A_0 \sqsubseteq_{\mathcal{T}} B_0$ because this implies $\epsilon \in A_0^{\mathcal{I}} \setminus B_0^{\mathcal{I}}$.

If $\mathit{Subs}(A_0, B_0, \mathcal{T})$ outputs "yes", then $B_0 \in \mathcal{Y}(\epsilon)$. It is easy to see that $Y(\sigma) \subseteq \mathcal{I}_{A_0,\mathcal{T}}(\sigma)$ holds for all $\sigma \in \mathsf{N_R}^*$. In fact, one can generate $\mathcal{I}_{A_0,\mathcal{T}}$ from $\mathcal{Y}_0$ by an infinite number of completion steps that also are applied to blocked nodes. Thus, whatever is added in the sequence $\mathcal{Y}_0 \vdash_{\mathcal{T}}{}^* \mathcal{Y}$ is also present in $\mathcal{I}_{A_0,\mathcal{T}}$. But then $B_0 \in \mathcal{Y}(\epsilon)$ yields $B_0 \in \mathcal{I}_{A_0,\mathcal{T}}(\epsilon)$, and this implies $A_0 \sqsubseteq_{\mathcal{T}} B_0$ by Theorem 2.1. $\square$

The algorithm $\mathit{Subs}(A_0, B_0, \mathcal{T})$ shares properties with the completion method for $\mathcal{EL}$ (Baader *et al.* 2005) as well as with tableau algorithms for expressive DLs (Baader and Sattler 2001). Every single $\mathcal{T}$-completion step extends the label set of at least one node in the tree. Intuitively, adding the concept name A to the label set of domain element σ corresponds to deriving $A_0 \sqsubseteq \forall\sigma.A$ as a consequence of $\mathcal{T}$. A single run of $\mathit{Subs}(A_0, B_0, \mathcal{T})$ not only decides whether $A_0 \sqsubseteq B_0$ is entailed by $\mathcal{T}$ but computes *all* named subsumers of A_0. This is similar to the $\mathcal{EL}$ completion method and other consequence-based calculi (Simančík *et al.* 2011). From tableau algorithms $\mathit{Subs}(A_0, B_0, \mathcal{T})$ inherits the blocking mechanism that ensures termination.

4 Horn and other fragments of $\mathcal{FL}_0$

Based on the algorithm presented in the last section, we show that subsumption between $\mathcal{FL}_0$ concepts becomes tractable if one restricts to the Horn logic Horn-$\mathcal{FL}_0$ introduced in Krötzsch *et al.* (2007). We then consider some extensions. In Horn-$\mathcal{FL}_0$, every GCI is of one of the following forms:

$$A \sqsubseteq C \quad A \sqcap B \sqsubseteq C \quad A \sqsubseteq \forall r.B, \tag{3}$$

where $A, B, C \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. Our definition differs slightly from that in Krötzsch *et al.* (2007), in that they allow $\top$ and $\bot$ to be used both in $\mathcal{FL}_0$ and Horn-$\mathcal{FL}_0$. To see that this is not a major restriction, we note that for the extension of Horn-$\mathcal{FL}_0$ that uses $\top$ and $\bot$ anywhere where a concept is used, the reduction presented in Section 2.3 can still be used to obtain a TBox fully in Horn-$\mathcal{FL}_0$ as it is presented here.

Krötzsch *et al.* (2007) only show the complexity for knowledge base consistency, which is PTime-complete in Horn-$\mathcal{FL}_0$. We improve upon these results by showing that subsumption between arbitrary $\mathcal{FL}_0$ concepts with respect to a Horn-$\mathcal{FL}_0$ TBox is tractable as well. Note that, whereas for $\mathcal{FL}_0$, subsumption between concepts can be reduced to knowledge base consistency, the restricted expressivity of Horn-$\mathcal{FL}_0$ does not allow for this in the general case.

Theorem 4.1
Concept subsumption of $\mathcal{FL}_0$ concepts with respect to general Horn-$\mathcal{FL}_0$ TBoxes is PTime-complete.

Proof
Hardness follows easily from PTime-hardness of satisfiability of propositional Horn formulae. Specifically, given a Horn formulae Φ over propositional variables $\{p_1, \ldots, p_m\}$, we

associate to each variable p_i a concept name A_i, translate clauses $p_{i_1} \wedge \ldots \wedge p_{i_m} \rightarrow p_j$ to GCIs $A_0 \sqcap A_{i_1} \sqcap \ldots \sqcap A_{i_m} \sqsubseteq A_j$, and clauses $p_{i_1} \wedge \ldots \wedge p_{i_m} \rightarrow \bot$ to $A_0 \sqcap A_{i_1} \sqcap \ldots A_{i_m} \sqsubseteq B_0$. Then, we transform these GCIs into ones with only binary conjunction on the left-hand sides by introducing auxiliary concept names. It is easy to see that the resulting TBox entails $A_0 \sqsubseteq B_0$ iff Φ is unsatisfiable.

For inclusion in PTime, we modify the procedure described in Section 3. In contrast to that procedure, we cannot reduce subsumption of the form $C \sqsubseteq_{\mathcal{T}} D$ to subsumptions of the form $A_0 \sqsubseteq_{\mathcal{T}} B_0$, since the axiom $D \sqsubseteq B_0$ need not be expressible in Horn-$\mathcal{FL}_0$. However, we can restrict ourselves to subsumptions of the form $A_0 \sqsubseteq_{\mathcal{T}} D$, where $A_0 \in \mathsf{N_C}$, as for subsumptions $C \sqsubseteq D$, we can add the axiom $A_0 \sqsubseteq C$ to the original TBox, which after normalization becomes an Horn-$\mathcal{FL}_0$ TBox $\mathcal{T}$ that entails $A_0 \sqsubseteq D$ iff the original ontology entails $C \sqsubseteq D$.

To decide $A_0 \sqsubseteq_{\mathcal{T}} D$ in polynomial time, we apply the algorithm described in Section 3 with two modifications:

1. the initial partial functional interpretation $\mathcal{Y}_0$ already contains several nodes which serve as a "skeleton" of D, and
2. expansions are only applied on nodes from that skeleton.

Specifically, for $D = \forall\sigma_1.A_1 \sqcap \ldots \sqcap \forall\sigma_n.A_n$, the initial partial functional interpretation $\mathcal{Y}_0$ is now defined as follows:

$$\Delta^{\mathcal{Y}_0} = \bigcup_{1 \leq i \leq n} \mathsf{prefix}(\sigma_i) \qquad A_0^{\mathcal{Y}_0} = \{\epsilon\} \qquad B^{\mathcal{Y}_0} = \emptyset \text{ for all } B \in \mathsf{N_C} \setminus \{A_0\}.$$

Furthermore, expansions are only applied on nodes $\sigma \in \Delta^{\mathcal{Y}_0}$, that is, new nodes may be introduced, but they are not further expanded. This restriction makes every completion sequence polynomially bounded, because we have at most one step per pair $(\alpha, \sigma) \in \mathcal{T} \times \Delta^{\mathcal{Y}_0}$. For the final interpretation $\mathcal{Z}$, we check whether $\sigma_i \in A_i^{\mathcal{Z}}$ for all $1 \leq i \leq n$, which corresponds to checking whether $\epsilon \in D^{\mathcal{Z}}$. To show that the resulting method is still sound and complete, we show that for the least functional model $\mathcal{I}_{A,\mathcal{T}}$, we have for every $\sigma \in \Delta^{\mathcal{Y}_0}$ that $\mathcal{Z}(\sigma) = I_{A,\mathcal{T}}(\sigma)$. For this, it suffices to show that, for every $d \in \mathcal{Y}^0$ and $C' \sqsubseteq D' \in \mathcal{T}$, $\sigma \in (C')^{\mathcal{Z}}$ implies $\sigma \in (D')^{\mathcal{Z}}$. Since $\mathcal{T}$ is in Horn-$\mathcal{FL}_0$, C' does not contain universal role restrictions. Consequently, if $\sigma \in \mathsf{match}(C', \mathcal{Z})$, the expansion already made sure that $\sigma \in \mathsf{match}(D', \mathcal{Z})$ and consequently that $\sigma \in (D')^{\mathcal{Z}}$. It follows that $\mathcal{Z}(\sigma) = I_{A,\mathcal{T}}(\sigma)$ for all $\sigma \in \Delta^{\mathcal{Y}_0}$. This means that $A \sqsubseteq_{\mathcal{T}} D$ iff $\epsilon \in D^{\mathcal{Z}}$. Our method runs in polynomial time and is sound and complete, and thus subsumption with Horn-$\mathcal{FL}_0$-TBoxes can be decided in polynomial time. $\square$

Remark 4.1
The proof of Theorem 4.1 uses the fact that we only need to consider role-successors of roles that occur on the left-hand side of a GCI (in case of Horn-$\mathcal{FL}_0$ there are no such roles to consider). We use this observation in an optimization of $\mathcal{FL}_o$*wer* to improve reasoning times.

For many DLs, such as $\mathcal{ALC}$ and $\mathcal{ALCI}$, it is common to define their Horn-fragments as their intersection with Horn-$\mathcal{SROIQ}$. If we define Horn-$\mathcal{FL}_\bot$ in this way, we obtain a DL in which value restrictions can occur on the left-hand side in axioms of the form

$A \sqcap \forall r.B \sqsubseteq \bot$, where $A, B \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. Specifically, in Horn-$\mathcal{FL}_\bot$, every axiom is of the form

$$A \sqsubseteq B \qquad A \sqcap B \sqsubseteq C \qquad A \sqsubseteq \forall r.A \qquad A \sqcap \forall r.B \sqsubseteq \bot, \tag{4}$$

where $A, B, C \in \mathsf{N_C} \cup \{\top, \bot\}$ and $r \in \mathsf{N_R}$.

Theorem 4.2
Subsumption between concept names is PSpace-complete for Horn-$\mathcal{FL}_\bot$.

Proof
Both directions can be shown by showing a relation to Horn-$\mathcal{FL}^-$, for which subsumption between concept names is also PSpace-complete (Krötzsch *et al.* 2007). Horn-$\mathcal{FL}^-$ is similar to Horn-$\mathcal{FL}_\bot$, but instead of axioms of the form $A \sqcap \forall r.B \sqsubseteq \bot$, it allows for axioms of the form $A \sqsubseteq \exists r$, where the semantics of $\exists r$ is defined by $(\exists r)^\mathcal{I} = \{d \mid \exists e \in \Delta^\mathcal{I}, (d,e) \in r^\mathcal{I}\}$. The Horn-$\mathcal{FL}^-$ axiom $A \sqsubseteq \exists r$ is equivalent to the Horn-$\mathcal{FL}_\bot$ axiom $A \sqcap \forall r.\bot \sqsubseteq \bot$, which means every Horn-$\mathcal{FL}^-$ ontology can be easily translated into Horn-$\mathcal{FL}_\bot$. This establishes PSpace-hardness of Horn-$\mathcal{FL}_\bot$.

For inclusion in PSpace, we show how every Horn-$\mathcal{FL}_\bot$ ontology can be translated in polynomial time into a Horn-$\mathcal{FL}^-$ ontology. For this, we replace every axiom α of the form $A \sqcap \forall r.B \sqsubseteq \bot$ by the axioms $A \sqsubseteq \exists r_\alpha$, $A \sqsubseteq \forall r_\alpha.\overline{B}$ and $B \sqcap \overline{B} \sqsubseteq \bot$, where r_α is fresh for every such axiom α. In addition, for every such fresh introduced role r_α and every axiom of the form $C \sqsubseteq \forall r.D$, we add $C \sqsubseteq \forall r_\alpha.D$. Intuitively, $A \sqcap \forall r.B \sqsubseteq \bot$ is satisfied iff every instance of A has some r-successor that does not satisfy B. As there may be several such axioms, we need to distinguish between different r-successors for each such axiom. Horn-$\mathcal{FL}^-$ is not expressive enough to do that directly, which is why we use a different role for every such axiom.

Let $\mathcal{T}$ be the TBox before this transformation and $\mathcal{T}'$ the result, and A, B be two concept names occurring in $\mathcal{T}$. We show that $\mathcal{T} \models A \sqsubseteq B$ iff $\mathcal{T}' \models A \sqsubseteq B$.

($\Rightarrow$) Assume $\mathcal{T}' \not\models A \sqsubseteq B$, which means there exists some model $\mathcal{I}'$ of $\mathcal{T}'$ s.t. $\mathcal{I}' \not\models A \sqsubseteq B$. We construct a model $\mathcal{I}$ of $\mathcal{T}$ s.t. $\mathcal{I} \not\models A \sqsubseteq B$ by setting $\Delta^\mathcal{I} = \Delta^{\mathcal{I}'}$, $A^\mathcal{I} = A^{\mathcal{I}'}$ for all $A \in \mathsf{N_C}$, and

$$r^\mathcal{I} = r^{\mathcal{I}'} \cup \bigcup_{\alpha = (A' \sqcap \forall r.B' \sqsubseteq \bot) \in \mathcal{T}} r_\alpha^{\mathcal{I}'}$$

for all $r \in \mathsf{N_R}$. For every introduced role name r_α and every axiom $A' \sqsubseteq \forall r.B' \in \mathcal{T}$, we have $\mathcal{I}' \models A' \sqsubseteq \forall r_\alpha.B'$, which yields $\mathcal{I} \models A' \sqsubseteq \forall r.B'$. Furthermore, for every $\alpha = A' \sqcap \forall r.B' \sqsubseteq \bot \in \mathcal{T}$ and $d \in (A')^\mathcal{I}$, there exists $(d,e) \in r^{\mathcal{I}'}$ s.t. $(d,e) \in r_\alpha^{\mathcal{I}'}$ and $e \in (\overline{B'})^{\mathcal{I}'}$, which implies $e \notin (B')^\mathcal{I}$ and $d \notin (\forall r.B')^\mathcal{I}$. Thus, we have show that $\mathcal{I}$ is a model of $\mathcal{T}$ and that $\mathcal{I} \not\models A \sqsubseteq B$, and thus $\mathcal{T} \not\models A \sqsubseteq B$.

($\Leftarrow$) Now let $\mathcal{I}$ be a model of $\mathcal{T}$ s.t. $\mathcal{I} \not\models A \sqsubseteq B$. Based on $\mathcal{I}$, we construct a model $\mathcal{I}'$ of $\mathcal{T}'$ s.t. $\mathcal{I}' \not\models A \sqsubseteq B$. For every $\alpha = A \sqcap \forall r.B \sqsubseteq \bot \in \mathcal{T}$ and $d \in A^\mathcal{I}$, there exists some $e \in \Delta^\mathcal{I}$ s.t. $(d,e) \in r^\mathcal{I}$ and $e \notin B^\mathcal{I}$. The interpretation $r_\alpha^{\mathcal{I}'}$ of the role r_α is defined as the set of all those pairs (d,e). All other concept and role names are interpreted as in $\mathcal{I}$. The resulting interpretation $\mathcal{I}'$ satisfies all axioms in $\mathcal{T}'$ and thus $\mathcal{T}' \not\models A \sqsubseteq B$.

Summing up, we have shown that $\mathcal{T} \not\models A \sqsubseteq B$ iff $\mathcal{T}' \not\models A \sqsubseteq B$, and thus that subsumption between concept names in Horn-$\mathcal{FL}_\bot$ can be polynomially reduced to subsumption between concept names in Horn-$\mathcal{FL}^-$. □

We have used a modification of the algorithm presented in Section 3 to show that subsumption in Horn-$\mathcal{FL}_0$ is PTime-complete, thus indicating optimality of our algorithm for this fragment. To deal with $\bot$, we could try to employ the reduction presented in Section 2.3, which introduces a concept name for $\bot$. Unfortunately, this approach cannot work for Horn-$\mathcal{FL}_\bot$. In fact, if we generalized axioms of the form $A \sqcap \forall r.B \sqsubseteq \bot$ to ones that use a concept name instead of $\bot$, we would have to allow axioms of the form $A \sqcap \forall r.B \sqsubseteq C$. This makes the logic powerful enough to cover the whole language of $\mathcal{FL}_0$, as we can represent axioms of the form $A \sqcap \forall r.B_1 \sqcap \forall s.B_2 \sqsubseteq C$ using $A \sqcap \forall r.B_1 \sqsubseteq D$ and $D \sqcap \forall r.B_2 \sqsubseteq C$, and axioms of the form $A \sqcap \forall r.B \sqsubseteq \forall s.C$ using $A \sqcap \forall r.B \sqsubseteq D$, $D \sqsubseteq \forall r.C$, where in each case, D is fresh. In fact, already allowing more than one value restriction on the left-hand increases the complexity.

If we further relax Horn-$\mathcal{FL}_\bot$ to allow several value restrictions on the left-hand side, the logic becomes again ExpTime-complete. In Horn-$\mathcal{FL}_\bot^+$, axioms are of the forms listed in (4) and the following form:

$$\forall \sigma_1.A_1 \sqcap \ldots \sqcap \forall \sigma_n.A_n \sqsubseteq \bot, \tag{5}$$

where for $1 \leq i \leq n$, $\sigma_i \in \mathsf{N_R}^*$ and $A_i \in \mathsf{N_C}$.

Hardness of Horn-$\mathcal{FL}_\bot^+$ can be shown based on the reduction used in the proof for Proposition 1 in Baader and Théron (2020) employed to show ExpTime-hardness of $\mathcal{FL}_0$. The reduction uses a TBox that is not in Horn-$\mathcal{FL}_\bot^+$ and does not even contain $\bot$. However, it uses a special concept name F which essentially mimics the behavior of $\bot$. Replacing F by $\bot$ creates a Horn-$\mathcal{FL}_\bot^+$ TBox with a similar behavior. Specifically, F occurs on the right-hand side of the subsumption test, in axioms of the form $A \sqcap B \sqcap \forall w_1.F \sqcap \ldots \sqcap w_n.F \sqsubseteq F$ (Axiom 2), $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \forall r.F$ (Axiom 7) and in axioms $F \sqsubseteq \forall r.F$, which are added for every role name r used in the reduction (Axioms 8 and 9). All other axioms are in Horn-$\mathcal{FL}_0$. Thus, replacing F by $\bot$ results in a TBox of the desired form. We argue that in the resulting TBox, $C \sqsubseteq \bot$ is entailed iff $C \sqsubseteq F$ is entailed in the original TBox, where C does not contain F. If $C \sqsubseteq F$ is entailed by the original TBox, clearly $C \sqsubseteq \bot$ is entailed by the transformed.

For the other direction, assume that $C \sqsubseteq F$ is not entailed by the original ontology, and let $\mathcal{I}$ be a witnessing model with $d \in C^\mathcal{I} \setminus F^\mathcal{I}$ such that every domain element is reachable by a path of role-successors from d. We transform $\mathcal{I}$ into $\mathcal{I}'$ by removing all elements in $F^\mathcal{I}$. Since $\mathcal{I} \models F \sqsubseteq \forall r.F$ for all $r \in \mathsf{N_R}$, we have for all domain elements $e \in \Delta^{\mathcal{I}'}$ and words $w \in \mathsf{N_R}^*$, $e \in (\forall w.F)^\mathcal{I}$ iff $e \in (\forall w.\bot)^\mathcal{I}$. It follows that for every axiom of the form $A \sqcap B \sqcap \forall w_1.F \sqcap \ldots \sqcap \forall w_n.F \sqsubseteq F$ in $\mathcal{T}$, $\mathcal{I}' \models A \sqcap \forall w_1.\bot \sqcap \ldots \sqcap \forall w_n.\bot \sqsubseteq \bot$, and for every axiom of the form $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \forall r.F$, $\mathcal{I}' \models A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \forall r.\bot$. The axioms $\bot \sqsubseteq \forall r.\bot \in \mathcal{T}$ are naturally entailed. None of the remaining axioms have value restrictions on the left-hand side, and are thus also entailed by $\mathcal{I}'$. Consequently, $\mathcal{I}'$ is a model of the transformed TBox.

Thus, we have shown that the reduction used in Baader and Théron (2020) to show ExpTime-hardness of $\mathcal{FL}_0$ can be adapted to show ExpTime-hardness of Horn-$\mathcal{FL}_\bot^+$.

Theorem 4.3
Deciding subsumption in Horn-$\mathcal{FL}_\bot^+$ is ExpTime-complete.

5 A Rete-based implementation

Our implementation of $Subs(A_0, B_0, \mathcal{T})$ in $\mathcal{FL}_o wer$ employs a variant of the algorithm for Rete networks (Forgy 1982) to allow for a fast generation of completions of the partial model to be constructed. Specifically, the Rete network tests on all domain elements satisfaction of all GCIs at the same time. It stores also partial matches so that they can be quickly continued once additional information is available. In addition, $\mathcal{FL}_o wer$ uses optimized data structures to allow for a fast and memory-efficient navigation in the current model, as well as to speed-up the implementation of blocking.

5.1 Rete network for the TBox to speed-up matching of GCIs

In order to compute a sequence of $\mathcal{T}$-completions $\mathcal{Y}_0 \vdash_\mathcal{T} \mathcal{Y}_1 \vdash_\mathcal{T} \mathcal{Y}_2 \vdash_\mathcal{T} \cdots$ starting from the initial partial functional interpretation $\mathcal{Y}_0$, we can employ a GCI $C \sqsubseteq D \in \mathcal{T}$ like a rule of the form

$$?\sigma \in \mathsf{match}(C, \mathcal{Y}_i) \to \; ?\sigma \in \mathsf{match}(D, \mathcal{Y}_i),$$

where $?\sigma$ ranges over the non-blocked domain elements of $\mathcal{Y}_i$, to obtain the next expansion. Overall, the rules corresponding to the GCIs from the TBox are applied during a run of $Subs(A_0, B_0, \mathcal{T})$ in a forward-chaining manner to yield the sequence of $\mathcal{T}$-completions.

In each expansion step i, one has to compute the elements that violate a GCI, i.e., the pairs $(\sigma, C \sqsubseteq D) \in \big(\Delta^{\mathcal{Y}_i} \cap \mathsf{nb}(\mathcal{Y}_i)\big) \times \mathcal{T}$ such that σ matches C but not D in $\mathcal{Y}_i$. Since there is potentially a large number of elements in $\Delta^{\mathcal{Y}_i}$ that has to be matched against a large number of left-hand sides of GCIs (patterns) in the TBox in each step, we have chosen to implement this task using the Rete algorithm for many pattern/many object matching (Forgy 1982), which is tailored to efficiently compute forward chaining rule applications. The general idea is to integrate the matching tests of all GCIs using a Rete network, which is in our case a compressed network-representation of the TBox. In each completion step, the extension of a tree $\mathcal{Y}_i$ only affects a small number of its elements: the matching element σ itself and/or its children. This makes the Rete-based algorithm particularly efficient in our setting, because it stores matching information across completion steps to avoid reiterating over the whole set of pairs $\big(\Delta^{\mathcal{Y}_i} \cap \mathsf{nb}(\mathcal{Y}_i)\big) \times \mathcal{T}$ in each step. Only those elements with changes have to be re-matched again in the next completion step.

For a given element, the network tests which left-hand sides of a GCI are matched and triggers the extension for the corresponding right-hand side. This Rete network corresponds to a graph using three kinds of nodes: a single root node, a set of intermediate nodes and a set of terminal nodes. Intuitively, the *intermediate nodes* check for matches of parts of the left-hand side of a GCI, while the *terminal nodes* hold the right-hand side of a GCI that is ready to be applied to an element. To process an element $\sigma \in \Delta^{\mathcal{Y}}$, a set of so-called tokens is passed from the root node through the intermediate nodes to the terminal nodes. Such a *token* is a pair of the form $(\sigma, r) \in \big(\mathsf{N_R}^*, \mathsf{N_R} \cup \{\epsilon\}\big)$. Intuitively, the token (σ, ϵ) is used to check whether σ matches the concept names on the left-hand side of a GCI, while a token of the form (σ, r) with $r \in \mathsf{N_R}$ is used to check whether σ matches value restrictions with the role name r.

There are the following three types of *intermediate nodes* that process tokens arriving from predecessor nodes in the network:

- A *concept node* is labeled with a concept name $B \in \mathsf{N_C}$ and sends an incoming token (σ, s) to all successor nodes iff $\sigma s \in B^{\mathcal{Y}_i}$.
- A *role node* is labeled with an $s \in \mathsf{N_R} \cup \{\epsilon\}$. An arriving token of the form (σ, s') is handled as follows. If $s \in \mathsf{N_R}$ and $s' = s$, then it sends (σ, s') to all successor nodes. If $s = \epsilon$, then it sends the token $(\sigma s', \epsilon)$ to all successor nodes.
- An *inter-element node* is labeled with a tuple $(s_1, \ldots, s_m) \in (\mathsf{N_R} \cup \{\epsilon\})^m$. It stores all arriving tokens and sends a token (σ, ϵ) to its successor nodes once *all* tokens of the form $(\sigma, s_1), \ldots, (\sigma, s_m)$ have arrived at this node.

The overall network is structured in layers. The root node with no incoming edges is on top. All successors of the root node are concept nodes. The root node takes an element of the form $\sigma = \rho r \in {\mathsf{N_R}}^*$ and sends the token (ρ, r) to all successor nodes. A successor of a concept node can only be another concept node or a role node. A role node leads directly to an inter-element node and inter-element nodes lead to terminal nodes. Intuitively, paths of concept nodes corresponds to conjunctions of concept names a token must satisfy in order to pass through them. These concept names either need to be matched on the current element or on its immediate role successors. If the path of concept names goes into a role node labeled with ϵ, this corresponds to a match on the current element. If it goes into a role node labeled with a role name r, this corresponds to a match on its r-successors. The inter-element nodes again correspond to a conjunction that combine the successful matches of the different role-successors.

Example 5.1
As an example of the structure of a Rete network compiled from a TBox, consider the following normalized TBox:

$$\begin{aligned}\mathcal{T}_{ex} = \{\ A_2 \sqcap A_4 \sqcap A_5 \sqcap \forall r_1.A_3 \sqcap \forall r_1.A_4 \sqcap \forall r_2.A_1 &\sqsubseteq B_7, \\ \forall r_2.A_3 \sqcap \forall r_2.A_4 &\sqsubseteq B_8, \\ \forall r_1.A_6 &\sqsubseteq \forall r_1.B_9\ \}.\end{aligned}$$

The corresponding Rete network is displayed in Figure 2 with the root node (Layer 1) and the three leaves being terminal nodes representing the left-hand sides of the three GCIs (Layer 5). The intermediate nodes are concept nodes representing (conjunctions of) named concepts (Layer 2), role nodes (Layer 3) or the inter-element node representing the conjunction of value restrictions for different roles from the first GCI in $\mathcal{T}_{ex}$ (Layer 4).

In the preprocessing phase, $\mathcal{FL}_o$*wer* compiles the normalized TBox $\mathcal{T}$ into the corresponding *Rete network*. In the main reasoning phase, $\mathcal{FL}_o$*wer* saturates the initial partial functional interpretation $\mathcal{Y}_0$ by the Rete algorithm. To unleash the full potential of this Rete-based approach, we need to store the current model in a way that allows for fast access of its successor nodes, which is discussed in the next subsection.

5.2 Numerical representation of partial functional interpretations

The operations $\mathcal{FL}_o$*wer* needs to perform repeatedly on the current partial functional interpretation $\mathcal{Y}_i$ for a given domain element are the following:

1. quickly access its direct successors when a GCI is applied,
2. quickly decide whether a smaller domain element with the same label set exists to test Condition **B2** for direct blocking, and

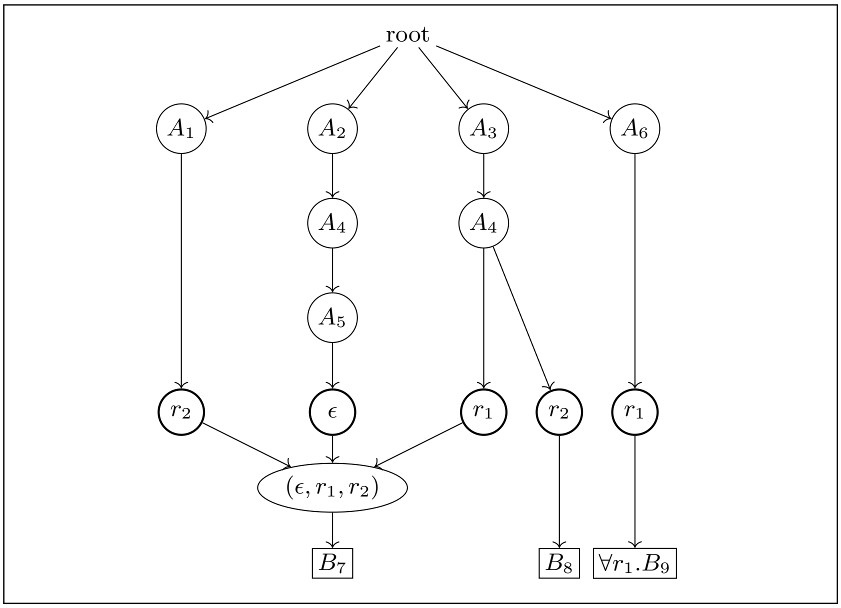


Fig. 2. Rete network for the TBox $\mathcal{T}_{ex}$ from Example 5.1. Intermediate nodes are drawn in round shapes: concept nodes in light circles, role nodes in dark circles, and the inter-element node in an ellipsis. Terminal nodes are in displayed as boxes.

3. quickly decide whether a domain element is an ancestor of another element to test Condition **B3** for indirect blocking.

To obtain a space-efficient representation of the partial functional interpretation that supports these operations with minimal overhead, we use an integer-based representation with the basis $|\mathsf{N_R}| + 1$. Specifically, we fix an enumeration of the role names in $\mathcal{T}$: $\mathsf{N_R} = \{r_1, \dots, r_n\}$. A word $\sigma = r_{i_1} r_{i_2} \dots r_{i_m} \in {\mathsf{N_R}}^*$ is then represented as

$$\mathsf{index}(\sigma) = \sum_{1 \le j \le m} i_j (n+1)^{m-j}.$$

This representation reduces various operations on words that are relevant for the algorithm to fast arithmetic operations in the following ways:

1. the length of σ is $\left|\sigma\right| = \left\lfloor \log_{n+1}(\mathsf{index}(\sigma)) \right\rfloor$,
2. the r_i-successor of σ has index: $(n+1) \cdot \mathsf{index}(\sigma)$,
3. the direct predecessor of σ has index $\left\lfloor \frac{\mathsf{index}(\sigma)}{n+1} \right\rfloor$, and
4. checking whether ρ is an ancestor of σ, that is whether $\sigma \in \mathsf{pprefix}(\rho)$, can be done by checking whether
$$\mathsf{index}(\sigma) = \left\lfloor \frac{\mathsf{index}(\rho)}{(n+1)^{(|\rho| - |\sigma|)}} \right\rfloor.$$

Note that this numerical encoding also directly provides an *ordering on elements* $\prec$ as required: specifically, we define this ordering by $\sigma \prec \rho$ iff $\mathsf{index}(\sigma) < \mathsf{index}(\rho)$.

The labels of each domain element are stored in a tree map, which is a data structure that associates each index with a non-empty label to its label set. The inverse of this

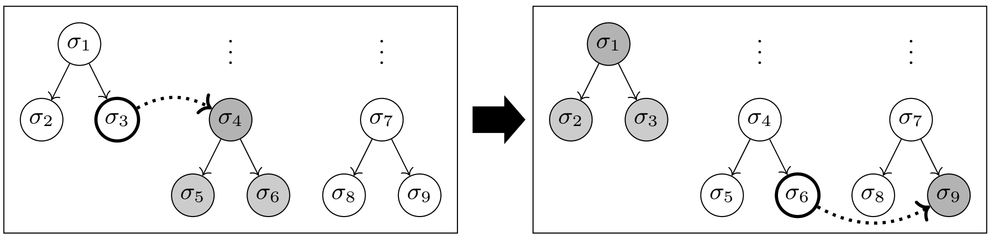


Fig. 3. Example of blocking interactions in a partial functional interpretation. Gray elements are blocked, and the dotted arrow indicates an element directly blocking another.

map is also stored, to quickly obtain which domain elements have a given label set. This operation is required to test the blocking Condition **B2**.

5.3 Implementation of blocking

After each expansion step it needs to be tested whether the blocking conditions **B1** to **B3** are fulfilled for the elements of the partial functional interpretation. Unfortunately, there can be intricate interactions between the blocking statuses of different elements. Although GCIs are only applied on elements that are not blocked, the labels of a blocked element can change if a GCI is applied on some of its predecessors. As elements that are themselves blocked cannot block other elements, such a change in the label of a blocked element can lead to chain-reactions where the blocking status of a number of elements changes once information is propagated into a single element. An example of this effect is visualized in Figure 3. On the left-hand side, σ_3 blocks σ_4, which makes the nodes σ_5 and σ_6 indirectly blocked. Thus, these nodes cannot block other nodes themselves. In our example, we assume σ_6 and σ_9 to have the same labels. σ_9 is not blocked by σ_6, since σ_6 is blocked. Blocking σ_9 would thus make the overall reasoning procedure incomplete. Now imagine some extension makes the node σ_1 blocked. The resulting situation is shown on the right-hand side. Since σ_3 becomes indirectly blocked, it cannot block σ_4 anymore. Consequently, also the descendants of σ_4 become unblocked, and now σ_9 becomes blocked by σ_6, even though there is no connection between these nodes and σ_1.

To determine directly blocked nodes, $\mathcal{FL}_o$*wer* uses two hash maps. One is mapping each node to its label set, and the other one is mapping each label set to a node. In addition, $\mathcal{FL}_o$*wer* stores for each node whether it is blocking another node, directly blocked, or indirectly blocked. If the label set of a node changes, $\mathcal{FL}_o$*wer* determines via the hash maps whether this change results in directly blocking or unblocking any nodes, and updates their blocking status accordingly. For every node whose blocking status changes, the indirect blocking status of their successors is recursively updated. If the indirect blocking status changes (as seen for σ_6 in the last example), $\mathcal{FL}_o$*wer* checks via the hash maps whether the blocking status of other nodes has to change as well, and invokes those changes. This process is continued recursively until all affected blocking statuses have been updated.

6 Evaluation of the $\mathcal{FL}_o$*wer* reasoner

The $\mathcal{FL}_o$*wer* reasoner is implemented in Java. It takes as input a general $\mathcal{FL}_\bot$ TBox $\mathcal{T}$ in OWL format (Cuenca Grau *et al.* 2008) and normalizes the input TBox. If the ontology uses $\top$ or $\bot$, the transformation rules from Section 2.3 are applied. $\mathcal{FL}_o$*wer* realizes the following reasoning tasks.

Subsumption: Given two OWL classes A and B, decide whether $A \sqsubseteq_\mathcal{T} B$ holds.

Subsumer set: Given an OWL class A, compute all classes B in $\mathcal{T}$ for which $A \sqsubseteq_\mathcal{T} B$ holds.

Classification: Decide for all pairs of named OWL classes A and B occurring in $\mathcal{T}$ whether the subsumption $A \sqsubseteq_\mathcal{T} B$ holds.

To decide subsumption $\mathcal{FL}_o$*wer* runs $\mathit{Subs}(A_0, B_0, \mathcal{T})$, but stops as soon as the subsumer candidate B_0 occurs at the root of the tree. For computing the whole subsumer set of A_0, a single complete run of $\mathit{Subs}(A_0, B_0, \mathcal{T})$ is sufficient, where the choice of B_0 is actually irrelevant. All subsumers of A_0 can be found at the root of the final tree. Classification is done by running $\mathit{Subs}(A_0, *, \mathcal{T})$ for each named class A_0 in $\mathcal{T}$ separately. (Again, the used subsumer candidate B_0 is irrelevant for this kind of reasoning task and can be replaced by any concept other than A_0 or $\top$. This is indicated here by the wildcard $*$.) The Rete network for $\mathcal{T}$ is created only once and is reused for the remaining runs of $\mathit{Subs}(*, *, \mathcal{T})$ during classification. Furthermore, $\mathcal{FL}_o$*wer* uses caching to reuse precomputed subsumer sets.

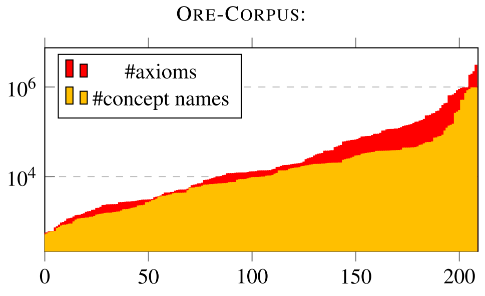
Our evaluation of $\mathcal{FL}_o$*wer* investigates two aspects. First, we wanted to see which optimizations implemented in $\mathcal{FL}_o$*wer* turned out to be effective and, second, we wanted to see how $\mathcal{FL}_o$*wer*'s performance compares to other state-of-the-art DL reasoner. As most other DL reasoners that can handle (extensions of) $\mathcal{FL}_0$ implement tableau-based methods, such a comparison would also tell us whether our new approach based on least functional models is competitive in terms of performance. We report on both kinds of evaluations in this section. In order to be able to assess the performance of $\mathcal{FL}_o$*wer*, we needed to find suitable test ontologies first.

6.1 Test data

We generated two corpora for our evaluation. The first corpus Ore-Corpus is based on the ontologies of the OWL EL classification track from the OWL Reasoner Evaluation 2015 (ORE 2015) (see Parsia *et al.* 2017). The benchmarks of the ORE 2015 have the advantage that they have been balanced according to different criteria such as size, expressivity and complexity, and consist of many application ontologies. However, unfortunately no track is dedicated to ontologies in $\mathcal{FL}_\bot$. We thus generated $\mathcal{FL}_\bot$ ontologies from ontologies written in $\mathcal{EL}$ by "flipping" the quantifier, that is, by replacing $\exists$ by $\forall$. We furthermore dropped axioms involving role inclusions, nominals, or other operators that cannot be expressed in $\mathcal{FL}_\bot$. From the resulting corpus, we removed all ontologies with less than 500 concept names, resulting in a set of 209 $\mathcal{FL}_\bot$ ontologies.[5]

While the ontologies used by the ORE do not contain a lot of $\mathcal{FL}_\bot$ axioms, we found larger usage of them in the Manchester Ontology Corpus (MOWLCorp), which is a large ontology corpus containing 34,741 OWL ontologies that were obtained by web-crawling (Matentzoglu *et al.* 2013). The second corpus, Mowl-Corpus, is based on

[5] In the initial study on $\mathcal{FL}_o$*wer*'s performance (Michel *et al.* 2019) was an undetected bug which lead to more ontologies being discarded. There we used only 159 ontologies.



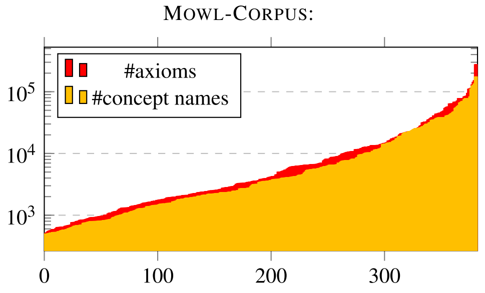


Fig. 4. Numbers of axioms and concept names in the ontologies under consideration. The y-axis shows in logarithmic scale the number of axioms and concept names of the respective ontology, for which we ordered the values along the x-axis.

MOWLCorp. From each ontology in MOWLCorp, we removed axioms that could not be expressed in $\mathcal{FL}_\bot$. If the resulting ontology contained at least 500 concept names, it was included in our corpus. This resulted in a set of 382 $\mathcal{FL}_\bot$ ontologies. While the Ore-Corpus contains more complex axioms and a more balanced set of ontologies, the Mowl-Corpus contains axioms that were obtained from application ontologies without modifications and thus preserves the original way of modeling.

Figure 4 shows the distribution of different parameters in the two corpora: number of concept names and number of axioms. The largest ontology in Ore-Corpus has 3,137,899 axioms, while the largest ontology in Mowl-Corpus has 279,682 axioms.

In order to evaluate the subsumption task, we generated 80 individual subsumption tests per ontology for the Ore-Corpus, composed of 40 tests with *positive* and 40 tests with *negative* outcome. Since this resulted in a large number of reasoning experiments to be performed, this experiment was only performed on Ore-Corpus, for which we assumed the most insights due to its more varied nature compared to Mowl-Corpus.

Positive tests were generated by randomly selecting a concept name, and then randomly selecting a subsumer of it. Negative tests were generated by randomly selecting a concept name, and then randomly selecting another concept name that does not subsume the first. For $\mathcal{FL}_o$*wer*, positive subsumption tests are easier, as the reasoner stops as soon as the subsumption relation has been proven. For tableau-based reasoning systems, the expected behavior is the other way around, as these reasoners try to create a counter-example to contradict the subsumption to be tested. Thus, evaluation results might not be as informative if one would just generate pairs for the subsumption test randomly without distinguishing between positive and negative tests as it was done in the earlier study (Michel *et al.* 2019).

6.2 Evaluation setup

In the initial study on $\mathcal{FL}_o$*wer*, we compared the performance for all three implemented reasoning tasks *Subsumption*, *Subsumer Set* and *Classification* (Michel *et al.* 2019). Although the OWL API supports computing subsumer sets and is implemented by all considered reasoner systems, it would not yield an informative test, since all three tableaux-based systems classify the entire ontology before returning the subsumer set, as inspection of their source code revealed. This makes a comparison simply unfair and less insightful, which is why we restricted this study to the tasks *Subsumption* and *Classification*.

For subsumption tests, we used the Ore-Corpus and the concept pairs for positive and negative subsumptions. For each subsumption test, the timeout was set to 1 min. Both Ore-Corpus and Mowl-Corpus were used to evaluate the classification task. For each classification reasoning task, the timeout was set to 10 min.

In addition to the running times measured for the two reasoning tasks, we also compared the computed results. While this comparison is easy for the *Subsumption* tests, for *Classification*, we computed a checksum for the classification result, and checked whether it was the same for every reasoner.

As a test system we have used an Intel Core i5-4590 CPU machine with 3.30GHz and 32 GB RAM, using Debian/GNU Linux 9 and OpenJDK 11.0.5. Java was called with *-Xmx8g* to set the maximum allocation pool (heap) size to 8 GB. We only measured the running time of the actual reasoning task and not the time for loading the ontology.

6.3 Evaluating $\mathcal{FL}_o$wer's optimizations

Although the current version of $\mathcal{FL}_o$*wer* is certainly not highly optimized, it implements several optimizations. We first evaluated the effect of the different optimizations within $\mathcal{FL}_o$*wer*:

Multithreading The main algorithm computes all subsumers for a given concept name. For the task of *classification*, we partition the concept names into batches of size 48 plus one partition for the rest, and compute the subsumers for each partition in a different thread.

Ancestor blocking Ancestor blocking corresponds to the blocking condition **B3**. While the method might not terminate without this condition, it is the interaction of this blocking condition with **B2** that makes the implementation of blocking more challenging (see Section 5.3). To assess the impact of this blocking condition, we allowed to deactivate ancestor blocking in the implementation.

Role filtering We do not generate r-successors for roles $r \in \mathsf{N_R}$ that do not occur on the left-hand side of a GCI. As pointed out in Remark 4.1, this optimization preserves soundness and completeness. Moreover, as shown in the proof for Theorem 4.1, reasoning in Horn-$\mathcal{FL}_0$ becomes polynomial with this optimization, so that one would expect a big impact of this optimization.

Global caching When performing classification, we store previously computed subsumer sets. If a node with a concept name is added for which we already have a subsumer set, we add all the subsumers to that node and block it. The node only becomes unblocked when new concept names are added to its label by subsequent reasoning steps.

	$\mathcal{FL}_o wer$	$\mathcal{FL}_o wer$-MT	$\mathcal{FL}_o wer$-MT-AB	$\mathcal{FL}_o wer$-MT-AB-GC	$\mathcal{FL}_o wer$-ALL
Timeouts:	0.40%	0.39%	0.00%	0.00%	0.00%

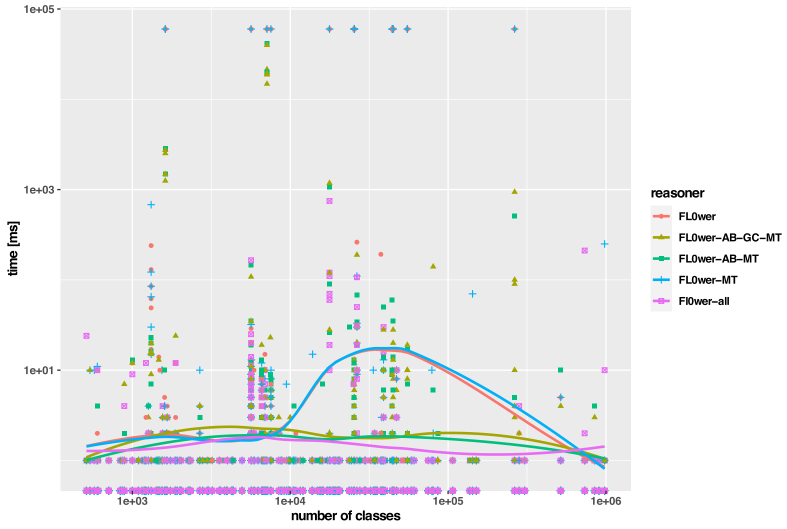


Fig. 5. Timeouts and running times for subsumption tests w.r.t. Ore-Corpus with the different optimizations.

We compared the following configurations of our reasoner:

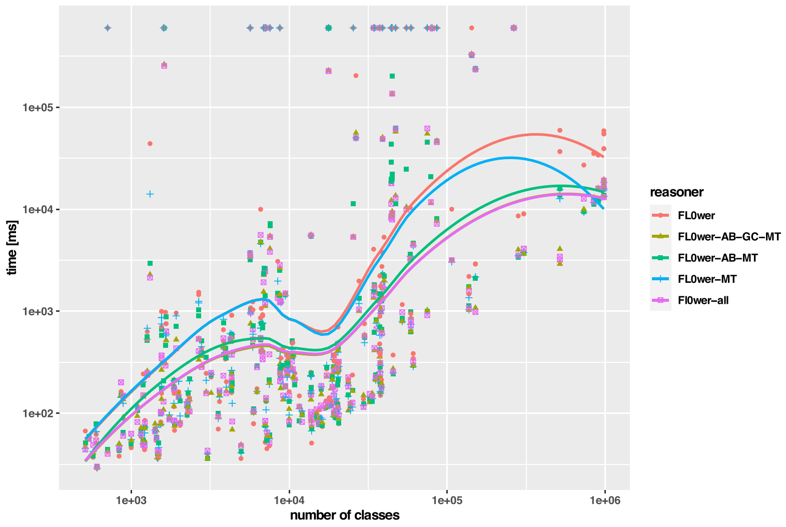
- $\mathcal{FL}_o wer$ with no optimizations,
- $\mathcal{FL}_o wer$-MT (multithreading activated),
- $\mathcal{FL}_o wer$-MT-AB (multithreading and ancestor blocking),
- $\mathcal{FL}_o wer$-MT-AB-GC (multithreading, ancestor blocking and global caching), and
- $\mathcal{FL}_o wer$-ALL with all four optimizations activated.

Figure 5 shows the results for the subsumption experiment, while Figure 6 shows the results for the classification experiments. Here and in the figures that follow, we use logarithmic scaling on both axes, and we show for the runs that caused a timeout the maximal value (1 minute for subsumption, and 10 min for classification).

For the subsumption tests, the biggest impact was caused by ancestor-blocking, despite the additional obstacles in the implementation. On the other hand, considering that termination can only be guaranteed with ancestor-blocking activated, and that the additional blocking condition may lead to fewer nodes being generated, a positive effect was to be expected. In fact, for Ore-Corpus, with ancestor blocking activated, the timeout rate dropped from 17.91% to 4.10%. However, this positive effect was only notable for the ontologies in Ore-Corpus, which can be explained by the simpler structure of the ontologies in Mowl-Corpus which in turn lead to simpler functional models. For a single subsumption task, the other optimizations merely seem to create an overhead and do not improve the performance in general. This is obvious for the caching procedure, which only brings a benefit if more than one subsumption task is performed. The largest impact here seems to be obtained by the role filtering. Interestingly, $\mathcal{FL}_o wer$'s reasoning time seems hardly correlated with the number of classes in the ontology – only if this number becomes very large, optimizations seem to have even a negative impact.

Ore-Corpus:

	$\mathcal{FL}_o wer$	$\mathcal{FL}_o wer$-MT	$\mathcal{FL}_o wer$-MT-AB	$\mathcal{FL}_o wer$-MT-AB-GC	$\mathcal{FL}_o wer$-ALL
Timeouts:	18.28%	17.91%	4.10%	2.99%	2.61%



Mowl-Corpus:

	$\mathcal{FL}_o wer$	$\mathcal{FL}_o wer$-MT	$\mathcal{FL}_o wer$-MT-AB	$\mathcal{FL}_o wer$-MT-AB-GC	$\mathcal{FL}_o wer$-ALL
Timeouts:	0.00%	0.00%	0.00%	0.00%	0.00%

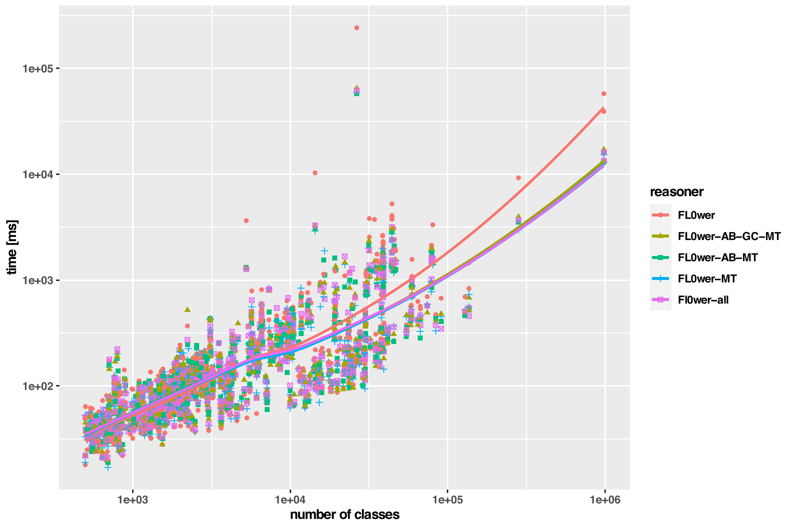


Fig. 6. Timeouts and running times for classification with the different optimizations. Note that the curve for $\mathcal{FL}_o wer$-MT-AB-GC is almost completely hidden under the curve for $\mathcal{FL}_o wer$-ALL.

For classification computed on the Ore-Corpus, besides ancestor blocking, global caching makes a noticeable impact, though it is not as large as one would expect for this task. In contrast, the impact of role filtering is not as strong, though it decreases the

	HermiT	JFact	Openllet	$\mathcal{FL}_o$*wer*-ALL
Timeouts:	0.001%	5.57%	1.53%	0.00%

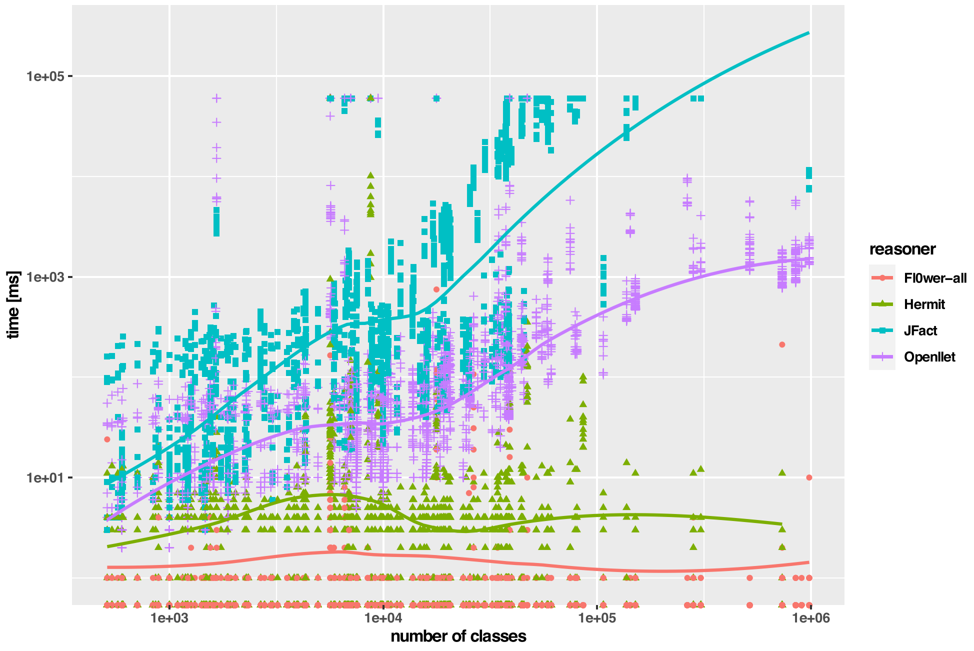


Fig. 7. Timeouts and running times for subsumption tests w.r.t. ORE-CORPUS for the different reasoners.

number of timeouts from 2.99% to 2.61%. We also observe that for MOWL-CORPUS, none of the optimizations apart from multithreading seem to be really indispensable. Again this can be explained by the simpler structure of the ontologies considered here.

6.4 Comparison with other DL reasoners

We evaluated $\mathcal{FL}_o$*wer* to see how its reasoning times compare with those of other state-of-the-art DL reasoners. We used the configuration of our DL reasoner with all optimizations active: $\mathcal{FL}_o$*wer*-ALL. Since there is no other dedicated reasoner for $\mathcal{FL}_0$, we used reasoner systems that can handle expressive DLs of which $\mathcal{FL}_0$ is a fragment. Here, we focused on reasoners which are implemented in Java just as $\mathcal{FL}_o$*wer*, and selected the following three state-of-the-art reasoning systems:

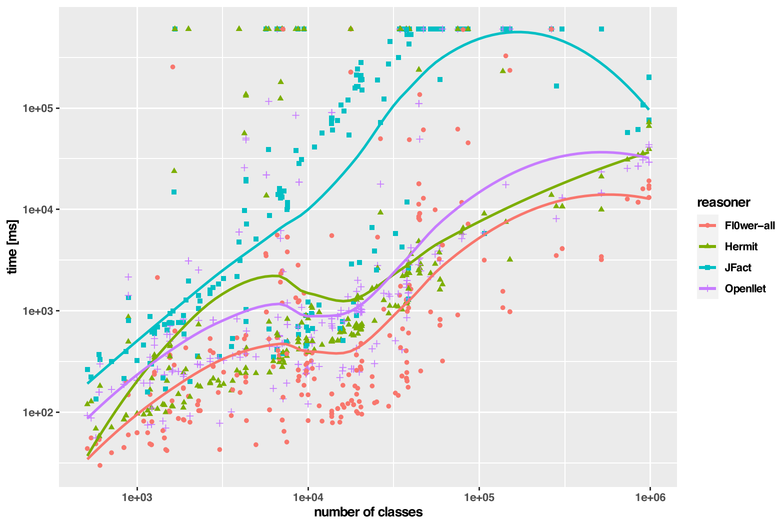
- HermiT,[6] version 1.3.8.510,
- Openllet,[7] version 2.6.3, and
- JFact,[8] version 5.0.1.

All three reasoners implement the OWL API (Horridge and Bechhofer 2011), which allows us to measure and compare the time needed for the reasoning tasks alone – excluding

[6] hermit-reasoner.com.
[7] github.com/Galigator/openllet.
[8] jfact.sourceforge.net.

ORE-CORPUS:

	HermiT	JFact	Openllet	$\mathcal{FL}_o wer$-ALL
Timeouts:	9.50%	21.01%	4.78%	2.61%



MOWL-CORPUS:

	HermiT	JFact	Openllet	$\mathcal{FL}_o wer$-ALL
Timeouts:	0.00%	2.19%	0.00%	0.00%

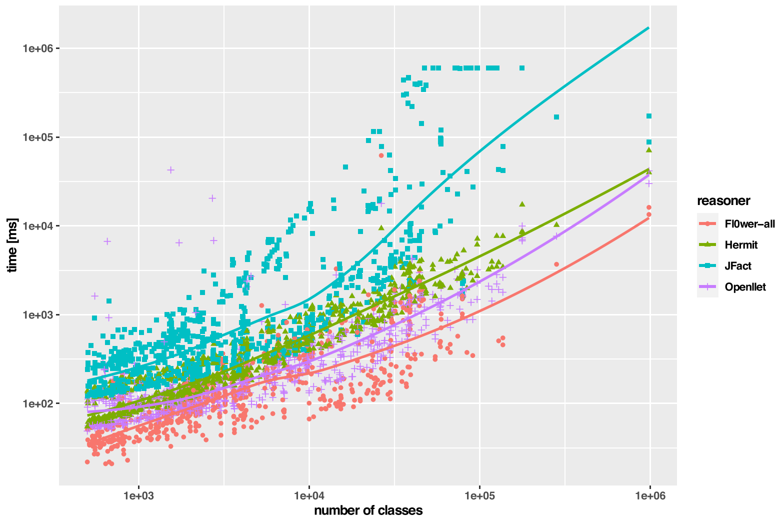


Fig. 8. Timeouts and running times for classification with the different reasoners.

the time for loading the ontologies using the OWL API. Note that furthermore, all three reasoners implement tableaux-based algorithms, so that this comparative evaluation also serves as a comparison of the different approaches: least functional model generation vs. tableaux-based approach. Regarding the actual reasoner implementations, we note that these are all complex and mature reasoning *systems* that come with more sophisticated optimizations than $\mathcal{FL}_o wer$, which makes it even more surprising that $\mathcal{FL}_o wer$ performs quite well in comparison. The timeouts and reasoning times of $\mathcal{FL}_o wer$ compared with those of the above three reasoners, are shown in Figure 7 for the subsumption experiment, and in Figure 8 for the two classification experiments.

Interestingly, for the subsumption tests, the performance of both $\mathcal{FL}_o wer$ and HermiT seems hardly affected by the number of concept names in the ontology. This number has a much bigger impact for JFact and Openllet which need orders of magnitude more running time than $\mathcal{FL}_o wer$ and HermiT to decide subsumption for the test cases. However, while some subsumption tasks still lead to timeouts for HermiT in 0.0047% of cases, no timeouts were observed by $\mathcal{FL}_o wer$. Generally, $\mathcal{FL}_o wer$ performs substantially better for this task (on this test set) than the other DL reasoners.

For classification on Ore-Corpus, our measurements indicate that $\mathcal{FL}_o wer$ performs the best among the four systems. JFact's running time is roughly an order of magnitude higher than the one of $\mathcal{FL}_o wer$. HermiT has twice as many timeouts as Openllet, but the picture on running times is more mixed, where HermiT often performed better than Openllet. $\mathcal{FL}_o wer$ again almost halved the number of timeouts compared to Openllet, but here, the performance looks consistently better than for all other reasoners. Interestingly, the (interpolated) performance curves of $\mathcal{FL}_o wer$, HermiT, and Openllet show very similar characteristics, as they develop almost synchronously. This may suggest that the same kind of ontology is difficult for all three systems and for both reasoning approaches. For Mowl-Corpus, the general picture is in principle similar. We can see a clear ranking between the reasoners, with $\mathcal{FL}_o wer$ performing generally the best. For this corpus there were only timeouts for JFact. Again, the (interpolated) performance curves of HermiT and Openllet are similar to the one of $\mathcal{FL}_o wer$– albeit less strongly as in the case of the Ore-Corpus. However, this may support the earlier finding that the same kind of ontology could be difficult for both reasoning approaches.

To sum up, the running time of $\mathcal{FL}_o wer$ for testing subsumption and for computing classification is on average substantially better than the one of JFact, Openllet, and even of HermiT. This is a remarkable result of a comparison between a newcomer system that implements only a few optimizations and well-established systems that have been developed for years. Our comparative evaluation suggests that the same kind of ontology may be difficult (or, alternatively, be easy) for reasoners based on the computation of least functional models as well as for tableaux-based reasoners.

7 Conclusions

The main contribution of this paper is a novel algorithm for deciding subsumption in the DL $\mathcal{FL}_0$ w.r.t. general TBoxes, and a practical demonstration that this algorithm is easy to implement and behaves surprisingly well on large ontologies. Our reasoner $\mathcal{FL}_o wer$ outperforms state-of-the art DL reasoners for testing subsumption and for classifying general TBoxes.

One may ask, however, why a dedicated reasoner for $\mathcal{FL}_0$ is needed, given the facts that the worst-case complexity of reasoning in $\mathcal{FL}_0$ is as high as for the considerably more expressive DL $\mathcal{ALC}$ and that there are very few pure $\mathcal{FL}_0$ ontologies available. We argue that such a dedicated reasoner may turn out to be very useful. First, the latter fact could be due to a chicken and egg problem: as long as no dedicated reasoner for $\mathcal{FL}_0$ is available, there is no incentive to restrict the expressiveness to $\mathcal{FL}_0$ when creating an ontology. When extracting our test ontologies, we observed that quite a number of application ontologies have large $\mathcal{FL}_0$ fragments. Second, regarding the former fact, it is well-known in the DL community that worst-case complexity results are not always a good indication for how hard reasoning turns out to be in practice. Third, some DL reasoners such as Konclude[9] and MORe (Romero *et al.* 2012) make use of specialized algorithms for certain language fragments as part of their overall reasoning approach, with impressive improvements of the performance. Our efficient subsumption algorithm for $\mathcal{FL}_0$ may turn out to be useful in this context. Finally, quite a number of non-standard reasoning tasks in $\mathcal{FL}_0$ w.r.t. general TBoxes have recently been investigated (Baader *et al.* 2016, 2018, 2018a, 2018b). The algorithms developed for solving these tasks usually depend on sub-procedures that perform subsumption tests or that use the least functional model directly. Our reasoner $\mathcal{FL}_o wer$ thus provides us with an efficient base for implementing such non-standard inferences.

Acknowledgments

We thank the anonymous reviewers for their helpful comments on an earlier draft of the manuscript. This work was partially supported by the DFG grant 389792660 as part of TRR 248 (`https://perspicuous-computing.science`).

Conflicts of interest

The authors declare none.

References

Baader, F. 1990. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI'90)*, Boston, MA, USA, 621–626.

Baader, F., Brandt, S. and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, L. P. Kaelbling and A. Saffiotti, Eds. Edinburgh, UK. Morgan Kaufmann, Los Altos, 364–369.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P. F., Eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York.

Baader, F., Fernandez Gil, O. and Marantidis, P. 2018. Matching in the description logic $\mathcal{FL}_0$ with respect to general TBoxes. In *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, G. Barthe, G. Sutcliffe and M. Veanes, Eds. vol. 57 of *EPiC Series in Computing*. EasyChair, 76–94.

Baader, F., Fernandez Gil, O. and Pensel, M. 2018a. Standard and non-standard inferences in the description logic $\mathcal{FL}_0$ using tree automata. In *GCAI-2018, 4th Global Conference on Artificial Intelligence*, D. D. Lee, A. Steen and T. Walsh, Eds. vol. 55 of *EPiC Series in Computing*. EasyChair, 1–14.

[9] `konclude.com`.

Baader, F., Horrocks, I., Lutz, C. and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Baader, F., Marantidis, P. and Okhotin, A. 2016. Approximate unification in the description logic $\mathcal{FL}_0$. In *Logics in Artificial Intelligence - 15th European Conference, JELIA 2016, Proceedings*, L. Michael and A. C. Kakas, Eds. vol. 10021 of *Lecture Notes in Computer Science*, 49–63.

Baader, F., Marantidis, P. and Pensel, M. 2018b. The data complexity of answering instance queries in $\mathcal{FL}_0$. In *Companion of the The Web Conference WWW*, P. Champin, F. L. Gandon, M. Lalmas and P. G. Ipeirotis, Eds. ACM, 1603–1607.

Baader, F. and Sattler, U. 2001. An overview of tableau algorithms for description logics. *Studia Logica 69*, 1, 5–40.

Baader, F. and Théron, C. 2020. Role-value maps and general concept inclusions in the minimal description logic with value restrictions – or revisiting old skeletons in the DL cupboard. *KI – Journal für Künstliche Intelligenz 34*, 3, 291–301.

Brachman, R. J., McGuinness, D. L., Patel-Schneider, P. F., Alperin Resnick, L. and Borgida, A. 1991. Living with CLASSIC: When and how to use a KL-ONE-like language. In *Principles of Semantic Networks*, J. F. Sowa, Eds. Morgan Kaufmann, Los Altos, 401–456.

Brachman, R. J. and Schmolze, J. G. 1985. An overview of the KL-ONE knowledge representation system. *Cognitive Science 9*, 2, 171–216.

Brandt, S. 2004. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, R. L. de Mántaras and L. Saitta, Eds., pp. 298–302.

Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P. F. and Sattler, U. 2008. OWL 2: The next step for OWL. *Journal of Web Semantics 6*, 4, 309–322.

Forgy, C. 1982. Rete: A fast algorithm for the many patterns/many objects match problem. *Artificial Intelligence 19*, 1, 17–37.

Hoehndorf, R., Schofield, P. N. and Gkoutos, G. V. 2015. The role of ontologies in biological and biomedical research: A functional perspective. *Briefings in Bioinformatics 16*, 6, 1069–1080.

Hofmann, M. 2005. Proof-theoretic approach to description-logic. In *Proc. of the 20th IEEE Symp. on Logic in Computer Science (LICS 2005)*, P. Panangaden, Ed. IEEE Computer Society Press, 229–237.

Horridge, M. and Bechhofer, S. 2011. The OWL API: A Java API for OWL ontologies. *Semantic Web 2*, 1, 11–21.

Horrocks, I., Patel-Schneider, P. F. and van Harmelen, F. 2003. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics 1*, 1, 7–26.

Kazakov, Y. 2009. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, C. Boutilier, Ed. IJCAI/AAAI, 2040–2045.

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

Krötzsch, M., Rudolph, S. and Hitzler, P. 2007. Complexity boundaries for Horn description logics. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22–26, 2007, Vancouver, British Columbia, Canada*. AAAI Press, 452–457.

Krötzsch, M., Rudolph, S. and Hitzler, P. 2013. Complexities of Horn description logics. *ACM Transactions on Computational Logic 14*, 1, 2:1–2:36.

Matentzoglu, N., Bail, S. and Parsia, B. 2013. A snapshot of the OWL Web. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21–25, 2013, Proceedings, Part I*, H. Alani, L. Kagal, A. Fokoue, P. T.

Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty and K. Janowicz, Eds. vol. 8218 of *Lecture Notes in Computer Science*. Springer, 331–346.

Mays, E., Dionne, R. and Weida, R. 1991. K-REP system overview. *SIGART Bulletin 2*, 3.

Michel, F., Turhan, A.-Y. and Zarriess, B. 2019. Efficient TBox reasoning with value restrictions—introducing the $\mathcal{FL}_o$*wer* reasoner. In *Proceedings of the 3rd International Joint Conference on Rules and Reasoning (RuleML+RR 2019)*, P. Fodor and M. Montali, Eds. LNCS, Bolzano, Italy. Springer.

Nebel, B. 1990. Terminological reasoning is inherently intractable. *Artificial Intelligence 43*, 235–249.

Parsia, B., Matentzoglu, N., Gonçalves, R. S., Glimm, B. and Steigmiller, A. 2017. The owl reasoner evaluation (ore) 2015 competition report. *Journal of Automated Reasoning 59*, 4, 455–482.

Peltason, C. 1991. The BACK system — an overview. *SIGART Bulletin 2*, 3, 114–119.

Romero, A. A., Cuenca Grau, B. and Horrocks, I. 2012. More: Modular combination of OWL reasoners for ontology classification. In *International Semantic Web Conference (1)*, vol. 7649 of *Lecture Notes in Computer Science*. Springer, 1–16.

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

Simančík, F., Kazakov, Y. and Horrocks, I. 2011. Consequence-based reasoning beyond Horn ontologies. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, T. Walsh, Ed. IJCAI/AAAI, 1093–1098.

Woods, W. A. and Schmolze, J. G. 1992. The KL-ONE family. In *Semantic Networks in Artificial Intelligence*, F. W. Lehmann, Ed., pp. 133–178. Pergamon Press. Published as a special issue of *Computers & Mathematics with Applications 23*, 2–9.