Combining Interval-based Temporal Reasoning with General TBoxes

Carsten Lutz

Institute for Theoretical Computer Science, TU Dresden, Germany

Abstract

While classical Description Logics (DLs) concentrate on the representation of static conceptual knowledge, recently there is a growing interest in DLs that, additionally, allow to capture the temporal aspects of conceptual knowledge. Such temporal DLs are based either on time points or on time intervals as the temporal primitive. Whereas point-based temporal DLs are well-investigated, this is not the case for interval-based temporal DLs: all known logics either suffer from rather limited expressive power or have undecidable reasoning problems. In particular, there exists no decidable interval-based temporal DL that provides for general TBoxes—one of the most important expressive means in modern description logics. In this paper, for the first time we define an interval-temporal DL that is equipped with general TBoxes and for which reasoning is decidable (and, more precisely, ExpTime-complete).

Key words: description logic, temporal reasoning, tree automata, complexity

1 Introduction

Description Logics (DLs) are a family of logics that originated in artificial intelligence as a tool for reasoning about conceptual knowledge, and are nowadays used in a broad spectrum of applications [6]. The fundamental notion of knowledge representation with DLs is that of a concept, where complex concepts are constructed from the following atoms: *concept names* (unary predicates),

Email address: lutz@tcs.inf.tu-dresden.de (Carsten Lutz).
URL: http://lat.inf.tu-dresden.de/~clu/ (Carsten Lutz).

role names (binary predicates), and a set of *concept constructors* that are provided by the chosen DL. For example, the following concept is formulated in the basic description logic $\mathcal{ALC}$ [34]:

$$\mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Human}$$

In this example, Human and Male are concept names while has-child is a role name. It should be easy to see that, intuitively, the above concept describes fathers.

Whereas classical DLs concentrate on the representation of "static" conceptual knowledge (such as in the above example), recently there is a growing interest in "dynamic" DLs that allow to incorporate, e.g., temporal and epistemic aspects of the application domain. Concerning temporal description logics, we can distinguish several quite different approaches (see e.g. the survey [2]). The most important decision to be made when devising a temporal DL is whether time points or time intervals should be the temporal primitive, since this decision has a serious ontological impact and may have dramatic consequences for issues of decidability and computational complexity.

In modal and temporal logics, to which description logics are very closely related [32,13], time points are the most popular temporal atom, see e.g. the handbook [12]. Consequently, there has been a series of papers on temporal DLs that use time points as their temporal primitive in the same spirit as modal and temporal logics do [33,41,36,27]. These logics offer an interesting expressivity and sometimes have quite attractive computational properties. However, their expressive power is not strong enough to talk about time intervals in a satisfying way.

In artificial intelligence, time intervals have a strong tradition as a temporal primitive since Allen's seminal 1983 paper [1], see e.g. the handbook [11]. As DLs form a subfield of artificial intelligence, it is hardly surprising that interval-based temporal DLs also received a considerable amount of attention. The expressive power of such DLs is usually based on concept constructors that refer to the 13 Allen relations, which describe all possible ways in which two time intervals can be related. While the advantage of interval-based temporal DLs is that they provide an appealing expressivity, their disadvantage is that it can be rather hard to avoid undecidability of reasoning. For example, the first interval-based temporal DL proposed by Schmiedel [35] can easily be proved to be undecidable by reduction of Halpern and Shoham's (undecidable) modal logic of time intervals [15]. Based on this observation, researchers have tried either to live with undecidability [8] or to find variants of Schmiedel's original logic that are still decidable [3].

The main obstacle for many potential applications of decidable interval-based temporal DLs is that, in order to avoid undecidability, these DLs do not pro-

vide for so-called *general TBoxes*. General TBoxes are finite sets of concept equations and a very important expressive means provided by all state-of-the-art "static" description logics, and by all modern DL reasoning systems such as FaCT and RACER [18,14]. The importance of TBoxes stems from the fact that they allow to capture terminological knowledge and background knowledge of an application domain. For example, the following concept equation defines the notion "father" (thus capturing terminological knowledge):

$$\mathsf{Father} \doteq \mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Human}$$

However, concept equations need not define notions. They can also describe background knowledge in the form of more general constraints:

$$\neg \exists \mathsf{has\text{-}child}.\mathsf{Human} \doteq \exists \mathsf{has\text{-}favorite}.\mathsf{Nightclub}$$

This concept equation states that people having no children are precisely those people having a favorite nightclub.

The contribution of this paper is to describe a *decidable temporal description logic $\mathcal{TDL}$ that provides for general TBoxes and allows interval-based temporal representation and reasoning*. Indeed, $\mathcal{TDL}$ is natively point-based, but admits a straightforward representation of Allen's interval relations, thus being a suitable tool for intermixed point- and interval-based temporal reasoning with general TBoxes. More precisely, $\mathcal{TDL}$ extends the basic propositionally closed description logic $\mathcal{ALC}$ with

- general TBoxes;
- *abstract features*, i.e. role names interpreted in functional relations;
- *temporal features*: a new syntactic type that allows to associate time points (rational numbers) with domain elements;
- a *temporal concept constructor* allowing to state that two time points attached via temporal features are in one of the relations $<, \leq, =, \neq, \geq, >$.

For example, the following $\mathcal{TDL}$-concept equation expresses that children are born after their parents were:

$$\top \doteq \mathsf{Human} \rightarrow \exists((\mathsf{mother}\,\mathsf{birthday}) < \mathsf{birthday}) \sqcap \exists((\mathsf{father}\,\mathsf{birthday}) < \mathsf{birthday})$$

In this equation, mother and father are abstract features, and birthday is a temporal feature whose value is the birthday of persons encoded as a rational number—a time *point*. The conjuncts in the consequence of the implication are both instantiations of the temporal concept constructor and must not be confused with the existential restriction as in $\exists \mathsf{child}.\mathsf{Human}$. Note that (mother birthday) denotes composition of the abstract feature mother with the temporal feature birthday.

We have claimed that Allen's interval relations can straightforwardly be en-

coded in $\mathcal{TDL}$. Assume, for example, that we want to represent the life-time of people by a time interval and then describe persons whose life-time is properly overlapping (this is one of the Allen relations) with that of their mother. The obvious idea is to represent intervals in terms of their start- and end-point and then to use the relations on time points $<, \leq, =, \neq, \geq, >$ to define Allen's interval relations. To represent the above example, we could thus write

$$\begin{aligned}&\mathsf{Human} \sqcap \exists((\mathsf{mother}\ \ell) < \ell) \sqcap \exists((\mathsf{mother}\ r) < r)\\ &\sqcap \exists(\ell < r) \sqcap \exists((\mathsf{mother}\ \ell) < (\mathsf{mother}\ r)),\end{aligned}$$

where the temporal feature ℓ represents left interval endpoints and the temporal feature r represents right interval endpoints. Since this concept requires close inspection to reveal that it talks about the Allen relation "overlaps", we will define a representation framework that builds on $\mathcal{TDL}$ and treats time intervals (*and* time points) as first-class citizens. In this framework, we can reformulate the above concept as

$$\mathsf{Human} \sqcap \exists(\mathsf{mother}\ \mathit{overlaps}\ \mathsf{self}).$$

Here, mother is still an abstract feature and self is a keyword of the framework. Concrete features do not appear explicitly in this abbreviated syntax. More details are provided in Section 3. It is interesting to note that interval-based temporal representation with $\mathcal{TDL}$ is similar in spirit to the Allen-based temporal constraint networks (see e.g. [1,40,38,29]) rather than to Schmiedel's or Halpern and Shoham's interval-based description/modal logics. A more detailed comparison of these two families of interval-based temporal description logics can be found in [4], which investigates the relationship between a relative of $\mathcal{TDL}$ and the logic $\mathcal{TL}$-$\mathcal{ALCF}$, a decidable and interval-based temporal DL that both restricts and extends Schmiedel's original proposal [3]. However, also $\mathcal{TL}$-$\mathcal{ALCF}$ does not provide for general TBoxes.

There exists a second, non-temporal view on the description logic $\mathcal{TDL}$ that we should also like to discuss. One shortcoming of simple description logics such as $\mathcal{ALC}$ is that they represent knowledge on an abstract logic level, thus prohibiting an adequate representation of "concrete knowledge" such as knowledge about sizes, weights, ages, or even spatial extensions. To eliminate this deficiency, DLs have been extended with so-called *concrete domains* as first proposed in [5], for a recent survey consult [25]. The relationship between $\mathcal{TDL}$ and description logics with concrete domains is a rather intimate one: indeed, $\mathcal{TDL}$ can be viewed as the extension of $\mathcal{ALC}$ with general TBoxes and a particular concrete domain (more details are provided in Section 2). Due to this fact, the results proved in this paper can be viewed in a different light. In [26], the extension of $\mathcal{ALC}$ with concrete domains *and* general TBoxes has been considered. As it turns out, the resulting logic is undecidable for many interesting concrete domains. It has been an open problem whether there

exist *any* useful concrete domains that can be combined with general TBoxes without loosing decidability. Since $\mathcal{TDL}$ can be viewed as being equipped with a concrete domain and, in our opinion, is a very useful DL, we answer this question to the affirmative.

This paper is organized as follows. In Section 2, we formally introduce the description logic $\mathcal{TDL}$ and discuss its relationship to concrete domains on more formal grounds. Section 3 starts with a description of the framework for representing mixed point- and interval-based temporal information. To illustrate the usefulness of both $\mathcal{TDL}$ and the representation framework, we then describe an example application from the area of process engineering. In Section 4, we use an approached based on automata on infinite trees to show that satisfiability and subsumption of $\mathcal{TDL}$-concepts w.r.t. general TBoxes is decidable. This proof also provides us with a tight ExpTime complexity bound. In Section 5, we consider another common DL reasoning problem: ABox consistency. By reduction to concept satisfiability, we prove that, in $\mathcal{TDL}$, ABox consistency is also ExpTime-complete. The reduction is much less straightforward than e.g. in the case of $\mathcal{ALC}$ due to the presence of temporal information. Finally, we conclude in Section 6.

All results in this article are from the PhD Thesis [22]. The results obtained in Section 4 have previously been published in the conference paper [20].

2 The Description Logic $\mathcal{TDL}$

We formally introduce the description logic $\mathcal{TDL}$, starting with the syntax. Examples are delayed to the subsequent section.

Definition 1 ($\mathcal{TDL}$ Syntax) *Let* $\mathsf{N_C}$, $\mathsf{N_R}$, *and* $\mathsf{N_{tF}}$ *be mutually disjoint and countably infinite sets of* concept names, role names, *and* temporal features. *We assume that* $\mathsf{N_R}$ *is partitioned into two countably infinite subsets* $\mathsf{N_{aF}}$ *and* $\mathsf{N_{rR}}$. *The elements of* $\mathsf{N_{aF}}$ *are called* abstract features *and the elements of* $\mathsf{N_{rR}}$ regular roles. *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 a temporal 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 name,* g *is a temporal feature,* u_1, u_2 *are paths, and* $P \in \{<, \leq, =, \neq, \geq, >\}$, *then the following expressions are also concepts:* $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$, $\exists(u_1 \, P \, u_2)$, *and* $g{\uparrow}$.

A concept equation *is an expression of the form* $C \doteq D$, *where* C *and* D *are concepts. A finite set of concept equations is called* TBox.

The TBox formalism introduced in Definition 1 is often called *general TBox* since it subsumes several other, much weaker variants [9,19]. Throughout this paper, we use $\top$ as abbreviation for an arbitrary propositional tautology, $\bot$ for $\neg\top$, and $u\uparrow$ for $\forall f_1.\cdots\forall f_k.g\uparrow$ if $u = f_1\cdots f_k g$. As most description logics, $\mathcal{TDL}$ is equipped with a Tarski-style set-based semantics.

Definition 2 ($\mathcal{TDL}$ Semantics) *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 temporal feature* g *to a partial function* $g^\mathcal{I}$ *from* $\Delta_\mathcal{I}$ *to the rational numbers* $\mathbb{Q}$*.* [1]

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

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

An interpretation $\mathcal{I}$ *is a* model *of a concept* C *iff* $C^\mathcal{I} \neq \emptyset$*.* $\mathcal{I}$ *is a* model *of a TBox* $\mathcal{T}$ *iff it satisfies* $C^\mathcal{I} = D^\mathcal{I}$ *for all concept equations* $C \doteq D$ *in* $\mathcal{T}$*.*

Note that the temporal constructor $\exists(u_1\, P\, u_2)$ has an existential semantics since it forces the interpretation of the paths u_1 and u_2 to be defined. The most important reasoning problems for description logics are concept satisfiability and concept subsumption, i.e. the questions whether a given concept can have any instances and whether one concept is more general than another one [6]. For both reasoning tasks, a TBox $\mathcal{T}$ is used to describe the "background theory".

Definition 3 (Reasoning Problems) *Let* C *and* D *be concepts and* $\mathcal{T}$ *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 common model of* C *and* $\mathcal{T}$*.*

[1] It would not make a difference to use real numbers instead of rational numbers as time points. This is discussed in more detail in Sections 4.1 and 6.

It is well-known that (un)satisfiability and subsumption can be mutually reduced to each other: $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$ and C is satisfiable w.r.t. $\mathcal{T}$ iff $C \not\sqsubseteq_{\mathcal{T}} \bot$. This fact allows us to concentrate on satisfiability since obtained decidability and complexity results are easily "pulled over" to subsumption.

Another very important reasoning problem is so-called ABox consistency [6]. Intuitively, an ABox describes the state of affairs in the real-world at a particular time, i.e. it is a "snapshot" of the real world. ABox consistency is then the ABox counterpart to concept satisfiability.

Definition 4 (ABox, ABox Consistency) *Let* $\mathsf{O_a}$ *and* $\mathsf{O_t}$ *be a countably infinite and mutually disjoint sets of* object names *and* time point names. *If* C *is a concept,* R *a role name,* g *a temporal feature,* $P \in \{<, \leq, =, \neq, \geq, >\}$, $a, b \in \mathsf{O_a}$, *and* $x, y \in \mathsf{O_t}$, *then the following are* ABox assertions*:*

$$a : C, \quad \langle a, b\rangle : R, \quad \langle a, x\rangle : g, \quad x\, P\, y.$$

A finite set of assertions is called an ABox. *Interpretations* $\mathcal{I}$ *can be extended to ABoxes by demanding that, additionally,* $\cdot^{\mathcal{I}}$ *maps every object name* a *to an element* $a^{\mathcal{I}}$ *of* $\Delta_{\mathcal{I}}$ *and every time point name* x *to a rational number* $x^{\mathcal{I}}$. *An interpretation* $\mathcal{I}$ *then* satisfies *an assertion*

$$\begin{aligned} a : C \quad &\textit{iff} \quad a^{\mathcal{I}} \in C^{\mathcal{I}} \\ \langle a, b\rangle : R \quad &\textit{iff} \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}} \\ \langle a, x\rangle : g \quad &\textit{iff} \quad g^{\mathcal{I}}(a^{\mathcal{I}}) = x^{\mathcal{I}} \\ x\, P\, y \quad &\textit{iff} \quad x^{\mathcal{I}}\, P\, y^{\mathcal{I}}. \end{aligned}$$

An interpretation is a model *of an ABox* $\mathcal{A}$ *iff it satisfies all assertions in* $\mathcal{A}$. *An ABox* $\mathcal{A}$ *is* consistent *w.r.t. a TBox* $\mathcal{T}$ *iff there exists a common model of* $\mathcal{A}$ *and* $\mathcal{T}$.

Let us view an example $\mathcal{TDL}$ ABox:

$$\begin{array}{rr} \mathsf{Mary} : \mathsf{Human} & \mathsf{John} : \mathsf{Human} \\ \langle \mathsf{Mary}, t_1\rangle : \mathsf{birthday} & \langle \mathsf{John}, t_2\rangle : \mathsf{birthday} \\ \langle \mathsf{John}, \mathsf{Mary}\rangle : \mathsf{father} & t_2 < t_1 \end{array}$$

where Human is a concept name, $\mathsf{birthday}$ a temporal feature, father an abstract feature, Mary and John are from $\mathsf{O_a}$, and t_1 and t_2 are from $\mathsf{O_t}$. Obviously, this ABox states that John is the father of Mary and that John was born before Mary was born.

Observe that concept satisfiability (and thus also concept subsumption) can be reduced to ABox consistency: a concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ iff the ABox $\{a : C\}$ is satisfiable w.r.t. $\mathcal{T}$, where $a \in \mathsf{O_a}$.

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

Definition 5 (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* $P^\mathcal{D} \subseteq \Delta_\mathcal{D}^n$.

In Baader and Hanschke's original proposal [5], concrete domains are integrated into the description logic by using a concept constructor $\exists u_0, \ldots, u_n.P$, where $u_0, \ldots, u_n$ are paths and $P \in \Phi_\mathcal{D}$ is a predicate of arity $n + 1$. The semantics of this concrete domain constructor is as follows:

$$(\exists u_0, \ldots, u_n.P)^\mathcal{I} := \{d \in \Delta_\mathcal{I} \mid \exists x_0, \ldots, x_n \in \mathbb{Q} : u_i^\mathcal{I}(d) = x_i \text{ for } i \leq n \text{ and } (x_0, \ldots, x_n) \in P^\mathcal{D}\}.$$

Hence, $\mathcal{TDL}$ can be viewed as being equipped with the concrete domain $\mathcal{D}_{\mathcal{TDL}} := (\mathbb{Q}, \{<, \leq, =, \neq, \geq, >\})$, where all predicates are binary and have the obvious semantics.

We should also like to comment on a difference between $\mathcal{TDL}$ and some other description logics with concrete domains: in their original proposal of concrete domains, Baader and Hanschke do not distinguish between abstract and temporal features (which are usually called "concrete features" in a non-temporal concrete domain context [25]), but rather provide only one type of feature interpreted as a partial function from $\Delta_\mathcal{I}$ to $\Delta_\mathcal{I} \cup \Delta_\mathcal{D}$. We prefer the separateness of features since, in our opinion, this yields a clearer formalism while the difference in expressive power is negligible.

3 Temporal Reasoning with $\mathcal{TDL}$

In this section, we introduce a general framework for the representation of temporal conceptual knowledge using the description logic $\mathcal{TDL}$. As sketched in the introduction, despite its point-based semantics $\mathcal{TDL}$ can be used as a full-fledged interval-based temporal description logic. This fact is reflected by our framework, which allows to freely combine point-based and interval-based temporal representation. The usefulness of our framework is illustrated by several examples from the area of process engineering. This application of description logics has already been considered, e.g., by Sattler and Moli-

$$
\begin{aligned}
\mathsf{ATemporal} &\doteq t{\uparrow} \sqcap \ell{\uparrow} \sqcap r{\uparrow} \\
\mathsf{Temporal} &\doteq \mathsf{Point} \sqcup \mathsf{Interval} \\
\mathsf{Point} &\doteq \exists(t = t) \\
\mathsf{Interval} &\doteq \exists(\ell < r) \\
\top &\doteq \exists(t = t) \rightarrow (\ell{\uparrow} \sqcap r{\uparrow}) \\
&\quad \sqcap\ (\exists(\ell = \ell) \sqcup \exists(r = r)) \rightarrow (\exists(\ell < r) \sqcap t{\uparrow})
\end{aligned}
$$

Fig. 1. TBox $\mathcal{T}^*$ with basic definitions of the framework.

tor [31,28]. However, in Sattler's and Molitor's approach only static knowledge about process engineering is considered, i.e., there is no explicit representation of temporal relationships. We use our framework to show how the temporal aspects of this application domain can be represented in $\mathcal{TDL}$, thus refining Sattler's and Molitor's model.

The representation framework 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 either a time point or a time interval. We generally assume that single time points are represented by the temporal feature t, left endpoints of intervals are represented by the temporal feature ℓ, and right endpoints of intervals are represented by the temporal feature r.[2] This is captured by the TBox $\mathcal{T}^*$ displayed in Figure 1. The first four concept equations in the TBox define the relevant notions while the fifth equation rules out pathological cases such as objects whose extension is both a point and an interval. Note that concepts of the form $\exists(t = t)$ are used only to express that there exists an associated value for the temporal feature t. The TBox clearly implies that the concepts ATemporal, Point, and Interval are mutually disjoint, and that their union is equivalent to $\top$.

As noted in the introduction, interval-based reasoning with $\mathcal{TDL}$ is based on the Allen interval relations [1], which are displayed in Figure 2. To keep concepts readable, we define a suitable abbreviation for each of the 13 relations. For example,

$$\exists(p\ \mathit{contains}\ p') \text{ abbreviates } \exists(p\ell < p'\ell) \sqcap \exists(pr > p'r)$$

where p and p' are sequences of abstract features. It is a straightforward job to derive similar abbreviations for the other Allen relations given their definition in Figure 2. In what follows, we use self to denote the empty sequence of abstract features. For example,

$$\exists(p\ \mathit{starts}\ \mathsf{self}) \text{ abbreviates } \exists(p\ell = \ell)\ \sqcap\ \exists(pr < r).$$

[2] It is only for simplicity that we assume temporal entities to have a *unique* temporal extension. In principle, we could also allow multiple extensions e.g. for lifetime, childhood, worktime, etc.

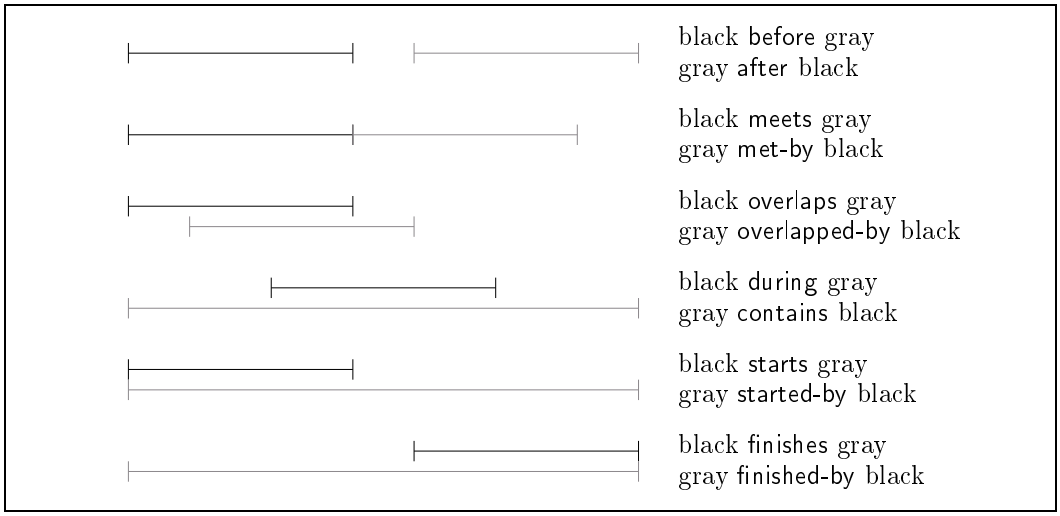

Fig. 2. The Allen relations (without equal).

Intuitively, self refers to the interval associated with the domain element at which the $\exists(p\ \mathit{starts}\ \mathsf{self})$ concept is "evaluated".

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

$$\begin{aligned}
\exists(p\ \mathit{beforep}\ p) &\text{ for } \exists(pt < p'\ell)\\
\exists(p\ \mathit{startsp}\ p') &\text{ for } \exists(pt = p'\ell)\\
\exists(p\ \mathit{duringp}\ p') &\text{ for } \exists(p'\ell < pt) \sqcap \exists(pt < p'r)\\
\exists(p\ \mathit{finishesp}\ p') &\text{ for } \exists(pt = p'r)\\
\exists(p\ \mathit{afterp}\ p') &\text{ for } \exists(p'r < pt)
\end{aligned}$$

where p and p' are again sequences of abstract features. We refrain from defining abbreviations for the inverses of these relations since they can easily be expressed by exchanging the arguments in the above abbreviations.

Now for the application of our framework in process engineering. 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 output quantity that is to be produced. It needs complex startup and shutdown phases before and after operation. Moreover, some weekly maintenance is needed to keep the device functional.

Let us first represent the underlying temporal structure consisting of weeks and days. The corresponding TBox can be found in Figure 3. In the figure, we use $C \sqsubseteq D$ as an abbreviation for $\top \doteq (C \to D)$. The first concept equation states that each week consists of seven days, where the i-th day

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

Fig. 3. Weeks and Days.

$$
\begin{aligned}
\mathsf{Day} \sqsubseteq\ & \exists \mathsf{start}.\mathsf{Startup} \sqcap \exists \mathsf{op}.\mathsf{Operation} \sqcap \exists \mathsf{shut}.\mathsf{Shutdown} \sqcap \\
& \exists(\mathsf{start} \circ \ell \ge \ell) \sqcap \\
& \exists(\mathsf{start}\ \mathit{meets}\ \mathsf{op}) \sqcap \\
& \exists(\mathsf{op}\ \mathit{meets}\ \mathsf{shut}) \sqcap \\
& \exists(\mathsf{shut} \circ r \le r) \\
\mathsf{Week} \sqsubseteq\ & \exists \mathsf{maint}.\mathsf{Maintenance} \sqcap \exists(\mathsf{self}\ \mathit{contains}\ \mathsf{maint}) \\
\mathsf{Interval} \sqsupseteq\ & \mathsf{Startup} \sqcup \mathsf{Operation} \sqcup \mathsf{Shutdown} \sqcup \mathsf{Maintenance}
\end{aligned}
$$

Fig. 4. Operation and Maintenance.

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. Moreover, each week has a successor week (accessible via the abstract feature next) that it temporally meets. The TBox clearly implies that days 2 to 6 are during the corresponding week although this is not explicitly stated.

Figure 4 defines the startup, operation, shutdown, and maintenance phases, where start, op, shut, and maint are abstract features and "$\circ$" is used as a separator for features that are used in sequences for better readability. In lines 2 to 5 of the concept equation for Day, we freely combine abbreviations from the framework with predicates from $\mathcal{TDL}$ to obtain succinct definitions. Taken together, these lines imply that phases are related to the corresponding day as follows: startup via *starts* or *during*, shutdown via *during* or *finishes*, and operation via *during*. Moreover, the startup phase meets the operation phase, which in turn meets the shutdown phase.

Until now, we did not say anything about the temporal relationship of maintenance and operation. This may be inadequate, if, for example, maintenance and operation are mutually exclusive. We can take this into account by using

the additional concept equation

$$\mathsf{Week} \sqsubseteq \bigsqcap_{1\leq i\leq 7} \Big(\begin{array}{l} \exists(\mathsf{maint}\ \mathit{before}\ \mathsf{day}_i \circ \mathsf{op}) \sqcup \\ \exists(\mathsf{maint}\ \mathit{after}\ \mathsf{day}_i \circ \mathsf{op}) \end{array} \Big) \qquad (*)$$

which expresses that the weekly maintenance phase must be either before or after the operation phase of every weekday. It is not hard to check that this is the case if and only if the weekly maintenance phase is strictly separated from the operation phase of any weekday.

This finishes the modeling of the basic properties of our production process. Let us define some more advanced concepts to illustrate reasoning with $\mathcal{TDL}$. For example, we can define a busy week as follows:

$$\mathsf{BusyWeek} \doteq \mathsf{Week} \sqcap \bigsqcap_{1\leq i\leq 7} \Big(\begin{array}{l} \exists(\mathsf{day}_i \circ \mathsf{start}\ \mathit{starts}\ \mathsf{day}_i) \sqcap \\ \exists(\mathsf{day}_i \circ \mathsf{shut}\ \mathit{finishes}\ \mathsf{day}_i) \end{array} \Big)$$

The concept equation says that on every day of a busy week, the startup phase starts at the beginning of the day and the shutdown finishes at the end of the day. Say now that it is risky to do maintenance during startup and shutdown phases and define

$$\mathsf{RiskyWeek} \doteq \mathsf{Week} \sqcap \neg \bigsqcap_{1\leq i\leq 7} \Big(\begin{array}{l} \exists(\mathsf{day}_i \circ \mathsf{start}\ \mathit{before}\ \mathsf{maint}) \sqcup \\ \exists(\mathsf{day}_i \circ \mathsf{shut}\ \mathit{after}\ \mathsf{maint}) \end{array} \Big)$$

expressing that, in a risky week, the maintenance phase is not strictly separated from the startup and shutdown phases. If $\mathcal{T}$ is the TBox obtained by taking the concept equations from Figures 1, 3, and 4, then a $\mathcal{TDL}$ reasoner can be used to deduce that $\mathsf{BusyWeek} \sqsubseteq_{\mathcal{T}} \mathsf{RiskyWeek}$, i.e., every busy week is a risky week: in a busy week, every day of the week is partitioned into startup, shutdown, and operation phases. Since maintenance may not overlap with operation phases by $(*)$, it must overlap with startup and/or shutdown phases, which means that the week is a risky week.

In order to demonstrate combined 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 phases. This is described by the concept equations in Figure 5, where up-int and down-int are abstract features. Note that the operator interaction is represented by a time point instead of a time interval. To illustrate reasoning, assume that, on Friday of calendar week 23, a shutdown interaction was performed by the maintenance team:

$$\mathsf{Week23} \sqsubseteq \mathsf{Week} \sqcap \exists(\mathsf{day}_5 \circ \mathsf{down\text{-}int}\ \mathit{duringp}\ \mathsf{maint}).$$

It is not hard to see that this is inconsistent with the description of faultless

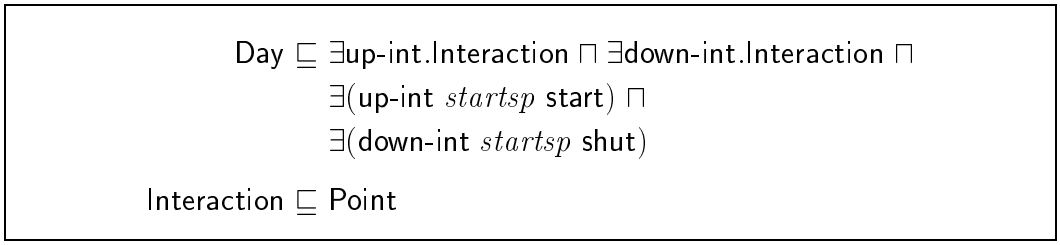

Fig. 5. Operator interaction.

operation from above, i.e., that Week23 is unsatisfiable: the shutdown interaction finishes the operation phase (since it starts the shutdown phase and the operation phase meets the shutdown phase), which means that the maintenance phase, during which the shutdown interaction was performed, is not strictly separated from the operation phase. This separateness, however, is required by $(*)$ since maintenance and operation are mutually exclusive. Hence, unsatisfiability of Week23 allows us to conclude that something went wrong on the Friday of calendar week 23.

It should be obvious how to extend the proposed framework to ABoxes and ABox reasoning. Details are left to the reader.

4 The Concept Satisfiability Algorithm

In this section, we prove the satisfiability of $\mathcal{TDL}$-concepts w.r.t. TBoxes to be decidable and obtain a tight ExpTime complexity bound for this reasoning task. By the reduction given in Section 2, we obtain ExpTime-completeness of $\mathcal{TDL}$-concept subsumption w.r.t. TBoxes as well. The upper bound is established using an automata-theoretic approach: first, models are abstracted 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 tree automaton $\mathcal{A}_{C,\mathcal{T}}$ (i.e., a Büchi tree automaton where every run is accepting) that accepts exactly the Hintikka-trees for C and $\mathcal{T}$. Hence, $\mathcal{A}_{C,\mathcal{T}}$ accepts the empty (tree-) language iff C is unsatisfiable w.r.t. $\mathcal{T}$. Since the translation produces at most an exponential blow-up in size and the emptiness-test for looping automata can be performed in polynomial time, we obtain the announced ExpTime upper bound.

Throughout this section, we assume that $\mathcal{TDL}$-concepts and TBoxes contain only the predicates $<$ and $=$. It is easy to see that this can be done without loss of generality since other predicates can be eliminated by exhaustively

applying the following rewrite rules:

$$\exists(u_1 \leq u_2) \rightsquigarrow \exists(u_1 < u_2) \sqcup \exists(u_1 = u_2)$$
$$\exists(u_1 \geq u_2) \rightsquigarrow \exists(u_1 > u_2) \sqcup \exists(u_1 = u_2)$$
$$\exists(u_1 \neq u_2) \rightsquigarrow \exists(u_1 > u_2) \sqcup \exists(u_1 < u_2)$$

For devising a satisfiability algorithm, it is interesting to note that $\mathcal{TDL}$ with general TBoxes lacks the finite model property since there exist satisfiable TBoxes such as $\top \doteq \exists(g < fg)$ having only infinite models (due to the semantics of the "<" predicate). Hence, Hintikka-trees and most other structures used for deciding satisfiability are (potentially) infinite.

4.1 Preliminaries

We introduce the basic notions needed for the automata-theoretic satisfiability algorithm like infinite trees, looping automata, and the language they accept. We also introduce constraint graphs which will be needed to take into account temporal information when defining Hintikka trees.

Definition 6 (Looping Automaton) *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 finite set Q of* states, *a finite 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 $r : \{1, \ldots, k\}^* \to Q$ 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 there exists a run, i.e., the language $L(\mathcal{A})$ of M-trees accepted by $\mathcal{A}$ is

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

Vardi and Wolper [39] show 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 T for C and $\mathcal{T}$ corresponds to a canonical model $\mathcal{I}$ of C and $\mathcal{T}$. Apart from representing the abstract domain $\Delta_{\mathcal{I}}$ together with the

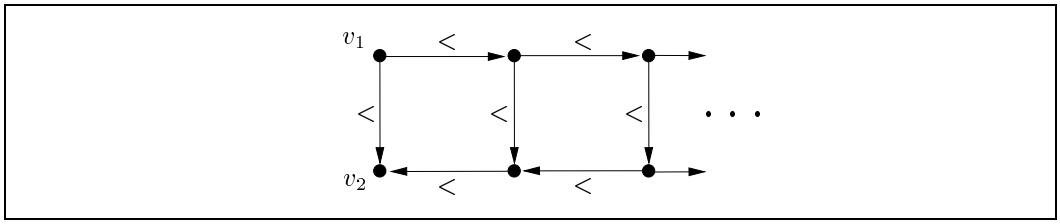

Fig. 6. A constraint graph containing no $<$-cycle that is unsatisfiable over $\mathbb{N}$.

interpretation of concepts and roles, T induces a directed graph whose edges are labeled with predicates from $\{<, =\}$. Such constraint graphs describe the "temporal part" of $\mathcal{I}$, i.e., temporal successors of elements of $\Delta_\mathcal{I}$ and their relationship by temporal predicates.

Definition 7 (Constraint Graph) *A* constraint graph *is a pair* $G = (V, E)$, *where*

- V *is a countable set of* nodes *and*
- $E \subseteq V \times \{=, <\} \times V$ *is a set of* edges *such that* $(v_1, =, v_2) \in E$ *implies* $(v_2, =, v_1) \in E$.

A constraint graph $G = (V, E)$ *is called* satisfiable *iff there exists a total mapping* δ *from* V *to* $\mathbb{Q}$ *such that* $\delta(v_1)\, P\, \delta(v_2)$ *for all* $(v_1, P, v_2) \in E$. *Such a mapping* δ *is called a* solution *for* G.

Let $G = (V, E)$ *be a constraint graph. A sequence of nodes* $v_0, \ldots, v_k \in V$ *is called a* cycle *in* G *if, for all* $i \leq k$, *we have* $(v_i, P, v_{(i+1) \bmod k}) \in E$ *for some* $P \in \{<, =\}$. *A cycle* $v_0, \ldots, v_k$ *is called a* $<$-cycle *if there is an* $i \leq k$ *with* $(v_i, <, v_{(i+1) \bmod k}) \in E$.

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 temporal part of the corresponding model, is satisfiable. The proof can be found in Appendix A.

Theorem 8 *A constraint graph is satisfiable iff it does not contain a* $<$*-cycle.*

Note that we use the rational numbers $\mathbb{Q}$ in the semantics of $\mathcal{TDL}$, and thus also for interpreting constraint graphs. All obtained results also apply if we choose $\mathbb{R}$ instead: the proof of Theorem 8 may remain unchanged and, intuitively, $\mathcal{TDL}$ does not "feel" the difference between $\mathbb{Q}$ and $\mathbb{R}$. However, it is interesting to note that Theorem 8 does *not* hold if satisfiability over non-dense structures such as $\mathbb{N}$ is considered: if there exist two nodes v_1 and v_2 such that the length of $<$-paths (which are defined in analogy to $<$-cycles) between v_1 and v_2 is unbounded, then a constraint graph is unsatisfiable over $\mathbb{N}$ even if it contains no $<$-cycle. Figure 6 shows such a constraint graph. And

indeed, $\mathcal{TDL}$ *does* feel the difference between $\mathbb{N}$ and dense structures such as $\mathbb{Q}$ and $\mathbb{R}$: 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 $\mathbb{R}$, but not over $\mathbb{N}$. Note that $\mathcal{T}$ enforces the constraint graph in Figure 6.

4.2 Path Normal Form

Apart from the assumption that only the predicates $<$ and $=$ occur in concepts and TBoxes, we require some more normalization as a prerequisite for the satisfiability algorithm. More specifically, we assume concepts and TBoxes to be in negation normal form (NNF) and, more importantly, restrict the length of paths, which will turn out to be rather convenient for some constructions like defining Hintikka-trees. We start with describing NNF conversion. A concept is said to be in *negation normal form* if negation occurs only in front of concept names. The following lemma shows that assuming NNF is not a restriction.

Lemma 9 (NNF Conversion) *Exhaustive application of the following rewrite rules translates $\mathcal{TDL}$-concepts to equivalent ones in NNF.*

$$\begin{aligned}
\neg\neg C &\rightsquigarrow C \\
\neg(C \sqcap D) &\rightsquigarrow \neg C \sqcup \neg D & \neg(C \sqcup D) &\rightsquigarrow \neg C \sqcap \neg D \\
\neg(\exists R.C) &\rightsquigarrow (\forall R.\neg C) & \neg(\forall R.C) &\rightsquigarrow (\exists R.\neg C) \\
\neg\exists(u_1\ P\ u_2) &\rightsquigarrow \exists(u_1\ \widetilde{P}\ u_2) \sqcup \exists(u_2 < u_1) \sqcup u_1{\uparrow} \sqcup u_2{\uparrow} & \neg(g{\uparrow}) &\rightsquigarrow \exists(g = g)
\end{aligned}$$

where $\widetilde{\cdot}$ denotes the exchange of predicates, i.e., $\widetilde{<}$ is $=$ and $\widetilde{=}$ is $<$. By $\mathsf{nnf}(C)$, we denote the result of converting C into NNF using the above rules.

We now introduce path normal form for $\mathcal{TDL}$-concepts and TBoxes.

Definition 10 (Path Normal Form) *A $\mathcal{TDL}$-concept C is in* path normal form (PNF) *iff it is in NNF and, for all subconcepts $\exists(u_1\ P\ u_2)$ of C, we have either*

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

A $\mathcal{TDL}$-TBox $\mathcal{T}$ is in path normal form iff all concepts 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 11 *Satisfiability of $\mathcal{TDL}$-concepts w.r.t. TBoxes can be reduced in polynomial time to satisfiability of $\mathcal{TDL}$-concepts in PNF w.r.t. TBoxes in PNF.*

Proof. 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 temporal features not used in C. We inductively define a mapping λ from paths u in C to concepts as follows:

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

For every $\mathcal{TDL}$-concept C, a corresponding concept $\rho(C)$ is obtained by replacing all subconcepts $\exists(u_1 \ P \ u_2)$ of C with $\exists([u_1] \ P \ [u_2]) \sqcap \lambda(u_1) \sqcap \lambda(u_2)$ and $g\!\uparrow$ with $[g]\!\uparrow$. We extend the mapping ρ to TBoxes in the obvious way, i.e., if

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

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. Using the rewrite rules from Lemma 9, we can convert C into an equivalent concept C' in NNF and $\mathcal{T}$ into an equivalent TBox $\mathcal{T}'$ in NNF. It is then easy to check that C' is satisfiable w.r.t. a TBox $\mathcal{T}'$ iff $\rho(C')$ is satisfiable w.r.t. $\rho(\mathcal{T}')$. Moreover, $\rho(C')$ and $\rho(\mathcal{T}')$ are clearly in PNF and the translation can be done in polynomial time. $\Box$

In what follows, we generally assume that all concepts and TBoxes are in path normal form. Moreover, we will often refer to TBoxes $\mathcal{T}$ in their *concept form* $C_\mathcal{T}$ which is defined as follows:

$$C_\mathcal{T} = \bigsqcap_{C \doteq D \in \mathcal{T}} \mathsf{nnf}(C \leftrightarrow D).$$

4.3 Defining Hintikka-trees

In this section, we define Hintikka-trees for $\mathcal{TDL}$-concepts C and TBoxes $\mathcal{T}$ (which are both required to be in PNF) and show that Hintikka-trees are proper abstractions of models, i.e., that there exists a Hintikka-tree for C and $\mathcal{T}$ iff there exists a model of C and $\mathcal{T}$.

Let C be a concept and $\mathcal{T}$ be a TBox. By $\mathsf{sub}(C, \mathcal{T})$, we denote the set of subconcepts of C and $C_\mathcal{T}$. We assume that existential concepts $\exists R.D \in \mathsf{sub}(C, \mathcal{T})$

with $R \in \mathsf{N_{rR}}$ are linearly ordered, and that $\mathcal{E}_i(C, \mathcal{T})$ yields the i-th such concept (starting with $i = 1$). Similarly, we assume the abstract features used in C or $\mathcal{T}$ to be linearly ordered and use $\mathcal{F}_i(C, \mathcal{T})$ to denote the i-th such feature (also starting with $i = 1$). The set of temporal features used in C or $\mathcal{T}$ is denoted by $\mathcal{G}(C, \mathcal{T})$.

We now define Hintikka-pairs which will be used as labels of nodes in Hintikka-trees.

Definition 12 (Hintikka-set, Hintikka-pair) *Let C be a concept and $\mathcal{T}$ a TBox. A set $\Psi \subseteq \mathsf{sub}(C, \mathcal{T})$ is a* Hintikka-set for C and $\mathcal{T}$ *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* $A \in \mathsf{sub}(C, \mathcal{T})$,
(H5) $g{\uparrow} \in \Psi$ *implies* $\exists(u_1\, P\, u_2) \notin \Psi$ *for all concepts* $\exists(u_1\, P\, u_2)$ *with* $g \in \{u_1, u_2\}$.

A Hintikka-pair (Ψ, χ) for C and $\mathcal{T}$ *consists of a Hintikka-set Ψ for C and $\mathcal{T}$ and a set χ of tuples (g_1, P, g_2) with $g_1, g_2 \in \mathcal{G}(C, \mathcal{T})$ such that*

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

By $\Gamma_{C,\mathcal{T}}$, we denote the set of all Hintikka-pairs for C and $\mathcal{T}$.

We say that an abstract feature $f \in \mathsf{N_{aF}}$ is enforced *by a Hintikka-pair (Ψ, χ) if $\exists f.C \in \Psi$ for some concept C or $\{\exists(f g_1\, P\, g_2), \exists(g_1\, P\, f g_2)\} \cap \Psi \neq \emptyset$ for some $g_1, g_2 \in \mathsf{N_{tF}}$ and $P \in \{<, =\}$. Similarly, a path u is* enforced *by (Ψ, χ) if u appears in χ or $\{\exists(u\, P\, u'), \exists(u'\, P\, u)\} \cap \Psi \neq \emptyset$ for some path u' and $P \in \{<, =\}$.*

Observe that, if a path u is enforced by a Hintikka-pair (Ψ, χ), then u has length 1 or 2: if u appears in χ, it has length 1 by definition; moreover, if $\{\exists(u\, P\, u'), \exists(u'\, P\, u)\} \cap \Psi \neq \emptyset$ for some u' and P, then u has length 1 or 2 since all concepts are in path normal form.

Intuitively, each node α of a (yet to be defined) Hintikka-tree T corresponds to a domain element d of the corresponding canonical model $\mathcal{I}$. The first component Ψ_α of the Hintikka-pair labeling α is the set of concepts from $\mathsf{sub}(C, \mathcal{T})$ satisfied by d. The second component χ_α states relationships between temporal successors of d. If, for example, $(g_1, <, g_2) \in \chi_\alpha$, then d must have g_1- and g_2-successors such that $g_1^\mathcal{I}(d) < g_2^\mathcal{I}(d)$. Note that the restrictions in χ_α are independent from concepts $\exists(g_1\, P\, g_2) \in \Psi_\alpha$, but rather describe "additional edges". As will be discussed below, these additional edges are used to ensure that the constraint graph induced by the Hintikka-tree T, which describes the

temporal 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 being the union of smaller constraint graphs, each one described by a Hintikka-pair labeling a node in T. These pair-graphs are defined next.

Definition 13 (Pair-graph) *Let C be a concept, $\mathcal{T}$ a TBox, and $p = (\Psi, \chi)$ a Hintikka-pair for C and $\mathcal{T}$. The* pair-graph $G(p) = (V, E)$ *of p is a constraint graph defined as follows:*

- *V is the set of paths enforced by p*
- $E = \mathsf{cl}_{=}(\chi \cup \{(u_1, P, u_2) \mid \exists(u_1\, P\, u_2) \in \Psi\})$,

where $\mathsf{cl}_{=}$ *is* equality closure, *i.e.* $\mathsf{cl}_{=}(E) = E \cup \{(v_2, =, v_1) \mid (v_1, =, v_2) \in E\}$.

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

Observe that, since all concepts are in path normal form and since no paths of length greater than one may appear in χ, we have $E' \cap E = \emptyset$ for every edge extension E' of pair-graphs (V, E).

There exists a close connection between completions of pair-graphs and the χ-component of Hintikka-pairs. Let α and β be nodes in a Hintikka-tree T representing domain elements d and e in the corresponding canonical model $\mathcal{I}$. Edges in Hintikka-trees represent role-relationships, i.e., if β is a successor of α in T, then there exists an $R \in \mathsf{N_R}$ such that $(d, e) \in R^{\mathcal{I}}$. Assume β is a successor of α and the edge between α and β represents relationship via the abstract feature f, i.e., we have $f^{\mathcal{I}}(d) = e$. The purpose of the second component χ_β of the Hintikka-pair labeling β is to fix the relationships between all temporal successors of e that "d 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 first component of the Hintikka-pair labeling α, then "d talks about" the temporal g_1-successor and the temporal g_3-successor of e. Hence, χ_β contains $(g_1, <, g_3)$, $(g_1, =, g_3)$, or $(g_3, <, g_1)$. This is formalized by demanding that the pair-graph $G(T(\alpha))$ of the Hintikka-pair labeling α together with all the edges from the χ-components of the successors of α are a completion of $G(T(\alpha))$. 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. As will be explained after the definition of Hintikka-trees, the purpose of all this is to achieve a "localized" detection of $<$-cycles in constraint-graphs induced by Hintikka-trees.

Definition 14 (Hintikka-tree) *Let C be a concept, $\mathcal{T}$ a TBox, k the number of existential subconcepts in $\mathsf{sub}(C, \mathcal{T})$, and ℓ the number of abstract features in $\mathsf{sub}(C, \mathcal{T})$. A $k + \ell + 1$-tuple of Hintikka-pairs $(p_0, \ldots, p_{k+\ell})$ with $p_i = (\Psi_i, \chi_i)$*

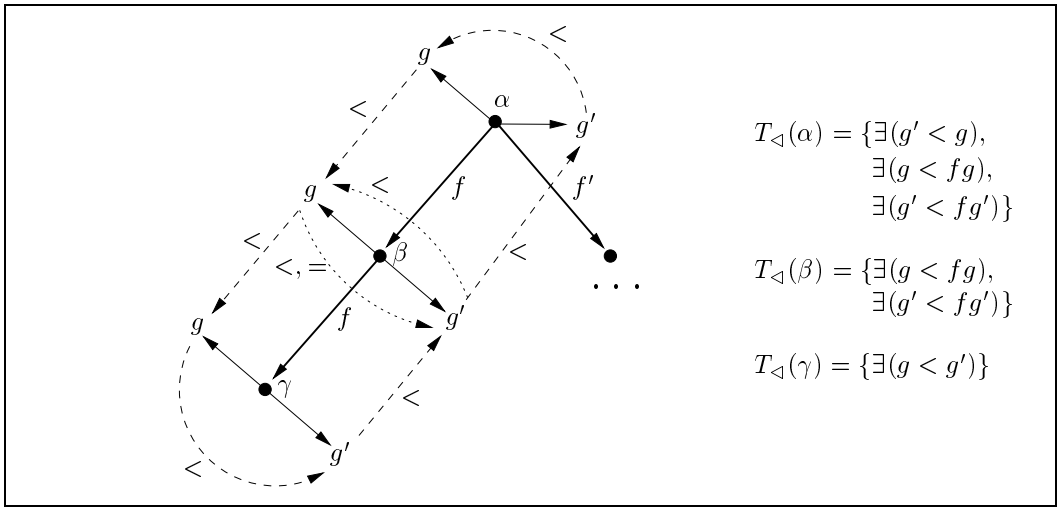

Fig. 7. Localized Cycle Detection.

and $G(p_0) = (V, E)$ *is called* matching *iff*

(H7) *if* $\exists R.D \in \Psi_0$ *and* $\mathcal{E}_i(C, \mathcal{T}) = \exists R.D$, *then* $D \in \Psi_i$,
(H8) *if* $\{\exists R.D, \forall R.E\} \subseteq \Psi_0$ *and* $\mathcal{E}_i(C, \mathcal{T}) = \exists R.D$, *then* $E \in \Psi_i$,
(H9) *if* $\exists f.D \in \Psi_0$ *and* $\mathcal{F}_i(C, \mathcal{T}) = f$, *then* $D \in \Psi_{k+i}$,
(H10) *if* f *is enforced by* p_0, $\mathcal{F}_i(C, \mathcal{T}) = f$, *and* $\forall f.D \in \Psi_0$, *then* $D \in \Psi_{k+i}$,
(H11) *the constraint graph* $(V, E \cup E')$ *with*

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

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

A $k + \ell$*-ary* $\Gamma_{C,\mathcal{T}}$*-tree* T *is a* Hintikka-tree for C *and* $\mathcal{T}$ *iff it 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* $(T(\alpha), T(\alpha 1), \ldots, T(\alpha j))$ *is matching, where* j *abbreviates* $k + \ell$.

For a Hintikka-tree T *and a 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)***.

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 suffices to check that the completions $\mathsf{cpl}(T, \alpha)$ are satisfiable as demanded by **(H11)**. An example for such a localization can be found in Figure 7. The Figure shows a non-local $<$-cycle (displayed as dashed edges)

in the constraint graph induced by a Hintikka-tree T (displayed as thickened solid edges). Assume for a moment that the dotted edges are not present, i.e. the relationship between the g-successor and the g'-successor of β is unknown. Then the constraint graphs $\mathsf{cpl}(T, \alpha)$, $\mathsf{cpl}(T, \beta)$, and $\mathsf{cpl}(T, \gamma)$ are all satisfiable if considered in isolation. Since **(H11)** indeed considers isolated graphs, the $<$-cycle cannot be detected. The problem is overcome as follows: let $G(T(\alpha)) = (V, E)$. Since $(V, E \cup E')$ with

$$E' = \{(fg_1, P, fg_2) \mid (g_1, P, g_2) \in T_{\triangleright}(\beta)\}$$

is required to be a completion of (V, E), we must have $(g, <, g') \in T_{\triangleright}(\beta)$, $(g, =, g') \in T_{\triangleright}(\beta)$, or $(g', <, g) \in T_{\triangleright}(\beta)$. In the first two cases, we obtain the lower dotted edge and $\mathsf{cpl}(T, \alpha)$ contains a $<$-cycle. In the third case, we obtain the upper dotted edge. Then we can repeat the whole process with α replaced by β and β replaced by γ such that, finally, either $\mathsf{cpl}(T, \beta)$ or $\mathsf{cpl}(T, \gamma)$ contains a $<$-cycle. Thus, the non-local $<$-cycle is broken down into smaller ones by "guessing" additional edges. The smaller $<$-cycles can then be detected by **(H11)**. Indeed, it is crucial that **(H11)** is a *local* condition since we need to define an automaton that accepts exactly Hintikka-trees, and automata work locally. It is worth noting that the localization of cycle detection as described above crucially depends on the path normal form.

The following lemma shows that Hintikka-trees are appropriate abstractions of models. This result is the main step towards devising a decision procedure since, as we shall see next, defining looping automata accepting exactly the Hintikka-trees for a given concept C and TBox $\mathcal{T}$ is a straightforward task. The proof can be found in Appendix A.

Lemma 15 *A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ iff there exists a Hintikka-tree for C and $\mathcal{T}$.*

4.4 Defining Looping Automata

To prove decidability of $\mathcal{TDL}$-concept satisfiability w.r.t. TBoxes, 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 and $\mathcal{T}$. Using the notion of matching tuples of Hintikka-pairs from Definition 14, this is rather straightforward.

Definition 16 *Let C be a concept, $\mathcal{T}$ a TBox, k the number of existential subconcepts in $\mathsf{sub}(C, \mathcal{T})$, and ℓ the number of abstract features in $\mathsf{sub}(C, \mathcal{T})$. The looping automaton $\mathcal{A}_{C,\mathcal{T}} = (Q, M, \Delta, I)$ is defined as follows:*

- $Q := M := \Gamma_{C,\mathcal{T}}$

- $I := \{(\Psi, \chi) \in Q \mid C \in \Psi\}$.
- $((\Psi, \chi), (\Psi', \chi'), (\Psi_1, \chi_1), \ldots, (\Psi_{k+\ell}, \chi_{k+\ell})) \in \Delta$ *iff*
  $(\Psi, \chi) = (\Psi', \chi')$ *and*
  $((\Psi, \chi), (\Psi_1, \chi_1), \ldots, (\Psi_{k+\ell}, \chi_{k+\ell}))$ *is matching.*

As a consequence of the following lemma and Lemmas 15, we can reduce satisfiability of concepts w.r.t. TBoxes (both in PNF) to the emptiness of the language accepted by looping automata.

Lemma 17 *T is a Hintikka-tree for C and $\mathcal{T}$ 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 16. For the "if" direction, let r be a run of $\mathcal{A}_{C,\mathcal{T}}$ on T. By definition of runs and of Δ, we have

$$r(\alpha) = T(\alpha) \text{ for all } \alpha \in \{1, \ldots, k+\ell\}^*.$$

Hence, it remains to be shown that r is a Hintikka-tree for C and $\mathcal{T}$, which is straightforward: (i) by definition of Q, r is a $\Gamma_{C,\mathcal{T}}$-tree; (ii) since, by definition of runs, $r(\epsilon) \in I$, **(H12)** is satisfied; and (iii) by definition of runs and of Δ, **(H13)** is satisfied.

Now for the "only if" direction. 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 $(r(\alpha), T(\alpha), r(\alpha_1), \ldots, r(\alpha_k)) \in \Delta$ for all $\alpha \in \{1, \ldots, k+\ell\}^*$. □

It is an immediate consequence of Lemmas 11, 15, and 17 and the decidability of the emptiness problem of looping automata [39] 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. In the following, we use $|C|$ to denote the length of the concept C and $\mathcal{T}$ to denote $\sum_{D \doteq E \in \mathcal{T}} |D| + |E|$.

Theorem 18 *Satisfiability of $\mathcal{TDL}$-concepts w.r.t. general TBoxes is* ExpTime*-complete.*

Proof. The lower bound is an immediate consequence of the fact that $\mathcal{ALC}$ with TBoxes is ExpTime-hard [32]. Hence, we may concentrate on the upper bound. We need to show that the size of $\mathcal{A}_{(C,\mathcal{T})}$ is exponential in $|C| + |\mathcal{T}|$ since, once that this is established, we can use Lemmas 11, 15, and 17 together with the fact that the emptiness problem for looping automata $\mathcal{A}_{(C,\mathcal{T})}$ is in PTime [39] to conclude that satisfiability of $\mathcal{TDL}$-concepts w.r.t. TBoxes can be decided in deterministic exponential time. Hence, let us investigate the size of $\mathcal{A}_{(C,\mathcal{T})} = (Q, M, \Delta, I)$. Obviously, the cardinality of $\mathsf{sub}(C, \mathcal{T})$ is linear in $|C| + |\mathcal{T}|$. Hence, by definition of $\mathcal{A}_{(C,\mathcal{T})}$ and Hintikka-pairs, the cardinality of Q, M, and I are exponential in $|C| + |\mathcal{T}|$. Together with the fact that Δ

contains $k+\ell$-tuples and $k+\ell$ is polynomial in $|C|+|\mathcal{T}|$, the exponential bound on the cardinality of Q implies that the cardinality of Δ is also exponential in $|C| + |\mathcal{T}|$. $\square$

Since subsumption can be reduced to (un)satisfiability, $\mathcal{TDL}$-concept subsumption w.r.t. TBoxes is also ExpTime-complete.

5 Deciding ABox Consistency

In this section, we extend the ExpTime upper bound just obtained to $\mathcal{TDL}$-ABox consistency w.r.t. TBoxes. The extended upper bound is established using a so-called precompletion algorithm [10,16]. The idea behind such algorithms is to proceed in two stages: first, a set of *completion rules* is exhaustively applied to the input ABox in order to make implicit information explicit. If an obvious contradiction is encountered during this process, then the input ABox is inconsistent and the second stage is not needed. If no contradiction is found, in the second stage we construct a reduction concept C_a for each object name a of the obtained ABox and check it for satisfiability w.r.t. the input TBox using the algorithm developed in the previous section. Then, the input ABox is satisfiable w.r.t. the input TBox if and only if all reduction concepts are satisfiable.

As in the previous section, we assume w.l.o.g. that all concepts (also inside TBoxes and ABoxes) contain only the predicates $<$ and $=$. Moreover, we require TBoxes and ABoxes to be in path normal form, where an ABox $\mathcal{A}$ is in PNF iff every concept occurring in $\mathcal{A}$ is in PNF. The next lemma shows that this assumption does not sacrifice generality.

Lemma 19 *Consistency of $\mathcal{TDL}$-ABoxes w.r.t. TBoxes can be reduced to consistency of $\mathcal{TDL}$-ABoxes in PNF w.r.t. TBoxes in PNF.*

Proof. Let $\mathcal{A}$ be an ABox and $\mathcal{T}$ a TBox, and let k be the length of the longest path occurring in $\mathcal{A}$ or $\mathcal{T}$. For every path $u = f_1 \cdots f_n g$ used in $\mathcal{A}$ or $\mathcal{T}$, we assume that $[g], [f_n g], \ldots, [f_1 \cdots f_n g]$ are temporal features not appearing in $\mathcal{A}$ or $\mathcal{T}$. Let ρ be the mapping from concepts to concepts in PNF and from TBoxes to TBoxes in PNF introduced in the proof of Lemma 11. Construct an ABox $\rho(\mathcal{A})$ from $\mathcal{A}$ by performing the following steps:

(1) Replace every assertion $a : C \in \mathcal{A}$ with $a : \rho(C)$;
(2) Replace every assertion $\langle a, x\rangle : g \in \mathcal{A}$ with $\langle a, x\rangle : [g]$;
(3) For $i = 1, \ldots, k-1$ do the following: for every pair of assertions

$$\langle a, b\rangle : f,\ \langle b, x\rangle : [u] \in \mathcal{A}$$

where the length of u is i and fu is a postfix of a path occurring in $\mathcal{A}$ or $\mathcal{T}$, add $\langle a, x\rangle : [fu]$ to $\mathcal{A}$.

It is straightforward to prove that $\mathcal{A}$ is satisfiable w.r.t. $\mathcal{T}$ iff $\rho(\mathcal{A})$ is satisfiable w.r.t. $\rho(\mathcal{T})$. Moreover, the size of $\rho(\mathcal{A})$ and $\rho(\mathcal{T})$ is polynomial in $n = |\mathcal{A}|+|\mathcal{T}|$ and $\rho(\mathcal{A})$ and $\rho(\mathcal{T})$ can be constructed in polynomial time. While $\rho(\mathcal{T})$ was treated in the proof of Lemma 11, for $\rho(\mathcal{A})$ this can be seen as follows. Since the number of postfixes of paths occurring in $\mathcal{A}$ and $\mathcal{T}$ is bounded by n, the number of object names and time point names in $\mathcal{A}$ is also bounded by n, and no new object names are introduced, the number of assertions of the form $\langle a, x\rangle : [u]$ generated in Step 3 is bounded by n^3. Since the number of assertions $\langle a, b\rangle : f$ is bounded by n, the number of pairs to be considered in each step of the "for" loop in Step 3 is thus bounded by n^4. Since we clearly have $k \leq n$, $\rho(\mathcal{A})$ can be computed in time n^5. $\square$

The completion rules can be found in Figure 8. In the formulation of the rules, we write $\mathcal{A}(a)$ for $\{C \mid a : C \in \mathcal{A}\}$ and call a time point name $x \in \mathsf{O_t}$ *fresh* in an ABox $\mathcal{A}$ if x does not occur in $\mathcal{A}$. Note that the rules $\mathsf{R}\sqcup$ and Rch yield more than one possible outcome. Intuitively, the precompletion algorithm has to explore *all* possible outcomes—more details are given later on. Note that we cannot use the usual non-deterministic "guessing" here since we are heading for a *deterministic* time bound.

While the rules $\mathsf{R}\sqcap$, $\mathsf{R}\sqcup$, $\mathsf{R}\forall$, and $\mathsf{R}\doteq$ are straightforward, the other rules deserve some comments. The $\mathsf{R}\exists\mathsf{f}$ rule deals with concepts $\exists f.C$, where $f \in \mathsf{N_{aF}}$. Since it is our goal to make explicit information for *existing* object names rather than generating new ones, this rule only applies to a concept $\exists f.C \in \mathcal{A}(a)$ if the object a already has an f-successor (i.e., an object name b with $\langle a, b\rangle : R \in \mathcal{A}$). For the same reason, concepts $\exists R.C$ with $R \in \mathsf{N_{rR}}$ are not expanded at all, but rather "treated" as part of the reduction concepts. The rules $\mathsf{Rc}1$, $\mathsf{Rc}2$ and $\mathsf{Rc}3$ deal with concepts $\exists(u_1\ P\ u_2)$: there exists one rule for each syntactic form that PNF allows. Observe that $\mathsf{Rc}2$ and $\mathsf{Rc}3$ generate new time point names, but, similar to the $\mathsf{R}\exists\mathsf{f}$ rule, none of the Rc rules generates new object names even if the paths u_1 and u_2 involve abstract features. Intuitively, if $\exists(fg_1\,P\,g_2) \in \mathcal{A}(a)$ and a has *no* f-successor, then it suffices to treat the concept $\exists(fg_1\,P\,g_2)$ in the reduction concept. The Rch rule has the character of a "choose rule" (c.f. for example [17]) and is needed to ensure that the relation between any two temporal successors of an object a is recorded as a concept of the form $\exists(g_1\,P\,g_2)$ in the node label of a. This is necessary since the relation between such temporal successors must be passed to the satisfiability algorithm as part of the reduction concept. Finally, the Rfe rule is a "fork elimination rule" (c.f. for example [5,23]) that is needed to enforce the functionality of abstract and temporal features.

If an ABox $\mathcal{A}'$ can be obtained from an ABox $\mathcal{A}$ by exhaustive rule appli-

R$\sqcap$	if $C_1 \sqcap C_2 \in \mathcal{A}(a)$ and $\{C_1, C_2\} \not\subseteq \mathcal{A}(a)$ then $\mathcal{A} := \mathcal{A} \cup \{a : C_1, a : C_2\}$
R$\sqcup$	if $C_1 \sqcup C_2 \in \mathcal{A}(a)$ and $\{C_1, C_2\} \cap \mathcal{A}(a) = \emptyset$ then $\mathcal{A}_1 := \mathcal{A} \cup \{a : C_1\}$ and $\mathcal{A}_2 := \mathcal{A} \cup \{a : C_2\}$
R$\exists$f	if $\exists f.C \in \mathcal{A}(a)$, $\langle a, b\rangle : f \in \mathcal{A}$, and $C \notin \mathcal{A}(b)$ then set $\mathcal{A} := \mathcal{A} \cup \{b : C\}$
R$\forall$	if $\forall R.C \in \mathcal{A}(a)$, $\langle a, b\rangle : R \in \mathcal{A}$, and $C \notin \mathcal{A}(b)$ then set $\mathcal{A} := \mathcal{A} \cup \{b : C\}$
Rc1	if $\exists(g_1\, P\, g_2) \in \mathcal{A}(a)$, $\{\langle a, x_1\rangle : g_1,\ \langle a, x_2\rangle : g_2\} \subseteq \mathcal{A}$, and $x_1\, P\, x_2 \notin \mathcal{A}$ then set $\mathcal{A} := \mathcal{A} \cup \{x_1\, P\, x_2\}$
Rc2	if $\exists(f\, g_1\, P\, g_2) \in \mathcal{A}(a)$, $\langle a, b\rangle : f \in \mathcal{A}$, and there are no $x_1, x_2 \in \mathsf{O_t}$ such that $\{\langle b, x_1\rangle : g_1,\ \langle a, x_2\rangle : g_2,\ x_1\, P\, x_2\} \subseteq \mathcal{A}$ then set $\mathcal{A} := \mathcal{A} \cup \{\langle b, x_1\rangle : g_1, \langle a, x_2\rangle : g_2, x_1\, P\, x_2\}$ where x_1 and x_2 are fresh in $\mathcal{A}$
Rc3	Symmetric to Rc2 but for concepts $\exists(g_1\, P\, f\, g_2) \in \mathcal{A}(a)$
Rch	if $\{\langle a, x_1\rangle : g_1,\ \langle a, x_2\rangle : g_2\} \subseteq \mathcal{A}$ and $\{\exists(g_1 < g_2),\ \exists(g_1 = g_2),\ \exists(g_2 < g_1)\} \cap \mathcal{A}(a) = \emptyset$ then set $\mathcal{A}_1 := \mathcal{A} \cup \{a : \exists(g_1 < g_2)\}$, $\mathcal{A}_2 := \mathcal{A} \cup \{a : \exists(g_1 = g_2)\}$, and $\mathcal{A}_3 := \mathcal{A} \cup \{a : \exists(g_2 < g_1)\}$.
R$\doteq$	if $C_{\mathcal{T}} \notin \mathcal{A}(a)$ then set $\mathcal{A} := \mathcal{A} \cup \{a : C_{\mathcal{T}}\}$
Rfe	if $\{\langle a, b\rangle : f, \langle a, c\rangle : f\} \subseteq \mathcal{A}$ and $b \neq c$ (resp. $\{\langle a, x\rangle : g, \langle a, y\rangle : g\} \subseteq \mathcal{A}$ and $x \neq y$) then replace b by c in $\mathcal{A}$ (resp. x by y)

Fig. 8. Completion rules for $\mathcal{TDL}$.

cation using a TBox $\mathcal{T}$, then $\mathcal{A}'$ is called *precomplete* and a *precompletion* of $\mathcal{A}$ w.r.t. $\mathcal{T}$. Interleaved with rule application, the precompletion checks for obvious contradictions. These are formalized as follows.

Definition 20 (Clash) *Let $\mathcal{A}$ be an ABox. $\mathcal{A}$ is called* temporally satisfiable *iff the constraint graph $G(\mathcal{A}) = (V, E)$ is satisfiable, where*

- $V = \{x \in \mathsf{O_t} \mid x \text{ occurs in } \mathcal{A}\}$;
- $E = \{(x_1, P, x_2) \mid x_1\, P\, x_2 \in \mathcal{A}\}$.

$\mathcal{A}$ is said to contain a clash *iff one of the following conditions applies:*

define procedure $\mathsf{cons}(\mathcal{A}, \mathcal{T})$
  while a rule $\mathsf{R} \in \{\mathsf{R}\sqcap, \mathsf{R}\exists\mathsf{f}, \mathsf{R}\forall, \mathsf{Rc1}, \mathsf{Rc2}, \mathsf{Rc3}, \mathsf{R}\doteq, \mathsf{Rfe}\}$ is applicable to $\mathcal{A}$ **do**
    apply R to $\mathcal{A}$
  if a completion rule $\mathsf{R} \in \{\mathsf{R}\sqcup, \mathsf{Rch}\}$ is applicable to $\mathcal{A}$ **then**
    apply R to $\mathcal{A}$ yielding $\mathcal{A}_1, \ldots, \mathcal{A}_k$ $(k \in \{2, 3\})$
    if $\mathsf{cons}(\mathcal{A}_i, \mathcal{T}) = \mathsf{consistent}$ for some $i \in \{1, \ldots, k\}$ **then**
      return consistent
    return inconsistent
  if $\mathcal{A}$ contains a clash **then**
    return inconsistent
  if $\mathsf{sat}(\bigsqcap_{C \in \mathcal{A}(a)} C, \mathcal{T}) = \mathsf{satisfiable}$ for every $a \in \mathsf{O_a}$ in $\mathcal{A}$ **then**
    return consistent
  return inconsistent

Fig. 9. The $\mathcal{TDL}$ precompletion algorithm.

(1) $\{A, \neg A\} \subseteq \mathcal{A}(a)$ *for a concept name* A *and object name* $a \in \mathsf{O_a}$,
(2) $g\!\uparrow \in \mathcal{A}(a)$ *for some* $a \in \mathsf{O_a}$ *and there exists an* $x \in \mathsf{O_t}$ *such that* $\langle a, x\rangle : g \in \mathcal{A}$, *or*
(3) $\mathcal{A}$ *is not temporally satisfiable.*

If $\mathcal{A}$ *does not contain a clash, then* $\mathcal{A}$ *is* clash-free.

The precompletion algorithm itself is given in Figure 9 in a pseudocode notation. In the formulation of the algorithm, we use $\mathsf{sat}(C, \mathcal{T})$ to denote the result of applying the satisfiability algorithm from the previous section to the concept C and TBox $\mathcal{T}$. The general idea behind the algorithm and the correctness proofs is that models of the reduction concepts can be "plugged together" to form a model of the input ABox.

We now prove termination and investigate the time complexity of the algorithm. In order to do this, we need a size function for ABoxes. To this end, set

$$|a : C| \;:=\; C$$
$$|\langle a, b\rangle : R| \;:=\; |\langle a, x\rangle : g| \;:=\; |x_1\, P\, x_2| \;:=\; 2$$

and $|\mathcal{A}| := \sum_{\alpha \in \mathcal{A}} |\alpha|$. First, we establish an upper bound for the number of rules that may be applied to a given ABox.

Lemma 21 *Let* $\mathcal{A}$ *be an ABox,* $\mathcal{T}$ *a TBox, and* $\mathcal{A}_0, \ldots, \mathcal{A}_k$ *with* $\mathcal{A}_0 = \mathcal{A}$ *a sequence of ABoxes obtained by repeated rule application. Then* $k \leq p(|\mathcal{A}| + |\mathcal{T}|)$ *for some polynomial* $p(n)$.

Proof. We abbreviate $|\mathcal{A}| + |\mathcal{T}|$ by n. Each of the rules $\mathsf{R}\sqcap$, $\mathsf{R}\sqcup$, $\mathsf{R}\exists\mathsf{f}$, $\mathsf{R}\forall$, Rch,

and $\mathsf{R}\doteq$ adds a new concept to the label of an object name. Since all added concepts are from the set

$$\chi := \mathsf{sub}(\mathcal{A},\mathcal{T}) \cup \{\exists(g_1\, P\, g_2) \mid P \in \{<,=\} \text{ and } g_1, g_2 \text{ used in } \mathsf{sub}(\mathcal{A},\mathcal{T})\}$$

and $|\chi| \leq 2n^2 + n$, the number of applications of the above rules per object name is also bounded by $2n^2 + n$ and their overall number of applications is bounded by $2n^3 + n^2$. There are four remaining rules:

- $\mathsf{Rc1}$, $\mathsf{Rc2}$, $\mathsf{Rc3}$. These rules are applied at most once per concept $\exists(u_1\, P\, u_2) \in \chi$ and object name a in $\mathcal{A}$. Since no new object names are introduced, there are at most $2n^3 + n^2$ applications of $\mathsf{Rc1}$, $\mathsf{Rc2}$ and $\mathsf{Rc3}$. Moreover, since each rule application introduces at most 2 new time point names and $\mathsf{Rc2}$ and $\mathsf{Rc3}$ are the only rules to introduce new time point names, it also follows that the number of newly introduced time point names is bounded by $4n^3 + 2n^2$.
- Rfe. The rule is applied at most once per object and time point name. The initial ABox contains at most n object and time point names, no new object names are generated, and at most $4n^3 + 2n^2$ new time point names are generated. Hence, the number of applications of Rfe is bounded by $4n^3 + 2n^2 + n$.

Taking together these observations, it is obvious that there exists a polynomial $p(n)$ as required. □

We can now prove termination.

Proposition 22 (Termination) *If started on an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, the precompletion algorithm terminates after time exponential in $|\mathcal{A}| + |\mathcal{T}|$.*

Proof. Assume that the precompletion algorithm is started on an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. The precompletion algorithm is a recursive procedure. In every recursion step, either several recursion calls or several calls to the sat algorithm are made. Obviously, a run of the algorithm induces a recursion tree, where nodes in the tree are recursion steps and edges are recursion calls. These recursion trees have the following properties:

(1) Since at most three recursion calls are made per recursion step, the out-degree is three.
(2) Every path of the recursion tree induces a sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \ldots$ with $\mathcal{A}_0 = \mathcal{A}$ that can be obtained by repeated rule application. By Lemma 21, the length of this sequence is bounded by $p(|\mathcal{A}| + |\mathcal{T}|)$, and, thus, the depth of recursion trees is also bounded by $p(|\mathcal{A}| + |\mathcal{T}|)$.

This implies that the total number of recursion steps made by the algorithm is bounded by $3^{p(|\mathcal{A}|+|\mathcal{T}|)}$. Since none of the rules introduces new object names, the number of sat calls per recursion step is bounded by $|\mathcal{A}|$ and the total

number of calls to sat by $3^{p(|\mathcal{A}|+|\mathcal{T}|)} \cdot |\mathcal{A}|$. Together with Theorem 18, we obtain termination and the exponential time bound. □

We now establish a series of lemmas that will finally allow to establish soundness and completeness of the precompletion algorithm. The proofs of all lemmas can be found in Appendix B. We start with showing that the construction of precompletions preserves (in)consistency.

Lemma 23 *Let $\mathcal{A}$ be an ABox and $\mathcal{T}$ a TBox. Then $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff there exists a precompletion $\mathcal{A}'$ of $\mathcal{A}$ w.r.t. $\mathcal{T}$ such that $\mathcal{A}'$ is consistent w.r.t. $\mathcal{T}$.*

Our next aim is to show that every clash-free precomplete ABox, for which all reduction concepts are satisfiable, is consistent. We start with a technical lemma which states that, intuitively, for every precomplete ABox $\mathcal{A}$ with satisfiable reduction concepts, we can find models for the reduction concepts such that these model's temporal parts can be "plugged into" solutions for the constraint graph $G(\mathcal{A})$ induced by $\mathcal{A}$. Recall that we use $\mathsf{con}(\mathcal{A}, a)$ to denote the reduction concept for $a \in \mathsf{O_a}$ in $\mathcal{A}$.

Lemma 24 *Let $\mathcal{A}$ be a precomplete ABox, δ a solution for $G(\mathcal{A})$, and $a \in \mathsf{O_a}$ used in $\mathcal{A}$. If $\mathsf{con}(\mathcal{A}, a)$ is satisfiable w.r.t. $\mathcal{T}$, then there exists a model $\mathcal{I}$ of $\mathsf{con}(\mathcal{A}, a)$ and $\mathcal{T}$ and a $d_a \in \mathsf{con}(\mathcal{A}, a)^{\mathcal{I}}$ such that, for all $\langle a, x\rangle : g \in \mathcal{A}$, we have $g^{\mathcal{I}}(d_a) = \delta(x)$.*

The following lemma is central for proving soundness and completeness. Its proof follows the intuition given above: models for the reduction concepts are "plugged together" in order to form a model for the ABox. To deal with the temporal parts of models, we relay on Lemma 24.

Lemma 25 *Let $\mathcal{A}$ be a precompletion of an ABox $\mathcal{A}'$ w.r.t. a TBox $\mathcal{T}$. $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}$ is clash-free and $\mathsf{con}(\mathcal{A}, a)$ is satisfiable w.r.t. $\mathcal{T}$ for every $a \in \mathsf{O_a}$ used in $\mathcal{A}$.*

Finally, we prove soundness and completeness.

Proposition 26 (Soundness and Completeness) *If the precompletion algorithm is started on an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, then it returns* $\mathsf{consistent}$ *if $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ and* $\mathsf{inconsistent}$ *otherwise.*

Proof. Let $\mathcal{A}$ and $\mathcal{T}$ be an input to the precompletion algorithm. Since the order of rule application is clearly irrelevant, the algorithm computes all clash-free precompletions of $\mathcal{A}$ w.r.t. $\mathcal{T}$. For each such precompletion $\mathcal{A}'$, it checks whether the reduction concept $\mathsf{con}(\mathcal{A}, a)$ is satisfiable for all $a \in \mathsf{O_a}$ occurring in $\mathcal{A}'$. It returns $\mathsf{consistent}$ if it finds a precompletion for which this is true and, by Proposition 22, $\mathsf{inconsistent}$ otherwise. Soundness and completeness

are thus an immediate consequence of Lemmas 23 and 25. □

Taking together Propositions 22 and 26, we obtain an ExpTime upper bound for $\mathcal{TDL}$-ABox consistency w.r.t. TBoxes. Together with the lower bound from Theorem 18, we obtain the following result.

Theorem 27 *$\mathcal{TDL}$-ABox consistency w.r.t. general TBoxes is* ExpTime*-complete.*

6 Discussion

In this paper, we have introduced the description logic $\mathcal{TDL}$ whose distinguishing feature is that it admits both interval-based temporal representation and general TBoxes, while still being decidable (and ExpTime-complete). As noted in the introduction, this result also shows that there exist interesting concrete domains whose combination with general TBoxes do not lead to undecidability of reasoning. Starting from the basic decidability results proved in the current paper, there are lots of options for promising future research. Let us discuss a few of them.

(1) It would be interesting to enhance the expressive power of both the temporal and the DL part of $\mathcal{TDL}$. Concerning the temporal part, one could add unary predicates P_q, where $P \in \{<, \leq, =, \neq, \geq, >\}$ and $q \in \mathbb{Q}$. This would allow *quantitative* temporal representation by referring to "concrete" time points. On the DL side, $\mathcal{ALC}$ can be extended by various means of expressivity that usually appear in state-of-the-art description logics such as inverse roles, qualifying numbers restrictions, and transitive roles. All these extensions have been realized in the recent paper [21], where it is shown that the popular DL $\mathcal{SHIQ}$ extended with a $\mathcal{TDL}$-style concrete domain and the afore mentioned unary predicates is still decidable and ExpTime-complete. The resulting logic is called $\mathbb{Q}$-$\mathcal{SHIQ}$ and has found interesting applications in reasoning about conceptual database schemas [24].

(2) The version of $\mathcal{TDL}$ defined in this paper uses $\mathbb{Q}$ as its temporal structure. As discussed in Section 4.1, it makes a difference whether dense temporal structures such as $\mathbb{Q}$ and $\mathbb{R}$ or non-dense structures such as $\mathbb{N}$ are used: there exist $\mathcal{TDL}$-concepts that are satisfiable over $\mathbb{Q}$ but not over $\mathbb{N}$. It would thus be interesting to consider a variant of $\mathcal{TDL}$ that is based on $\mathbb{N}$, or even to add to the current version of $\mathcal{TDL}$ a unary predicate int stating that a time point/rational number is an integer. In this case, the algorithm would, additionally, have to detect unsatisfiable constraint graphs such as the one in Figure 6. We conjecture that this cannot be done without adding a non-trivial acceptance condition to our automata model, e.g. switching from looping to

Büchi automata. Using $\mathbb{N}$ as the temporal structure would be useful if discreteness of time is assumed (e.g. a time point is viewed as a reference to some particular second in time), and if the resulting logic is used in non-temporal applications: we may, e.g., use temporal features to store the number of children that a person has, rather than storing a time point. Clearly, fractional numbers make no sense in this context.

(3) It would be natural to define a spatial description logic $\mathcal{SDL}$ by replacing the temporal predicates of $\mathcal{TDL}$ with spatial ones. For example, one could use the set set of eight "topological" relations called RCC-8 [30,7], which describe all possible ways in which two regions can be related in topological spaces, and which in many aspects resemble the Allen relations. Our guess is that again a decidable formalism is obtained, but many proof techniques would clearly have to be reworked. For example, the RCC-8 relations cannot be broken down to the predicates $\{<, =\}$.

It is also interesting to note that there are certain well-known limitations for extending the temporal part of $\mathcal{TDL}$. For example, if we think of the $\exists(u_1\, P\, u_2)$ constructor as a means for talking about rational numbers rather than about temporal information, then it seems natural to add predicates for arithmetics such as a ternary addition predicate. However, it is shown in [26] that it suffices to add to $\mathcal{TDL}$ a unary predicate for equality to zero and a binary predicate for incrementation in order to make reasoning undecidable.

References

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

[2] A. Artale and E. Franconi. Temporal description logics. In Gabbay et al. [11]. To appear.

[3] A. Artale and E. Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research (JAIR)*, 9:463–506, 1998.

[4] A. Artale and C. Lutz. A correspondence between temporal description logics. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, number 22 in CEUR-WS (http://ceur-ws.org/), pages 145–149, 1999.

[5] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, Australia, 1991.

[6] F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications.* Cambridge University Press, 2003.

[7] B. Bennett. Modal logics for qualitative spatial reasoning. *Journal of the Interest Group in Pure and Applied Logic*, 4(1), 1997.

[8] C. Bettini. Time-dependent concepts: representation and reasoning using temporal description logics. *Data & Knowledge Engineering*, 22:1–38, 1997.

[9] D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 303–307, 1996.

[10] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: from subsumption to instance checking. *Journal of Logic and Computation*, 4(4):423–452, 1994.

[11] D. Gabbay, M. Fisher, and L. Vila, editors. *Handbook of Time and Temporal Reasoning in Artificial Intelligence.* MIT Press. To appear.

[12] D. M. Gabbay, I. M. Hodkinson, and M. A. Reynolds, editors. *Temporal Logic: Mathematical Foundations and Computational Aspects, Volume 1.* Oxford University Press, Logic Guides 28, 1994.

[13] G. D. Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94). Volume 1*, pages 205–212. AAAI Press, 1994.

[14] V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, number 2083 in Lecture Notes in Artifical Intelligence, pages 701–705. Springer-Verlag, 2001.

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

[16] B. Hollunder. Consistency checking reduced to satisfiability of concepts in terminological systems. *Annals of Mathematics and Artificial Intelligence*, 18:133–157, 1996.

[17] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 335–346, Boston, MA, USA, 1991.

[18] I. Horrocks. Using an expressive description logic: Fact or fiction? In *Proceedings of the Sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR98)*, pages 636–647, 1998.

[19] 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, 1999.

[20] C. Lutz. Interval-based temporal reasoning with general TBoxes. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 89–94. Morgan-Kaufmann, 2001.

[21] C. Lutz. Adding numbers to the $\mathcal{SHIQ}$ description logic—First results. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 191–202. Morgan Kaufman, 2002.

[22] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2002.

[23] C. Lutz. PSpace reasoning with the description logic $\mathcal{ALCF}(\mathcal{D})$. *Logic Journal of the IGPL*, 10(5):535–568, 2002.

[24] C. Lutz. Reasoning about entity relationship diagrams with complex attribute dependencies. In I. Horrocks and S. Tessaris, editors, *Proceedings of the International Workshop in Description Logics 2002 (DL2002)*, number 53 in CEUR-WS (http://ceur-ws.org/), pages 185–194, 2002.

[25] C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*. World Scientific Publishing Co. Pte. Ltd., 2003. To appear.

[26] C. Lutz. NExpTime-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 2003. To appear.

[27] C. Lutz, H. Sturm, F. Wolter, and M. Zakharyaschev. Tableaux for temporal description logic with constant domain. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, number 2083 in Lecture Notes in Artifical Intelligence, pages 121–136. Springer-Verlag, 2001.

[28] R. Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000.

[29] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43–66, 1995.

[30] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176. Morgan Kaufman, 1992.

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

[32] K. D. Schild. A correspondence theory for terminological logics: Preliminary report. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471. Morgan Kaufmann, 1991.

[33] 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*, volume 727 of *Lecture Notes in Artificial Intelligence*, pages 105–120. Springer-Verlag, 1993.

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

[35] A. Schmiedel. Temporal terminological logic. In W. Dietterich, Tom; Swartout, editor, *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI'90)*, pages 640–645. MIT Press, 1990.

[36] H. Sturm and F. Wolter. A tableau calculus for temporal description logic: The expanding domain case. *Journal of Logic and Computation*, 2001.

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

[38] 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.

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

[40] 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. Morgan Kaufmann, 1990.

[41] 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.

A Proofs for Section 4

The first task is to prove Theorem 8. For the proof, it will be helpful to define the notion of paths in constraint graphs G: a *path* Q in G is a finite non-empty sequence of nodes $v_0, \ldots, v_k \in V$ such that, for all i with $i < k$, we have $(v_i, P, v_{i+1}) \in E$ for some $P \in \{<, =\}$. A path $v_0, \ldots, v_k$ is a $=$*-path* iff $(v_i, =, v_{i+1}) \in E$ for $i < k$.

Theorem 8 A constraint graph is satisfiable iff it does not contain a $<$-cycle.

Proof. Since the "only if" direction is trivial, we concentrate on the "if" 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 $v \in V$, we denote the equivalence class of v w.r.t. $\sim$ by $[v]_\sim$. Define a new constraint graph $G' = (V', E')$ as follows:

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

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) [37]. 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 $\prec$ with a countable domain D can be embedded into $\mathbb{Q}$ such that the ordering is preserved. This suffices to complete the proof since it implies that that there exists a total mapping δ from V to $\mathbb{Q}$ such that $v_1 \prec_{E'} v_2$ implies $\delta(v_1) < \delta(v_2)$. It is obvious that δ is a solution for G' and it is straightforward to use δ to construct a solution for G.

Let $d_0, d_1, \ldots$ be an enumeration of D. We use induction on this enumeration to define a function δ from D to $\mathbb{Q}$ such that $d_1 \prec d_2$ implies $\delta(d_1) < \delta(d_2)$ for all $d_1, d_2 \in D$.

(1) For the induction start, set $\delta(d_0)$ to some $r \in \mathbb{Q}$.
(2) Assume that $\delta(d_i)$ is defined for all $i < k$. We distinguish three cases:
   (a) $d_i \prec d_k$ for all $i < k$. Since $\mathbb{Q}$ has no maximum, there exists an $r \in \mathbb{Q}$ such that $r > \delta(d_i)$ for all $i < k$. Set $\delta(d_k) := r$.
   (b) $d_k \prec d_i$ for all $i < k$. Since $\mathbb{Q}$ has no minimum, there exists an $r \in \mathbb{Q}$ such that $r < \delta(d_i)$ for all $i < k$. Set $\delta(d_k) := r$.
   (c) Neither of the previous two cases holds. Since $\mathbb{Q}$ is dense, there exists

an $r \in \mathbb{Q}$ such that

$\max\{\delta(d_i) \mid i < k \text{ and } d_i \prec d_k\} < r < \min\{\delta(d_i) \mid i < k \text{ and } d_k \prec d_i\}$.

Set $\delta(d_k) := r$.

It is readily checked that δ is as required. $\square$

Since the proof of **Lemma 15** is rather involved, we treat the "if" and the "only-if" direction in two separate lemmas. We again use the notions of paths and =-paths in constraint graphs. Moreover, we use the following notation: if $v_0, \ldots, v_k$ is a path or a cycle, we use i_O^+ to denote $(i+1) \bmod k + 1$, i.e., i_O^+ denotes the index following i in the path/cycle O. The index $\cdot_O$ is omitted if clear from the context.

Lemma 28 *A concept C is satisfiable w.r.t. a general TBox $\mathcal{T}$ if there exists a Hintikka-tree for C and $\mathcal{T}$.*

Proof. Let C be a concept, $\mathcal{T}$ a TBox, k the number of existential subconcepts in $\mathsf{sub}(C, \mathcal{T})$, and ℓ the number of abstract features in $\mathsf{sub}(C, \mathcal{T})$. Moreover, let T be a Hintikka-tree for C and $\mathcal{T}$. We define an interpretation $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ as follows:

$$
\begin{aligned}
\Delta_\mathcal{I} &= \{1, \ldots, k+\ell\}^* \\
A^\mathcal{I} &= \{\alpha \mid A \in T_\lhd(\alpha)\} \text{ for all } A \in C_N \\
R^\mathcal{I} &= \{(\alpha, \beta) \mid \beta = \alpha i,\ \mathcal{E}_i(C, \mathcal{T}) = \exists R.E \text{ for some concept } E, \text{ and} \\
&\qquad\qquad \exists R.E \in T_\lhd(\alpha)\} \text{ for all } R \in \mathsf{N_{rR}} \\
f^\mathcal{I} &= \{(\alpha, \beta) \mid \beta = \alpha i,\ \mathcal{F}_{i-k}(C, \mathcal{T}) = f, \text{ and } f \text{ is enforced by } T(\alpha)\} \\
&\quad \text{for all } f \in \mathsf{N_{aF}}
\end{aligned}
$$

It remains to define the interpretation of temporal features, which is done as follows: we define an (infinite) constraint graph $G(T)$ induced by T, show that $G(T)$ is satisfiable, and define the interpretation of temporal 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 $(V, \mathsf{cl}_=(E))$, where

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

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.

By Theorem 8, it suffices to show that $G(T)$ contains no $<$-cycle. Assume to the contrary that $G(T)$ contains a $<$-cycle and that $O = \alpha_0|u_0, \ldots, \alpha_n|u_n$ is the $<$-cycle in $G(T)$ with minimal length. Fix a $t \leq n$ such that

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

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 $s \leq n$ such that we have $(\alpha_s|u_s, <, \alpha_{s^+}|u_{s^+}) \in E$. 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 $O' = \alpha'_0|u'_0, \ldots, \alpha'_m|u'_m$. By definition of $G(T)$, the fact that $(\alpha_s|u_s, <, \alpha_{s^+}|u_{s^+}) \in E$ implies $\alpha_{s^+} = \alpha_s$. Since $\alpha_s \neq \alpha_t$, $\alpha_s|u_s$ and $\alpha_{s^+}|u_{s^+}$ are in O' and it remains to show that O' is a cycle in $G(T)$, i.e., for all $i \leq m$, we have $(\alpha'_i|u'_i, P, \alpha'_{i^+}|u'_{i^+}) \in E$ for some $P \in \{<, =\}$.

  Let $\alpha'_i|u'_i$ and $\alpha'_{i^+}|u'_{i^+}$ 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

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

  in $G(T)$, which is at most as long as O. Since $\alpha'_i \neq \alpha_t$ and $\alpha'_{i^+} \neq \alpha_t$, by construction of $G(T)$ and by $(*)$, this implies that there exist $\beta \in \{1, \ldots, k+\ell\}^*$, $f \in N_{aF}$, $j \in \{1, \ldots, k+\ell\}$, and $g, g' \in \mathcal{G}(C, \mathcal{T})$ such that the following conditions are satisfied:

(1) $\alpha'_i = \alpha'_{i^+} = \beta$,
(2) $\alpha_t = \beta j$ and $\mathcal{F}_{j-k}(C, \mathcal{T}) = f$,
(3) $u'_i = fg$, $u^*_1 = g$, $u^*_x = g'$, and $u'_{i^+} = fg'$, 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) $(fg, P, fg') \in E'$ for some $P \in \{<, =\}$.

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

  Let $\mathsf{cpl}(T, \beta j) = (V'', E'')$. 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 $(\alpha_t|u^*_x, <, \alpha_t|u^*_1) \in E$. Hence, the path $\alpha_t|u^*_1, \ldots, \alpha_t|u^*_x$ is a $<$-cycle in $G(T)$,

which contradicts the minimality of O.

- $\alpha_s = \alpha_t$. We first show that there exists a node $\alpha_z|u_z$ in O such that $\alpha_z \neq \alpha_t$. For suppose that no such node exists. Then, by definition of $G(T)$, $u_0, \ldots, u_n$ is a $<$-cycle in $\mathsf{cpl}(T, \alpha_t)$. This is clearly a contradiction to the fact that T is a Hintikka-tree. Hence, we may conclude the existence of an α_z as above. 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 $\alpha_s = \alpha_t$. Moreover, O' is shorter than O due to the existence of α_z. We show that O' is a $<$-cycle in $G(T)$, which is a contradiction to the minimality of O. Let $O' = \alpha_t|u'_0, \ldots, \alpha_t|u'_m$. By definition of $G(T)$, the fact that $(\alpha_s|u_s, <, \alpha_{s^+}|u_{s^+}) \in E$ implies $\alpha_{s^+} = \alpha_s = \alpha_t$. Hence, it remains to show that O' is a cycle in $G(T)$, i.e., that, for all $i \leq m$, we have $(\alpha_t|u'_i, P, \alpha_t|u'_{i^+}) \in E$ for some $P \in \{<, =\}$.

  Let $\alpha_t|u'_i$ and $\alpha_t|u'_{i^+}$ 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

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

  in $G(T)$, which is at most as long as O, such that $\alpha^*_i \neq \alpha_t$ for all i with $1 \leq i \leq x$. By construction of $G(T)$ and by $(*)$, this implies that there exist $\beta \in \{1, \ldots, k+\ell\}^*$, $f \in \mathsf{N_{aF}}$, $j \in \{1, \ldots, k+\ell\}$, and $g, g' \in \mathcal{G}(C, \mathcal{T})$ such that the following conditions are satisfied:

(1) $\alpha^*_1 = \alpha^*_x = \beta$,
(2) $\alpha_t = \beta j$ and $\mathcal{F}_{j-k}(C, \mathcal{T}) = f$,
(3) $u'_i = g$, $u^*_1 = fg$, $u^*_x = fg'$, and $u'_{i^+} = g'$, 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) $(fg, P, fg') \in E'$ for some $P \in \{<, =\}$.

  Case (a) is impossible, which can be seen as follows: 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.

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

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

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

To show that there exists a $d \in \Delta_{\mathcal{I}}$ such that $d \in C^{\mathcal{I}}$, 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 $D \in \mathsf{sub}(C, \mathcal{T})$.

Proof: The claim is proved by induction on the structure of D. First for the induction start, which splits into several subcases:

- D is a concept name. Immediate by definition of $\mathcal{I}$.
- $D = \neg E$. Since C is in NNF, D is also 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 $\alpha \in (\neg E)^{\mathcal{I}}$.
- $D = \exists(u_1\, P\, u_2)$. Let $G(T) = (V, E)$ and $\mathsf{cpl}(T, \alpha) = (V', E')$. By definition of pair-graphs and $\mathsf{cpl}()$, we have $(u_1, P, u_2) \in E'$. Moreover, by definition of $G(T)$, we have $(\alpha|u_1, P, \alpha|u_2) \in E$. It thus remains to show that $u_1^{\mathcal{I}}(\alpha) = \delta(\alpha|u_1)$, and $u_2^{\mathcal{I}}(\alpha) = \delta(\alpha|u_2)$: since δ is a solution for $G(T)$, this clearly implies $u_1^{\mathcal{I}}(\alpha) P u_2^{\mathcal{I}}(\alpha)$.

  First, assume $u_i = g$ for some $g \in \mathsf{N_{tF}}$. By definition of $g^{\mathcal{I}}$ and since g is enforced by $T(\alpha)$, we have $u_i^{\mathcal{I}}(\alpha) = \delta(\alpha|u_i)$ as required. Now let $u_i = fg$ with $\mathcal{F}_{j-k}(C, \mathcal{T}) = f$. Since fg is a node in $\mathsf{cpl}(T, \alpha)$, we have $(\alpha|fg, =, \alpha j|g) \in E$. Hence, $\delta(\alpha j|g) = \delta(\alpha|fg)$. By definition of $f^{\mathcal{I}}$ and since f is obviously enforced by $T(\alpha)$, we have $f^{\mathcal{I}}(\alpha) = \alpha j$. By definition of cpl and of pair-graphs, $fg \in V'$ implies that g appears in $T_{\triangleright}(\alpha j)$: 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 $\mathcal{F}_{j-k}(C, \mathcal{T}) = f$, we must have $(g, =, g) \in T_{\triangleright}(\alpha j)$, i.e., g appears in $T_{\triangleright}(\alpha j)$. Since g appears in $T_{\triangleright}(\alpha j)$ and is thus enforced by $T(\alpha j)$, we have $g^{\mathcal{I}}(\alpha j) = \delta(\alpha j|g)$ by definition of $g^{\mathcal{I}}$. Summing up, we obtain $(fg)^{\mathcal{I}}(\alpha) = \delta(\alpha j|g) = \delta(\alpha|fg)$.
- $D = g{\uparrow}$. If $g^{\mathcal{I}}(\alpha)$ is defined, then g is enforced by $T(\alpha)$. We show that this implies $g{\uparrow} \notin T_{\triangleleft}(\alpha)$. If g is enforced by $T(\alpha)$, then either (i) g appears in $T_{\triangleright}(\alpha)$ or (ii) $\{\exists(g\, P\, u'), \exists(u'\, P\, g)\} \cap T_{\triangleleft}(\alpha) \neq \emptyset$ for some path u' and $P \in \{<, =\}$. In case (i), **(H6)** yields $g{\uparrow} \notin T_{\triangleleft}(\alpha)$. In case (ii), **(H5)** yields the same result.

For the induction step, we make a case distinction according to the topmost operator in D. Assume $D \in T_{\triangleleft}(\alpha)$.

- $D = C_1 \sqcap C_2$ or $D = C_1 \sqcup C_2$. Straightforward by **(H2)** and **(H3)** of Hintikka-sets and by induction hypothesis.
- $D = \exists R.E$ with $R \in \mathsf{N_{rR}}$. By definition of $R^{\mathcal{I}}$, we have $(\alpha, \beta) \in R^{\mathcal{I}}$ for $\beta = \alpha i$ and $\mathcal{E}_i(C, \mathcal{T}) = \exists R.E$. By **(H7)**, we have $E \in T_{\triangleleft}(\beta)$, and, by induction, $\beta \in E^{\mathcal{I}}$.
- $D = \exists f.E$ with $f \in \mathsf{N_{aF}}$. Hence, f is enforced by $T(\alpha)$. By definition of

$f^{\mathcal{I}}$, we have $f^{\mathcal{I}}(\alpha) = \beta$ for $\beta = \alpha i$ and $\mathcal{F}_{i-k}(C, \mathcal{T}) = f$. By **(H9)**, we have $E \in T_{\triangleleft}(\beta)$, and, by induction, $\beta \in E^{\mathcal{I}}$.

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

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

Lemma 29 *A concept C is satisfiable w.r.t. a general TBox $\mathcal{T}$ only if there exists a Hintikka-tree for C and $\mathcal{T}$.*

Proof. Let C be a concept, $\mathcal{T}$ a TBox, and k and ℓ as in the proof of Lemma 28. Moreover, let $\mathcal{I}$ be a model of C w.r.t. $\mathcal{T}$, i.e., there exists a $d_0 \in \Delta_{\mathcal{I}}$ such that $d_0 \in C^{\mathcal{I}}$ and $D^{\mathcal{I}} = E^{\mathcal{I}}$ for all $D \doteq E \in \mathcal{T}$. We inductively define a Hintikka-tree T for C and $\mathcal{T}$, 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

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

For the induction start, set

$$\tau(\epsilon) := d_0, \quad T_{\triangleleft}(\epsilon) := \{D \in \mathsf{sub}(C, \mathcal{T}) \mid d_0 \in D^{\mathcal{I}}\}, \quad \text{and} \quad T_{\triangleright}(\epsilon) := \emptyset.$$

Obviously, $(*)$ is satisfied. Now for the induction step. Let $\alpha \in \{1, \ldots, k + \ell\}^*$ be a word of minimal length such that $\tau(\alpha)$ is defined and $\tau(\alpha i)$ is undefined for some $i \in \{1, \ldots, k + \ell\}$. We make a case distinction as follows:

(1) $\mathcal{E}_i(C, \mathcal{T}) = \exists R.D \in T_{\triangleleft}(\alpha)$. By $(*)$, we have $\tau(\alpha) \in (\exists R.D)^{\mathcal{I}}$. Thus, there exists some $e \in \Delta_{\mathcal{I}}$ such that $(\tau(\alpha), e) \in R^{\mathcal{I}}$ and $e \in D^{\mathcal{I}}$. Set $\tau(\alpha i) := e$, $T_{\triangleleft}(\alpha i) := \{E \in \mathsf{sub}(C, \mathcal{T}) \mid e \in E^{\mathcal{I}}\}$, and $T_{\triangleright}(\alpha i) := \emptyset$.

(2) $\mathcal{F}_{i-k}(C, \mathcal{T}) = f$, and f is enforced by $T(\alpha)$. By $(*)$ and the definition of "enforced", there exists an $e \in \Delta_{\mathcal{I}}$ such that $f^{\mathcal{I}}(\tau(\alpha)) = e$. Set

$$\begin{aligned}
\tau(\alpha i) &:= e \\
T_{\triangleleft}(\alpha i) &:= \{E \in \mathsf{sub}(C, \mathcal{T}) \mid e \in E^{\mathcal{I}}\} \\
T_{\triangleright}(\alpha i) &:= \{(g_1, P, g_2) \mid fg_1,\ fg_2 \text{ are enforced by } T(\alpha) \text{ and } g_1^{\mathcal{I}}(e) P g_2^{\mathcal{I}}(e)\}
\end{aligned}$$

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

Clearly, $(*)$ is satisfied after each induction step, and hence T is well-defined. Intuitively, Case 3 applies if the i-th successor of α is not needed to satisfy the Properties of Hintikka-trees. In this case, the choice of $\tau(\alpha i)$ is arbitrary: we could have defined $\tau(\alpha i)$ as any element of $\Delta_{\mathcal{I}}$ (instead of choosing $\tau(\epsilon)$).

We must show that T is a Hintikka-tree for C and $\mathcal{T}$. From $(*)$ together with the semantics of concepts and TBoxes, it is clear that $T_{\triangleleft}(\alpha)$ is a Hintikka-set for C and $\mathcal{T}$ 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 $C_{\mathcal{T}} \notin T_{\triangleleft}(\alpha)$. By $(*)$, we have $\tau(\alpha) \notin C_{\mathcal{T}}^{\mathcal{I}}$. By definition of $C_{\mathcal{T}}$, this implies the existence of $D \doteq E \in \mathcal{T}$ such that $\tau(\alpha) \notin (D \leftrightarrow E)^{\mathcal{I}}$, i.e., $\tau(\alpha) \in D^{\mathcal{I}} \setminus E^{\mathcal{I}}$ or $\tau(\alpha) \in E^{\mathcal{I}} \setminus D^{\mathcal{I}}$. Hence, we do not have $D^{\mathcal{I}} = E^{\mathcal{I}}$ and obtain a contradiction to the fact that $\mathcal{I}$ is a model of $\mathcal{T}$.

Now we 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 $(g_1, P, g_2) \in T_{\triangleright}(\alpha)$ and $g_j\!\uparrow \in T_{\triangleleft}(\alpha)$ for $j \in \{1, 2\}$. By definition of $T_{\triangleright}$, $(g_1, P, g_2) \in T_{\triangleright}(\alpha)$ implies that $g_j^{\mathcal{I}}(\tau(\alpha))$ is defined. But from $g_j\!\uparrow \in T_{\triangleleft}(\alpha)$ and $(*)$, we obtain that $g_j^{\mathcal{I}}(\tau(\alpha))$ is undefined: contradiction.

It remains to show that T satisfies **(H12)** and **(H13)**. The former is simple due to the definition of T (induction start) and the fact that $d_0 \in C^{\mathcal{I}}$. The latter amounts to showing that, for each $\alpha \in \{1, \ldots, k+\ell\}^*$, the tuple $(T(\alpha), T(\alpha 1), \ldots, T(\alpha j))$ satisfies **(H7)** to **(H11)**, where j abbreviates $k+\ell$.

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

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

(H9) Let $\exists f.D \in T_{\triangleleft}(\alpha)$ and $\mathcal{F}_i(C, \mathcal{T}) = f$. Hence, f is enforced by $T(\alpha)$. By definition of τ (Case 2), we have $\tau(\alpha j) = e$ for $e = f^{\mathcal{I}}(\tau(\alpha))$ and $j = k + i$. From $\exists f.D \in T_{\triangleleft}(\alpha)$ and $(*)$, we obtain $\tau(\alpha) \in (\exists f.D)^{\mathcal{I}}$ and thus $e \in D^{\mathcal{I}}$. Again by $(*)$, we get $D \in T_{\triangleleft}(\alpha j)$.

(H10) Let f be enforced by $T(\alpha)$, $\mathcal{F}_i(C, \mathcal{T}) = f$, and $\forall f.D \in T_{\triangleleft}(\alpha)$. By definition of τ (Case 2), we have $\tau(\alpha j) = e$ for $e = f^{\mathcal{I}}(\tau(\alpha))$ and $j = k + i$. From $\forall f.D \in T_{\triangleleft}(\alpha)$ and $(*)$, we obtain $\tau(\alpha) \in (\forall f.D)^{\mathcal{I}}$ and thus $e \in D^{\mathcal{I}}$. Again by $(*)$, we get $D \in T_{\triangleleft}(\alpha j)$.

(H11) Let $G(T(\alpha)) = (V, E)$ and E' be defined as in **(H11)**. To prove that **(H11)** is satisfied, we show that

(1) E' is an edge extension of $G(T(\alpha))$, which implies that $(V, E \cup E')$ 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 $fg_1, fg_2 \in V$, $\{(fg_1, <, fg_2), (fg_1, =, fg_2), (fg_2, <, fg_1)\} \cap E' \neq \emptyset$. By definition of $G(T(\alpha))$, fg_1 and fg_2 are enforced by $T(\alpha)$. Since $T_\triangleright(\alpha)$ may only contain paths of length 1, we have $\{\exists(fg_1\, P'\, u), \exists(u\, P'\, fg_1)\} \cap T_\triangleleft(\alpha) \neq \emptyset$ 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 $\mathcal{F}_{i-k}(C, \mathcal{T}) = f$. Hence, $g_1^{\mathcal{I}}(\tau(\alpha i))$ and $g_2^{\mathcal{I}}(\tau(\alpha i))$ are defined. By the semantics, we have $g_1^{\mathcal{I}}(\tau(\alpha i)) < g_2^{\mathcal{I}}(\tau(\alpha i))$, $g_1^{\mathcal{I}}(\tau(\alpha i)) = g_2^{\mathcal{I}}(\tau(\alpha i))$, or $g_2^{\mathcal{I}}(\tau(\alpha i)) < g_1^{\mathcal{I}}(\tau(\alpha i))$. By definition of $T_\triangleright$, this yields $\{(g_1, <, g_2), (g_1, =, g_2), (g_2, <, g_1)\} \cap T_\triangleright(\alpha i) \neq \emptyset$. Hence, by definition of E' we have $\{(fg_1, <, fg_2), (fg_1, =, fg_2), (fg_2, <, fg_1)\} \cap E' \neq \emptyset$.

We now prove Point 2 from above. Define a mapping δ from V to $\mathbb{Q}$ as follows: $\delta(u) := u^{\mathcal{I}}(\tau(\alpha))$. 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 $T_\triangleright(\alpha)$ or
(ii) $\{\exists(u\, P\, u'), \exists(u'\, P\, u)\} \cap T_\triangleleft(\alpha) \neq \emptyset$ for some path u' and $P \in \{<, =\}$.
In Case (i), we have $u = g$ for some $g \in \mathsf{N_{tF}}$. By definition of T, there exists a predecessor β of α in T such that $\alpha = \beta i$, $\mathcal{F}_{i-k}(C, \mathcal{T}) = f$ for some $f \in \mathsf{N_{aF}}$, and fg is enforced by $T(\beta)$. Since $T_\triangleright(\beta)$ contains only paths of length 1, we have $\{\exists(fg\, P\, u), \exists(u\, P\, fg)\} \cap T_\triangleleft(\beta) \neq \emptyset$ 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), it follows from $(*)$ that $u^{\mathcal{I}}(\tau(\alpha))$ is defined.

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

(1) $(u_1, P, u_2) \in E \cap T_\triangleright(\alpha)$. Then there exist $g_1, g_2 \in \mathsf{N_{tF}}$ such that $u_1 = g_1$ and $u_2 = g_2$. By definition of $T_\triangleright$, we have $g_1^{\mathcal{I}}(\tau(\alpha)) P g_2^{\mathcal{I}}(\tau(\alpha))$, and thus, by definition of δ, $\delta(g_1) P \delta(g_2)$.

(2) $(u_1, P, u_2) \in E$ and $\exists(u_1\, P\, u_2) \in T_\triangleleft(\alpha)$. Then $(*)$ implies that $\tau(\alpha) \in \exists(u_1\, P\, u_2)^{\mathcal{I}}$. Hence, $u_1^{\mathcal{I}}(\tau(\alpha)) P u_2^{\mathcal{I}}(\tau(\alpha))$. By definition of δ, we thus obtain $\delta(u_1) P \delta(u_2)$.

(3) $(u_1, P, u_2) \in E'$. By definition of E', we have $u_1 = fg_1$, $u_2 = fg_2$, and $(g_1, P, g_2) \in T_\triangleright(\alpha i)$ where $\mathcal{F}_{k-i}(C, \mathcal{T}) = f$. By definition of $T_\triangleright$, this yields that fg_1 and fg_2 are enforced by $T(\alpha)$ and $g_1^{\mathcal{I}}(\tau(\alpha i)) P g_2^{\mathcal{I}}(\tau(\alpha i))$. From this and the definition of T (Case 2), it follows that $f^{\mathcal{I}}(\tau(\alpha)) = \tau(\alpha i)$. We conclude $\delta(u_1) P \delta(u_2)$.

To sum up, we have shown that **(H13)** holds. □

B Proofs for Section 5

Lemma 23 Let $\mathcal{A}$ be an ABox and $\mathcal{T}$ a TBox. Then $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff there exists a precompletion $\mathcal{A}'$ of $\mathcal{A}$ w.r.t. $\mathcal{T}$ such that $\mathcal{A}'$ is consistent w.r.t. $\mathcal{T}$.

Proof. Recall that $\mathcal{A}'$ is a precompletion of $\mathcal{A}$ w.r.t. $\mathcal{T}$ if $\mathcal{A}'$ can be obtained from $\mathcal{A}$ by exhaustive rule application (using the TBox $\mathcal{T}$). By Lemma 21, exhaustive rule application always terminates. Hence, we only need to show that, if a precompletion rule R is applicable to an ABox $\mathcal{A}$, then $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff one of the outcomes $\mathcal{A}'$ of applying R to $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$.

We make a case distinction according to the rule R. For all rules except Rfe, the "if" direction is trivial since $\mathcal{A} \subseteq \mathcal{A}'$ if $\mathcal{A}'$ is obtained from $\mathcal{A}$ by rule application. Hence, every model of $\mathcal{A}'$ and $\mathcal{T}$ is clearly also a model of $\mathcal{A}$ and $\mathcal{T}$. Thus, we concentrate on the "only if" direction in all cases except Rfe. Assume that $\mathcal{I}$ is a model of $\mathcal{A}$ and $\mathcal{T}$.

- $\mathsf{R} = \mathsf{R}\sqcap$. Assume that the $\mathsf{R}\sqcap$ rule is applied to a concept $C \sqcap D \in \mathcal{A}(a)$. Since $\mathcal{I}$ is a model of $\mathcal{A}$, we have $a^\mathcal{I} \in (C \sqcap D)^\mathcal{I}$ and thus $a^\mathcal{I} \in C^\mathcal{I}$ and $a^\mathcal{I} \in D^\mathcal{I}$. Rule application adds $a : C$ and $a : D$ to $\mathcal{A}$. Thus, $\mathcal{I}$ is a model of the resulting ABox $\mathcal{A}'$.
- $\mathsf{R} = \mathsf{R}\sqcup$. Assume that the $\mathsf{R}\sqcup$ rule is applied to a concept $C \sqcup D \in \mathcal{A}(a)$. Since $\mathcal{I}$ is a model of $\mathcal{A}$, we have $a^\mathcal{I} \in (C \sqcup D)^\mathcal{I}$ and thus $a^\mathcal{I} \in C^\mathcal{I}$ or $a^\mathcal{I} \in D^\mathcal{I}$. The rule application yields ABoxes $\mathcal{A}_1 = \mathcal{A} \cup \{a : C\}$ and $\mathcal{A}_2 = \mathcal{A} \cup \{a : D\}$. Thus, $\mathcal{I}$ is a model of $\mathcal{A}_1$ or $\mathcal{A}_2$.
- $\mathsf{R} = \mathsf{R}\exists\mathsf{f}$. Assume that the rule is applied to a concept $\exists f.C \in \mathcal{A}(a)$ adding $b : C$ to $\mathcal{A}$ for some b with $\langle a, b\rangle : f \in \mathcal{A}$. Since $\mathcal{I}$ is a model of $\mathcal{A}$, we have $a^\mathcal{I} \in (\exists f.C)^\mathcal{I}$ and $f^\mathcal{I}(a^\mathcal{I}) = b^\mathcal{I}$. Due to the functionality of $f^\mathcal{I}$, this yields $b^\mathcal{I} \in C^\mathcal{I}$. Thus, $\mathcal{I}$ is a model of the resulting ABox $\mathcal{A}'$.
- $\mathsf{R} = \mathsf{R}\forall$. Similar to the previous case.
- $\mathsf{R} = \mathsf{Rc1}$. Analogous to the next case, only simpler.
- $\mathsf{R} = \mathsf{Rc2}$. Assume that the $\mathsf{Rc2}$ rule is applied to a concept $\exists(f\, g_1\, P\, g) \in \mathcal{A}(a)$ and a node b with $\langle a, b\rangle : f \in \mathcal{A}$. Since $\mathcal{I}$ is a model of $\mathcal{A}$, we have $a^\mathcal{I} \in \exists(f\, g_1\, P\, g)^\mathcal{I}$, $f^\mathcal{I}(a^\mathcal{I}) = b^\mathcal{I}$, and $g_1^\mathcal{I}(b^\mathcal{I})\, P\, g_2^\mathcal{I}(a^\mathcal{I})$. Rule application introduces two new time point names x_1 and x_2 and adds $\{\langle b, x_1\rangle : g_1,\ \langle a, x_2\rangle : g_2,\ x_1\, P\, x_2\}$ to $\mathcal{A}$. Let $\mathcal{I}'$ be obtained from $\mathcal{I}$ by setting $x_1^\mathcal{I} = g_1^\mathcal{I}(b^\mathcal{I})$ and $x_2^\mathcal{I} = g_2^\mathcal{I}(a^\mathcal{I})$. Clearly, $\mathcal{I}'$ is a model of the resulting ABox $\mathcal{A}'$.
- $\mathsf{R} = \mathsf{Rc3}$. Analogous to the previous case.
- $\mathsf{R} = \mathsf{Rch}$. Assume that the Rch rule is applied to the assertions $\langle a, x_1\rangle : g_1$ and $\langle a, x_2\rangle : g_2$. Since $\mathcal{I}$ is a model of $\mathcal{A}$, there exist $q_1, q_2 \in \mathbb{Q}$ such that $x_1^\mathcal{I} = q_1$ and $x_2^\mathcal{I} = q_2$. Trivially, we have either $q_1 < q_2$, $q_1 = q_2$, or $q_2 < q_1$. We obtain three new ABoxes by adding one of the assertions $a : \exists(g_1 < g_2)$, $a : \exists(g_1 = g_2)$, and $a : \exists(g_2 < g_1)$. Thus, $\mathcal{I}$ is a model of $\mathcal{A}_i$ for some $i \in \{1, 2, 3\}$.

- $\mathsf{R} = \mathsf{R}\doteq$. The rule application adds $a : C_{\mathcal{T}}$ for some $a \in \mathsf{O_a}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$, we have $d \in C^{\mathcal{I}}_{\mathcal{T}}$ for every $d \in \Delta_{\mathcal{I}}$. Thus, $\mathcal{I}$ is also a model of the resulting ABox $\mathcal{A}'$.
- $\mathsf{R} = \mathsf{Rfe}$. Assume that the rule is applied to assertions $\langle a, b\rangle : f$ and $\langle a, c\rangle : f$. Since $\mathcal{I}$ is a model of $\mathcal{A}$, we have $f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}} = c^{\mathcal{I}}$. Thus, $\mathcal{I}$ is also a model of the ABox $\mathcal{A}'$ obtained from $\mathcal{A}$ by replacing the object name b with c.

  For this rule, we also treat the "if" direction. Hence assume that the rule has applied to two assertions $\langle a, b\rangle : f$ and $\langle a, c\rangle : f$ in $\mathcal{A}$, and let $\mathcal{I}'$ be a model of the resulting ABox $\mathcal{A}'$ that has been obtained by replacing b with c. It is easily seen that we can construct a model $\mathcal{I}''$ of $\mathcal{A}$ by setting $b^{\mathcal{I}} = c^{\mathcal{I}}$.

  The replacement of time point names rather than object names can be treated analogously. □

Lemma 24 Let $\mathcal{A}$ be a precomplete ABox, δ a solution for $G(\mathcal{A})$, and $a \in \mathsf{O_a}$ used in $\mathcal{A}$. If $\mathsf{con}(\mathcal{A}, a)$ is satisfiable w.r.t. $\mathcal{T}$, then there exists a model $\mathcal{I}$ of $\mathsf{con}(\mathcal{A}, a)$ and $\mathcal{T}$ and a $d_a \in \mathsf{con}(\mathcal{A}, a)^{\mathcal{I}}$ such that, for all $\langle a, x\rangle : g \in \mathcal{A}$, we have $g^{\mathcal{I}}(d_a) = \delta(x)$.

Proof. Let $\mathcal{A}$, $G(\mathcal{A})$, and δ be as in the lemma, and let $\mathcal{I}$ be a model of $\mathsf{con}(\mathcal{A}, a)$ and $\mathcal{T}$. Moreover, let d_a be an arbitrary element of $\mathsf{con}(\mathcal{A}, a)^{\mathcal{I}}$. We show that $\mathcal{I}$ can be transformed into a model $\mathcal{J}$ such that $\mathcal{J}$ and d_a are as required.

In the following, we assume that there exists a well-founded linear ordering on the set $\Delta_{\mathcal{I}} \times \mathsf{N_{tF}}$. This can be done w.l.o.g. since it is a byproduct of the proof of Lemma 28 that, if a concept C is satisfiable w.r.t. a TBox $\mathcal{T}$, then there exist a model of C and $\mathcal{T}$ (the one constructed in the proof) for which such an ordering exists. We construct the model $\mathcal{J}$ from $\mathcal{I}$ by modifying the interpretations of temporal features in an appropriate way. To do this, we successively "mark" pairs in $\Delta_{\mathcal{I}} \times \mathsf{N_{tF}}$ such that a pair (d, g) is marked iff $g^{\mathcal{J}}(d)$ has already been determined. During the construction of $\mathcal{J}$, the following invariant will always hold:

$$\begin{array}{l}\text{if } (d_1, g_1), (d_2, g_2) \in \Delta_{\mathcal{I}} \times \mathsf{N_{tF}} \text{ are marked, then} \\ \quad g_1^{\mathcal{I}}(d_1)\, P\, g_2^{\mathcal{I}}(d_2) \text{ with } P \in \{<, =, >\} \text{ implies } g_1^{\mathcal{J}}(d_1)\, P\, g_2^{\mathcal{J}}(d_2)\end{array} \qquad (*)$$

Initially, each pair in $\Delta_{\mathcal{I}} \times \mathsf{N_{tF}}$ is unmarked. The construction of $\mathcal{J}$ consists of an initial step and an inductive step.

(1) Initial step. For all $\langle a, x\rangle : g \in \mathcal{A}$, set $g^{\mathcal{J}}(d_a) := \delta(x)$ and mark (d_a, g).

We need to show that $(*)$ is satisfied. Hence, fix two marked pairs (d_a, g_1) and (d_a, g_2) from $\Delta_{\mathcal{I}} \times \mathsf{N_{tF}}$. Then we have $\{\langle a, x_1\rangle : g_1, \langle a, x_2\rangle : g_2\} \subseteq \mathcal{A}$ for some $x_1, x_2 \in \mathsf{O_t}$. Since neither the Rch nor the $\mathsf{Rc1}$ rule is applicable, we have either (i) $\exists(g_1 < g_2) \in \mathcal{A}(a)$ and $x_1 < x_2 \in \mathcal{A}$, (ii) $\exists(g_1 = g_2) \in \mathcal{A}(a)$

and $x_1 = x_2 \in \mathcal{A}$, or (iii) $\exists(g_2 < g_1) \in \mathcal{A}(a)$ and $x_2 < x_1 \in \mathcal{A}$. We only treat case (i) exemplarily. By definition of $\mathsf{con}(\mathcal{A}, a)$ and since $d_a \in \mathsf{con}(\mathcal{A}, a)^{\mathcal{I}}$, we have $d_a \in \exists(g_1 < g_2)^{\mathcal{I}}$ and thus $g_1^{\mathcal{I}}(d_a) < g_2^{\mathcal{I}}(d_a)$. From $x_1 < x_2 \in \mathcal{A}$ and the definition of $G(\mathcal{A})$, it follows that $\delta(x_1) < \delta(x_2)$ and hence $g_1^{\mathcal{J}}(d_a) < g_2^{\mathcal{J}}(d_a)$ as required. Cases (ii) and (iii) are analogous.

(2) Inductive step. Choose the least unmarked pair (d, g) from $\Delta_{\mathcal{I}} \times \mathsf{N_{tF}}$ (w.r.t. the assumed ordering) for which $g^{\mathcal{I}}(d)$ is defined. For $P \in \{<, =, >\}$, let Ψ_P be the set of marked pairs $(d_1, g_1) \in \Delta_{\mathcal{I}} \times \mathsf{N_{tF}}$ for which $g_1^{\mathcal{I}}(d_1)\, P\, g^{\mathcal{I}}(d)$. By $(*)$, we have
- $g_1^{\mathcal{J}}(d_1) = g_2^{\mathcal{J}}(d_2)$ for all $(d_1, g_1), (d_2, g_2) \in \Psi_=$,
- $g_1^{\mathcal{J}}(d_1) < g_2^{\mathcal{J}}(d_2)$ for all $(d_1, g_1) \in \Psi_<$ and $(d_2, g_2) \in \Psi_= \cup \Psi_>$, and
- $g_1^{\mathcal{J}}(d_1) < g_2^{\mathcal{J}}(d_2)$ for all $(d_1, g_1) \in \Psi_=$ and $(d_2, g_2) \in \Psi_>$.

Hence, due to the density of $\mathbb{Q}$ there is a $q \in \mathbb{Q}$ such that
- $q > \max\{g_1^{\mathcal{J}}(d_1) \mid (d_1, g_1) \in \Psi_<\}$,
- $q = g_1^{\mathcal{J}}(d_1)$ for all $(d_1, g_1) \in \Psi_=$, and
- $q < \min\{g_1^{\mathcal{J}}(d_1) \mid (d_1, g_1) \in \Psi_>\}$.

Set $g^{\mathcal{J}}(d) := q$. Obviously, $(*)$ is satisfied.

It is straightforward to show by structural induction that $d \in C^{\mathcal{I}}$ iff $d \in C^{\mathcal{J}}$ for all $d \in \Delta_{\mathcal{I}}$ and all $\mathcal{TDL}$-concepts C. Hence, $\mathcal{J}$ is a model of $\mathsf{con}(\mathcal{A}, a)$ and $\mathcal{T}$. By the initial step of its construction, $\mathcal{J}$ is as required. □

Lemma 25 Let $\mathcal{A}$ be a precompletion of an ABox $\mathcal{A}'$ w.r.t. a TBox $\mathcal{T}$. $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}$ is clash-free and $\mathsf{con}(\mathcal{A}, a)$ is satisfiable w.r.t. $\mathcal{T}$ for every $a \in \mathsf{O_a}$ used in $\mathcal{A}$.

Proof. Since the "only if" direction is straightforward, we concentrate on the "if" direction. Let $\mathfrak{A}$ denote the set of object names $a \in \mathsf{O_a}$ appearing in $\mathcal{A}$. Since $\mathcal{A}$ is clash-free, there exists a solution δ for $G(\mathcal{A})$. For every $a \in \mathfrak{A}$, fix a model $\mathcal{I}_a$ of $\mathsf{con}(\mathcal{A}, a)$ and $\mathcal{T}$ and a domain element $d_a \in \Delta_{\mathcal{I}_a}$ such that $d_a \in \mathsf{con}(\mathcal{A}, a)^{\mathcal{I}_a}$. By Lemma 24, we may assume w.l.o.g. that, for all $a \in \mathfrak{A}$,

$$\langle a, x\rangle : g \in \mathcal{A} \text{ implies } g^{\mathcal{I}_a}(d_a) = \delta(x). \qquad (*)$$

Moreover, we assume that (i) $a \neq b$ implies $\Delta_{\mathcal{I}_a} \cap \Delta_{\mathcal{I}_b} = \emptyset$ and (ii) none of the d_a has incoming edges, i.e., $(d, d_a) \notin R^{\mathcal{I}_a}$ for all $d \in \Delta_{\mathcal{I}_a}$ and $R \in \mathsf{N_R}$. It is straightforward to prove that none of these assumptions restricts generality: for example, take for each $a \in \mathfrak{A}$ the canonical model constructed from a Hintikka-tree for $\mathsf{con}(\mathcal{A}, a)$ and $\mathcal{T}$ as in the proof of Lemma 28. Then apply the modification from the proof of Lemma 24 and finally make all domains $\mathcal{I}_a$ disjoint by renaming. Clearly, $(*)$, (i), and (ii) are satisfied for the resulting set of models. In the following, we define an interpretation $\mathcal{I}$ by taking the "union" of the models $\mathcal{I}_a$ with $a \in \mathfrak{A}$ and the relational structure defined by the ABox. However, we have to be careful not to obtain too many abstract feature successors and prefer successors from the ABox over successors from

the models.

(1) $\Delta_{\mathcal{I}} := \bigcup_{a\in\mathfrak{A}} \Delta_{\mathcal{I}_a}$,
(2) $A^{\mathcal{I}} := \bigcup_{a\in\mathfrak{A}} A^{\mathcal{I}_a}$ for all $A \in \mathsf{N_C}$,
(3) $R^{\mathcal{I}} := \{(d_a, d_b) \mid \langle a, b\rangle : R \in \mathcal{A}\} \cup \bigcup_{a\in\mathfrak{A}} R^{\mathcal{I}_a}$ for all $R \in \mathsf{N_{rR}}$,
(4) $f^{\mathcal{I}} := \{(d_a, d_b) \mid \langle a, b\rangle : f \in \mathcal{A}\} \cup \bigcup_{a\in\mathfrak{A}}\{(d, e) \in f^{\mathcal{I}_a} \mid d \neq d_a$ or $\langle a, b\rangle : f \notin \mathcal{A}$ for all $b \in \mathsf{O_a}\}$ for all $f \in \mathsf{N_{aF}}$,
(5) $g^{\mathcal{I}} := \bigcup_{a\in\mathfrak{A}} g^{\mathcal{I}_a}$ for all $g \in \mathsf{N_{tF}}$,
(6) $a^{\mathcal{I}} := d_a$ for all $a \in \mathfrak{A}$, and
(7) $x^{\mathcal{I}} := \delta(x)$ for all $x \in \mathsf{O_t}$ appearing in $\mathcal{A}$.

Note that, for all $f \in \mathsf{N_{aF}}$, $f^{\mathcal{I}}$ is functional since the Rfe rule is not applicable to $\mathcal{A}$. Since none of the d_a has incoming edges, the following claim can be proved straightforwardly by structural induction:

Claim 1: For all objects $a \in \mathcal{A}$, domain elements $d \in \Delta_{\mathcal{I}_a}$ with $d \neq d_a$, and $\mathcal{TDL}$-concepts C, we have $d \in C^{\mathcal{I}_a}$ iff $d \in C^{\mathcal{I}}$.

However, we still need to deal with the elements d_a themselves.

Claim 2: For all objects $a \in \mathcal{A}$, $C \in \mathcal{A}(a)$ implies $d_a \in C^{\mathcal{I}}$.

The proof is by induction on the structure of C. The induction start consists of three cases:

- $C \in \mathsf{N_C}$. Straightforward by definition of $\mathsf{con}(\mathcal{A}, a)$, the choice of $\mathcal{I}_a$ and d_a, and the construction of $\mathcal{I}$.
- $C = \exists(u_1\, P\, u_2)$. By definition of $\mathsf{con}(\mathcal{A}, a)$ and choice of $\mathcal{I}_a$ and d_a, $C \in \mathcal{A}(a)$ implies $d_a \in C^{\mathcal{I}_a}$. We make a case distinction according to the form of u_1 and u_2 (recall that all concepts are assumed to be in path normal form).

(1) $u_1 = g_1$ and $u_2 = g_2$. Since $d_a \in \exists(g_1\, P\, g_2)^{\mathcal{I}_a}$, there exist $q_1, q_2 \in \mathbb{Q}$ such that $g_1^{\mathcal{I}_a}(d_a) = q_1$, $g_2^{\mathcal{I}_a}(d_a) = q_2$, and $q_1 P q_2$. By definition of $\mathcal{I}$, this implies $g_1^{\mathcal{I}}(d_a) = q_1$, $g_2^{\mathcal{I}}(d_a) = q_2$ and thus $d_a \in \exists(g_1\, P\, g_2)^{\mathcal{I}}$.
(2) $u_1 = fg_1$ and $u_2 = g_2$. We have to distinguish two subcases. First assume that $\langle a, b\rangle : f \in \mathcal{A}$ for some $b \in \mathsf{O_a}$. Since the $\mathsf{Rc2}$ rule is not applicable, there exist $x_1, x_2 \in \mathsf{O_t}$ such that $\{\langle a, x_1\rangle : g_1, \langle b, x_2\rangle : g_2, x_1\, P\, x_2\} \in \mathcal{A}$. Since δ is a solution for $G(\mathcal{A})$, there are $q_1, q_2 \in \mathbb{Q}$ such that $q_1 = \delta(x_1)$, $q_2 = \delta(x_2)$, and $q_1 P q_2$. Since $\mathcal{I}_a$ and $\mathcal{I}_b$ satisfy $(*)$, we have $g_1^{\mathcal{I}_b}(d_b) = q_1$ and $g_2^{\mathcal{I}_a}(d_a) = q_2$. By construction of $\mathcal{I}$, we have $f^{\mathcal{I}}(d_a) = d_b$, $g_1^{\mathcal{I}}(d_a) = g_1^{\mathcal{I}_a}(d_a)$, and $g_2^{\mathcal{I}}(d_b) = g_2^{\mathcal{I}_b}(d_b)$. Hence, $g_1^{\mathcal{I}}(d_a) P g_2^{\mathcal{I}}(d_b)$ and $d_a \in \exists(fg_1\, P\, g_2)^{\mathcal{I}}$.
Now assume that there is no $b \in \mathsf{O_a}$ such that $\langle a, b\rangle : f \in \mathcal{A}$. From $d_a \in \exists(fg_1\, P\, g_2)^{\mathcal{I}_a}$ and the construction of $\mathcal{I}$, it follows straightforwardly (similar to Case 1) that $d_a \in \exists(fg_1\, P\, g_2)^{\mathcal{I}}$.
(3) $u_1 = g_1$ and $u_2 = fg_2$. Analogous to the previous case using $\mathsf{Rc3}$ instead of $\mathsf{Rc2}$.

- $C = g{\uparrow}$. As in the previous case, $C \in \mathcal{A}(a)$ implies $d_a \in C^{\mathcal{I}_a}$. Hence, $g^{\mathcal{I}_a}(d_a)$ is undefined. By definition of $\mathcal{I}$, $g^{\mathcal{I}}(d_a)$ is also undefined and thus $d_a \in (g{\uparrow})^{\mathcal{I}}$.

For the induction step, we make a case distinction according to the topmost constructor in C:

- $C = C_1 \sqcap C_2$. Since the $\mathsf{R}\sqcap$ rule is not applicable to $\mathcal{A}$ and $C \in \mathcal{A}(a)$, we have $\{C_1, C_2\} \subseteq \mathcal{A}(a)$. The induction hypothesis yields $d_a \in C_1^{\mathcal{I}}$ and $d_a \in C_2^{\mathcal{I}}$. By the semantics, we obtain $d_a \in C^{\mathcal{I}}$.
- $C = C_1 \sqcup C_2$. Similar to the previous case.
- $C = \exists R.D$ with $R \in \mathsf{N_{rR}}$. By definition of $\mathsf{con}(\mathcal{A}, a)$ and choice of $\mathcal{I}_a$ and d_a, $C \in \mathcal{A}(a)$ implies $d_a \in C^{\mathcal{I}_a}$. Hence, there exists an $e \in \Delta_{\mathcal{I}_a}$ such that $(d_a, e) \in R^{\mathcal{I}_a}$ and $e \in D^{\mathcal{I}_a}$. Since d_a has no incoming edges in $\mathcal{I}_a$ (see above), we have $d_a \neq e$. Hence, by Claim 1, $e \in D^{\mathcal{I}_a}$ implies $e \in D^{\mathcal{I}}$. By construction of $\mathcal{I}$, we additionally have $(d_a, e) \in R^{\mathcal{I}}$ and thus $d_a \in (\exists R.D)^{\mathcal{I}}$.
- $C = \exists f.D$. If there is no $b \in \mathsf{O_a}$ such that $\langle a, b\rangle : f \in \mathcal{A}$, then we can argue as in the previous case. Hence assume that such a b exists. Since the $\mathsf{R}\exists\mathsf{f}$ rule is not applicable, we have $b : D \in \mathcal{A}$. By induction, we have $d_b \in D^{\mathcal{I}}$. Since we have $f^{\mathcal{I}}(d_a) = d_b$ by construction of $\mathcal{I}$, we obtain $d_a \in (\exists f.D)^{\mathcal{I}}$.
- $C = \forall R.D$. Fix a pair $(d_a, e) \in R^{\mathcal{I}}$. By definition of $\mathcal{I}$, we have either $(d_a, e) \in R^{\mathcal{I}_a}$ or $e = d_b$ and $\langle a, b\rangle : R \in \mathcal{A}$. In the first case, we have $e \neq d_a$ since d_a has no incoming edges in $\mathcal{I}_a$ and $e \in D^{\mathcal{I}}$ by the semantics and Claim 1. In the second case, we have $D \in \mathcal{A}(b)$ since the $\mathsf{R}\forall$ rule is not applicable to $\mathcal{A}$. Hence, by induction, $e \in D^{\mathcal{I}}$ and thus $d_a \in (\forall R.D)^{\mathcal{I}}$.

This finishes the proof of Claim 2.

Using the two claims, it is easy to show that $\mathcal{I}$ is a model of $\mathcal{A}$ and $\mathcal{T}$. We first show that $\mathcal{I}$ satisfies every assertion in $\mathcal{A}$. For assertions of the form $a : C$, we have $a^{\mathcal{I}} = d_a \in C^{\mathcal{I}}$ by Claim 2. Assertions $\langle a, b\rangle : R$ are obviously satisfied by definition of $\mathcal{I}$. Assertions $\langle a, x\rangle : g$ are satisfied by construction of $\mathcal{I}$ and since the models $\mathcal{I}_b$ (for $b \in \mathfrak{A}$) satisfy $(*)$. Finally, assertions $x_1\, P\, x_2$ are satisfied since δ is a solution for $G(\mathcal{A})$.

It remains to show that $\mathcal{I}$ is a model of $\mathcal{T}$. Fix a concept equation $C \doteq D \in \mathcal{T}$ and a $d \in \Delta_{\mathcal{I}}$. First assume that $d \neq d_a$ for all $a \in \mathfrak{A}$. Let $d \in \Delta_{\mathcal{I}_a}$. Then $d \in C^{\mathcal{I}}$ iff $d \in D^{\mathcal{I}}$ by Claim 1 and since $\mathcal{I}_a$ is a model of $\mathcal{T}$. Now assume $d = d_a$. Since the $\mathsf{R}\doteq$ rule is not applicable to $\mathcal{A}$, we have $a : C_{\mathcal{T}} \in \mathcal{A}$. Hence, by Claim 2, $d_a \in C_{\mathcal{T}}^{\mathcal{I}}$. By definition of $C_{\mathcal{T}}$, this clearly implies $d_a \in C^{\mathcal{I}}$ iff $d_a \in D^{\mathcal{I}}$. $\square$