LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

Interval-based Temporal Reasoning with General TBoxes

Carsten Lutz

LTCS-Report 00-06

RWTH Aachen
LuFg Theoretische Informatik
http://www-lti.informatik.rwth-aachen.de

Ahornstr. 55
52074 Aachen
Germany

Interval-based Temporal Reasoning with General TBoxes

Carsten Lutz
RWTH Aachen, LuFG Theoretical Computer Science
Ahornstr. 55, 52074 Aachen

September 7, 2001

Contents

1 Motivation

Description Logics (DLs) are a family of formalisms well-suited for the representation of and reasoning about knowledge. Whereas most Description Logics represent only static aspects of the application domain, recent research resulted in the exploration of various Description Logics that allow to, additionally, represent temporal information, see [4] for an overview. The approaches to integrate time differ in at least two important aspects: First, the basic temporal entity may be a time point or a time interval. Second, the temporal structure may be part of the semantics (yielding a multi-dimensional semantics) or it may be integrated as a so-called concrete domain. Examples for multi-dimensional point-based logics can be found in, e.g., [21; 29], while multi-dimensional interval-based logics are used in, e.g., [23; 2]. The concrete domain approach needs some more explanation. Concrete domains have been proposed by Baader and Hanschke as an extension of Description Logics that allows reasoning about "concrete qualities" of the entities of the application domain such as sizes, length, or weights of real-worlds objects [5]. Description Logics with concrete domains do usually not use a fixed concrete domain; instead the concrete domain can be thought of as a parameter to the logic. As was first described in [16], if a "temporal" concrete domain is employed, then concrete domains are very useful for temporal reasoning. Temporal reasoning with concrete domains may be point-based, interval-based, or both.

In this paper, we define a temporal Description Logic based on concrete domains which uses points as its basic temporal entity, but which may also be used as a full-fledged interval-based temporal DL. More precisely, the presented logic $\mathcal{TDL}$ extends the basic Description Logic $\mathcal{ALC}$ [22] with a concrete domain that is based on the rationals and predicates $<$ and $=$. This allows to represent point-based temporal knowledge, e.g., the $\mathcal{TDL}$ concept

$$\mathsf{Student} \sqcap \exists \mathsf{graduation}, \mathsf{21birthday}.<$$

describes students who graduated before their 21'st birthday. For interval-based reasoning, the well-known Allen relations can be defined in terms of their endpoints [1]. Of course, point-based and interval-based temporal reasoning may be used in combination. Since it is an important feature of DLs that reasoning should be decidable, we prove decidability of the standard reasoning tasks by using an automata-theoretic approach which also yields a tight ExpTime complexity bound.

Most DLs allow for some kind of TBox formalism that is used to represent terminological knowledge as well as background knowledge about the application domain. However, there exist various flavours of TBoxes with vast differences in expressivity [17; 18; 14]. To the best of our knowledge, all interval-based DLs and all DLs with concrete domains defined in the literature admit only a very restricted form of TBox, i.e., sets of acyclic macro definitions. Compared to existing Description Logics that are interval-based or include concrete domains, the distinguishing feature of our logic is that it is equipped with a very general form of TBoxes that allows arbitrary equations over concepts. Thus, the presented work overcomes a major limitation of both families of Description Logics.

Our results can be viewed from the perspective of interval-based temporal reasoning and from the perspective of concrete domains. For the temporal perspective, we claim that the combination of general TBoxes and interval-based temporal reasoning is important for many application areas. In this paper, we present process engineering as an example. From the concrete domain perspective, our results can be viewed as follows: In [15], it is shown that, even for very simple concrete domains, reasoning with general TBoxes is undecidable. Obvious solutions, which include the restriction of the concrete domain to unary predicates and the restriction of the concrete domain concept constructor to features instead of feature chains, are not really convincing since the expressive power of the resulting formalism is very limited (readers not familiar with these notions are referred to [5]). It was an open question whether there exist interesting concrete domains for which reasoning with general TBoxes is decidable. The results presented in this paper answer this question to the a rmative. This paper is accompanied by a technical report containing the proofs of theorems.

2 Syntax and Semantics

In this section, we introduce syntax and semantics of the Description Logic $\mathcal{TDL}$. As mentioned in the introduction, this logic is from the family of DLs with concrete domains. However, since we only consider a single concrete domain, we do not explicitly refer to concrete domains in the definition of $\mathcal{TDL}$. The exact connection between concrete domains and our DL is discussed at the end of this section.

Definition 1. Let N_C, N_R, and N_{cF} be mutually disjoint and countably infinite sets of *concept names*, *roles*, and *concrete features*. Furthermore, let N_{aF} be a countably infinite subset of N_R. The elements of N_{aF} are called *abstract features*. A *path* u is a composition $f_1 \cdots f_n g$ of n abstract features $f_1, \ldots, f_n$ ($n \geq 0$) and one concrete feature g. The set of $\mathcal{TDL}$-concepts is the smallest set such that

1. every concept name is a concept
2. if C and D are concepts, R is a role, g is a concrete feature, u_1, u_2 are paths, and $P \in \{<, =\}$, then the following expressions are also concepts:
   (a) $\neg C$, $C \sqcap D$, $C \sqcup D$,
   (b) $\exists R.C$, $\forall R.C$,
   (c) $\exists u_1, u_2.P$, and $g\uparrow$.

An axiom is an expression of the form $C \sqsubseteq D$, where C and D are concepts. A finite set of axioms is called a *TBox*.

Throughout this paper, we will denote atomic concepts by the letter A, (possibly complex) concepts by the letters $C, D, \ldots$, roles by the letter R, abstract features by the letter f, concrete features by the letter g, paths by the letter u, and elements of the set $\{<, =\}$ by the letter P. We defer a discussion of how $\mathcal{TDL}$ can be used for interval-based temporal reasoning until Section 3. We will sometimes call the TBox formalism

introduced above *general TBoxes* to distinguish it from other, weaker formalisms such as the ones in [17; 14]. As most Description Logics, $\mathcal{TDL}$ is equipped with a Tarski-style semantics which is introduced next.

Definition 2. An *interpretation* $\mathcal{I}$ is a pair $(\Delta_\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta_\mathcal{I}$ is a set called the *domain* and $\cdot^\mathcal{I}$ is the *interpretation function*. The interpretation function maps

- each concept name C to a subset $C^\mathcal{I}$ of $\Delta_\mathcal{I}$,
- each role name R to a subset $R^\mathcal{I}$ of $\Delta_\mathcal{I} \times \Delta_\mathcal{I}$,
- each abstract feature f to a partial function $f^\mathcal{I}$ from $\Delta_\mathcal{I}$ to $\Delta_\mathcal{I}$, and
- each concrete feature g to a partial function $g^\mathcal{I}$ from $\Delta_\mathcal{I}$ to $\mathbb{Q}$.

For paths $u = f_1 \cdots f_n g$, we set $u^\mathcal{I}(a) := g^\mathcal{I}(f_n^\mathcal{I}(\cdots(f_1^\mathcal{I}(a))\cdots))$. The interpretation function is extended to arbitrary concepts as follows:

$$
\begin{aligned}
(C \sqcap D)^\mathcal{I} &:= C^\mathcal{I} \cap D^\mathcal{I} \\
(C \sqcup D)^\mathcal{I} &:= C^\mathcal{I} \cup D^\mathcal{I} \\
(\neg C)^\mathcal{I} &:= \Delta_\mathcal{I} \setminus C^\mathcal{I} \\
(\exists R.C)^\mathcal{I} &:= \{a \in \Delta_\mathcal{I} \mid \{b \mid (a,b) \in R^\mathcal{I}\} \cap C^\mathcal{I} \neq \emptyset\} \\
(\forall R.C)^\mathcal{I} &:= \{a \in \Delta_\mathcal{I} \mid \{b \mid (a,b) \in R^\mathcal{I}\} \subseteq C^\mathcal{I}\} \\
(\exists u_1, u_2.P)^\mathcal{I} &:= \{a \in \Delta_\mathcal{I} \mid u_1^\mathcal{I}(a) = x_1, u_2^\mathcal{I}(a) = x_2, \text{ and } x_1 P x_2\} \\
(g\uparrow)^\mathcal{I} &:= \{a \in \Delta_\mathcal{I} \mid g^\mathcal{I}(a) \text{ undefined}\}
\end{aligned}
$$

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff it satisfies $C^\mathcal{I} \subseteq D^\mathcal{I}$ for all axioms $C \sqsubseteq D$ in $\mathcal{T}$. $\mathcal{I}$ is a *model* of a concept C w.r.t. a TBox $\mathcal{T}$ iff $\mathcal{I}$ is a model of $\mathcal{T}$ and $C^\mathcal{I} \neq \emptyset$.

If $g(a) = x$ for some $a \in \Delta_\mathcal{I}$ and $x \in \mathbb{Q}$, then we call x a *concrete successor* of a in $\mathcal{I}$. In what follows, we write $\top$ for $A \sqcup \neg A$ and $\bot$ for $A \sqcap \neg A$, where A is a concept name. Moreover, we write $u\uparrow$ with $u = f_1 \cdots f_k g$ for $\forall f_1. \cdots \forall f_k. g\uparrow$.

How do $\mathcal{TDL}$-models look like? It is not hard to see that $\mathcal{TDL}$ does not have the finite model property: The concept $\top$ is obviously satisfiable w.r.t. the TBox $\{\top \sqsubseteq \exists g, fg. <\}$; however, it is not hard to see that there exists no *finite* model for this concept and this TBox. Furthermore, $\mathcal{TDL}$ has the tree model property, i.e., if a concept C is satisfiable w.r.t. a TBox $\mathcal{T}$, then there exists a tree-shaped model $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ of C and $\mathcal{T}$ where the elements of $\Delta_\mathcal{I}$ are the nodes of the tree and $\bigcup_{R \in N_R} R^\mathcal{I}$ is the set of edges. The proof of this tree model property is a byproduct of some results on Hintikka-trees obtained in Section 4.3.

In this paper, the following inference problems are considered.

Definition 3 (Inference Problems). Let C and D be concepts and $\mathcal{T}$ be a TBox. C *subsumes* D *w.r.t.* $\mathcal{T}$ (written $D \sqsubseteq_\mathcal{T} C$) iff $D^\mathcal{I} \subseteq C^\mathcal{I}$ for all models $\mathcal{I}$ of $\mathcal{T}$. C is *satisfiable w.r.t.* $\mathcal{T}$ iff there exists a model of both $\mathcal{T}$ and C. Moreover, C and D are *equivalent w.r.t.* $\mathcal{T}$ (written $C \equiv_\mathcal{T} D$) iff $C \sqsubseteq_\mathcal{T} D$ and $D \sqsubseteq_\mathcal{T} C$.

We are primarily interested in deciding satisfiability and subsumption. It is well-known that (un)satisfiability and subsumption can be mutually reduced to each other, i.e., $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$ and C is satisfiable w.r.t. $\mathcal{T}$ iff we do not have $C \sqsubseteq_{\mathcal{T}} \bot$. Sometimes, the inference problems are considered without reference to a TBox. In this case, we omit the index $\mathcal{T}$.

The use of $\mathbb{Q}$ in the semantics of $\mathcal{TDL}$-concepts and TBoxes is not crucial: $\mathbb{Q}$ may be replaced by any set with a dense linear ordering "$<$" such as, e.g., $\mathbb{R}$. Such a change in the semantics does not affect the satisfiability and subsumption of concepts. $\mathbb{Q}$ may, however, *not* be replaced by $\mathbb{N}$ since, on $\mathbb{N}$, the usual "$<$" ordering is not dense, and, as we will see later, there exist concepts which are satisfiable if $\mathbb{Q}$ is employed in the semantics but unsatisfiable if $\mathbb{N}$ is used.

We now discuss the relationship of $\mathcal{TDL}$ and Description Logics with concrete domains. To this end, let us introduce concrete domains formally.

Definition 4 (Concrete Domain). A *concrete domain* $\mathcal{D}$ is a pair $(\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$, where $\Delta_{\mathcal{D}}$ is a set called the domain, and $\Phi_{\mathcal{D}}$ is a set of predicate names. Each predicate name $P \in \Phi_{\mathcal{D}}$ is associated with an arity n and an n-ary predicate $P^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}^n$.

The concrete domain is usually integrated into the logic by a concept constructor $\exists u_1, \ldots, u_n.P$, where $u_1, \ldots, u_n$ are paths and $P \in \Phi_{\mathcal{D}}$ (there also exist other constructors, see, e.g., [10]). The semantics of this constructor is as follows:

$$(\exists u_1, \ldots, u_n.P)^{\mathcal{I}} := \{a \in \Delta_{\mathcal{I}} \mid u_i^{\mathcal{I}}(a) = x_i \text{ for } 1 \leq i \leq n \text{ and } (x_1^{\mathcal{I}}, \ldots, x_n^{\mathcal{I}}) \in P^{\mathcal{D}}\}.$$

Hence, it is obvious that $\mathcal{TDL}$ can be viewed as being equipped with the concrete domain $\mathcal{D}_< := (\mathbb{Q}, \{<, =\})$, where $<$ and $=$ are binary predicates with the usual semantics. For most DLs with concrete domains, it is required that the set of predicates is closed under negation and contains a name $\top_{\mathcal{D}}$ for $\Delta_{\mathcal{D}}$. This property ensures that every concept can be converted into an equivalent one in the so-called negation normal form (NNF). The NNF of concepts, in turn, is used as a starting point for devising satisfiability algorithms. It is not hard to see that $\mathcal{D}_<$ is not admissible in this sense. However, as we will see in Section 4.2, the conversion of $\mathcal{TDL}$-concepts into equivalent ones in NNF is nevertheless possible.

One technical difference between $\mathcal{TDL}$ and most Description Logics with concrete domains such $\mathcal{ALC}(\mathcal{D})$, which is defined by Baader and Hanschke in [5], should be mentioned: $\mathcal{ALC}(\mathcal{D})$ does not distinguish between abstract and concrete features, but provides only one type of feature interpreted as a partial function from $\Delta_{\mathcal{I}}$ to $\Delta_{\mathcal{I}} \cup \Delta_{\mathcal{D}}$. Obviously, using these "combined" features instead of separated ones slightly increases expressive power. However, it seems rather hard to find any cases in which the additional expressivity is crucial. Furthermore, separating concrete and abstract features allows a clearer algorithmic treatment and clearer proofs. Apart from this difference, $\mathcal{TDL}$ is just $\mathcal{ALC}(\mathcal{D}_<)$ enriched with general TBoxes, where $\mathcal{ALC}(\mathcal{D}_<)$ is $\mathcal{ALC}(\mathcal{D})$ instantiated with the concrete domain $\mathcal{D}_<$.

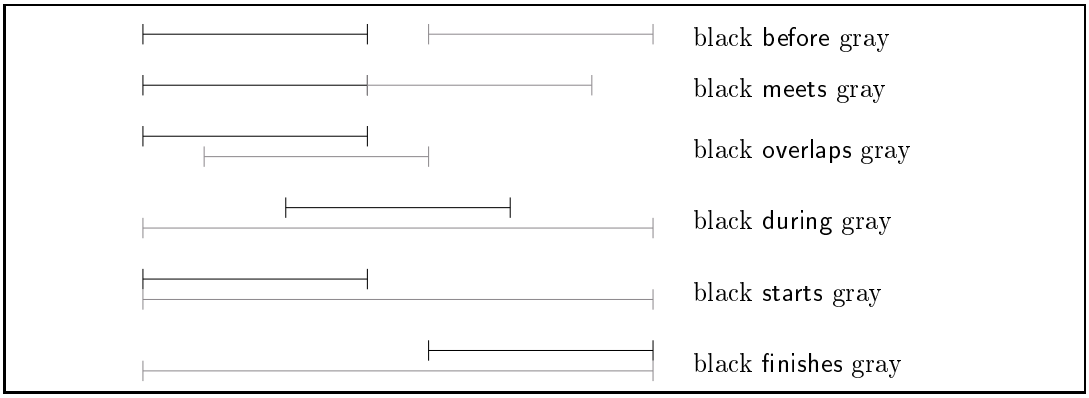

Figure 1: The Allen relations (without equal and inverses).

3 Temporal Reasoning with $\mathcal{TDL}$

Although $\mathcal{TDL}$ does only provide the relations "=" and "<" on time points, it is not hard to see that the remaining relations can be defined:

$$\begin{aligned}
\exists u_1, u_2. > &\equiv \exists u_2, u_1.< \\
\exists u_1, u_2. \leq &\equiv \exists u_1, u_2.< \sqcup \exists u_1, u_2.= \\
\exists u_1, u_2. \geq &\equiv \exists u_2, u_1.< \sqcup \exists u_1, u_2.=
\end{aligned}$$

However, we claim that $\mathcal{TDL}$ cannot only be used for point-based temporal reasoning but also as a full-fledged interval-based temporal Description Logic. Reasoning with time intervals has a considerable tradition in artificial intelligence, see, e.g., [1; 23; 28; 9; 12; 26; 2]. The foundation of interval-based reasoning in AI is Allen's observation that there are 13 possible relationships between two intervals [1] such as, for example, the meets relation: Two intervals i_1 and i_2 are related by meets iff the right endpoint of i_1 is identical to the left endpoint of i_2. Figure 1 illustrates meets and the other relations omitting equal and inverses. The inverse relations corresponding to the relations displayed in Figure 1 are called after, met-by, overlapped-by, contains, started-by, and finished-by (from top to bottom).

In this section, we first introduce a variant of $\mathcal{TDL}$ that is purely interval-based, then present a framework for mixed interval- and point-based reasoning in $\mathcal{TDL}$ itself and finally apply this framework in the application area of process engineering

3.1 An Interval-based Variant of $\mathcal{TDL}$

It is straightforward to define a variant of $\mathcal{TDL}$ whose semantics is interval-based instead of point-based and which offers Allen's relations as predicates.

Definition 5 ($\mathcal{TDL}^I$). A $\mathcal{TDL}^I$-concept is a $\mathcal{TDL}$-concept in which each occurrence of predicates $<$ and $=$ have been replaced by Allen relations equal, before, after, meets, met-by, overlaps, overlapped-by, during, contains, starts, started-by, finishes, finished-by. $\mathcal{TDL}^I$-TBoxes are defined analogously.

Let $I_{\mathbb{Q}}$ be the set $\{(q_1, q_2) \in \mathbb{Q}^2 \mid q_1 < q_2\}$, i.e., the set of all intervals over $\mathbb{Q}$. An $\mathcal{TDL}^I$-interpretation $\mathcal{I}$ is a $\mathcal{TDL}$-interpretation that maps each concrete feature g to partial $g^{\mathcal{I}}$ functions from $\Delta_{\mathcal{I}}$ to $I_{\mathbb{Q}}$. The extension to complex $\mathcal{TDL}^I$ concepts is as for $\mathcal{TDL}$ (now using Allen relations instead of the predicates $<$ and $=$).

For example, the following is a $\mathcal{TDL}^I$-concept:

$$\exists fg_1, g_2.\mathsf{during} \sqcup \exists R.\exists g_1, g_2.\mathsf{after}$$

Just as $\mathcal{TDL}$ can (apart from some minor differences) be viewed as $\mathcal{ALC}(\mathcal{D}_<)$ enriched with general TBoxes, $\mathcal{TDL}^I$ can be seen as $\mathcal{ALC}(\mathcal{D}_I)$ enriched with general TBoxes, where $\mathcal{D}_I$ is a concrete domain based on the set of intervals (over $\mathbb{Q}$) and Allen's relations. Such a concrete domain has, e.g., been defined in [16].

We show that $\mathcal{TDL}^I$ is indeed just a variant of $\mathcal{TDL}$ by giving a very simple translation of $\mathcal{TDL}^I$-concepts (resp. TBoxes) to corresponding $\mathcal{TDL}$ concepts (resp. TBoxes). W.l.o.g., we assume that the concrete features appearing in $\mathcal{TDL}^I$-concepts and TBoxes are called $i_1, i_2, \ldots$. In the target language $\mathcal{TDL}$, we use concrete features $\ell_1, r_1, \ell_2, r_2, \ldots$, where, intuitively, ℓ_j-successors in $\mathcal{TDL}$ represent the left endpoints of the intervals represented by i_j-successors in $\mathcal{TDL}^I$ and r_j-successors represent the corresponding right endpoints of these intervals.

Definition 6 ($\mathcal{TDL}^I$ translation). For each $\mathcal{TDL}^I$-concept C, we define a corresponding $\mathcal{TDL}$-concept $\tau(C)$ that is obtained by exhaustively applying the following rewrite rules to C:

$$\begin{aligned}
\exists Fi_1, F'i_2.\mathsf{equal} &\rightsquigarrow \exists F\ell_1, F'\ell_2.= \sqcap \exists Fr_1, Fr_2.= \\
\exists Fi_1, F'i_2.\mathsf{before} &\rightsquigarrow \exists Fr_1, F'\ell_2.< \\
\exists Fi_1, F'i_2.\mathsf{after} &\rightsquigarrow \exists F'r_2, F\ell_1.< \\
\exists Fi_1, F'i_2.\mathsf{meets} &\rightsquigarrow \exists Fr_1, F'\ell_2.= \\
\exists Fi_1, F'i_2.\mathsf{met\text{-}by} &\rightsquigarrow \exists F\ell_1, F'r_2.= \\
\exists Fi_1, F'i_2.\mathsf{overlaps} &\rightsquigarrow \exists F\ell_1, F'\ell_2.< \sqcap \exists F'\ell_2, Fr_1.< \sqcap \exists Fr_1, F'r_2.< \\
\exists Fi_1, F'i_2.\mathsf{overlapped\text{-}by} &\rightsquigarrow \exists F'\ell_2, F\ell_1.< \sqcap \exists F\ell_1, F'r_2.< \sqcap \exists F'r_2, Fr_1.< \\
\exists Fi_1, F'i_2.\mathsf{during} &\rightsquigarrow \exists F'\ell_2, F\ell_1.< \sqcap \exists Fr_1, F'r_2.< \\
\exists Fi_1, F'i_2.\mathsf{contains} &\rightsquigarrow \exists F\ell_1, F'\ell_2.< \sqcap \exists F'r_2, Fr_1.< \\
\exists Fi_1, F'i_2.\mathsf{starts} &\rightsquigarrow \exists F\ell_1, F'\ell_2.= \sqcap \exists Fr_1, F'r_2.< \\
\exists Fi_1, F'i_2.\mathsf{started\text{-}by} &\rightsquigarrow \exists F\ell_1, F'\ell_2.= \sqcap \exists F'r_2, Fr_1.< \\
\exists Fi_1, F'i_2.\mathsf{finishes} &\rightsquigarrow \exists F'\ell_2, F\ell_1.= \sqcap \exists Fr_1, F'r_2.= \\
\exists Fi_1, F'i_2.\mathsf{finished\text{-}by} &\rightsquigarrow \exists F\ell_1, F'\ell_2.= \sqcap \exists Fr_1, F'r_2.=
\end{aligned}$$

where $F, F' \in (N_{aF})^*$, i.e., F and F' are words over the alphabet N_{aF}. For each $\mathcal{TDL}^I$-TBox $\mathcal{T}$, we define a corresponding $\mathcal{TDL}$-TBox $\tau(\mathcal{T})$ analogously.

Using this translation, satisfiability (and hence also subsumption) of $\mathcal{TDL}^I$-concepts can be reduced to satisfiability of $\mathcal{TDL}$-concepts.

$$
\begin{array}{rcl}
\mathsf{ATemporal} & \doteq & t{\uparrow} \sqcap \ell{\uparrow} \sqcap r{\uparrow} \\
\mathsf{Temporal} & \doteq & \mathsf{Point} \sqcup \mathsf{Interval} \\
\mathsf{Point} & \doteq & \exists t, t.= \sqcap \ell{\uparrow} \sqcap r{\uparrow} \\
\mathsf{Interval} & \doteq & \exists \ell, r.< \sqcap t{\uparrow} \\
\mathsf{Interval} & \sqsupseteq & \exists \ell, \ell, = \sqcup \exists r, r.=
\end{array}
$$

Figure 2: Basic definitions of the framework $\mathcal{T}^*$.

Proposition 7. *A $\mathcal{TDL}^I$-concept C is satisfiable w.r.t. a $\mathcal{TDL}^I$-TBox $\mathcal{T}$ iff $\tau(C)$ is satisfiable w.r.t. $\tau(\mathcal{T}) \cup \mathcal{T}''$, where*

$$\mathcal{T}'' = \bigcup_{1 \leq i \leq k} \{\top \sqsubseteq \exists \ell_i, r_i.<\}.$$

Hence, the ExpTime decidability result for $\mathcal{TDL}$ obtained in Section 4 implies that $\mathcal{TDL}^I$ is also decidable in ExpTime. Again, it is worth to view this result from the concrete domain point of view: The above translation shows that the concrete domain $\mathcal{D}_I$ based on Allen's relations (which is formally defined in [16]) is a concrete domain for which reasoning with general TBoxes is decidable. Hence, we have shown that, despite the discouraging results given in [15]—where it is shown that the combination of general TBoxes and certain very simple concrete domains already leads to undecidability—there exist interesting concrete domains for which reasoning with general TBoxes is decidable.

3.2 A Representation Framework

As an alternative to using the logic $\mathcal{TDL}^I$ for interval-based temporal reasoning, one can directly employ $\mathcal{TDL}$ and represent intervals as pairs of endpoints. This has the advantage that mixed point- and interval-based reasoning becomes possible. This approach is pursued in the following section, where we give an application example. To simplify presentation, in this section we introduce an abstract framework for temporal reasoning with $\mathcal{TDL}$ that consists of several conventions and abbreviations.

We assume that each entity of the application domain is either temporal or atemporal. If it is temporal, its temporal extension may be a time point or an interval but not both. We generally assume that left endpoints of intervals aree represented by the concrete feature ℓ, right endpoints of intervals are represented by the concrete feature r, and time-points not related to intervals are represented by the concrete feature t. All this can be expressed by the TBox $\mathcal{T}^*$ displayed in Figure 2. In the figure, $C \doteq D$ is an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. The TBox implies that the concepts ATemporal, Point, and Interval are mutually disjoint. To keep concepts readable, we introduce abbreviations for Allen's relations. For example,

$$\exists (F, F').\mathsf{contains}$$

is an abbreviation for

$$\exists F\ell, F'\ell.< \ \sqcap\ \exists F'r, Fr.<$$

where $F, F' \in (N_{aF})^*$. Note that we have

$$\exists(F, F').\mathsf{contains} \sqsubseteq_{\mathcal{T}^*} \exists F.\mathsf{Interval} \sqcap \exists F'.\mathsf{Interval}.$$

Similar abbreviations are introduced for the other Allen relations, where the defining concepts can be read off from Definition 6. For better readability, we use self to denote the empty word. For example,

$$\exists(F, \mathsf{self}).\mathsf{starts}$$

is an abbreviation for

$$\exists F\ell, \ell.= \ \sqcap\ \exists Fr, r.<.$$

Intuitively, self refers to the interval associated with the abstract object at which the $\exists(F, \mathsf{self}).\mathsf{starts}$ concept is "evaluated".

Since we have intervals *and* points available, we should also be able to talk about the relationship of points and intervals. More precisely, there exist 5 possible relations between a point and an interval and we introduce the following abbreviations for them:

$$\begin{array}{rcl}
\exists(Fp, F').\mathsf{beforep} & \text{for} & \exists Fp, F'\ell.< \\
\exists(Fp, F').\mathsf{startsp} & \text{for} & \exists Fp, F'\ell.= \\
\exists(Fp, F').\mathsf{duringp} & \text{for} & \exists F'\ell, Fp.< \ \sqcap\ \exists Fp, F'r.< \\
\exists(Fp, F').\mathsf{finishesp} & \text{for} & \exists Fp, F'r.= \\
\exists(Fp, F').\mathsf{afterp} & \text{for} & \exists F'r, Fp.<
\end{array}$$

where again $F, F' \in (N_{aF})^*$ and $p \in N_{cF}$. We refrain from defining the inverses of these relations. The usefulness of the introduced framework is demonstrated in the next section.

3.3 An Application Example

Interval-based temporal Description Logics have been used in various application areas such as disaster management [13] and reasoning about action and plans [4]. We claim that $\mathcal{TDL}$ is a contribution to most of these application areas since, unlike existing interval-based Description Logics, it admits general TBoxes. To substantiate this claim, we motivate $\mathcal{TDL}$ as an appropriate tool for temporal reasoning in the area of process engineering. In [19], Sattler describes how Description Logics can be used for representation and reasoning in this application domain. However, in Sattler's approach, only static knowledge about process engineering is considered, i.e., there is no explicit representation of the temporal relationships between described entities. We use the framework presented in the previous section to show how the temporal aspects of this application domain can be represented in $\mathcal{TDL}$ thus refining Sattler's model.

Assume that our goal is to represent information about an automated chemical production process that is carried out by some complex technical device. The device operates each day for some time depending on the number of orders. It needs a

$$
\begin{aligned}
\mathsf{Week} \quad \doteq \quad & \mathsf{Interval} \sqcap \\
& \bigsqcap_{1 \leq i \leq 7} \exists \mathsf{day}_i.\mathsf{Day} \sqcap \\
& \exists(\mathsf{day}_1, \mathsf{self}).\mathsf{starts} \sqcap \\
& \exists(\mathsf{day}_7, \mathsf{self}).\mathsf{finishes} \sqcap \\
& \bigsqcap_{1 \leq i < 7} \exists(\mathsf{day}_i, \mathsf{day}_{i+1}).\mathsf{meets} \sqcap \\
& \exists \mathsf{next}.\mathsf{Week} \sqcap \\
& \exists(\mathsf{self}, \mathsf{next}).\mathsf{meets}
\end{aligned}
$$

Figure 3: Weeks and days.

$$
\begin{aligned}
\mathsf{Day} \quad \doteq \quad & \mathsf{Interval} \sqcap \\
& \exists \mathsf{start}.\mathsf{Startup} \sqcap \\
& \exists \mathsf{op}.\mathsf{Operation} \sqcap \\
& \exists \mathsf{shut}.\mathsf{Shutdown} \sqcap \\
& \exists(\mathsf{start}, \mathsf{self}).\mathsf{during} \sqcap \\
& \exists(\mathsf{start}, \mathsf{op}).\mathsf{meets} \sqcap \\
& \exists(\mathsf{op}, \mathsf{shut}).\mathsf{meets} \sqcap \\
& \exists(\mathsf{shut}, \mathsf{self}).\mathsf{during} \\
\mathsf{Week} \quad \sqsubseteq \quad & \exists \mathsf{maint}.\mathsf{Maintenance} \sqcap \\
& \exists(\mathsf{self}, \mathsf{maint}).\mathsf{contains} \\
\mathsf{Interval} \quad \sqsupseteq \quad & \mathsf{Startup} \sqcup \mathsf{Operation} \sqcup \mathsf{Shutdown} \sqcup \mathsf{Maintenance}
\end{aligned}
$$

Figure 4: Operation and maintenance.

complex startup and shutdown process before resp. after operation. Moreover, some weekly maintenance is needed to keep the device functional. Let us first represent the underlying temporal structure that, in our case, consists of weeks and days. The corresponding TBox can be found in Figure 3. In the figure, concepts are written capitalized while roles (as well as predicates) start with a lowercase letter. The TBox in the figure states that each week consists of seven days, where the i'th day is accessible from the corresponding week via the abstract feature day_i. The temporal relationship between the days are as expected: Monday starts the week, Sunday finishes it, and each day temporally meets the succeeding one. This implies that each of the seven days is during the corresponding week although this is not explicitly stated. Moreover, each week has a successor week that it temporally meets. Note that the TBox is cyclic, i.e., Week is defined in terms of itself. This is already more than simple, acyclic TBoxes are able to express [18].

We can now describe the startup, operation, shutdown, and maintenance phases,

$$
\begin{aligned}
\mathsf{Day} &\sqsubseteq \exists \mathsf{up\text{-}int}.\mathsf{Operator\text{-}interaction} \sqcap \\
&\quad\ \exists \mathsf{down\text{-}int}.\mathsf{Operator\text{-}interaction} \sqcap \\
&\quad\ \exists (\mathsf{up\text{-}int} \quad t, \mathsf{start}).\mathsf{startsp} \\
&\quad\ \exists (\mathsf{down\text{-}int} \quad t, \mathsf{shut}).\mathsf{startsp} \\
\mathsf{Operator\text{-}interaction} &\sqsubseteq \mathsf{Point}
\end{aligned}
$$

Figure 5: Operator interaction.

see Figure 4. Here start, op, shut, and maint are abstract features. The definition implies that operation phases are temporally during the corresponding day. Our current model does not say anything about the temporal relationship of maintenance and operation. This may be inadequate, if, for example, maintenance and operation are mutually exclusive since maintenance prevents operation or is too dangerous during the operation phase. We can take this into account by using additional axioms

$$\mathsf{Week} \sqcap \bigsqcup_{1 \leq i \leq 7} \exists (\mathsf{maint}, \mathsf{day}_i \quad \mathsf{op}).\mathsf{REL} \sqsubseteq \bot \qquad (*)$$

where (1) " " is used for better readability (i.e., sequences of features $f_1 \cdots f_k$ are written as $f_1 \quad \cdots \quad f_k$) and (2) REL is equal, overlaps, overlapped-by, during, contains, starts, started-by, finishes, or finished-by.

We may view the knowledge modeled so far as the specification of a faultless operation. To illustrate reasoning with $\mathcal{TDL}$ in this application domain, we can now check facts about specific weeks or days against the specification. For example, say that in the 23rd calendar week, the maintenance took extremely long: it started on Tuesday night and wasn't finished until Thursday morning. This is expressed by the axiom

$$\mathsf{Week23} \sqsubseteq \mathsf{Week} \sqcap \exists (\mathsf{day}_3, \mathsf{maint}).\mathsf{during}.$$

The $\mathcal{TDL}$ reasoner can be used to check whether there was a problem in Week 23. This is obviously the case if the concept Week23 is not satisfiable w.r.t. the TBox that is obtained from the TBox expressing faultless operation by adding the definitorial axiom for Week23. It is not hard to see that this TBox is indeed unsatisfiable: The definition of Week23 implies that the operation phase of wednesday is during the weekly maintenance phase which is a contradiction to $(*)$. Hence we can deduce that, in the 23rd calendar week, the specification of faultless operation was not met.

In order to demonstrate mixed reasoning with time points and intervals, we propose a further refinement of our model. Assume that the production process is fully automated except that an operator interaction is necessary to initiate the startup and shutdown processes. These facts can be expressed using the axiom shown in Figure 5. In the figure, up-int and down-int are abstract features. We may now check specific weeks or days against our refined specification of faultless operation. For example, it may be the case that, on November 13, the operation continued after the shutdown

interaction which is obviously not a faultless operation. This can be described by

$$\mathsf{Nov13} \sqsubseteq \mathsf{Day} \sqcap \exists(\mathsf{down\text{-}int}, \mathsf{op}).\mathsf{duringp}.$$

It is not hard to see that Nov13 is unsatisfiable: the shutdown interaction cannot start the shutdown phase and simultaneously be during the operation phase since the operation phase must meet the shutdown phase. As another example, assume that, in calendar week 11, the shutdown interaction of some (unspecified) day occurred during the weekly maintenance phase. Is this compatible with a faultless operation? To check this, we can add the axiom

$$\mathsf{Week11} \doteq \mathsf{Week} \sqcap \bigsqcup_{1 \leq i \leq 7} \exists(\mathsf{day}_i \quad \mathsf{down\text{-}int}, \mathsf{maint}).\mathsf{duringp}$$

to our TBox. A close look reveals that Week11 is also unsatisfiable: The shutdown interaction starts the shutdown phase that is met by the operation phase. Hence, if the shutdown interaction of some day occurs during the weekly maintenance phase, then the temporal relation between this day's operation phase and the maintenance phase is either during, overlaps, or starts. All three possibilities conflict with $(*)$.

The discussed examples do not exploit all the expressive power of $\mathcal{TDL}$ because of simplicity. Nevertheless, they demonstrate that $\mathcal{TDL}$ is a powerful tool for representing temporal knowledge.

4 The Decision Procedure

In this section, we prove satisfiability of $\mathcal{TDL}$-concepts w.r.t. TBoxes to be decidable and obtain a tight ExpTime complexity bound for this problem. Decidability is proved using an automata-theoretic approach: This is done using an automata-theoretic approach: first, we abstract models to so-called Hintikka-trees such that there exists a model for a concept C and a TBox $\mathcal{T}$ iff there exists a Hintikka-tree for C and $\mathcal{T}$. Then, we build, for each $\mathcal{TDL}$-concept C and TBox $\mathcal{T}$, a looping automaton $\mathcal{A}_{(C,\mathcal{T})}$ that accepts exactly the Hintikka-trees for $(C, \mathcal{T})$. In particular, this implies that $\mathcal{A}_{(C,\mathcal{T})}$ accepts the empty (tree-) language iff C is unsatisfiable w.r.t. $\mathcal{T}$.

4.1 Preliminaries

In this section, we introduce the basic notions needed for proving decidability of $\mathcal{TDL}$ like trees, looping automata, and the language they accept.

Definition 8. Let M be a set and $k \geq 1$. A *k-ary M-tree* is a mapping $T : \{1, \ldots, k\}^* \to M$ that labels each node $\alpha \in \{1, \ldots, k\}^*$ with $T(\alpha) \in M$. Intuitively, the node αi is the i-th child of α. We use ϵ to denote the empty word (corresponding to the root of the tree).

A *looping automaton* $\mathcal{A} = (Q, M, I, \Delta)$ for k-ary M-trees is defined by a set Q of states, an alphabet M, a subset $I \subseteq Q$ of initial states, and a transition relation $\Delta \subseteq Q \times M \times Q^k$.

A *run* of $\mathcal{A}$ on an M-tree T is a mapping $r : \{1, \ldots, k\}^* \mapsto Q$ with $r(\epsilon) \in I$ and

$$(r(\alpha), T(\alpha), r(\alpha 1), \ldots, r(\alpha k)) \in \Delta$$

for each $\alpha \in \{1, \ldots, k\}^*$. A looping automaton accepts all those M-trees for which a run exists, i.e., the language $\mathcal{L}(\mathcal{A})$ of M-trees accepted by $\mathcal{A}$ is

$$L(\mathcal{A}) = \{T \mid \text{there is a run of } \mathcal{A} \text{ on } T\}.$$

In [27], it is proved that the emptiness problem for looping automata, i.e., the problem to decide whether the language $L(\mathcal{A})$ accepted by a given looping automaton $\mathcal{A}$ is empty, is decidable in polynomial time. A Hintikka-tree for C and $\mathcal{T}$ corresponds to a canonical model for C and $\mathcal{T}$. Apart from describing the abstract domain $\Delta_{\mathcal{I}}$ of the corresponding canonical model $\mathcal{I}$ together with the interpretation of concepts and roles, each Hintikka-tree induces a directed graph whose edges are labelled with predicates from $\{<, =\}$. These constraint graphs describe the "concrete part" of $\mathcal{I}$ (i.e., concrete successors of domain objects and their relationships) and are defined in the following.

Definition 9. A *constraint graph* is a pair $G = (V, E)$, where V is a countable set of *nodes* and $E \subseteq V \times V \times \{=, <\}$ a set of *edges*. We generally assume that constraint graphs are *equality closed*, i.e., that $(v_1, v_2, =) \in E$ implies $(v_2, v_1, =) \in E$. A constraint graph $G = (V, E)$ is called *satisfiable over* M—where M is a set equipped with a total ordering $<$—iff there exists a total mapping from V to M such that $(v_1)\, P\, (v_2)$ for all $(v_1, v_2, P) \in E$. Such a mapping is called a *solution* for G.

A *path* Q in G is a finite non-empty sequence of nodes $v_0, \ldots, v_{k-1} \in V$ such that, for all i with $0 \leq i < k$, we have $(v_i, v_{i+1}, P) \in E$, where $P \in \{<, =\}$. Such a path is also called a path *from* v_0 to v_{k-1}. Q is called a $=$-path iff $(v_i, v_{i+1}, =) \in E$ for $0 \leq i < k - 1$. A *cycle* O in G is a *path* $v_0, \ldots, v_{k-1}$, such that $(v_{k-1}, v_0, P) \in E$ for some $P \in \{<, =\}$. O is a $<$-*cycle* iff O is a cycle such that $(v_i, v_{i \oplus_k 1}, <) \in E$ for some i with $0 \leq i < k$, where $\oplus_k$ denotes addition modulo k.

The following theorem will be crucial for proving that, for every Hintikka-tree, there exists a corresponding canonical model. More precisely, it will be used to ensure that the constraint graph induced by a Hintikka-tree, which describes the concrete part of the corresponding model, is satisfiable.

Theorem 10. *A constraint graph G is satisfiable over M with $M \in \{\mathbb{Q}, \mathbb{R}\}$ iff G does not contain a $<$-cycle.*

Proof Since the "$\Rightarrow$" direction is trivial, we concentrate on the "$\Leftarrow$" direction. Let G be a constraint graph not containing a $<$-cycle. Let $\sim$ be the relation on V with $v_1 \sim v_2$ iff $v_1 = v_2$ or there exists a $=$-path between v_1 and v_2. Since constraint graphs are assumed to be equality closed, $\sim$ is an equivalence relation. For $v \in V$, denote the equivalence class of v w.r.t. $\sim$ with $[v]_\sim$. Define a new constraint graph $G' = (V', E')$ as follows:

$$\begin{aligned} V' &:= \{[v]_\sim \mid v \in V\} \\ E' &:= \{([v_1]_\sim, [v_2]_\sim, <) \mid \exists v_1', v_2' \in V \text{ such that} \\ &\qquad v_1' \in [v_1]_\sim, v_2' \in [v_2]_\sim, \text{ and } (v_1', v_2', <) \in E\} \end{aligned}$$

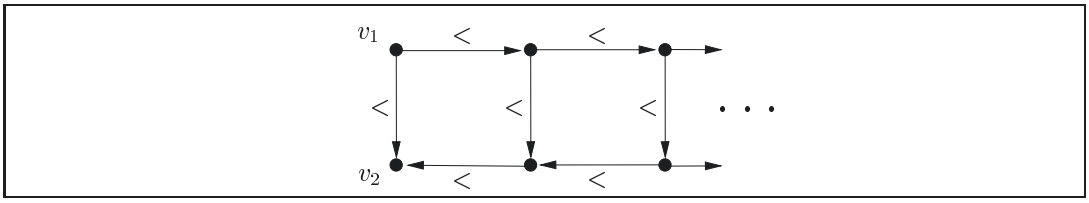

Figure 6: A constraint graph containing no $<$-cycle that is unsatisfiable over $\mathbb{N}$.

Using the fact that G does not contain a $<$-cycle, it is straightforward to prove that G' does not contain a $<$-cycle. Since G' does not contain a $<$-cycle, E' induces a partial order with domain V'. By Szpilrajn's Theorem, every partial order can be extended to a total order (on the same domain) [24]. Let $\prec_{E'}$ be a total order obtained in this way from the partial order induced by E'. In the following, we show that every total order with a countable domain can be embedded into $\mathbb{Q}$ (resp. $\mathbb{R}$) such that the ordering is preserved. This su ces to complete the proof since it implies that that there exists a total mapping σ from V to $\mathbb{Q}$ (resp. $\mathbb{R}$) such that $v_1 \prec_{E'} v_2$ implies $\sigma(v_1) < \sigma(v_2)$. It is obvious that σ is a solution for G' and it is straightforward to use σ to construct a solution for G.

Hence, it remains to show that every total order $\prec$ with a countable domain D can be embedded into $\mathbb{Q}$ (resp. $\mathbb{R}$) such that the ordering is preserved. Let $d_0, d_1, \ldots$ be an enumeration of D. We use induction over this enumeration to define a function σ from D to $\mathbb{Q}$ (resp. $\mathbb{R}$) such that $d_1 \prec d_2$ implies $\sigma(d_1) < \sigma(d_2)$ for all $d_1, d_2 \in D$.

1. For the induction start, set $\sigma(d_0)$ to some $q \in \mathbb{Q}$.

2. Assume that $\sigma(d_i)$ is defined for $0 \leq i < k$. We distinguish three cases:

   (a) $d_i \prec d_k$ for $0 \leq i < k$. Since $\mathbb{Q}$ has no maximum, there exists a $q \in \mathbb{Q}$ such that $q > \sigma(d_i)$ for $0 \leq i < k$. Set $\sigma(d_k) := q$.

   (b) $d_k \prec d_i$ for $0 \leq i < k$. Since $\mathbb{Q}$ has no minimum, there exists a $q \in \mathbb{Q}$ such that $q < \sigma(d_i)$ for $0 \leq i < k$. Set $\sigma(d_k) := q$.

   (c) Neither of the previous two cases holds. Since $\mathbb{Q}$ is dense, there exists a $q \in \mathbb{Q}$ such that $\max\{\sigma(d_i) \mid 0 \leq i < k \text{ and } d_i \prec d_k\} < q$ and $q < \min\{\sigma(d_i) \mid 0 \leq i < k \text{ and } d_k \prec d_i\}$. Set $\sigma(d_k) := q$.

It is readily checked that σ is as required. ❑

In the succeeding sections, we deal with the satisfiability of constraint graphs over $\mathbb{Q}$. However, all obtained results also apply if we choose $\mathbb{R}$ instead. Note that Theorem 10 does *not* hold if satisfiability over $\mathbb{N}$ is considered due to the absence of density: If there exist two nodes v_1 and v_2 such that the length of $<$-paths (which are defined in the obvious way) between v_1 and v_2 is unbounded, a constraint graph is unsatisfiable over $\mathbb{N}$ even if it contains no $<$-cycle. Figure 6 shows such a constraint graph.

4.2 A $\mathcal{TDL}$ Normal Form

The decidability procedure works on $\mathcal{TDL}$-concepts and TBoxes of a certain syntactic form. This greatly simplifies some constructions like defining Hintikka-trees. Let us first introducing the well-known negation normal form.

Definition 11 (NNF). A concept C is in *negation normal form* (NNF) if negation occurs only in front of concept names. Exhaustive application of the following rewrite rules translates concepts to equivalent concepts in NNF.

$$\begin{array}{rcl} \neg\neg C & \Longrightarrow & C \\ \neg(C \sqcap D) & \Longrightarrow & \neg C \sqcup \neg D \\ \neg(\exists R.C) & \Longrightarrow & (\forall R.\neg C) \\ \neg(\exists u_1, u_2.P) & \Longrightarrow & \exists u_1, u_2.\widetilde{P} \sqcup \exists u_2, u_1.< \sqcup u_1\uparrow \sqcup u_2\uparrow \end{array} \qquad \begin{array}{rcl} \neg(C \sqcup D) & \Longrightarrow & \neg C \sqcap \neg D \\ \neg(\forall R.C) & \Longrightarrow & (\exists R.\neg C) \\ \neg(g\uparrow) & \Longrightarrow & \exists g, g.= \end{array}$$

where $\widetilde{\cdot}$ denotes the exchange of predicates, i.e., $\widetilde{<}$ is $=$ and $\widetilde{=}$ is $<$. With $\mathsf{nnf}(C)$, we denote the equivalent of C in NNF which can be obtained by applying the above rules. Furthermore, we use $\sim C$ as a shorthand for $\mathsf{nnf}(\neg C)$. A TBox $\mathcal{T}$ is in NNF iff all concepts in $\mathcal{T}$ are in NNF.

We can now extend NNF to an even more convenient normal form.

Definition 12 (Path Normal Form). A $\mathcal{TDL}$-concept C is in *path normal form* (PNF) iff it is in NNF and, for all subconcepts $\exists u_1, u_2.P$ of C, we have either

1. $u_1 = g_1$ and $u_2 = g_2$ for some $g_1, g_2 \in N_{cF}$,
2. $u_1 = fg_1$ and $u_2 = g_2$ for some $f \in N_{aF}$ and $g_1, g_2 \in N_{cF}$, or
3. $u_1 = g_1$ and $u_2 = fg_2$ for some $f \in N_{aF}$ and $g_1, g_2 \in N_{cF}$.

A $\mathcal{TDL}$ TBox $\mathcal{T}$ is in path normal form iff it is in NNF and all concepts appearing in $\mathcal{T}$ are in PNF.

The following lemma shows that it is not a restriction to consider only concepts and TBoxes in PNF.

Lemma 13. *Satisfiability of $\mathcal{TDL}$-concepts w.r.t. $\mathcal{TDL}$-TBoxes can be reduced to satisfiability of $\mathcal{TDL}$-concepts in PNF w.r.t. $\mathcal{TDL}$-TBoxes in PNF.*

Proof We start with defining a function ρ that converts $\mathcal{TDL}$-concepts (resp. $\mathcal{TDL}$-TBoxes) to $\mathcal{TDL}$-concepts (resp. $\mathcal{TDL}$-TBoxes) containing only paths of a restricted length. This mapping will then be used to convert $\mathcal{TDL}$-concepts and TBoxes into PNF.

Let C be a $\mathcal{TDL}$-concept. For every path $u = f_1 \cdots f_n g$ used in C, we assume that $[g], [f_n g], \ldots, [f_1 \cdots f_n g]$ are concrete features. We inductively define a mapping λ from paths u in C to concepts as follows:

$$\begin{array}{rcl} \lambda(g) & = & \top \\ \lambda(fu) & = & (\exists [fu], f[u].=) \sqcap \exists f.\lambda(u) \end{array}$$

Now, $\rho(C)$ is obtained from C by replacing all subconcepts $\exists u_1, u_2.P$ of C with $\exists[u_1], [u_2].P \sqcap \lambda(u_1) \sqcap \lambda(u_2)$ and $g\uparrow$ with $[g]\uparrow$. Moreover, if

$$\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_k \sqsubseteq D_k\},$$

is a $\mathcal{TDL}$-TBox, then

$$\rho(\mathcal{T}) = \{\rho(C_1) \sqsubseteq \rho(D_1), \ldots, \rho(C_k) \sqsubseteq \rho(D_k)\}.$$

Now let C be a $\mathcal{TDL}$-concept and $\mathcal{T}$ a $\mathcal{TDL}$-TBox. By Definition 11, we can convert C to a concept C' in NNF and $\mathcal{T}$ to a concept $\mathcal{T}'$ in NNF such that C is satisfiable w.r.t. $\mathcal{T}$ iff C' is satisfiable w.r.t. $\mathcal{T}'$. Moreover, we can clearly translate C' to $\rho(C')$ and $\mathcal{T}'$ to $\rho(\mathcal{T}')$ in polynomial time and, obviously, $\rho(C')$ and $\rho(\mathcal{T}')$ are in PNF. Hence, it remains to show that C' is satisfiable w.r.t. $\mathcal{T}'$ iff $\rho(C')$ is satisfiable w.r.t. $\rho(\mathcal{T}')$.

First for the "if" direction. Let $\mathcal{I}$ be a model for $\rho(C')$ and $\rho(\mathcal{T}')$. We extend $\mathcal{I}$ to an interpretation $\mathcal{J}$ by setting $g^{\mathcal{J}} := [g]^{\mathcal{I}}$ for all concrete features g used in C' or $\mathcal{T}'$. It is not hard to show by structural induction that, for all subconcepts D of C' or $\mathcal{T}'$ and all $a \in \Delta_{\mathcal{I}}$, we have $a \in \rho(D)^{\mathcal{I}} \rightarrow a \in D^{\mathcal{J}}$. Since C' and $\mathcal{T}'$ are in NNF, the only non-trivial cases are:

- $D = \exists u_1, u_2.P$. Then $\rho(D) = \exists[u_1], [u_2].P \sqcap \lambda(u_1) \sqcap \lambda(u_2)$. For $i \in \{1, 2\}$, let $u_i = f_1^{(i)}, \ldots, f_{k_i}^{(i)} g_i$. It is easy to show by induction on n that, for each $i \in \{1, 2\}$ and every n with $1 \leq n \leq k_i$, there exist $b_1, \ldots, b_n \in \Delta_{\mathcal{I}}$ such that $(a, b_1) \in (f_1^{(i)})^{\mathcal{I}}$ and $[u_i]^{\mathcal{I}}(a) = [f_2^{(i)}, \ldots, f_{k_i}^{(i)} g_i]^{\mathcal{I}}(b_1)$, and, for each $1 < j \leq n$, we have $(b_{j-1}, b_j) \in (f_j^{(i)})^{\mathcal{I}}$ and $[f_j^{(i)}, \ldots, f_{k_i}^{(i)} g_i]^{\mathcal{I}}(b_{j-1}) = [f_{j+1}^{(i)}, \ldots, f_{k_i}^{(i)} g_i]^{\mathcal{I}}(b_j)$. Hence, for each $i \in \{1, 2\}$, we have $f_{k_i}^{(i)}(\cdots(f_1^{(i)}(a))\cdots) = b_{k_i}$ and $[u_i]^{\mathcal{I}}(a) = [g_i]^{\mathcal{I}}(b_{k_i})$. Since $\exists[u_1], [u_2].P$ is a conjunct of $\rho(D)$, it is thus clear that $[g_1]^{\mathcal{I}}(b_{k_1})\, P\, [g_2]^{\mathcal{I}}(b_{k_2})$. It is now immediate by definition of $\mathcal{J}$ that $a \in (\exists u_1, u_2.P)^{\mathcal{J}}$.

- $D = g\uparrow$. Then $\rho(D) = [g]\uparrow$. Obvious by definition of $\mathcal{J}$.

It is easily seen that the claim just proved by induction implies that $\mathcal{J}$ is a model for C' and $\mathcal{T}'$.

Now for the "only if" direction. Let $\mathcal{I}$ be a model for C' and $\mathcal{T}'$. We extend $\mathcal{I}$ to an interpretation $\mathcal{J}$ by, for each path u used in C' and $\mathcal{T}'$, and each postfix $u' = f_1 \cdots f_k g$ of u, setting $[f_1 \cdots f_k g]^{\mathcal{J}} := (u')^{\mathcal{I}}$. It is not hard to show by structural induction that, for all subconcepts D of C' or $\mathcal{T}'$ and all $a \in \Delta_{\mathcal{I}}$, we have $a \in D^{\mathcal{I}} \rightarrow a \in \rho(D)^{\mathcal{J}}$. The only non-trivial cases are the same as in the "if" direction. However, both cases are straightforward by definition of ρ and $\mathcal{J}$. Thus, $\mathcal{J}$ is clearly a model for $\rho(C')$ and $\rho(\mathcal{T}')$. ❑

Hence, it su ces to prove that satisfiability of concepts in PNF w.r.t. TBoxes in PNF is decidable. In what follows, we generally assume that all concepts and TBoxes are in path normal form. We will often refer to TBoxes $\mathcal{T}$ in their *concept form* $C_{\mathcal{T}}$ which is defined as follows:

$$C_{\mathcal{T}} = \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \mathsf{nnf}(\neg C \sqcup D).$$

4.3 Defining Hintikka-trees

In this section, we define Hintikka-trees for $\mathcal{TDL}$-concepts C and TBoxes $\mathcal{T}$ in path normal form and show that there exists Hintikka-tree for C and $\mathcal{T}$ iff there exists a model for C and $\mathcal{T}$.

Let C be a concept and $\mathcal{T}$ be a TBox. With $\mathsf{cl}(C, \mathcal{T})$, we denote the set of subconcepts of C and $C_{\mathcal{T}}$. We assume that existential concepts $\exists R.D$ in $\mathsf{cl}(C, \mathcal{T})$ with $R \in N_R \setminus N_{aF}$ are linearly ordered, and that $\mathcal{E}(C, \mathcal{T}, i)$ yields the i-th existential concept in $\mathsf{cl}(C, \mathcal{T})$. Furthermore, we assume the abstract features used in $\mathsf{cl}(C, \mathcal{T})$ to be linearly ordered and use $\mathcal{F}(C, \mathcal{T}, i)$ to denote the i-th abstract feature in $\mathsf{cl}(C, \mathcal{T})$. The set of concrete features used in $\mathsf{cl}(C, \mathcal{T})$ is denoted with $\mathcal{G}(C, \mathcal{T})$.

We now define Hintikka-pairs which are used as labels of the nodes in Hintikka-trees (recall that Hintikka-trees are abstractions of models).

Definition 14 (Hintikka-set, Hintikka-pair). Let C be a concept and $\mathcal{T}$ be a TBox. A set $\Psi \subseteq \mathsf{cl}(C, \mathcal{T})$ is a *Hintikka-set for* $(C, \mathcal{T})$ iff it satisfies the following conditions:

(H1) $C_{\mathcal{T}} \in \Psi$,

(H2) if $C_1 \sqcap C_2 \in \Psi$, then $\{C_1, C_2\} \subseteq \Psi$,

(H3) if $C_1 \sqcup C_2 \in \Psi$, then $\{C_1, C_2\} \cap \Psi \neq \emptyset$,

(H4) $\{A, \neg A\} \not\subseteq \Psi$ for all concept names $A \in \mathsf{cl}(C, \mathcal{T})$,

(H5) if $g\uparrow \in \Psi$, then $\exists u_1, u_2.P \notin \Psi$ for all concepts $\exists u_1, u_2.P$ with $u_1 = g$ or $u_2 = g$.

We say that $f \in N_{aF}$ is *enforced* by a Hintikka-set Ψ iff either $\exists f.C \in \Psi$ for some concept C or $\{\exists f g_1, g_2.P, \exists g_1, f g_2.P\} \cap \Psi \neq \emptyset$ for some $g_1, g_2 \in N_{cF}$ and $P \in \{<, =\}$. A *Hintikka-pair* (Ψ, χ) *for* $(C, \mathcal{T})$ consists of a Hintikka-set Ψ for $(C, \mathcal{T})$ and a set χ of tuples (g_1, g_2, P) with $g_1, g_2 \in \mathcal{G}(C, \mathcal{T})$ such that

(H6) if $(g_1, g_2, P) \in \chi$, then $\{g_1\uparrow, g_2\uparrow\} \cap \Psi = \emptyset$.

With $\Gamma_{(C,\mathcal{T})}$, we denote the set of all Hintikka-pairs for $(C, \mathcal{T})$. A path u is *enforced* by (Ψ, χ) iff either u appears in χ or $\{\exists u, u'.P, \exists u', u.P\} \cap \Psi \neq \emptyset$ for some path u' and $P \in \{<, =\}$.[1]

Intuitively, each node α of a (yet to be defined) Hintikka-tree T corresponds to a domain object a of the corresponding canonical model $\mathcal{I}$. The first component Ψ_α of the Hintikka-pair labelling α is the set of concepts from $\mathsf{cl}(C, \mathcal{T})$ satisfied by a. The second component χ_α states restrictions on the relationship between concrete successors of a. If, for example, $(g_1, g_2, <) \in \chi_\alpha$, then we must have $g_1^{\mathcal{I}}(a) < g_2^{\mathcal{I}}(a)$. Note that the restrictions in χ_α are independent from concepts $\exists g_1, g_2.P \in \Psi_\alpha$. As will become clear when Hintikka-trees are defined, the restrictions in χ_α are used to

[1] Note: by definition of Hintikka-pairs, the path u has length 1 if it appears in χ, and, since all concepts are in path normal form, u has length 1 or 2 if $\{\exists u, u'.P, \exists u', u.P\} \cap \Psi \neq \emptyset$ for some u' and P.

ensure that the constraint graph induced by the Hintikka-tree T, which describes the concrete part of the model $\mathcal{I}$, does not contain a $<$-cycle, i.e., that it is satisfiable. This induced constraint graph can be thought of as the union of smaller constraint graphs, each one being described by a Hintikka-pair labelling a node in T. These pair-graphs are defined next.

Definition 15 (Pair-graph). Let C be a concept, $\mathcal{T}$ a TBox, and $p = (\Psi, \chi)$ a Hintikka-pair for $(C, \mathcal{T})$. The *pair-graph* $G(p) = (V, E)$ of p is a constraint graph defined as follows:

1. V is the set of paths enforced by p
2. $E = \chi \cup \{(u_1, u_2, P) \mid \exists u_1, u_2.P \in \Psi\}$.

An *edge extension* of $G(p)$ is a set $E' \subseteq V \times V \times \{<, =\}$ such that for all $fg_1, fg_2 \in V$, we have either $(fg_2, fg_1, <) \in E'$ or $(fg_1, fg_2, P) \in E'$ for some $P \in \{<, =\}$. If E' is an edge extension of $G(p)$, then the graph $(V, E \cup E')$ is a *completion* of $G(p)$.

Note that, since all concepts are in path normal form and due to the definitions of Hintikka-pairs and pair-graphs, we have $E' \cap E = \emptyset$ for every edge extension E' of a pair-graph (V, E). As all constraint graphs, we assume pair-graphs to be equality closed.

We briefly comment on the connection of completions and the χ-component of Hintikka-pairs. Let α and β be nodes in a Hintikka-tree T and let a and b be the corresponding domain objects in the corresponding model $\mathcal{I}$. Edges in Hintikka-trees represent role-relationships, i.e., if β is successor of α in T, then there exists an $R \in N_R$ such that $(a, b) \in R^{\mathcal{I}}$. Assume β is successor of α and the edge between α and β represents relationship via the abstract feature f, i.e., we have $f^{\mathcal{I}}(a) = b$. The second component χ_β of the Hintikka-pair labelling β fixes the relationships between all concrete successors of b that "a talks about". For example, if $(\exists fg_1, g_2.=) \in \Psi_\alpha$ and $(\exists fg_3, g_2. <) \in \Psi_\alpha$, where Ψ_α is the firsto component of the Hintikka-pair labelling α, then "a talks about" the concrete g_1-successor and the concrete g_3-successor of b. Hence, χ_β either contains $(g_3, g_1, <)$ or (g_1, g_3, P) for some $P \in \{<, =\}$. This is formalized by demanding that the pair-graph $G(T(\alpha))$ of the Hintikka-pair labelling α together with all the edges from the χ-components of the successors of α are a completion of $G(T(\alpha))$. Moreover, this completion has to be satisfiable, which is necessary to ensure that the constraint graph induced by T does not contain a $<$-cycle. An appropriate way of thinking about the χ-components is as follows: at α, a completion of $G(T(\alpha))$ is "guessed". The additional edges are then "recorded" in the χ-components of the successor-nodes of α. We now define Hintikka-trees formally.

Definition 16 (Hintikka-tree). Let C be a concept, $\mathcal{T}$ be a TBox, k the number of existential subconcepts in $\mathsf{cl}(C, \mathcal{T})$, and ℓ be the number of abstract features in $\mathsf{cl}(C, \mathcal{T})$. A $1 + k + \ell$-tuple of Hintikka-pairs $(p_0, \ldots, p_{k+\ell})$ with $p_i = (\Psi_i, \chi_i)$ and $G(p_0) = (V, E)$ is called *matching* iff

(H7) if $\exists R.D \in \Psi_0$ and $\mathcal{E}(C, \mathcal{T}, i) = \exists R.D$, then $D \in \Psi_i$

(H8) if $\{\exists R.D, \forall R.E\} \subseteq \Psi_0$ and $\mathcal{E}(C, \mathcal{T}, i) = \exists R.D$, then $E \in \Psi_i$

(H9) if $\exists f.D \in \Psi_0$ and $\mathcal{F}(C, \mathcal{T}, i) = f$, then $D \in \Psi_{k+i}$.

(H10) if f is enforced by Ψ_0, $\mathcal{F}(C, \mathcal{T}, i) = f$, and $\forall f.D \in \Psi_0$, then $D \in \Psi_{k+i}$.

(H11) the constraint graph $(V, E \cup E')$, where

$$E' = \bigcup_{1 \leq i \leq \ell} \{(fg_1, fg_2, P) \mid \mathcal{F}(C, \mathcal{T}, i) = f, (g_1, g_2, P) \in \chi_{k+i}\}$$

is a satisfiable completion of $G(p_0)$.

A $k + \ell$-ary $\Gamma_{(C,\mathcal{T})}$-tree T is a *Hintikka-tree for* $(C, \mathcal{T})$ iff $T(\alpha)$ is a Hintikka-pair for $(C, \mathcal{T})$ for each node α in T, and T satisfies the following conditions:

(H12) $C \in \Psi_\epsilon$, where $T(\epsilon) = (\Psi_\epsilon, \chi_\epsilon)$,

(H13) for all $\alpha \in \{1, \ldots, k + \ell\}^*$, the tuple $(T(\alpha), T(\alpha 1), \ldots, T(\alpha j))$ with $j = k + \ell$ is matching.

For a Hintikka-tree T and node $\alpha \in \{1, \ldots, k + \ell\}^*$ with $T(\alpha) = (\Psi, \chi)$, we use $T_\triangleleft(\alpha)$ to denote Ψ and $T_\triangleright(\alpha)$ to denote χ. Moreover, if $G(\alpha) = (V, E)$, we use $\mathsf{cpl}(T, \alpha)$ to denote the constraint graph $(V, E \cup E')$ as defined in **(H11)**.[2]

Whereas most properties of Hintikka-trees deal with concepts, roles, and abstract features and are hardly surprising, **(H11)** ensures that constraint graphs induced by Hintikka-trees contain no $<$-cycle. By "guessing" a completion as explained above, possible $<$-cycles are anticipated and can be detected locally, i.e., it then su ces to check that the completions $\mathsf{cpl}(T, \alpha)$ are satisfiable as demanded by **(H11)**. Indeed, it is crucial that the cycle detection is done by a *local* condition since we need to define an automaton which accepts exactly Hintikka-trees and automata work locally. It is worth noting that the localization of cycle detection as expressed by **(H11)** crucially depends on path normal form.

The following two lemmas show that Hintikka-trees are appropriate abstractions of models. This lemma is central since, as we will see, defining looping automata accepting exactly Hintikka-trees is a straightforward task.

Lemma 17. *A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if there exists a Hintikka-tree for $(C, \mathcal{T})$.*

Proof Let C be a concept, $\mathcal{T}$ a TBox, and k and ℓ as in Definition 16. Moreover, let T be a Hintikka-tree for $(C, \mathcal{T})$. We define an interpretation $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ as follows:

$$\begin{array}{rcl}
\Delta_\mathcal{I} & = & \{1, \ldots, k + \ell\}^* \\
A^\mathcal{I} & = & \{\alpha \mid A \in T_\triangleleft(\alpha)\} \text{ for all } A \in C_N \\
R^\mathcal{I} & = & \{(\alpha, \beta) \mid \beta = \alpha i \text{ and } \mathcal{E}(C, \mathcal{T}, i) = \exists R.E \in T_\triangleleft(\alpha)\} \text{ for all } R \in N_R \setminus N_{aF} \\
f^\mathcal{I} & = & \{(\alpha, \beta) \mid \beta = \alpha i,\ \mathcal{F}(C, \mathcal{T}, i - k) = f, \text{ and } f \text{ is enforced by } T_\triangleleft(\alpha)\} \\
 & & \text{for all } f \in N_{aF}
\end{array}$$

[2] more precisely its equality closure.

It remains to define the interpretation of concrete features. We define an (infinite) constraint graph $G(T)$ induced by T, show that $G(T)$ is satisfiable, and define the interpretation of concrete features from a solution of $G(T)$. The nodes of $G(T)$ have the form $\alpha|u$, where α is a node in T and u is a path in C or $\mathcal{T}$. More precisely, $G(T)$ is defined as (V, E), where

1. $V = \{\alpha|u \mid \alpha \in \{1, \ldots, k+\ell\}^*, u \text{ appears in } C \text{ or } \mathcal{T}\}$
2. $E = \bigcup_{\alpha \in \{1,\ldots,k+\ell\}^*} \{(\alpha|u, \alpha|u', P) \mid (u, u', P) \in \mathsf{cpl}(T, \alpha)\}$
   $\cup \{(\alpha|fg), \alpha i|g, =) \mid \mathcal{F}(C, \mathcal{T}, i-k) = f,\ fg \text{ is a node in } \mathsf{cpl}(T, \alpha)\}$

As always, we assume that $G(T)$ is equality closed. It is not hard to see that $G(T)$ really is a constraint graph, i.e., the node set of $G(T)$ is countable. Next, we show the following claim:

Claim 1: $G(T)$ is satisfiable.

Proof: By Theorem 10, it su ces to show that $G(T)$ contains no $<$-cycle. Assume to the contrary that $G(T)$ contains a $<$-cycle and that $O = \alpha_0|u_0, \ldots, \alpha_{n-1}|u_{n-1}$ is the $<$-cycle in $G(T)$ with minimal length. Fix a t with $0 \leq t < n$ such that

$$\text{for each } i \text{ with } 0 \leq i < n \text{ and each } \beta \in \{1, \ldots, k+\ell\}^+, \text{ we have } \alpha_i \neq \alpha_t\beta, \qquad (*)$$

i.e., there exist no α_i in O such that α_t is a true prefix of α_i (such a t exists since O is of finite length). Since O is a $<$-cycle, there exists an s with $0 \leq s < n$ such that $(\alpha_s|u_s, \alpha_{s\oplus_n 1}|u_{s\oplus_n 1}, <) \in E$. We make a case distinction and derive a contradiction in either case.

- $\alpha_s \neq \alpha_t$. Define a sequence of nodes O' from O by deleting all nodes $\alpha_i|u_i$ with $\alpha_i = \alpha_t$. O' is non-empty since $\alpha_s \neq \alpha_t$. We show that O' is a $<$-cycle in $G(T)$ which is a contradiction to the minimality of O. Let $O' = \alpha'_0|u'_0, \ldots, \alpha'_{m-1}|u'_{m-1}$. By definition of $G(T)$, the fact that $(\alpha_s|u_s, \alpha_{s\oplus_n 1}|u_{s\oplus_n 1}, <) \in E$ implies $\alpha_{s\oplus_n 1} = \alpha_s$. Since $\alpha_s \neq \alpha_t$, $\alpha_s|u_s$ and $\alpha_{s\oplus_n 1}|u_{s\oplus_n 1}$ are in O' and it remains to show that O' is a cycle in $G(T)$, i.e., for all i with $0 \leq i < m$, we have $(\alpha'_i|u'_i, \alpha'_{i\oplus_m 1}|u'_{i\oplus_m 1}, P) \in E$ for some $P \in \{<, =\}$.

  Let $\alpha'_i|u'_i$ and $\alpha'_{i\oplus_m 1}|u'_{i\oplus_m 1}$ be nodes in O'. If these two nodes are already neighbor nodes in O, we are obviously done. Hence, assume that this is not the case. By construction of O', this implies the existence of a path

  $$\alpha'_i|u'_i, \alpha_t|u^*_1, \ldots, \alpha_t|u^*_x, \alpha'_{i\oplus_m 1}|u'_{i\oplus_m 1}$$

  in $G(T)$ which is a subpath of O.[3] Since $\alpha'_i \neq \alpha_t$ and $\alpha'_{i\oplus_m 1} \neq \alpha_t$, by construction of $G(T)$ and by $(*)$, this implies that

  1. there exists a $\beta \in \{1, \ldots, k+\ell\}^*$ such that $\alpha'_i = \alpha'_{i\oplus_m 1} = \beta$,
  2. there exists an $f \in N_{aF}$ such that $\alpha_t = \beta j$ where $\mathcal{F}(C, \mathcal{T}, j-k) = f$,

[3] Where "subpath" is defined in the obvious way.

3. $u'_i = fg$, $u^*_1 = g$, $u^*_x = g'$, and $u'_{i\oplus_m 1} = fg'$ for some $g, g' \in \mathcal{G}(C, \mathcal{T})$, and
4. $(\beta|fg, \beta j|g, =) \in E$ and $(\beta|fg', \beta j|g', =) \in E$.

By definition of $G(T)$ and by Point 4, both fg and fg' are nodes in $\mathsf{cpl}(T, \beta) = (V', E')$. By definition of cpl, this implies that either

(a) $(fg', fg, <) \in E'$ or
(b) $(fg, fg', P) \in E'$ for some $P \in \{<, =\}$.

Together with Point 1 and 3 and the definition of $G(T)$, (b) obviously implies $(\alpha'_i|u'_i, \alpha'_{i\oplus_m 1}|u'_{i\oplus_m 1}, P) \in E$ and we are done. Moreover, in the following we show that case (a) cannot occur.

Let $\mathsf{cpl}(\beta j) = (V'', E'')$. In case (a), we have $(g', g, <) \in E''$: Let $G(\beta) = (V'_*, E'_*)$; by definition of pair-graphs and since all concepts are in path normal form, $(fg', fg, <) \in E'$ implies $(fg', fg, <) \in E' \setminus E'_*$; by definition of cpl and by Point 2, this means that $(g', g, <) \in T_\triangleright(\beta)$. Hence, $(g', g, <) \in E''$. By definition of $G(T)$ and Point 1 and 3, $(g', g, <) \in E''$ implies that we have $(\alpha_t|u^*_x, \alpha_t|u^*_1, <) \in E$. Hence, the path $\alpha_t|u^*_1, \ldots, \alpha_t|u^*_x$ is a $<$-cycle in $G(T)$ which contradicts the minimality of O since this path is a true subpath of O.

- $\alpha_s = \alpha_t$. Define a sequence of nodes O' from O by deleting all nodes $\alpha_i|u_i$ with $\alpha_i \neq \alpha_t$. O' is non-empty since $\alpha_s = \alpha_t$. We show that O' is a $<$-cycle in $G(T)$ which is a contradiction to the minimality of O. Let $O' = \alpha_t|u'_0, \ldots, \alpha_t|u'_{m-1}$. By definition of $G(T)$, the fact that $(\alpha_s|u_s, \alpha_{s\oplus_n 1}|u_{s\oplus_n 1}, <) \in E$ implies $\alpha_{s\oplus_n 1} = \alpha_s = \alpha_t$. Hence, it remains to show that O' is a cycle in $G(T)$, i.e., that, for all i with $0 \leq i < m$, we have $(\alpha_t|u'_i, \alpha_t|u'_{i\oplus_m 1}, P) \in E$ for some $P \in \{<, =\}$.

  Let $\alpha_t|u'_i$ and $\alpha_t|u'_{i\oplus_m 1}$ be nodes in O'. If these two nodes are already neighbor nodes in O, we are obviously done. Hence, assume that this is not the case. By construction of O', this implies the existence of a subpath

$$\alpha_t|u'_i, \alpha^*_1|u^*_1, \ldots, \alpha^*_x|u^*_x, \alpha_t|u'_{i\oplus_m 1}$$

  of O in $G(T)$ with $\alpha^*_i \neq \alpha_t$ for $1 \leq i \leq x$. By construction of $G(T)$ and by $(*)$, this implies that

  1. there exists a $\beta \in \{1, \ldots, k + \ell\}^*$ such that $\alpha^*_1 = \alpha^*_x = \beta$,
  2. there exists an $f \in N_{aF}$ such that $\alpha_t = \beta j$ where $\mathcal{F}(C, \mathcal{T}, j - k) = f$,
  3. $u'_i = g$, $u^*_1 = fg$, $u^*_x = fg'$, and $u'_{i\oplus_m 1} = g'$ for some $g, g' \in \mathcal{G}(C, \mathcal{T})$, and
  4. $(\beta j|g, \beta|fg, =) \in E$ and $(\beta|fg', \beta j|g', =) \in E$.

  By definition of $G(T)$ and by Point 4, both fg and fg' are nodes in $\mathsf{cpl}(T, \beta) = (V', E')$.[4] By definition of cpl, this implies that either

  (a) $(fg', fg, <) \in E'$ or

[4] Here we exploit that $G(T)$ is equality closed.

(b) $(fg, fg', P) \in E'$ for some $P \in \{<, =\}$.

Together with Point 1 and 3 and the definition of $G(T)$, (a) obviously implies $(\alpha_x^*|u_x^*, \alpha_1^*|u_1^*, <) \in E$. Hence, the path $\alpha_1^*|u_1^*, \ldots, \alpha_x^*|u_x^*$ is a $<$-cycle in $G(T)$ which contradicts the minimality of O since this path is a true subpath of O.

Hence, let us assume that (b) holds. Moreover, let $\mathsf{cpl}(\beta j) = (V'', E'')$. We have $(g, g', <) \in E''$: Let $G(\beta) = (V'_*, E'_*)$; by definition of pair-graphs and since all concepts are in path normal form, $(fg, fg', P) \in E'$ implies $(fg, fg', P) \in E' \setminus E'_*$; by definition of cpl and by Point 2, this means that $(g, g', P) \in T_\triangleright(\beta)$. Hence, $(g, g', P) \in E''$. By definition of $G(T)$ and Point 1 and 3, $(g, g', P) \in E''$ implies that we have $(\alpha_t|u'_i, \alpha_t|u'_{i\oplus_m 1}, P) \in E$ what was to be shown.

This finishes the proof of Claim 1. We may now define the interpretation of concrete features. Let be a solution for $G(T)$. We set

$$g^{\mathcal{I}} \quad = \quad \{(\alpha, x) \mid g \text{ is enforced by } T(\alpha) \text{ and } \ (\alpha|g) = x\} \text{ for all } g \in N_{cF}.$$

To show that there exists an $a \in \Delta_{\mathcal{I}}$ such that $a \in C^{\mathcal{I}}$, we prove the following claim:

Claim 2: $D \in T_\triangleleft(\alpha)$ implies $\alpha \in D^{\mathcal{I}}$ for all $\alpha \in \Delta_{\mathcal{I}}$ and $D \in \mathsf{cl}(C, \mathcal{T})$.

Proof: The claim is proved by induction over the structure of D. The induction start, i.e., the case that D is a concept name, is an immediate consequence of the definition of $\mathcal{I}$. For the induction step, we make a case distinction according to the topmost operator in D. Assume $D \in T_\triangleleft(\alpha)$.

- $D = \neg E$. Since C is in NNF and by definition of $\mathsf{cl}()$, D is in NNF. Hence, E is a concept name. By definition of $\mathcal{I}$ and since $T(\alpha)$ is a Hintikka-set and thus satisfies **(H4)**, we have $\alpha \in (\neg E)^{\mathcal{I}}$.

- $D = C_1 \sqcap C_2$ or $D = C_1 \sqcup C_2$. Straightforward by **(H2)** and **(H3)** of Hintikka-sets and by induction hypothesis.

- $D = \exists R.E$ with $R \in N_R \setminus N_{aF}$. By definition of $R^{\mathcal{I}}$, we have $(\alpha, \beta) \in R^{\mathcal{I}}$ for $\beta = \alpha i$ and $\mathcal{E}(C, \mathcal{T}, i) = \exists R.E$. By **(H7)**, we have $E \in T_\triangleleft(\beta)$, and, by induction, $\beta \in E^{\mathcal{I}}$.

- $D = \exists f.E$ with $f \in N_{aF}$. Hence, f is enforced by $T_\triangleleft(\alpha)$. By definition of $f^{\mathcal{I}}$, we have $f^{\mathcal{I}}(\alpha) = \beta$ for $\beta = \alpha i$ and $\mathcal{F}(C, \mathcal{T}, i - k) = f$. By **(H9)**, we have $E \in T_\triangleleft(\beta)$, and, by induction, $\beta \in E^{\mathcal{I}}$.

- $D = \forall R.E$ with $R \in N_R \setminus N_{aF}$. Let $(\alpha, \beta) \in R^{\mathcal{I}}$. By definition of $R^{\mathcal{I}}$, there exists an i such that $\mathcal{E}(C, \mathcal{T}, i) = \exists R.D \in T_\triangleleft(\alpha)$ and $\beta = \alpha i$. By **(H8)**, we have $E \in T_\triangleleft(\beta)$, and, by induction, $\beta \in E^{\mathcal{I}}$. Since this holds independently of the choice of β, we have $\alpha \in (\forall R.E)^{\mathcal{I}}$.

- $D = \forall f.E$ with $f \in N_{aF}$. Let $f^{\mathcal{I}}(\alpha) = \beta$. By definition of $f^{\mathcal{I}}$, we have $\beta = \alpha i$, $\mathcal{F}(C, \mathcal{T}, i - k) = f$, and f is enforced by $T_\triangleleft(\alpha)$. By **(H10)**, we have $E \in T_\triangleleft(\beta)$, and, by induction, $\beta \in E^{\mathcal{I}}$.

- $D = \exists u_1, u_2.P$. Let $G(T) = (V, E)$ and $\mathsf{cpl}(T, \alpha) = (V', E')$. By definition of pair-graphs and $\mathsf{cpl}()$, we have $(u_1, u_2, P) \in E'$. We show that there exist nodes $n_1, n_2 \in V$ such that $(n_1, n_2, P) \in E$, $u_1^{\mathcal{I}}(\alpha) = (n_1)$, and $u_2^{\mathcal{I}}(\alpha) = (n_2)$. Since is a solution for $G(T)$, this implies $u_1^{\mathcal{I}}(\alpha) P u_2^{\mathcal{I}}(\alpha)$.

  For $i \in \{1, 2\}$, set $n_i := \alpha | u_i$. By definition of $G(T)$ and since $(u_1, u_2, P) \in E'$, we have $(n_1, n_2, P) \in E$. Fix an $i \in \{1, 2\}$. We need to show that $u_i^{\mathcal{I}}(\alpha) = (n_i)$. In the case $u_i = g$, this is obvious by definition of $g^{\mathcal{I}}$ (since g is obviously enforced by $T(\alpha)$).

  Hence let $u_i = fg$ and $\mathcal{F}(C, \mathcal{T}, j - k) = f$. Since fg is a node in $\mathsf{cpl}(T, \alpha)$, we have $(\alpha | fg, \alpha j | g, =) \in E$. Hence, $(\alpha j | g) = (\alpha | fg)$. By definition of $f^{\mathcal{I}}$ and since f is clearly enforced by $T_{\lhd}(\alpha)$, we have $f^{\mathcal{I}}(\alpha) = \alpha j$. By definition of cpl and of pair-graphs, $fg \in V'$ implies that g appears in $T_{\rhd}(\alpha j)$: Since $\mathsf{cpl}(T, \alpha)$ is both a completion of $G(\alpha)$ and satisfiable, $fg \in V'$ implies $(fg, fg, =) \in E'$; due to the definition of pair graphs and since all concepts are in path normal form, $(fg, fg, =)$ is not an edge of $G(\alpha)$; hence, by definition of cpl and since $\mathcal{F}(C, \mathcal{T}, j - k) = f$, we must have $(g, g, =) \in T_{\rhd}(\alpha j)$, i.e., g appears in $T_{\rhd}(\alpha j)$. Since g appears in $T_{\rhd}(\alpha j)$ and thus enforced by $T(\alpha j)$, we have $g^{\mathcal{I}}(\alpha j) = (\alpha j | g)$ by definition of $g^{\mathcal{I}}$. Summing up, $(fg)^{\mathcal{I}}(\alpha) = (\alpha j | g) = (\alpha | fg)$.

- $D = g{\uparrow}$. If $g^{\mathcal{I}}(\alpha)$ is defined, then g is enforced by $T(\alpha)$. We show that this implies $g{\uparrow} \notin T_{\lhd}(\alpha)$. If g is enforced by $T(\alpha)$, then either (i) g appears in $T_{\rhd}(\alpha)$ or (ii) $\{\exists g, u'.P, \exists u', g.P\} \cap T_{\lhd}(\alpha) \neq \emptyset$ for some path u' and $P \in \{<, =\}$. In case (i), **(H6)** yields $g{\uparrow} \notin T_{\lhd}(\alpha)$. In case (ii), **(H5)** yields the same result.

This completes the proof of the claim. Since $C \in T_{\lhd}(\epsilon)$ by **(H12)** and, for all $\alpha \in \Delta_{\mathcal{I}}$, we have $C_{\mathcal{T}} \in T_{\lhd}(\alpha)$ by **(H1)**, it is an immediate consequence of the semantics and Claim 2 that $\mathcal{I}$ is a model of C w.r.t. $\mathcal{T}$. ❑

Lemma 18. *A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ only if there exists a Hintikka-tree for $(C, \mathcal{T})$.*

Proof Let C be a concept, $\mathcal{T}$ a TBox, and k and ℓ as in Definition 16. Moreover, let $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model for C w.r.t. $\mathcal{T}$, i.e., there exists an $a_0 \in \Delta_{\mathcal{I}}$ such that $a_0 \in C^{\mathcal{I}}$ and $D^{\mathcal{I}} \subseteq E^{\mathcal{I}}$ for all $D \sqsubseteq E \in \mathcal{T}$. We inductively define a Hintikka-tree T for $(C, \mathcal{T})$, i.e., a $k + \ell$-ary $\Gamma_{(C,\mathcal{T})}$-tree that satisfies **(H12)** and **(H13)**. Along with T, we define a mapping τ from $\{1, \ldots, k + \ell\}^*$ to $\Delta_{\mathcal{I}}$ in such a way that

$$T_{\lhd}(\alpha) = \{D \in \mathsf{cl}(C, \mathcal{T}) \mid \tau(\alpha) \in D^{\mathcal{I}}\} \qquad (*)$$

For the induction start, set

$$\tau(\epsilon) := a_0, \quad T_{\lhd}(\epsilon) := \{D \in \mathsf{cl}(C, \mathcal{T}) \mid a_0 \in D^{\mathcal{I}}\}, \quad \text{and} \quad T_{\rhd}(\epsilon) := \emptyset.$$

Now for the induction step. Let $\alpha \in \{1, \ldots, k + \ell\}^*$ such that $\tau(\alpha)$ is already defined, and let $i \in \{1, \ldots, k + \ell\}$. We make a case distinction as follows:

1. $\mathcal{E}(C, \mathcal{T}, i) = \exists R.D \in T_{\lhd}(\alpha)$. By $(*)$, we have $\tau(\alpha) \in (\exists R.D)^{\mathcal{I}}$. By the semantics, there exists some $b \in \Delta_{\mathcal{I}}$ such that $(\tau(\alpha), b) \in R^{\mathcal{I}}$ and $b \in D^{\mathcal{I}}$. Set $\tau(\alpha i) := b$, $T_{\lhd}(\alpha_i) := \{E \in \mathsf{cl}(C, \mathcal{T}) \mid b \in E^{\mathcal{I}}\}$, and $T_{\rhd}(\alpha i) := \emptyset$.

2. $\mathcal{F}(C, \mathcal{T}, i - k) = f$, and f is enforced by $\tau(\alpha)$. By $(*)$, the semantics, and the definition of "enforced", $f^{\mathcal{I}}(\tau(\alpha))$ is defined. Let $f^{\mathcal{I}}(\tau(\alpha)) = b$. Set $\tau(\alpha i) := b$, $T_{\lhd}(\alpha i) := \{E \in \mathsf{cl}(C, \mathcal{T}) \mid b \in E^{\mathcal{I}}\}$, and

   $$T_{\rhd}(\alpha i) := \quad \{(g_1, g_2, P) \mid fg_1 \text{ and } fg_2 \text{ are enforced by } T(\alpha) \text{ and } g_1^{\mathcal{I}}(b) P g_2^{\mathcal{I}}(b)\}$$

3. α, i do not match the above cases. Then set $\tau(\alpha i) := \tau(\epsilon)$ and $T(\alpha i) := T(\epsilon)$.

It is readily checked that the $k + \ell$-ary tree T just defined does satisfy $(*)$. We need to prove that T is a Hintikka-tree for $(C, \mathcal{T})$. From $(*)$ together with the semantics of concepts and TBoxes, it is clear that $T_{\lhd}(\alpha)$ is a Hintikka-set for $(C, \mathcal{T})$ for each $\alpha \in \{1, \ldots, k+\ell\}^*$. Let us show exemplarily that **(H1)** holds. Assume to the contrary that there exists an $\alpha \in \{1, \ldots, k + \ell\}^*$ such that $C_{\mathcal{T}} \notin T_{\lhd}(\alpha)$. Since $C_{\mathcal{T}} \in \mathsf{cl}(C, \mathcal{T})$ and by $(*)$, we have $\tau(\alpha) \notin (C_{\mathcal{T}})^{\mathcal{I}}$, and, by the semantics, $\tau(\alpha) \in (\sim C_{\mathcal{T}})^{\mathcal{I}}$. By definition of $C_{\mathcal{T}}$ and semantics, this implies the existence of $D \sqsubseteq E \in \mathcal{T}$ such that $\tau(\alpha) \in (\neg \mathsf{nnf}(\neg D \sqcup E))^{\mathcal{I}}$, i.e., $\tau(\alpha) \in D^{\mathcal{I}}$ and $\tau(\alpha) \notin E^{\mathcal{I}}$. Hence, we do not have $D^{\mathcal{I}} \subseteq E^{\mathcal{I}}$ and obtain a contradiction to the fact that $\mathcal{I}$ is a model for $\mathcal{T}$.

We now show that $T(\alpha)$ is a Hintikka-pair for each node α, i.e., that **(H6)** is satisfied. The proof is by contradiction. Assume that there exists an $\alpha \in \{1, \ldots, k+\ell\}^*$ such that $(g_1, g_2, P) \in T_{\rhd}(\alpha)$ and $g_j{\uparrow} \in T_{\lhd}(\alpha)$ where $j \in \{1, 2\}$. Since $(g_1, g_2, P) \in T_{\rhd}(\alpha)$, $g_j^{\mathcal{I}}(\tau(\alpha))$ is defined by definition of $T_{\rhd}$. Since $g_j{\uparrow} \in T_{\lhd}(\alpha)$ and by $(*)$, $g_j^{\mathcal{I}}(\tau(\alpha))$ is undefined, which is a contradiction.

It remains to show that T satisfies **(H12)** and **(H13)**, where the latter amounts to showing that, for each $\alpha \in \{1, \ldots, k + \ell\}^*$, the tuple $(T(\alpha), T(\alpha 1), \ldots, T(\alpha j))$ with $j = k + \ell$ satisfies **(H7)** to **(H11)**.

(H7) Let $\exists R.D \in T_{\lhd}(\alpha)$ and $\mathcal{E}(C, \mathcal{T}, i) = \exists R.D$. By definition of τ (Case 1), we have $\tau(\alpha i) = b$ for some $b \in \Delta_{\mathcal{I}}$ with $(\tau(\alpha), b) \in R^{\mathcal{I}}$ and $b \in D^{\mathcal{I}}$. By $(*)$, we thus have $D \in T_{\lhd}(\alpha i)$.

(H8) Let $\{\exists R.D, \forall R.E\} \subseteq T_{\lhd}(\alpha)$ and $\mathcal{E}(C, \mathcal{T}, i) = \exists R.D$. By definition of τ (Case 1), we have $\tau(\alpha i) = b$ for some $b \in \Delta_{\mathcal{I}}$ with $(\tau(\alpha), b) \in R^{\mathcal{I}}$. By $(*)$, we have $\tau(\alpha) \in (\forall R.E)^{\mathcal{I}}$. The semantics implies $b \in E^{\mathcal{I}}$, and, by $(*)$, we thus have $E \in T_{\lhd}(\alpha i)$.

(H9) Let $\exists f.D \in T_{\lhd}(\alpha)$ and $\mathcal{F}(C, \mathcal{T}, i) = f$. Hence, f is enforced by $T(\alpha)$. By definition of τ (Case 2), we have $\tau(\alpha j) = b$ for $b = f^{\mathcal{I}}(\tau(\alpha))$ and $j = k + i$. The semantics implies $b \in D^{\mathcal{I}}$, and, by $(*)$, we thus have $D \in T_{\lhd}(\alpha j)$.

(H10) Let f be enforced by $T(\alpha)$, $\mathcal{F}(C, \mathcal{T}, i) = f$, and $\forall f.D \in T_{\lhd}(\alpha)$. By definition of τ (Case 2), we have $\tau(\alpha j) = b$ for $b = f^{\mathcal{I}}(\tau(\alpha))$ and $j = k + i$. The semantics implies $b \in D^{\mathcal{I}}$, and, by $(*)$, we thus have $D \in T_{\lhd}(\alpha j)$.

(H11) Let $G(T(\alpha)) = (V, E)$ and $\mathsf{cpl}(T, \alpha) = (V, E \cup E')$. To prove that **(H11)** is satisfied, we show that

1. E' is an edge extension of $G(T(\alpha))$, i.e., $(V, E \cup E')$ is a completion of $G(T(\alpha))$ and
2. $(V, E \cup E')$ is satisfiable.

We first prove Point 1. It needs to be shown that, for each $fg_1, fg_2 \in V$, either $(fg_2, fg_1, <) \in E'$ or $(fg_1, fg_2, P) \in E'$ for some $P \in \{<, =\}$. By definition of $G(T(\alpha))$, fg_1 and fg_2 are enforced by $T(\alpha)$. Since $T_{\rhd}(\alpha)$ may only contain paths of length 1, we have $\{\exists fg_1, u.P', \exists u, fg_1.P'\} \cap T_{\lhd}(\alpha) \neq \emptyset$ for some path u and $P' \in \{<, =\}$ and similarly for fg_2. By $(*)$, this implies that $f^{\mathcal{I}}(g_1^{\mathcal{I}}(\tau(\alpha)))$ and $f^{\mathcal{I}}(g_2^{\mathcal{I}}(\tau(\alpha)))$ are defined. By definition of T (Case 2) and since f is obviously enforced by $T(\alpha)$, we have $f^{\mathcal{I}}(\tau(\alpha)) = \tau(\alpha i)$ with $\mathcal{F}(C, \mathcal{T}, i - k) = f$. Hence, $g_1^{\mathcal{I}}(\tau(\alpha i))$ and $g_2^{\mathcal{I}}(\tau(\alpha i))$ are defined. By semantics, we either have (i) $g_2^{\mathcal{I}}(\tau(\alpha i)) < g_1^{\mathcal{I}}(\tau(\alpha i))$ or (ii) $g_1^{\mathcal{I}}(\tau(\alpha i)) P g_2^{\mathcal{I}}(\tau(\alpha i))$ for some $P \in \{<, =\}$. By definition of $T_{\rhd}$, (i) implies $(g_2, g_1, <) \in T_{\rhd}(\alpha i)$ and (ii) implies $(g_1, g_2, P) \in T_{\rhd}(\alpha i)$. Hence, by definition of E', we have either $(fg_2, fg_1, <) \in E'$ or $(fg_1, fg_2, P) \in E'$.

We now prove point 2. Define a mapping from V to $\mathbb{Q}$ as follows: $(u) := u^{\mathcal{I}}(\tau(\alpha))$. This mapping is well-defined, which can be seen as follows. Fix a $u \in V$. Since u is enforced by $T(\alpha)$, either

(i) u occurs in $T_{\rhd}(\alpha)$ or

(ii) $\{\exists u, u'.P, \exists u', u.P\} \cap T_{\lhd}(\alpha) \neq \emptyset$ for some path u' and $P \in \{<, =\}$.

In Case (i), we have $u = g$ for some $g \in N_{cF}$. By definition of T, there exists a predecessor β of α in T such that $\alpha = \beta i$, $\mathcal{F}(C, \mathcal{T}, i - k) = f$ for some $f \in N_{aF}$, and fg is enforced by $T(\beta)$. Since $T_{\rhd}(\beta)$ contains only paths of length 1, we have $\{\exists fg, u.P, \exists u, fg.P\} \cap T_{\lhd}(\beta) \neq \emptyset$ for some path u and $P \in \{<, =\}$. By $(*)$, $g^{\mathcal{I}}(f^{\mathcal{I}}(\tau(\beta)))$ is defined. Since, by definition of T, we have $f^{\mathcal{I}}(\tau(\beta))) = \tau(\alpha)$, $g^{\mathcal{I}}(\tau(\alpha))$ is defined. In Case (ii), definedness of $u^{\mathcal{I}}(\tau(\alpha))$ follows from $(*)$ and the semantics.

We show that is a solution for $(V, E \cup E')$ by distinguishing the following cases:

1. $(u_1, u_2, P) \in E$ and $(u_1, u_2, P) \in T_{\rhd}(\alpha)$. Then there exist $g_1, g_2 \in N_{cF}$ such that $u_1 = g_1$ and $u_2 = g_2$. By definition of $T_{\rhd}$, we have $g_1^{\mathcal{I}}(\tau(\alpha)) P g_2^{\mathcal{I}}(\tau(\alpha))$, and, by definition of , $(g_1) P (g_2)$.[5]
2. $(u_1, u_2, P) \in E$ and $\exists u_1, u_2.P \in T_{\lhd}(\alpha)$. By $(*)$, we have $\tau(\alpha) \in (\exists u_1, u_2.P)^{\mathcal{I}}$. Hence, $u_1^{\mathcal{I}}(\tau(\alpha)) P u_2^{\mathcal{I}}(\tau(\alpha))$. By definition of , we obtain $(u_1) P (u_2)$.
3. $(u_1, u_2, P) \in E'$. By definition of E', we have $u_1 = fg_1$, $u_2 = fg_2$, and $(g_1, g_2, P) \in T_{\rhd}(\alpha i)$ where $g_1, g_2 \in N_{cF}$ and $\mathcal{F}(C, \mathcal{T}, k - i) = f$. By definition of $T_{\rhd}$, this implies that fg_1 and fg_2 are enforced by $T(\alpha)$ and that

[5] We need not consider the case "$(u_1, u_2, =) \in E$ and $(u_2, u_1.=) \in T_{\rhd}(\alpha)$" since constraint graphs are assumed to be equality closed. A similar note applies to Case 2.

$g_1^{\mathcal{I}}(\tau(\alpha i))Pg_2^{\mathcal{I}}(\tau(\alpha i))$. From this and the definition of T (Case 2), it follows that $f^{\mathcal{I}}(\tau(\alpha)) = \tau(\alpha i)$. We conclude $(u_1)P\ (u_2)$.

(H12) By definition of T (induction start) and since $a_0 \in C^{\mathcal{I}}$ by assumption.

❑

Note that Lemma 18 together with the proof of Lemma 17 implies that the PNF fragment of $\mathcal{TDL}$ has the tree model property: By Lemma 18, the satisfiability of a concept C w.r.t. a TBox $\mathcal{T}$ implies the existence of a Hintikka-tree T for $(C, \mathcal{T})$. Using the construction from the proof of Lemma 17, we can construct a canonical model from T. It is not hard to see that this canonical model is a tree model in the sense of Section 2. In view of Lemma 13 and its proof, it is not hard to show that $\mathcal{TDL}$ also has the tree model property.

4.4 Defining looping automata

To prove decidability, it remains to define a looping automaton $\mathcal{A}_{(C,\mathcal{T})}$ for each concept C and TBox $\mathcal{T}$ such that $\mathcal{A}_{(C,\mathcal{T})}$ accepts exactly the Hintikka-trees for $(C, \mathcal{T})$. Using the notion of matching tuples of Hintikka-pairs, this is rather straightforward.

Definition 19. Let C be a concept, $\mathcal{T}$ be a TBox, k the number of existential subconcepts in $\mathsf{cl}(C, \mathcal{T})$, and ℓ be the number of abstract features in $\mathsf{cl}(C, \mathcal{T})$. The looping automaton $\mathcal{A}_{(C,\mathcal{T})} = (Q, \Gamma_{(C,\mathcal{T})}, \Delta, I)$ is defined as follows:

- $Q = \Gamma_{(C,\mathcal{T})}$
- $I = \{(\Psi, \chi) \in Q \mid C \in \Psi\}$.
- $((\Psi, \chi), (\Psi', \chi'), (\Psi_1, \chi_1), \ldots, (\Psi_k, \chi_{k+\ell})) \in \Delta$ iff

  $(\Psi, \chi) = (\Psi', \chi')$ and
  $((\Psi, \chi), (\Psi_1, \chi_1), \ldots, (\Psi_k, \chi_{k+\ell}))$ is matching.

As a consequence of the following lemma and Lemmas 17 and 18, we can reduce satisfiability of concepts w.r.t. TBoxes (in PNF) to the emptiness of the language accepted by looping automata.

Lemma 20. *T is a Hintikka-tree for $(C, \mathcal{T})$ iff $T \in L(\mathcal{A}_{C,\mathcal{T}})$.*

Proof Let C be a concept, $\mathcal{T}$ a TBox, and k, ℓ, and $\mathcal{A}_{(C,\mathcal{T})}$ as in Definition 19.

"$\Rightarrow$" It is straightforward to check that the function r defined by $r(\alpha) := T(\alpha)$ is a run of $\mathcal{A}_{C,\mathcal{T}}$ on T: (i) By definition of Hintikka-trees and $\mathcal{A}_{C,\mathcal{T}}$, $r(\alpha) \in Q$ for all $\alpha \in \{1, \ldots, k+\ell\}^*$; (ii) by **(H12)** and definition of I, we have $r(\epsilon) \in I$; (iii) by **(H13)** and by definition of r and of Δ, we have $(r(\alpha), T(\alpha), r(\alpha_1), \ldots, r(\alpha_k)) \in \Delta$.

"$\Leftarrow$" Let r be a run of $\mathcal{A}_{(C,\mathcal{T})}$ on T. It is straightforward to show that T is a Hintikka-tree for $(C, \mathcal{T})$: (i) by definition of runs and of Q, r is a $\Gamma_{(C,\mathcal{T})}$-tree; (ii) since, by definition of runs, $r(\epsilon) \in I$, **(H12)** is satisfied by definition of I; and (iii), by definition of runs and of Δ, **(H13)** is satisfied. ❑

It is an immediate consequence of Lemmas 13, 17, 18, and 20, and the decidability of the emptiness problem of looping automata [27] that satisfiability of $\mathcal{TDL}$-concepts w.r.t. TBoxes is decidable. However, the presented automata-based algorithm has the nice property of additionally providing us with a tight complexity bound.

Theorem 21. *Satisfiability and subsumption of $\mathcal{TDL}$-concepts w.r.t. TBoxes are decidable in deterministic exponential time.*

Proof The lower bound is an immediate consequence of the fact that $\mathcal{ALC}$ with general TBoxes is ExpTime-hard [20]. For the upper bound, we need to show that the size of $\mathcal{A}_{(C,\mathcal{T})} = (Q, M, \Delta, I)$ is exponential in the size of C and $\mathcal{T}$ (it is then obvious that $\mathcal{A}_{(C,\mathcal{T})}$ can be computed in exponential time).[6] Obviously, the cardinality of $\mathsf{cl}(C, \mathcal{T})$ is linear in the size of C and $\mathcal{T}$. Hence, by definition of $\mathcal{A}_{(C,\mathcal{T})}$ and Hintikka-pairs, the cardinality of Q and M are exponential in the size of C and $\mathcal{T}$. Again by definition of $\mathcal{A}_{(C,\mathcal{T})}$, this implies that the cardinalities of I and Δ are also exponential in the size of C and $\mathcal{T}$. Hence, the size of $\mathcal{A}_{(C,\mathcal{T})}$ is exponential in the size of C and $\mathcal{T}$. This fact together with Lemmas 13, 17, 18, and 20, and the fact that emptiness of the language accepted by a looping automaton $\mathcal{A}_{(C,\mathcal{T})}$ can be tested in time polynomial in the size of $\mathcal{A}_{(C,\mathcal{T})}$ [27], we have that satisfiability of $\mathcal{TDL}$-concepts w.r.t. TBoxes is in ExpTime. It remains to remind the reader that subsumption can be reduced to satisfiability. ❑

4.5 Connection to Tableau Algorithms

We conclude this section with some remarks on the connection of the presented algorithm with so-called tableau algorithms. This informal discussion assumes some familiarity with tableau algorithms, see, e.g., [6] for more information on this topic.

As empirical results have shown, tableau algorithms are amenable to optimizations that allow for e cient implementations of these algorithms, see, e.g., [11]. To the contrary, e cient implementations of automata-based algorithms like the one presented in this paper are—as of now—unknown. Hence, it would be interesting to define a tableau algorithm for $\mathcal{TDL}$. We argue that there exists a close connection between the results obtained in this paper and certain di culties encountered in correctness proofs for tableau algorithm for $\mathcal{TDL}$.

Since $\mathcal{TDL}$ admits general TBoxes, a tableau algorithm for $\mathcal{TDL}$ would have to use a technique called *blocking*. This means that such an algorithm would not try to directly construct a model for the input concept but it would try to construct a "pre-model", i.e., a finite representation of a "real" model. In the correctness proof, the real model can then be obtained from the pre-model by a technique called *unravelling*. It

[6] When talking of the size of C and $\mathcal{T}$, we refer to the sum of the lengths of C and $C_{\mathcal{T}}$.

is not hard to see that unravelling yields only models that are in some sense periodic. Unfortunately, it is outside the scope of this paper to give a formal definition of this kind of periodicity (or unravelling itself). Roughly spoken, unravelling yields an infinite tree whose paths are built according to a specific pattern induced by the blocking-relationships in the pre-model. The existence of this pattern is what is meant by "periodic". This implies that a tableau algorithm for $\mathcal{TDL}$ does only construct finite representations of periodic models (and not of arbitrary models).

This is not a problem for most Description Logics: It is a (usually unnoticed) byproduct of the standard proof technique used for showing correctness of DL tableau algorithms that, for most logics, satisfiability of a concept implies satisfiability in a periodic model. However, in $\mathcal{TDL}$, this issue cannot be treated "implicitly" because of the global nature of the "concrete part" of models. Assume that we want to prove that the existence of a model for a concept C implies that a $\mathcal{TDL}$ tableau algorithm reports satisfiable. To do this, we could, for example, fix a model $\mathcal{I}$ for C and use it to "guide" the application of the completion rules. We will end up with a pre-model that induces a model $\mathcal{I}'$. As we argued above, this model is periodic and so is its local part. To complete our proof, we need to show that the concrete part of $\mathcal{I}$ is identical to the concrete part of $\mathcal{I}'$. But this may not be the case since the concrete part of $\mathcal{I}$ may be aperiodic. Hence, we want to choose a periodic model of $\mathcal{I}$ to guide our completion rules which can only be done safely if we first establish a "periodic model property". Summing up, in the case of $\mathcal{TDL}$, the things are a little bit different than usual: The periodic model property is not a byproduct of the correctness proof but it needs to be proved in advance and is then *used* in the correctness proof. We claim that this effect is not due to the specific proof technique sketched above but rather a general problem. Unfortunately, it is not at all obvious how a periodic model property can be established.

Fortunately, the periodic model property is a byproduct of the automata-based approach presented in this paper: As is, e.g., noted by Thomas [25], there exists a so-called "regular" tree in every nonempty Büchi-recognizable set of trees. Obviously, the set of Hintikka-trees for a formula C and a TBox $\mathcal{T}$ is such a set. The notion of regularity of trees is very closely related to the periodicity of models. More precisely, we conjecture that a proof of the periodic model property could work as follows: By Lemma 20, the fact that C is satisfiable w.r.t. a TBox $\mathcal{T}$ implies that the set of Hintikka-trees accepted by $\mathcal{A}_{(C,\mathcal{T})}$ is nonempty. Hence, it contains a regular tree T. If we construct a model of T as in the proof of Lemma 17, then this model is periodic. We conclude that the algorithm presented in this paper provides an important building block for proving the correctness of tableau algorithms for $\mathcal{TDL}$.

5 Conclusion

In this paper, we presented the temporal Description Logic $\mathcal{TDL}$ that combines interval-based reasoning with reasoning about general TBoxes. An automata-based decision procedure was devised and a tight ExpTime-complexity bound was obtained. We demonstrated the usefulness of our logic by giving examples from the application

domain of process engineering. Since there exists a very close connection between $\mathcal{TDL}$ and Description Logics with concrete domains, our results can also be viewed from a different perspective: Despite the discouraging results given in [15], we were able to show that there exist interesting concrete domains for which reasoning with general TBoxes is decidable.

There are several perspectives for future work of which we highlight two rather interesting ones: First, the presented decision procedure is only valid if a dense strict linear order is assumed as the underlying temporal structure. For example, the concept $\top$ is satisfiable w.r.t. the TBox

$$\mathcal{T} = \{\top \sqsubseteq \exists g_1, g_2, < \sqcap \exists g_1, fg_1, < \sqcap \exists fg_2, g_2, <\}$$

over the temporal structures $\mathbb{Q}$ and $\mathbb{R}$ (with the natural orderings) but not over $\mathbb{N}$. To see this, note that $\mathcal{T}$ induces a constraint graph as in Figure 6. Hence, it would be interesting to investigate how the presented algorithm has to be modified for reasoning with the temporal structure $\mathbb{N}$. We conjecture that a constraint graph G is satisfiable over $\mathbb{N}$ iff there exists an upper bound on the length of $<$-paths between any two nodes in G.[7] It is, however, not immediately clear how Hintikka-trees and automata can be modified to account for this stronger condition.

Second, it would be interesting to extend $\mathcal{TDL}$ to make it suitable for reasoning about entity relationship (ER) diagrams with temporal integrity constraints. As demonstrated by Calvanese et al. in [7; 8], Description Logics are well-suited for reasoning about ER diagrams with integrity constraints and thus are a valuable tool for database design. Artale and Franconi propose a temporalization of Calvanese's approach that can be used for reasoning about temporal ER diagrams [3]. They use a point-based logic and focus on temporal databases, i.e., they admit reference to previous database states in the ER model. By using an appropriate extension of $\mathcal{TDL}$, one should be able to capture a different kind of temporal reasoning with ER diagrams, namely reasoning over ER diagrams with integrity constraints for databases that store temporal data. Such an extension would allow to formulate *temporal* integrity constraints, i.e., integrity constraints that take into account the temporal semantics of the data in the database. For example, a temporal integrity constraint could state that employees birthdays should be before their employment date. But what is an appropriate extension of $\mathcal{TDL}$ for reasoning in this domain? Given the results in [7], it is clear that we need (unqualified) number restrictions and inverse roles. For the temporal aspects, we need a generalized version of the concrete domain constructor $\exists u_1, u_2.P$ that allows quantification over role paths instead of feature paths and has a universal instead of an existential semantics. An extension of the presented automata-theoretic decision procedure to this more complex logic seems possible.

Acknowledgements The author would like to thank Franz Baader, Ulrike Sattler, and Stephan Tobies for fruitful discussions. The author was supported by the DFG Project BA1122/3-1 "Combinations of Modal and Description Logics".

[7] This also implies that G contains no $<$-cycle.

References

[1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

[2] A. Artale and E. Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research (JAIR)*, (9), 1998.

[3] A. Artale and E. Franconi. Temporal ER modeling with description logics. In *Proceedings of the International Conference on Conceptual Modeling (ER'99)*, Paris, France, November 1999. Springer–Verlag.

[4] A. Artale and E. Franconi. Temporal description logics. In D. Gabbay, M. Fisher, and L. Vila, editors, *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. MIT Press, 2001. To appear.

[5] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence IJCAI-91*, pages 452–457, Sydney, Australia, August 24–30, 1991. Morgan Kaufmann Publ. Inc., San Mateo, CA, 1991.

[6] F. Baader and U. Sattler. Tableau algorithms for description logics. In R. Dyckhoff, editor, *Proceedings of the International Conference on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 1–18, St Andrews, Scotland, UK, 2000. Springer-Verlag.

[7] D. Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms*. Dottorato di ricerca in informatica, Università degli Studi di Roma "La Sapienza", Italia, 1996.

[8] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

[9] J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of ACM*, 38(4):935–962, 1991.

[10] P. Hanschke. Specifying role interaction in concept languages. In W. Nebel, Bernhard; Rich, Charles; Swartout, editor, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 318–329, Cambridge, MA, Oct. 1992. Morgan Kaufmann.

[11] I. Horrocks, P. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8(3):293–323, 2000.

[12] H. A. Kautz and P. B. Ladkin. Integrating metric and qualitative temporal reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence AAAI-91*, pages 241–246, Anaheim, California, July 14 - 19, 1991. AAAI-Press/The MIT Press, Menlo Park – Cambridge – London, 1991.

[13] M. Kullmann, F. de Bertrand de Beuvron, and F. Rousselot. A description logic model for reacting in a dynamic environment. In F. Baaer and U. Sattler, editors, *Proceedings of the 2000 International Workshop in Description Logics (DL2000)*, number 33 in CEUR-WS, pages 203–212, Aachen, Germany, August 2000. RWTH Aachen. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-33/.

[14] C. Lutz. Complexity of terminological reasoning revisited. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 181–200. Springer-Verlag, Sept. 1999.

[15] C. Lutz. NExpTime-complete description logics with concrete domains. LTCS-Report 00-01, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[16] C. Lutz, V. Haarslev, and R. Möller. A concept language with role-forming predicate restrictions. Technical Report FBI-HH-M-276/97, University of Hamburg, Computer Science Department, Hamburg, 1997.

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

[18] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[19] U. Sattler. *Terminological knowledge representation systems in a process engineering application.* PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, 1998.

[20] K. D. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 13th IJCAI*, pages 466–471, Sidney, Australia, 1991.

[21] K. D. Schild. Combining terminological logics with tense logic. In M. Filgueiras and L. Damas, editors, *Progress in Artificial Intelligence – 6th Portuguese Conference on Artificial Intelligence, EPIA'93*, Lecture Notes in Artificial Intelligence, pages 105–120, Porto, Portugal, Oct. 1993. Springer-Verlag.

[22] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[23] A. Schmiedel. Temporal terminological logic. In W. Dietterich, Tom; Swartout, editor, *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 640–645. MIT Press, July 29–Aug. 3 1990.

[24] E. Szpilrajn. Sur l'extension de l'ordre partiel. *Fundamenta Mathematica*, 16:386–389, 1930.

[25] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V., 1990.

[26] P. van Beek and D. W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research (JAIR)*, (4):1–18, 1996.

[27] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

[28] M. Vilain, H. Kautz, and P. Van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Kaufmann, San Mateo, CA, 1990.

[29] F. Wolter and M. Zakharyaschev. Temporalizing description logic. In D. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems*, pages 379 – 402. Studies Press/Wiley, 1999.