Temporal Description Logics

Alessandro Artale
Department of Computation
UMIST, Manchester, UK
artale@co.umist.ac.uk

Enrico Franconi
Computer Science Dept.
Univ. of Manchester, UK
franconi@cs.man.ac.uk

http://www.cs.man.ac.uk/~franconi/

Abstract

This chapter will survey the temporal extensions of Description Logics. These formalisms give an emphasis to reasoning tasks such as satisfiability, subsumption, classification, and instance recognition. The analysis will include the whole spectrum of approaches used within the Temporal Description Logics area: from the loosely coupled approaches – which comprise, for example, the enhancement of simple Description Logics with a constraint based mechanism – to the most principled ones – considering a combined semantics for the abstract and the temporal domains. We wil show how these latter approaches described shares many similarities with approaches based on modal temporal logics, since Description Logics share many features with modal logic.

1 Introduction

Description Logics are formalisms designed for a logical reconstruction of representation tools such as frames, object-oriented and semantic data models, semantic networks, KL-One-like languages [Woods and Schmolze, 1992], type systems, and feature logics. Nowadays, Description Logics are also considered the most important unifying formalism for the many object-centered representation languages used in areas other than Knowledge Representation. Important characteristics of Description Logics are high expressivity together with decidability, which guarantee that reasoning algorithms always terminate with the correct answers.

In the Description Logic literature, several approaches for representing and reasoning with time dependent concepts have been proposed. These temporal extensions differ from each others in different ways. They differ on the ontology of time, whether they adopt an interval-based or a point-based notion of time. They differ on the way of adding the temporal dimension, i.e., whether an explicit notion of time is adopted in which temporal operators are used to build new formulae, or temporal information is only implicit in the language by resorting to a STRIPS-like style of representation to represent plans made by sequence of actions. In the case of an explicit representation of time, there is a further distinction between an external and an internal point of view; this distinction has been introduced by Finger and Gabbay [Finger and Gabbay, 1993]. In the external method the very same individual can have different “snapshots” in different moments of time that describe the various states of the individual at these times. In this case, a temporal logic can be seen in a modular way: while an atemporal part of the language describes the “static”

1Description Logics have been also called Frame-Based Description Languages, Term Subsumption Languages, Terminological Logics, Taxonomic Logics, Concept Languages or KL-One-like languages.
Aspects, the temporal part relates the different snapshots describing in such a way the “dynamic” aspects. In the internal method the different states of an individual are seen as different individual components: an individual is a collection of temporal “parts” each one holding at a particular moment.

This chapter is organized as follows. After introducing the main features of Description Logics in section 2, and the basic interval temporal modal logic $\mathcal{HS}$ in section 3, we propose to group the surveyed papers in four different areas: Sections 4 and 5 illustrate what we called the external method where an interval-based or a point-based temporal modal logic is embedded into a Description Logic. Section 6 describes the main languages devised using an internal representation of time, and Section 7 presents the extensions which avoid any explicit temporal representation – like the state-change based approaches. Section 7.2 concludes the chapter.

2 Description Logics

In this section we give a brief introduction to the formal framework of Description Logics. The presentation of the formal apparatus will strictly follow the formalism introduced by [Schmidt-Schauss and Smolka, 1991] and further elaborated by [Hollunder and Nutt, 1990, Donini et al., 1991, Donini et al., 1992, Donini et al., 1994, Donini et al., 1995, Buchheit et al., 1993, De Giacomo and Lenzerini, 1995, De Giacomo and Lenzerini, 1996]: in this perspective, Description Logics are considered as a structured fragment of predicate logic. $\mathcal{ALCF}$ [Schmidt-Schauss and Smolka, 1991] is the minimal Description Logic including full negation and disjunction – i.e., propositional calculus.

The basic types of a concept language are concepts, roles, and features. A concept is a description gathering the common properties among a collection of individuals; from a logical point of view it is a unary predicate ranging over the domain of individuals. Properties are represented either by means of roles – which are interpreted as binary relations associating to individuals of a given class values for that property – or by means of features – which are interpreted as functions associating to individuals of a given class a single value for that property. In the following, we will consider the language $\mathcal{ALCF}$ [Hollunder and Nutt, 1990], extending $\mathcal{ALC}$ with features (i.e., functional roles).

According to the syntax rules of Figure 1, $\mathcal{ALCF}$ concepts (denoted by the letters $C$ and $D$) are built out of atomic concepts (denoted by the letter $A$), atomic roles (denoted by the letter
\[\tau I = \Delta I\]
\[\bot I = \emptyset\]
\[\neg C I = \Delta I \setminus C I\]
\[(C \cap D) I = C I \cap D I\]
\[(C \cup D) I = C I \cup D I\]
\[(\exists P, C) I = \{a \in \Delta I \mid \exists b(a, b) \in P I \land b \in C I\}\]
\[(\forall P, C) I = \{a \in \Delta I \mid \forall b(a, b) \in P I \Rightarrow b \in C I\}\]
\[p \downarrow q I = \{a \in \text{dom} p I \cap \text{dom} q I \mid p I(a) = q I(a)\}\]
\[p \uparrow q I = \{a \in \text{dom} p I \cap \text{dom} q I \mid p I(a) \neq q I(a)\}\]
\[(p \uparrow) I = \Delta I \setminus \text{dom} p I\]
\[(p \circ q) I = p I \circ q I\]

\[\neg F_C(\gamma)\]
\[F_C(\gamma) \land F_D(\gamma)\]
\[F_C(\gamma) \lor F_D(\gamma)\]
\[\exists x. F_P(\gamma, x) \land F_C(x)\]
\[\forall x. F_P(\gamma, x) \Rightarrow F_C(x)\]
\[\forall x. F_P(\gamma, x) \land F_C(x)\]

Figure 2: The extensional and transformational semantics in \(\mathcal{ALCF}\).

\(P\), and atomic features (denoted by the letter \(f\)). The syntax rules are expressed following the tradition of Description Logics [Baader et al., 1990]: they can be read as, e.g., if \(C\) and \(D\) are concept expressions then \(C \cap D\) is a concept expression, too.

Let us now consider the formal semantics of the Description Logic. We define the meaning of concept expressions as sets of individuals – as for unary predicates – and the meaning of roles as sets of pairs of individuals – as for binary predicates. Formally, an interpretation is a pair \(I = (\Delta I, \tau I)\) consisting of a set \(\Delta I\) of individuals (the domain of \(I\)) and a function \(\tau I\) (the interpretation function of \(I\)) mapping every concept to a subset of \(\Delta I\), every role to a subset of \(\Delta I \times \Delta I\), and every feature to a partial function from \(\Delta I\) to \(\Delta I\), such that the equations of the left column in Figure 2 are satisfied. The semantics of the language can also be given by stating equivalences among expressions of the language and First Order Logic formulae. An atomic concept \(A\), an atomic role \(P\), and an atomic feature \(f\), are mapped respectively to the open formulae \(A(\gamma)\), \(P(\alpha, \beta)\), and \(f(\alpha, \beta)\) – with \(f\) a functional relation, also written \(f(\alpha) = \beta\).

The rightmost column of Figure 2 gives the transformational semantics of \(\mathcal{ALCF}\) expressions in terms of equivalent FOL well-formed formulae. A concept \(C\), a role \(P\) and a path \(p\) correspond to the FOL open formulae \(F_C(\gamma)\), \(F_P(\alpha, \beta)\), and \(F_p(\alpha, \beta)\), respectively. It is worth noting that, using the standard model-theoretic semantics, the extensional semantics of the left column can be derived from the transformational semantics of the right column.

A terminology or TBox is a finite set of terminological axioms. For an atomic concept \(A\) – called defined concept – and a (possibly complex) concept \(C\), a terminological axiom is of the form \(A \equiv C\). An atomic concept not appearing on the left-hand side of any terminological axiom is called a primitive concept. We consider acyclic simple TBoxes only: a defined concept may appear at most once as the left-hand side of an axiom, and no terminological cycles are allowed, i.e., no defined concept may occur – neither directly nor indirectly – within its own definition [Nebel, 1991]. An interpretation \(I\) satisfies \(A \equiv C\) if and only if \(A I = C I\).

As an example of the expressive power of \(\mathcal{ALCF}\), we can consider the unary relation (i.e., a concept) denoting the class of happy fathers, defined using the atomic predicates Man, Woman, Doctor, Rich, Famous (concepts), CHILD, FRIEND (roles) and WIFE (feature):
i.e., the men with a wife (exactly one) whose children are doctors having some rich or famous friend.

An ABox is a finite set of assertional axioms, i.e., predications on individual objects. Let \( \mathcal{O} \) be the alphabet of symbols denoting individuals, an assertion is an axiom of the form \( C(a) \), \( R(a, b) \) or \( p(a, b) \), where \( a \) and \( b \) denote individuals in \( \mathcal{O} \). The interpretation \( \mathcal{I} \) is extended over individuals in such a way that \( a^\mathcal{I} \in \Delta^{\mathcal{I}} \) for each individual \( a \in \mathcal{O} \), and \( a^\mathcal{I} \neq b^\mathcal{I} \) if \( a \neq b \) (unique name assumption). \( C(a) \) is satisfied by an interpretation \( \mathcal{I} \) iff \( a^\mathcal{I} \in C^\mathcal{I} \), \( P(a, b) \) is satisfied by \( \mathcal{I} \) iff \( (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I} \), and \( p(a, b) \) is satisfied by \( \mathcal{I} \) iff \( p^\mathcal{I}(a^\mathcal{I}) = b^\mathcal{I} \).

A knowledge base is a finite set \( \Sigma \) of terminological and assertional axioms (i.e., \( \Sigma = \langle TBox, ABox \rangle \)). An interpretation \( \mathcal{I} \) is a model of a knowledge base \( \Sigma \) iff every axiom of \( \Sigma \) is satisfied by \( \mathcal{I} \).

Let us describe now the basic reasoning services provided by a DL-system. From \( \Sigma \) does not logically follows that \( C \equiv \bot \) (written \( \Sigma \not\models C \equiv \bot \)) if there exists a model \( \mathcal{I} \) of \( \Sigma \) such that \( C^\mathcal{I} \neq \emptyset \); we say that \( C \) is satisfiable and we indicate this reasoning problem as concept satisfiability. \( \Sigma \) logically implies \( D \subseteq C \) (written \( \Sigma \models D \subseteq C \)) if \( D^\mathcal{I} \subseteq C^\mathcal{I} \) for every model of \( \Sigma \); we say that \( D \) is subsumed by \( C \) in \( \Sigma \). The reasoning problem of checking whether \( D \) is subsumed by \( C \) in \( \Sigma \) is called subsumption checking. We write \( \Sigma \not\models \) to indicate the problem of checking whether \( \Sigma \) has a model, a problem called knowledge base consistency. \( \Sigma \) logically implies \( C(a) \) (written \( \Sigma \models C(a) \)) if \( a^\mathcal{I} \in C^\mathcal{I} \) for every model of \( \Sigma \); we say that \( a \) is an instance of \( C \) in \( \Sigma \). The reasoning problem of checking whether \( a \) is an instance of \( C \) in \( \Sigma \) is called instance checking. Notice that for propositionally complete languages we have that \( \Sigma \models D \subseteq C \) if and only if \( \Sigma \models D \cap \neg C \equiv \bot \), and \( \Sigma \models C(a) \) if and only if \( \Sigma \cup \{ \neg C(a) \} \not\models \). In other words, subsumption can be reduced to satisfiability and instance checking to knowledge base consistency.

An acyclic simple TBox can be transformed into an expanded TBox having the same models, where no defined concept makes use in its definition of any other defined concept. In this way, the interpretation of a defined concept in an expanded TBox does not depend from any other defined concept. It is easy to see that \( D \) is subsumed by \( C \) in \( \Sigma \) with an acyclic simple TBox if and only if the expansion of \( D \) (w.r.t. \( \Sigma \)) is subsumed by the expansion of \( C \) (w.r.t. \( \Sigma \)) in \( \Sigma' = \langle \emptyset, ABox \rangle \) (i.e., the knowledge base with an the empty TBox). The expansion procedure recursively substitutes every defined concept occurring in a definition with its defining expression; such a procedure may generate a TBox exponential in size, but it was proven [Nebel, 1990] that it works in polynomial time under reasonable restrictions. In the following we will interchangeably refer either to reasoning with respect to a TBox or to reasoning involving expanded concepts with respect to an empty TBox.

### 2.1 Correspondence with Modal Logics

[Schild, 1991] proved the correspondence between \( \mathcal{ALC} \) and the propositional multi-modal logic \( K(m) \) [Halpern and Moore, 1985]. \( K(m) \) is the simplest multi-modal logic interpreted over Kripke structures: there are no restrictions on the accessibility relations. Informally, a concept corresponds to a propositional formula, and it is interpreted as the set of possible worlds over which the formula holds. The existential and universal quantifiers correspond to the possibility and necessity operators over different accessibility relations: \( \Box \phi \) is interpreted as the set of all the possible worlds such that in every \( r \)-accessible world \( C \) holds; \( \Diamond \phi \) is interpreted as the set of all the possible worlds such that in some \( r \)-accessible world \( C \) holds. Thus, roles are interpreted as the accessibility relations between worlds. A knowledge base includes also constraints on the Kripke structures, by stating which are the necessary relations between worlds, and which are
the formulas necessarily holding in some world. Thus, we can speak of satisfiability of a formula \( \phi \) of \( K_{\text{fin}} \) with respect to a set of world constraints \( \Sigma \).

Starting from [Schild, 1991], [Calvanese et al., 1998] have defined a very expressive modal logic \( \mathcal{ALCFQT}_{\text{reg}} \) – which extends the expressivity of converse-PDL, i.e. propositional dynamic logic with the converse operator. They have proven the decidability of satisfiability in \( \mathcal{ALCFQT}_{\text{reg}} \) and its correspondence with a very expressive description logic, which includes \( \mathcal{ALC} \), functional and general cardinality restrictions, inverse roles, and regular expressions over roles. In \( \mathcal{ALCFQT}_{\text{reg}} \), full cyclic terminological axioms can be expressed.

3 The Modal Temporal Logic \( \mathcal{HS} \)

In this section we briefly introduce the interval-based modal temporal logic \( \mathcal{HS} \) [Halpern and Shoham, 1991]. As Description Logics are in a strict correspondence with propositional modal logics, interval-based temporal Description Logics have in the \( \mathcal{HS} \) logic their natural ancestor.

Well-formed formulae of \( \mathcal{HS} \) are built augmenting the propositional calculus with the modal temporal operators for \textit{starts} and \textit{finishes} and their inverses: \( \langle \text{starts} \rangle \), \( \langle \text{finishes} \rangle \), \( \langle \text{started-by} \rangle \), \( \langle \text{finished-by} \rangle \). Formulae are interpreted as the set of intervals where they hold true. In carrying

the interpretation process there is an implicit use of the reference interval, i.e., the actual evaluation interval – called \textit{now}. The modal operators relate the \textit{now} interval with other intervals. Intuitively, the meaning of an \( \mathcal{HS} \) formula is the following:

\( \langle \text{starts} \rangle \phi \) is true iff \( \phi \) holds at some interval starting the reference interval

\( \langle \text{finishes} \rangle \phi \) is true iff \( \phi \) holds at some interval finishing the reference interval.

while the modal operators \( \langle \text{started-by} \rangle \), \( \langle \text{finished-by} \rangle \) introduce the inverse temporal relations. It also possible to define the duals of these operators as usual: \( [X] \equiv \neg \langle X \rangle \neg \phi \) – where \( X \) stands for a temporal relation. For example, \( \langle \text{starts} \rangle \phi \) says that \( \phi \) is true at all beginning intervals.

It is worth noting that, the thirteen temporal relations can be simulated by using the above mentioned four modal operators provided that the temporal structure allow for point intervals [Venema, 1990]\(^2\). Indeed, if this is the case, we can define both a \textit{beginning point} and an \textit{ending point} modal operators in the following way:

\[
\begin{align*}
[BP] \phi & \equiv (\langle \text{starts} \rangle \perp \phi) \lor \langle \text{starts} \rangle (\langle \text{starts} \rangle \perp \phi) \\
[EP] \phi & \equiv (\langle \text{finishes} \rangle \perp \phi) \lor \langle \text{finishes} \rangle (\langle \text{finishes} \rangle \perp \phi)
\end{align*}
\]

which allow to define the \textit{meets} and \textit{met-by} modal operators as: \( \langle \text{meets} \rangle \phi \equiv [EP] \langle \text{started-by} \rangle \phi \) and \( \langle \text{met-by} \rangle \phi \equiv [BP] \langle \text{finished-by} \rangle \phi \). For the other temporal relations we have the following definitions:

\( \langle \text{during} \rangle \phi \equiv \langle \text{starts} \rangle \langle \text{finishes} \rangle \phi \)

\( \langle \text{before} \rangle \phi \equiv \langle \text{meets} \rangle \langle \text{meets} \rangle \phi \)

\( \langle \text{overlaps} \rangle \phi \equiv \langle \text{starts} \rangle \langle \text{finished-by} \rangle \phi \)

The notion of \textit{Mortal} can be expressed in this logic as: \( \text{LivingBeing} \land \langle \text{after} \rangle \text{– LivingBeing} \), with the meaning of a \text{LivingBeing} who will not be alive in some future interval.

For what concerns the semantics of this tense logic, an interpretation is a pair \( \langle S, V \rangle \). \( S \) is a temporal structure \( \langle P, \leq \rangle \), where \( P \) is a set of time points and \( \leq \) is a partial order on \( P \). \( V \) is a

\(^2\)If the temporal structure allows only for proper intervals you need also the \textit{meets} as a basic modal operator.
\(\langle S, V \rangle, \langle t_1, t_2 \rangle \models p\) iff \(\langle t_1, t_2 \rangle \in V(p), p \in \Phi_0\)

\(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \neg \phi\) iff \(\langle S, V \rangle, \langle t_1, t_2 \rangle \not\models \phi\)

\(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \phi \land \psi\) iff \(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \phi\) and \(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \psi\)

\(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \langle \text{starts} \rangle \phi\) iff \(\exists u. t_1 \leq u < t_2\) and \(\langle S, V \rangle, \langle t_1, u \rangle \models \phi\)

\(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \langle \text{started-by} \rangle \phi\) iff \(\exists u. t_2 < u < \langle S, V \rangle, \langle t_1, u \rangle \models \phi\)

\(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \langle \text{finishes} \rangle \phi\) iff \(\exists u. \langle s, u \rangle \leq t_2\) and \(\langle S, V \rangle, \langle u, t_2 \rangle \models \phi\)

\(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \langle \text{finished-by} \rangle \phi\) iff \(\exists u. u < t_1\) and \(\langle S, V \rangle, \langle u, t_2 \rangle \models \phi\).

Figure 3: \(\mathcal{HS}\)'s semantics

valuation function which maps each primitive proposition into a set of closed intervals where it is true. Thus, if \(\Phi_0\) denotes the alphabet of primitive propositions, \(V : \Phi_0 \rightarrow 2^I\), where \(I\) is the interval domain: \(I = \{\langle t_1, t_2 \rangle \mid t_1 \leq t_2, t_1, t_2 \in P\}\). The truth relation \(\models\) is inductively defined as reported in figure 3. As usual, a formula \(\phi\) is said satisfiable if there exists an interpretation \(\langle S, V \rangle\) where \(\langle S, V \rangle, \langle t_1, t_2 \rangle \models \phi\) for some \(\langle t_1, t_2 \rangle \in I\). A formula \(\phi\) is said valid with respect to a class of temporal structures \(\mathcal{A}\) if \(\neg \phi\) is not satisfiable in \(\mathcal{A}\).

Halpern and Shoham prove many interesting complexity results concerning the validity and satisfiability problems. It is worth to note how these results depend from the underlying temporal structure. Depending on the class of temporal structures, the validity problem range from being decidable to being \(\Pi^1_1\)-hard (for what concerns the satisfiability problem its complexity class can be obtained by observing that it is the complement of the validity problem). The bad result is summarized by the following statement:

"One gets decidability only in very restricted cases, such as when the set of temporal models considered is a finite collection of structures, each consisting of a finite set of natural numbers (since in this case one can simply perform an exhaustive check on all structures)." [Halpern and Shoham, 1991]

To present the complexity results in a formal way we introduce the notion of infinitely ascending sequence. A temporal structure is said to contain an infinitely ascending sequence if it contains an infinite sequence of points \(t_0, t_1, \ldots\) such that \(t_i < t_{i+1}\). The critical complexity result is stated by the following proposition.

**Proposition 3.1** The validity problem for any class of temporal structures that contains an infinitely ascending sequence is r.e.-hard.

Which was proven by constructing tense formulae that encode the computation of a Turing machine. The next theorem summarizes the complexity results provided by Halpern and Shoham.

**Theorem 3.1** The validity problem for all dense, linear and unbounded classes of temporal structures is r.e.-complete. The validity problem for \(Q\) is r.e.-complete. The validity problem for \(N\) is \(\Pi^1_1\)-complete. The validity problem for \(R\) is in \(\Pi^1_2\).
\[
\begin{align*}
<\text{concept}> & ::= <\text{atomic-concept}> \\
& \quad | \quad \text{(and} <\text{concept}>^+) \\
& \quad | \quad \text{(all} <\text{role}> <\text{concept}>) \\
& \quad | \quad \text{(atleast} \ \text{min} <\text{role}>) \\
& \quad | \quad \text{(atmost} \ \text{max} <\text{role}>) \\
& \quad | \quad \text{(at} <\text{interval}> <\text{concept}>) \\
& \quad | \quad \text{(sometime} <\text{interval-variable}^+) <\text{time-net}> <\text{concept}>) \\
& \quad | \quad \text{(alltime} <\text{interval-variable}^+) <\text{time-net}> <\text{concept}>) \\
<\text{atomic-concept}> & ::= \text{symbol} \\
<\text{role}> & ::= <\text{atomic-role}> \\
& \quad | \quad \text{(and} <\text{role}>^+) \\
& \quad | \quad \text{(domain} <\text{concept}>) \\
& \quad | \quad \text{(range} <\text{concept}>) \\
& \quad | \quad \text{(at} <\text{interval}> <\text{role}>) \\
& \quad | \quad \text{(sometime} <\text{interval-variable}^+) <\text{time-net}> <\text{role}>) \\
& \quad | \quad \text{(alltime} <\text{interval-variable}^+) <\text{time-net}> <\text{role}>) \\
<\text{atomic-role}> & ::= \text{symbol} \\
<\text{time-net}> & ::= <\text{time-constraint}> \\
& \quad | \quad \text{(and} <\text{time-constraint}>^+) \\
<\text{time-constraint}> & ::= ( <\text{interval-relation}> <\text{interval}> <\text{interval}>), \quad <\text{interval}> \\
& \quad | \quad ( <\text{comparison}> <\text{interval}> <\text{duration-constant}>), \quad <\text{duration-constant}> \\
& \quad | \quad ( <\text{granularity}> <\text{interval}>), \quad <\text{granularity}> \\
<\text{interval-relation}> & ::= \text{equal} \quad | \quad \text{meets} \quad | \quad \text{met-by} \quad | \quad \text{after} \quad | \quad \text{before} \\
& \quad | \quad \text{overlaps} \quad | \quad \text{overlapped-by} \quad | \quad \text{starts} \quad | \quad \text{started-by} \\
& \quad | \quad \text{finishes} \quad | \quad \text{finished-by} \quad | \quad \text{during} \quad | \quad \text{contains} \\
& \quad | \quad (\text{or} \quad <\text{interval-relation}>^+) \\
<\text{comparison}> & ::= < \quad | \quad \leq \quad | \quad = \quad | \quad \geq \quad | \quad > \\
<\text{granularity}> & ::= \text{sec} \quad | \quad \text{min} \quad | \quad \text{hour} \quad | \quad . . . \\
<\text{interval}> & ::= <\text{interval-variable}> <\text{interval-constant}> \quad | \quad \text{NOW} \\
<\text{interval-variable}> & ::= \text{symbol} \\
<\text{interval-constant}> & ::= \text{symbol} \\
<\text{duration-constant}> & ::= \text{symbol} \\
\end{align*}
\]

Figure 4: Syntax rules for the Schmiedel proposal.

4 Interval-based Temporal Description Logics

4.1 Schmiedel’s Formalism

Schmiedel [Schmiedel, 1990] is the first attempt to extend Description Logics with an interval-based temporal logic. The temporal variant of the Description Logic is equipped with a model-
Theorict semantics. The underlying Description Logic is the $\mathcal{FLN}^R$ language\textsuperscript{3} [Donini et al., 1995], while the new term-forming operators are the temporal qualifier at, the existential and universal temporal quantifiers sometime and alltime. The at operator specifies the time at which a concept holds while sometime and alltime are temporal quantifiers introducing temporal variables. Temporal variables are constrained by means of temporal relationships based on Allen’s intervals algebra extended with metric constraints in order to deal with durations, absolute times and granularities of intervals. Figure 4 shows the syntax of the temporal extension proposed by Schmiedel. To give an example of this temporal Description Logic, the concept of Mortal can be defined as:

\textbf{Mortal} $\equiv$ \textbf{LivingBeing} and (\textbf{sometime}(x) (after x \textbf{NOW}) (at x (not \textbf{LivingBeing})))

with the meaning of a \textbf{LivingBeing} at the reference interval \textbf{NOW}, who will not be alive at an interval \(x\) sometime after the reference interval \textbf{NOW}. A concept denotes a set of pairs of temporal intervals and individuals \((i, a)\). With the use of the at temporal operator it is possible to bind the evaluation time of a concept to a particular interval of time:

\((\text{at } '1993' \text{ Student})\)

denotes the set of persons that were students during the 1993.

The expressive power of the language is a direct consequence of the introduction of temporal variables constrained by temporal relations. In this way the logic is able to express abstract temporal patterns. Temporal variables are introduced by the quantifiers sometime and alltime together with a set of constraints – indicated as \(<\text{time-net}>\) in the syntax. There are three kind of temporal constraints: qualitative relations in single intervals by using the Allen algebra, metric constraints on a single interval, and granularity constraints where an interval is required to take values that are multiples of some time unit. The following example shows a \(<\text{time-net}>\) which makes use of the three kind of constraints:

\[(\text{and } (\text{day } x) (= x '24h') \text{ (day } y)(= y '24h') \text{ (meets } x y)\text{ (or starts finishes during) } x \text{ NOW})\text{ (or starts finishes during) } y \text{ NOW})\]

where \(x\) and \(y\) are two consecutive days within \textbf{NOW}. The two constraints (day \(x\)) and (\(= x '24h'\)) restrict \(x\) to be coincident with a calendar day. Note that, without the constraint (day \(x\)), \(x\) could be any interval spanning 24 hours, but not necessarily a full day of the calendar. On the other hand, leaving away the duration constraint, \(x\) could take any value that is started and finished by a full day.

Let us show now the model-theoretic semantics. The author assumes a discrete temporal structure over the integers \(\mathcal{T} = (\mathbb{Z}, <)\), where \(<\) is a strict partial order over \(\mathbb{Z}\). The interval set \(\mathcal{T}_2^\mathbb{Z}\) is defined as the set of closed intervals \([u, v] = \{x \in \mathbb{Z} \mid u \leq x \leq v, u \neq v\}\) in \(\mathcal{T}\). For the temporal relations is assumed a fixed temporal model \(\mathcal{M} = \{M_1, M_2, M_3, M_4, M_5\}\) such that:

\textsuperscript{3}Note that \(\mathcal{FLN}^R\) differs from \(\mathcal{ALC}\) in that neither it contains the concepts \(\top\) and \(\bot\) nor it allows for complement or disjunction; the letter \(\mathbb{Z}\) stands for cardinality restrictions on roles, while \(\mathcal{R}\) indicates the role conjunction operator.
\[(C_1 \ldots C_n)^\exists \mathcal{V}, t = \bigcap_{i=1}^{n} (C_i)^\exists \mathcal{V}, t\]

\[\text{(all } P\text{) } (C_i)^\exists \mathcal{V}, t = \{d \in \Delta^\mathcal{I} \mid (P)^\exists \mathcal{V}, t(d) \subseteq C_i^\exists \mathcal{V}, t\}\]

\[\text{(atleast } m\text{) } (P)^\exists \mathcal{V}, t = \{d \in \Delta^\mathcal{I} \mid \left| \left( P \right)^\exists \mathcal{V}, t(d) \right| \geq m\}\]

\[\text{(atmost } m\text{) } (P)^\exists \mathcal{V}, t = \{d \in \Delta^\mathcal{I} \mid \left| \left( P \right)^\exists \mathcal{V}, t(d) \right| \leq m\}\]

\[\text{(at } x\text{) } (C_i)^\exists \mathcal{V}, t = \begin{cases} 
C_i^\exists \mathcal{V}(x) & \text{if } x \neq \text{NOW} \\
C_i^\exists \mathcal{V}, t & \text{if } x = \text{NOW}
\end{cases}\]

\[(\text{sometime } X\text{) } TC\text{ ) } (C_i)^\exists \mathcal{V}, t = \{d \in \Delta^\mathcal{I} \mid \exists W. W \in \langle X, TC \rangle^{\mathcal{E}} \wedge d \in C_i^\exists \mathcal{V}, t\}\]

\[\text{(alltime } X\text{) } TC\text{ ) } (C_i)^\exists \mathcal{V}, t = \{d \in \Delta^\mathcal{I} \mid \forall W. W \in \langle X, TC \rangle^{\mathcal{E}} \wedge d \in C_i^\exists \mathcal{V}, t\}\]

\[\text{(domain) } (C_i)^\exists \mathcal{V}, t = C_i^\exists \mathcal{V}, t \times \Delta^\mathcal{I}\]

\[\text{(range) } (C_i)^\exists \mathcal{V}, t = \Delta^\mathcal{I} \times C_i^\exists \mathcal{V}, t\]

Figure 5: Semantics for composed terms.

\[\mathcal{M}_1 : \text{Interval-constant } \mapsto T_\mathcal{X}^\ast\]

\[\mathcal{M}_2 : \text{Duration-constant } \mapsto 2T_\mathcal{X}^\ast\]

\[\mathcal{M}_3 : \text{Comparison-operator } \mapsto 2T_\mathcal{X}^\ast \times T_\mathcal{X}^\ast\]

\[\mathcal{M}_4 : \text{Interval-relation } \mapsto 2T_\mathcal{X}^\ast \times T_\mathcal{X}^\ast\]

\[\mathcal{M}_5 : \text{Granularity-predicates } \mapsto 2T_\mathcal{X}^\ast\]

Such an interpretation preserves the intuitive meaning of the various temporal constructs – e.g., \langle M_1['August 1990'], M_3['September 1990']) \rangle \in \mathcal{M}_4[\text{meets}] \text{ and } M_3['3/12/1990'] \in \mathcal{M}_5[\text{day}]. A variable assignment is a function \(\mathcal{V} : X \mapsto T_\mathcal{X}^\ast\). The time-net interpretation, \(\langle X, TC \rangle^{\mathcal{E}}\) – where \(TC\) is a set of temporal constraints and \(X\) is a set of temporal variables – is the set of all possible variable assignments which satisfy the temporal relations in \(TC\). As an example, let \(X = \{x, y\}\) and \(TC = \{\{\text{meets} \ x \ y\}\}\) then \(\forall \mathcal{V} \in \langle X, TC \rangle^{\mathcal{E}}, \langle \mathcal{V}(x), \mathcal{V}(y) \rangle \in \mathcal{M}_4[\text{meets}]\). Furthermore, \(\langle X, TC \rangle^{\mathcal{E}}_{x \mapsto t}\) denotes the set of interpretations of a time-net where \(x\) is mapped to \(t\).

The interpretation of temporal conceptual expressions is a triple \(\mathcal{I} = \langle T_\mathcal{X}^\ast, \Delta^\mathcal{I}, ^\exists \mathcal{I}\rangle\), with the interval domain \(T_\mathcal{X}^\ast\); a generic individual domain \(\Delta^\mathcal{I}\) and an interpretation function \(^\exists \mathcal{I}\) which fixes the extension of primitive concepts and roles – denoted with the letters \(A\) and \(P\) respectively – in such a way that:

\[A^\mathcal{I} \mapsto (T_\mathcal{X}^\ast \mapsto 2\Delta^\mathcal{I})\]
\[P^\mathcal{I} \mapsto (T_\mathcal{X}^\ast \mapsto 2\Delta^\mathcal{I} \times \Delta^\mathcal{I})\]

Furthermore, the interpretation function has to satisfy the equations showed in figure 5 (the equations for the analogous operators on roles are left to the intuition of the reader). Thus, each concept (role) is mapped to a function that assigns set of individuals (pairs of individuals) to each time interval. The notation \((P)^\exists \mathcal{V}, t(d)\) stands for the set \(\{d' \in \Delta^\mathcal{I} \mid (d, d') \in (P)^\exists \mathcal{V}, t\}\), while \(C_i^\exists \mathcal{V}, t\) stands for \(C_i^\exists \mathcal{V}(t)\) in which all free variables present in \(C\) are evaluated using \(\mathcal{V}\). Thus, the interpretation of a generic expression depends both from a given time interval \(t\) and from an assignment \(\mathcal{V}\) for the free variables.
If we consider just closed concept expressions the interpretation does not depend on $\mathcal{V}$. An interpretation $I$ is a model for a concept $C$ if $C^I \neq \emptyset$ for some $t$. A concept $C$ is subsumed by the concept $D$ ($C \sqsubseteq D$) if $C^I \subseteq D^I$ for all interpretation $I$ and all time intervals $t$.

The Schmiedel's work does not propose any algorithm for computing subsumption for this temporal variant of Description Logics, but he only gives some preliminary hints. Schmiedel's temporal Description Logic can be formally related to the interval-based modal temporal logic $\mathcal{HS}$ proposed by Halpern and Shoham. As Bettini [Bettini, 1993] shows (see lemma 4.2 in the next Section), Schmiedel's logic when closed under complementation contains the $\mathcal{HS}$ logic as a proper fragment. Unfortunately, the $\mathcal{HS}$ logic is shown to be undecidable, at least for most interesting classes of temporal structures. Actually, Schmiedel's logic is argued to be undecidable, sacrificing the main benefit of Description Logics, i.e., the possibility to have decidable inference techniques (see the next Section for more details).

### 4.2 The Undecidable Realm

Bettini [Bettini, 1993, Bettini, 1997] suggests a variable-free extension with both existential and universal temporal quantification. He gives undecidability results for the class of temporal languages proposed – resorting to the undecidability results of Halpern and Shoham’s temporal logic – and investigates approximated reasoning algorithms. Starting from the language $\mathcal{ALCCN}$ two concept expressions are introduced: $\diamond T E$, $C$ and $\Box T E$, $C$. The $\diamond$ and $\Box$ operators are respectively the existential and universal temporal quantifiers, but, unlike Schmiedel’s formalism, they do not allow for explicit interval variables. The temporal expression, $T E$, is a set of temporal constraints on two implicit intervals: the reference interval and the current one. This makes the language very close to the $\mathcal{HS}$ logic if we note that each $T E$ can be simulated by an appropriate combination of modal temporal operators. Bettini presents a hierarchy of temporal expressions $T E_i$, with $i = 1 \ldots 5$, with higher temporal expressiveness. $T E_1$ allows to express single basic temporal relations, i.e., meets, starts, finishes and their converses. They are called basic since the other Allen’s relations can be expressed by a combination of them. $T E_2$ can be any of the thirteen Allen’s relations. $T E_3$ allows all those combination of temporal relations which give rise to the pointisable interval set. $T E_4$ allows the arbitrary disjunction of temporal relations. $T E_5$, in addition to qualitative relations between the current and the reference interval, allows the specification of metric constraints for the current interval by bounding its length by means of duration intervals or by relating it with specific constant intervals – reaching in this latter case an expressive power close to the one of the time-net of Schmiedel. Every language is indicated by prefixing the name of the non-temporal Description Logic composing it with the letter $\mathcal{T}$ and by adding a numerical subscript which denotes the kind of temporal expressions allowed. For example, $\mathcal{TALCC_5}$ is the temporal extension of $\mathcal{ALCC}$ allowing $T E_5$ as temporal expressions in $\diamond T E$, $C$ and $\Box T E$, $C$ operators. In this framework the concept of Mortal can be defined as:

$$\text{Mortal} = \text{LivingBeing} \sqcap \diamond (\text{after}) \cdot (\not\text{LivingBeing})$$

We observe that, depending on the underlying non-temporal Description Logic, there are some expressiveness equivalence between apparently different languages due to the interaction between temporal and non-temporal operators. As an example, when the languages are built upon $\mathcal{ALC}$ the following equivalences hold: $\mathcal{TALC}_1 \equiv \mathcal{TALC}_2 \equiv \mathcal{TALC}_3 \equiv \mathcal{TALC}_4$. These equivalences can be easily proved by recovering to the equivalences showed by Halpern and Shoham on the reducibility of the global Allen’s relations to a combination of the basic one, and by noting that:

$$\diamond (rel_1, rel_2), C \equiv (\diamond (rel_1), C \sqcup \diamond (rel_2), C)$$

$$\Box (rel_1, rel_2), C \equiv (\Box (rel_1), C \sqcap \Box (rel_2), C)$$
whenever you read \((rel_1, rel_2)\) as the disjunction (or \(rel_1 \lor rel_2\)).

As in the case of Schmieder's formalism, the time is part of the semantic structure. A concept denotes a set of pairs of temporal intervals and individuals \(\langle i, a \rangle\). Intuitively, a given individual belongs to the extension of a concept at certain time intervals. The temporal operators allow to relate the current interval to other intervals. The expression \(\diamond TE. C\) denotes the set of pairs \(\langle i, a \rangle\) such that the individual \(a\) belongs to the extension of \(C\) at the interval \(i'\) which satisfies \(i' \cap TE.i\). For example, \(\langle 1990, a_1 \rangle\) belongs to the set \(\diamond after. Engineer\) if there exists an interval that is after 1990 in which the individual \(a_1\) is an Engineer -- this will be the case if \(\langle 1991, a_1 \rangle\) would be an instance of the concept Engineer. The \(\Box\) operator works in a similar way, but it qualifies universally the implicit temporal variable that satisfies the temporal constraints.

Let us now briefly introduce the model-theoretic semantics as illustrated by Bettini. Given an unbound and untunable temporal structure - reported as \(\langle \mathcal{UL}, < \rangle\) - the domain of temporal intervals, \(\mathcal{T}_\mathcal{U}\), is defined, as usual, as the set of pairs of points in \(\mathcal{UL}\). A fixed temporal model, \(\mathcal{M}\), is assumed in the same spirit of Schmieder, while \(\mathcal{E}\) is the temporal interpretation function that maps a temporal expression \(TE\) and a reference interval \(i\) into a set of intervals in \(\mathcal{T}_\mathcal{U}\). \(\mathcal{E}\) must satisfy the following equations:

\[
[rel]^\mathcal{E}_i = \{i' | \langle i', i \rangle \in \mathcal{M}[rel]\}
\]

\[
[rel_1, \ldots, rel_n]^\mathcal{E}_i = \bigcup_{j=1}^n [rel_j]^\mathcal{E}_i
\]

along with other equations taking into account the interaction between qualitative and metric temporal constraints. An interpretation structure is a tuple \(\mathcal{I} = (\mathcal{T}_\mathcal{U}, \Delta^\mathcal{I}, \mathcal{I})\), where \(\Delta^\mathcal{I}\) is a set of individuals, \(\mathcal{T}_\mathcal{U}\) is the time interval set and \(\mathcal{I}\) an interpretation function. It assigns a meaning to generic concept expressions by mapping each primitive concept into a set of pairs \(\langle i, a \rangle\), and each role into a set of triples \(\langle i, a, b \rangle\). For the complex temporal concept expressions the function \(\mathcal{I}\) has to satisfy the usual equations for DL constructs -- only the time-dependent constructs are reported here:

\[
(\diamond TE. C)^\mathcal{I} = \{\langle i, a \rangle | \exists i'. i' \in (TE)^\mathcal{E}_i \land \langle i', a \rangle \in C^\mathcal{I}\}
\]

\[
(\Box TE. C)^\mathcal{I} = \{\langle i, a \rangle | \forall i'. i' \in (TE)^\mathcal{E}_i \rightarrow \langle i', a \rangle \in C^\mathcal{I}\}
\]

An interpretation \(\mathcal{I}\) is a model for a concept \(C\) if \(C^\mathcal{I} \neq \emptyset\). If a concept has a model, then it is satisfiable, otherwise it is unsatisfiable. A concept \(C\) is subsumed by a concept \(D\) (written \(C \sqsubseteq D\)) if \(C^\mathcal{I} \subseteq D^\mathcal{I}\) for every interpretation \(\mathcal{I}\). Two concepts \(C, D\) are equivalent (written \(C \equiv D\)) if \(C^\mathcal{I} = D^\mathcal{I}\) for every interpretation \(\mathcal{I}\).

Let us comment now on the temporal expressivity of this family of languages. The absence of explicit temporal variables weakens the temporal structure of a concept since arbitrary relationships between more than two intervals could not be represented anymore. For example, having only implicit intervals it is not possible to describe the situation where two concept expressions, say \(C\) and \(D\), hold at two meeting intervals (say \(x, y\)) with the first interval starting and the second finishing the reference interval (i.e., the temporal pattern \((x \text{ meets } y)(x \text{ starts } y)(y \text{ finishes } y)\) cannot be represented). More precisely, it is not possible to represent temporal relations between more than two intervals if they are not derivable by the temporal propagation of the constraints imposed on pairs of variables. While explicit variables go against the general trust of Description Logics, the gained expressive power together with the observation that the variables are limited only to the temporal part of the language are the main motivation for using them.

This limited temporal expressive power is motivated by the need to study the complexity of the inference machinery in such kind of hybrid languages. As a matter of fact, the merit of this
work is the deep analysis of the computational properties of such Description Logics extended with temporal operators on intervals. Bettini starts from the analysis presented in the work of Halpern and Shoham by discovering precise equivalences between some of these Description Logics and the logic $\mathcal{HS}$.

**Lemma 4.1 (Correspondence between $\mathcal{HS}$ and $\mathcal{TALC}_1$)** A formula $\phi$ in the logic $\mathcal{HS}$ is satisfiable in a linear and unbounded temporal structure if and only the correspondent concept $C$ is a satisfiable term in $\mathcal{TALC}_1(\mathcal{UL}, \leq)$.

The following theorem easily follows from the above reduction, the results provided for $\mathcal{HS}$, and by observing that in a propositionally complete language subsumption reduces to unsatisfiability.

**Theorem 4.1** The problem of determining the satisfiability of terms in $\mathcal{TALC}_1(\mathcal{UL}, \leq)$ is co-r.e.-hard, for all $i$ with $1 \leq i \leq 5$. The subsumption problem in $\mathcal{TALC}_1(\mathcal{UL}, \leq)$ is r.e.-hard.

These complexity results were obtained for $\mathcal{HS}$ but they were limited to temporal structures that allowed durationless intervals. Bettini extends these results to temporal structures which allow only for proper intervals. The following theorem considers the realm of integer numbers as a model for a discrete temporal structure where you cannot express durationless intervals.

**Theorem 4.2** The satisfiability and subsumption problems for the languages $\mathcal{TALC}_i(\mathbb{Z}, <)$ with $i = 1, \ldots, 5$ are undecidable. In particular, they belong to the classes $\Sigma_1^1$-hard and $\Pi_1^1$-hard, respectively.

The author then shows that the Schmiedel’s formalism, indicated as $\mathcal{TB}$, is strictly related to the class of languages $\mathcal{TALC}_i$ (and so to the modal logic $\mathcal{HS}$). In particular, he considers the language $\mathcal{TB}$ extended with the complement operator, and called $\mathcal{TB}_{\text{neg}}$.

**Lemma 4.2 (Correspondence between $\mathcal{TALC}_1$ and $\mathcal{TB}_{\text{neg}}$)** Every term $\phi$ in $\mathcal{TALC}_1(\mathbb{Z}, <)$ can be translated into a term in $\mathcal{TB}_{\text{neg}}$, in such a way that $\phi$ is satisfiable if and only if its translation is a $\mathcal{TB}_{\text{neg}}$ satisfiable term.

From this equivalence follows that for the language $\mathcal{TB}_{\text{neg}}$ the very same complexity results of theorem 4.2 apply. The deep analysis led by Bettini leaves some important open problems, as declared by the author himself. It remains an open problem the decidability of satisfiability and subsumption when considering temporal structures which allow only for proper intervals and which are different from the structure of integer numbers. An example of such a structure is the one that interprets intervals on the rational numbers, $\mathbb{Q}$. A critical point that will deserve a deep investigation is the decidability of satisfiability and subsumption with respect to languages without negation. Since in this case the two problems are no more each other reducible they belong to different complexity classes. In particular, it remains an open problem whether reasoning in the language $\mathcal{TB}$, as presented by Schmiedel, is decidable.

### 4.3 Towards Decidable Logics

Artale and Franconi [Artale and Franconi, 1994, Artale and Franconi, 1998, Artale, 1994] consider a class of interval-based temporal Description Logics by reducing the expressivity of [Schmiedel, 1990]. While Schmiedel’s work lacks of a computational machinery, and Halpern and Shoham’s logic is undecidable, Artale and Franconi present different decidable logics, providing for them sound, complete and terminating reasoning algorithms.
The language $\mathcal{TL}-\mathcal{ALCF}$ is presented here. $\mathcal{TL}-\mathcal{ALCF}$ is composed by the temporal Logic $\mathcal{TL}$ - able to express temporally quantified terms - and the non-temporal Description Logic $\mathcal{ALCF}$. Concept expressions (denoted by $C, D$) are syntactically built following the syntax rules of figure 6. Temporal concepts ($C, D$) are distinct from non-temporal concepts ($E, F$). Names for atomic features and atomic parametric features are from the same alphabet of symbols; the $\star$ symbol is not intended as operator, but only as differentiating the two syntactic types. For the basic interval relations the Allen notation [Allen, 1991] is adopted. Temporal variables are
introduced by the temporal existential quantifier “◊” – excluding the special temporal variable \( \tau \), usually called NOW, and intended as the reference interval.

Let us now informally see the intended meaning of the terms of the language \( \mathcal{TL-ALC}_F \). Concept expressions are interpreted over pairs of temporal intervals and individuals \( \langle i, a \rangle \), meaning that the individual \( a \) is in the extension of the concept at the interval \( i \). If a concept is intended to describe an action, then its interpretation can be seen as the set of individual actions of that type occurring at some interval.

Within a concept expression, the special “\( \tau \)” variable denotes the current interval of evaluation; in the case of actions, we can say that it refers to the temporal interval at which the action itself occurs. The temporal existential quantifier introduces interval variables, related to each other and possibly to the \( \tau \) variable in a way defined by the set of temporal constraints. In order to evaluate a concept at an interval \( X \), different from the current one, we need to temporally qualify it at \( X \) – written \( C \@ X \); in this way, every occurrence of \( \tau \) embedded within the concept expression \( C \) is interpreted as the \( X \) variable\(^4\). The informal meaning of a concept with a temporal existential quantification can be understood with the following examples in the action domain.

\[
\text{Basic-Stack} \triangleq \Diamond (x, y) \left( x \text{ meets } \tau \right) \left( \tau \text{ meets } y \right). \left( \text{\texttt{+BLOCK}: OnTable}@x \quad \land \quad \left( \text{\texttt{+BLOCK}: OnBlock}@y \right) \right)
\]

Figure 7 shows the temporal dependencies of the intervals in which the concept Basic-Stack holds. Basic-Stack denotes, according to the definition (a terminological axiom), any action occurring at some interval involving a \texttt{+BLOCK} that was once OnTable and then OnBlock. The \( \tau \) interval could be understood as the occurring time of the action type being defined: referring to it within the definition is an explicit way to temporally relate states and actions occurring in the world with respect to the occurrence of the action itself. The temporal constraints \( (x \text{ m } \tau) \) and \( (\tau \text{ m } y) \) state that the interval denoted by \( x \) should meet the interval denoted by \( \tau \) – the occurrence interval of the action type Basic-Stack – and that \( \tau \) should meet \( y \). The parametric feature \texttt{+BLOCK} plays the role of formal parameter of the action, mapping any individual action of type Basic-Stack to the block to be stacked, independently from time. Please note that, whereas the existence and identity of the \texttt{+BLOCK} of the action is time invariant, it can be qualified differently in different intervals of time, e.g., the \texttt{+BLOCK} is necessarily OnTable only during the interval denoted by \( x \).

In this framework, the concept defining a Mortal is

\[
\text{Mortal} \triangleq \Diamond (x) \left( \tau \text{ before } x \right). \text{LivingBeing} \quad \sqcap \quad \neg \text{LivingBeing}@x.
\]

\( \mathcal{TL-ALC}_F \) is provided with a Tarski-style extensional semantics. A linear, unbounded, and dense temporal structure \( \mathcal{T} = (\mathcal{P}, <) \) is assumed, where \( \mathcal{P} \) is a set of time points and \( < \) is a strict partial order on \( \mathcal{P} \). The interval set of a structure \( \mathcal{T} \) is defined as the set \( \mathcal{I}_\tau \) of all closed proper intervals \( [u, v] = \{ x \in \mathcal{P} \mid u \leq x \leq v, u \neq v \} \) in \( \mathcal{T} \). A primitive interpretation \( \mathcal{I} \triangleq \left< \mathcal{I}_\tau, \Delta^\mathcal{I}, \mathcal{I} \right> \) consists of a set \( \mathcal{I}_\tau \) (the interval set of the selected temporal structure \( \mathcal{T} \)), a set \( \Delta^\mathcal{I} \) (the domain of \( \mathcal{I} \)), and a function \( \mathcal{I} \) (the primitive interpretation function of \( \mathcal{I} \)) which gives a meaning to

\(^4\)Since any concept is implicitly temporally qualified at the special \( \tau \) variable, it is not necessary to explicitly qualify concepts at \( \tau \).
atomic concepts, features and parametric features:

\[
\begin{align*}
A^\mathcal{I} & \subseteq \tau_+^\mathcal{I} \times \mathcal{I}^\mathcal{I} & P^\mathcal{I} & \subseteq \tau_+^\mathcal{I} \times \mathcal{I}^\mathcal{I} \times \mathcal{I}^\mathcal{I} & f^\mathcal{I} : (\tau_+^\mathcal{I} \times \mathcal{I}^\mathcal{I})^\text{partial} \xrightarrow{\sim} \mathcal{I} \quad \ast g^\mathcal{I} : \mathcal{I}^\mathcal{I}^\text{partial} \xrightarrow{\sim} \mathcal{I}
\end{align*}
\]

Atomic parametric features are interpreted as partial functions; they differ from atomic features for being independent from time. In order to give a meaning to temporal expressions present in generic concept expressions, figure 8 defines the \textit{temporal interpretation function}. The \textit{temporal interpretation function} \( \mathcal{E} \) depends only on the temporal structure \( \mathcal{T} \). The labeled directed graph \( \langle X, \mathcal{E} \rangle \) – where \( X \) is the set of variables representing the nodes, and \( \mathcal{E} \) is the set of temporal constraints representing the arcs – is called \textit{temporal constraint network}. The interpretation of a temporal constraint network is a set of variable assignments which satisfy the temporal constraints. A \textit{variable assignment} is a function \( V : X \mapsto \tau_+^\mathcal{I} \) associating an interval value to a variable temporal. A temporal constraint network is \textit{consistent} if it admits a non empty interpretation. The notation \( \langle X, \mathcal{E} \rangle^\mathcal{E} \{ x_1 \mapsto t_1, x_2 \mapsto t_2, \ldots \} \), used to interpret concept expressions, denotes the subset of \( \langle X, \mathcal{E} \rangle^\mathcal{E} \) where the variable \( x_i \) is mapped to the interval value \( t_i \).

At this point we are able to interpret generic concept expressions. An \textit{interpretation function} \( \mathcal{I}^{\mathcal{E}}_{\mathcal{V}, \mathcal{I}, \mathcal{H}} \) based on a variable assignment \( \mathcal{V} \), an interval \( t \) and a set of constraints \( \mathcal{H} = \{ x_1 \mapsto t_1, \ldots \} \) over the assignments of inner variables, extends the primitive interpretation function in such a way that the equations of the figure 9 are satisfied – we do not report the constructors that can be obtained by complementation. Intuitively, the interpretation of a concept \( C^{\mathcal{E}}_{\mathcal{V}, \mathcal{I}, \mathcal{H}} \) the set of entities of the domain which are of type \( C \) at the time interval \( t \), with the assignment for the free temporal variables in \( C \) given by \( \mathcal{V} \) – see \( (C@X)^\mathcal{I}_{\mathcal{V}, \mathcal{I}, \mathcal{H}} \) – and with the constraints for the assignment of variables in the scope of the outermost temporal quantifiers given by \( \mathcal{H} \). Notice that, \( \mathcal{H} \) interprets the variable renaming due to the temporal substitutive qualifier – see \( (C[Y]@X)^\mathcal{I}_{\mathcal{V}, \mathcal{I}, \mathcal{H}} \) – and it takes effect during the choice of a variable assignment, as the equation \( (\diamond (X)^\mathcal{E}_{\mathcal{I}, \mathcal{I}, \mathcal{H}}, C)^\mathcal{I}_{\mathcal{V}, \mathcal{I}, \mathcal{H}} \).

In absence of free variables in the concept expression – with the exception of \( \# \) – it is introduced as a notational simplification the \textit{natural interpretation function} \( C^{\mathcal{I}}_{\mathcal{I}, \mathcal{H}} \) being equivalent to the interpretation function \( C^{\mathcal{I}}_{\mathcal{V}, \mathcal{I}, \mathcal{H}} \) with any \( \mathcal{V} \) such that \( \mathcal{V}(\#) = t \), and \( \mathcal{H} = \emptyset \). The set of interpretations \( \{ C^{\mathcal{I}}_{\mathcal{V}, \mathcal{I}, \mathcal{H}} \} \) obtained by varying \( \mathcal{I}, \mathcal{V}, t \) with a fixed \( \mathcal{H} \) is maximal wrt set inclusion if \( \mathcal{H} = \emptyset \), i.e., the set of natural interpretations includes any set of interpretations with a fixed \( \mathcal{H} \). In fact, since \( \mathcal{H} \) represents a constraint in the assignment of variables, the unconstrained set is the larger one. Note that, the feature’s interpretation uses just the natural one since it is not admitted to temporally qualify them.

An interpretation \( \mathcal{I} \) is a \textit{model} for a concept \( C \) if \( C^{\mathcal{I}}_{\mathcal{U}} \neq \emptyset \) for some \( t \). If a concept has a model, then it is \textit{satisfiable}, otherwise it is \textit{unsatisfiable}. A concept \( C \) is \textit{subsumed} by a concept
Figure 9: The interpretation function.

$D$ (written $C \subseteq D$) if $C^T \subseteq D^T$ for every interpretation $I$ and every interval $t$. Two concepts $C, D$ are equivalent (written $C \equiv D$) if $C^T = D^T$ for every interpretation $I$ and every interval $t$.

Similar to the case for the logic $\mathcal{HS}$, only the relations $s, f, m_i$ are really necessary (note that, the temporal structure does not allow durationless intervals), because it is possible to express any temporal relationship between two distinct intervals using only these three relations and their transposes $\bar{s}, \bar{f}, m_i$. In fact, the following equivalences hold:

\[
\begin{align*}
\Diamond x (x a \#), \, C@x & \equiv \Diamond xy (y \bar{a} \#)(x \bar{m} \#), \, C@x \\
\Diamond x (x d \#), \, C@x & \equiv \Diamond xy (y s \#)(x \bar{f} \#), \, C@x \\
\Diamond x (x o \#), \, C@x & \equiv \Diamond xy (y s \#)(x \bar{f} \#), \, C@x
\end{align*}
\]

We report here how the authors propose to represent a more complex example of Stack action. A stacking action involves two blocks, which should be both clear at the beginning; the central part of the action consists of holding one block; at the end, the blocks are one on top of the other, and the bottom one is no longer clear (figure 10).

\[
\begin{align*}
\text{Stack} & \equiv \Diamond (xy z v w) (x \bar{f} \#)(y \bar{m} \#)(z \bar{m} \#)(v \bar{f} \#)(w \bar{f} \#)(w \bar{m} \#).
\end{align*}
\]

The definition makes use of temporal qualified concept expressions: the expression $(\bullet \text{OBJECT2} : \text{Clear-Block})@x$ means that the second parameter of the action should be a Clear-Block at the interval denoted by $x$; while $(\bullet \text{OBJECT1} \circ \text{ON} \circ \downarrow \bullet \text{OBJECT2})@y$ indicates that at the interval $y$ the object on which $\bullet \text{OBJECT1}$ is placed is $\bullet \text{OBJECT2}$. The above defined concept does not state which properties are the prerequisites for the stacking action or which properties must be true whenever
Figure 10: Temporal dependencies in the definition of the Stack action.

the action succeeds. What this action intuitively states is that *OBJECT1 will be on *OBJECT2 in a situation where both objects are clear at the start of the action.

Artale and Franconi contribute to explore the decidable realm of interval-based temporal Description Logics by presenting sound, complete and terminating procedures for subsumption reasoning. The key idea in order to obtain decidable languages is the restriction posed on the temporal expressivity by eliminating the universal quantification on temporal variables. The main results are proven starting with the simplest language, $\mathcal{TL}-\mathcal{F}$, where $\mathcal{F}$ is feature language with neither negation nor disjunction and lacking roles, too. Then, the authors show how to reason with more expressive languages such as $\mathcal{TLU}$-$\mathcal{FU}$, which adds disjunction both at the temporal and non-temporal sides of the language, and $\mathcal{TL}$-$\mathcal{ALCF}$. The subsumption procedures are based on a normalization procedure, i.e., an interpretation preserving transformation which operates a separation between the temporal and the non-temporal part of the formalism. A concept in normal form can be seen as a conceptual temporal constraint network, i.e., a labeled directed graph $\langle \bar{X}, \bar{T}, \bar{Q} @ \bar{X} \rangle$ (in $\mathcal{TL}$-$\mathcal{ALCF}$ syntax: $\Diamond(\bar{X}) \bar{T}, (Q^0 \cap Q^1 @ X^1 \cap \ldots \cap Q^n @ X^n)$) where arcs are labeled with a set of arbitrary temporal relationships – representing their disjunction, and temporal nodes are labeled with non-temporal concepts (i.e., each $Q^i$ is an $\mathcal{ALCF}$ concept expression). The subsumption procedure checks whether there is a mapping function between conceptual temporal constraint network (i.e., a form of subgraph isomorphism, called by the authors $s$-mapping) such that a subsumption relation holds both among the non-temporal concepts labeling the corresponding nodes in the mapping function, and among the temporal relations of the corresponding arcs. Then the calculus can adopt standard procedures developed both in the Description Logics community and in the temporal constraints community. Algorithms to compute subsumption between non-temporal concepts are well known and based on a notational variant of the first-order tableaux calculus [Schmidt-Schauss and Smolka, 1991, Hollunder and Nutt, 1990, Donini et al., 1995]. A standard technique for computing subsumption between temporal networks relies on methods for checking the satisfiability of such networks [van Beek and Manchak, 1996]. The following theorem summarizes the main results proved by Artale and Franconi:

**Theorem 4.3** Let $C_1$ and $C_2$ be $\mathcal{TL}$-$\mathcal{F}$ or $\mathcal{TL}$-$\mathcal{ALCF}$ concepts in normal form, then $C_1$ subsumes $C_2$ ($C_2 \sqsubseteq C_1$) if and only if there exists an $s$-mapping from $C_1$ to $C_2$.

Let $C = C_1 \sqcup \ldots \sqcup C_m$ and $D = D_1 \sqcup \ldots \sqcup D_n$ be $\mathcal{TLU}$-$\mathcal{FU}$ concepts in normal form; then $D$ subsumes $C$ if and only if $\forall i \exists j, C_i \sqsubseteq D_j$.

Concept subsumption between $\mathcal{TL}$-$\mathcal{F}$ or $\mathcal{TLU}$-$\mathcal{FU}$ concept expressions in normal form is an NP-complete problem.
5 Point-based Temporal Description Logics

5.1 Combining Description and Tense Logics

Schild [Schild, 1993] combines the Description Logic ALC with point-based modal temporal operators. The new language is called ALC\(T\), and the temporal operators used are those of tense logics [Burgess, 1984] as illustrated in figure 11.

As in the case of Schmiedel, Bettini, Artale and Franconi’s formalisms, the time is part of the semantic structure. A concept denotes a set of pairs of temporal points and individuals \(\langle t, a \rangle\), while a role denotes a triple \(\langle t, a, b \rangle\). The operator existential future denotes those individuals that belong to \(C\) at some time coincident or successive to the reference time. As an example, the concept of Mortal can be defined in ALC\(T\) as:

\[
\text{Mortal} \equiv \text{LivingBeing} \sqcap \Diamond \neg \text{LivingBeing}
\]

which denotes the set of pairs \(\langle t, a \rangle\) where \(a\) is a kind of LivingBeing at the time \(t\), and there exists an instant \(t' \geq t\) where \(a\) is not more a LivingBeing. We observe that, in this example, the time point \(t'\) cannot be coincident with \(t\), since an individual cannot belong to disjoint concepts, as is the case for LivingBeing and \(\neg\)LivingBeing. Thus, a better definition for Mortal makes use of the next instant operator, \(\bigcirc\). Intuitively, given a time \(t\), the concept \(\bigcirc C\) denotes the set of individuals that belong to \(C\) at the point in time that is the immediate successor of \(t\). In order to interpret this operator we need a discrete temporal structure where the immediate successor is definable. The new definition of Mortal is the following:

\[
\text{Mortal} \equiv \text{LivingBeing} \sqcap \bigcirc \Diamond \neg \text{LivingBeing}
\]

The operator universal future, \(\Box\), is the dual of \(\Diamond\). Given a time point \(t\), the concept \(\Box C\) denotes the set of individuals which are of kind \(C\) at every time \(t' \geq t\). With this operator, we can refine the definition of a mortal by saying that whenever he dies he will never be alive again:

\[
\text{Mortal} \equiv \text{LivingBeing} \sqcap \Diamond \Box \neg \text{LivingBeing}
\]

This definition is still incomplete since does not tell anything about the time between \(t\) – when the mortal is alive – and \(t'\) – when a mortal dies. At that time the being can be dead or alive. For this reason the binary operator until, \(\sqcup\), is introduced. At time \(t\), the concept \(C \sqcup D\) denotes all those individuals which are of kind \(D\) at some time \(t' > t\) and which are of kind \(C\) for all time \(t''\) with \(t < t'' < t'\). Thus, we can redefine a mortal as a living being who is alive until he dies:

\[
\text{Mortal} \equiv \text{LivingBeing} \sqcap (\text{LivingBeing} \sqcup \neg \text{LivingBeing})
\]

A slight variant of the until operator is the constructor \(\mathcal{U}\). At time \(t\) the concept \(C \mathcal{U} D\) denotes all those individuals which are of kind \(D\) at some time \(t' \geq t\) and which are of kind \(C\) for every time \(t''\) with \(t \leq t'' < t'\). In the last definition of mortal, \(\sqcup\) can be substituted by \(\mathcal{U}\) without changing its meaning.
\((\bigcirc C)^I = \{ \langle t, a \rangle | \exists t'. t < t' \land \langle t', a \rangle \in C^I \land \neg \exists t''. t < t'' < t' \} \)
\((\bigotimes C)^I = \{ \langle t, a \rangle | t' \leq t \land \langle t', a \rangle \in C^I \} \)
\((\square C)^I = \{ \langle t, a \rangle | \forall t'. t < t' \land \langle t', a \rangle \in C^I \} \)
\((C U D)^I = \{ \langle t, a \rangle | \exists t'. t < t' \land \langle t', a \rangle \in D^I \land \forall t''. t < t'' < t' \rightarrow \langle t'', a \rangle \in C^I \} \)
\((C U D)^I = \{ \langle t, a \rangle | \exists t'. t \leq t' \land \langle t', a \rangle \in D^I \land \forall t''. t \leq t'' < t' \rightarrow \langle t'', a \rangle \in C^I \} \)

Figure 12: \(\text{ALC}T\) semantics.

Although the examples can be formulated with an ordinary tense logic you can imagine to further structure what appears as a propositional constant using Description Logic constructors. More formally, we can define complex temporal concepts using the following syntax.

**Definition 5.1** The tense-logical extension of a concept language \(\mathcal{L}\), called \(\mathcal{L}T\), is the least set containing all concepts of \(\mathcal{L}\) such that \(\bigcirc C, \bigotimes C, C U D, C U D\) are concepts of \(\mathcal{L}T\) if \(C\) and \(D\) are concepts of \(\mathcal{L}\).

Note that, as far as we drop the object facet we get a pure propositional tense logic. The figure 12 shows the model-theoretic semantics for the temporal constructs in \(\text{ALC}T\).

Schild deeply analyses the computational property of the temporal language \(\text{ALC}T\). He proves that reasoning in \(\text{ALC}T\) is of the same complexity class that reasoning in \(\text{ALC}\), when interpreted over linear, unbounded and discrete time structures like the natural numbers. Then, reasoning in \(\text{ALC}T(\mathbb{N})\), which denotes the language \(\text{ALC}T\) interpreted only considering the natural numbers as temporal structure, is a PSPACE-complete problem. These important results show that adding a point-based time dimension to \(\text{ALC}\) does not alter its computational behavior. However, since for branching, discrete and unbounded time reasoning in classical tense logic is an EXPTIME-hard problem then the same lower complexity bound carries over \(\text{ALC}T\). Schild also proposes a sound and complete tableau-based decision procedure for \(\text{ALC}T(\mathbb{N})\) by coupling tableau techniques for linear tense logic [Wolper, 1985] to that for \(\text{ALC}\). When \(\text{ALC}\) is extended with an interval-based time dimension (let us call it \(\text{ALC-INT}\)) the undecidability results showed by Halpern and Shoham for the logic \(\mathcal{HS}\), and the one presented by Bettini apply also for \(\text{ALC-INT}\). Interesting open problems remain. One concerns the complexity of \(\text{ALC}T(\mathbb{N})\) when extended with past tense operators. It is also unknown whether \(\text{ALC}T\) is still decidable when interpreted on the structure of real numbers.

### 5.2 Description Logics with Modal Operators

Starting from the correspondence between description and modal logics many recent works investigate various way of combining modal operators within a description language. As observed by Wolter and Zakharyaschev [Wolter and Zakharyaschev, 1998], different design choices have to be taken concerning the integration of modal operators. First of all, modal operators can be applied in different places. They can be used not only to form new concept terms but, as proposed by Baader and Laux [Baader and Laux, 1995], also in front of concept definitions and assertions. They following example shows the notion of Happy-father, where different modalities interact:

\[
\text{[BEL-JOH]}(\text{Happy-father} \triangleq \exists \text{MARRIED-TO} , (\text{Woman} \land \text{[BEL-JOH]} \text{Pretty}) \land \text{future}) \lor \text{CHILD.Graduate})
\]
In this case, it is in the John's belief that a Happy-father is someone married to a woman believed to be pretty by John, and whose children will be graduate sometime in the feature.

Baader and Olhbach [Baader and Olhbach, 1995] study the case where modalities can also modify roles. For example, we can build a complex role [always] loves which restrict the role loves to the pairs \( (x, y) \) which are in the loves relation in every future time (in some sense, this combination uncouple the intepretation of a role from the temporal dimension).

Another parameter concerns the different dimensions we want to model. If, for example, we are interested in modeling both temporal and belief aspects the next choice is the number of modalities which can be used in usual box and diamond operators. Thus, in the temporal dimension we could have future and past modalities as well as one for the next instant, while in the dimension of belief we could be interested in both belief-John and belief-Mary.

If we consider the case where modal operators can be applied to concepts and to both terminological and assertional axioms, but not to roles we obtain the language \( \mathcal{ALC}_M \) [Baader and Laux, 1995, Wolter and Zakharyaschev, 1998].

**Definition 5.2** \( \mathcal{ALC}_M \) concepts are defined inductively as follows: All concept names as well as \( \bot \) and \( \top \) are concepts. If \( C \) and \( D \) are concepts, \( R \) is a role name and \( m \) is a modality then \( C \cap D, \neg C, \exists R, C, \langle m \rangle C \) are concepts.

Let \( C \) and \( D \) be concepts, \( R \) a role name, \( a, b \) object names. Then expressions of the form \( C \equiv D, a R b, a : C \) are (atomic) formulae. If \( \phi \) and \( \psi \) are formulae then so are \( \langle m \rangle \phi, \neg \phi, \phi \land \psi \).

The semantics has a Kripke-style: each modal operator is interpreted as an accessibility relation on a set of possible worlds. Thus we need a set of possible worlds \( D_i \) for each dimension \( i \) (in the case of time and believe we would need a temporal structure and a belief world as possible worlds). To each world correspond a classical \( \mathcal{ALC} \) interpretation structure where concepts and roles are interpreted.

**Definition 5.3** An interpretation of \( \mathcal{ALC}_M \) consists of a Kripke structure \( \langle \mathcal{W}, \Gamma, \mathcal{I} \rangle \) such that:
- \( \mathcal{W} \), the set of possible worlds, is the cartesian product of non-empty domains \( D_1, \ldots, D_n \), one for each dimension; \( \Gamma \) contains for each modality \( m \) of dimension \( i \) an accessibility relation \( \gamma_m \), which is a function \( \gamma_m : \mathcal{W} \rightarrow 2^{D_i} \) (whenever \( d^i \in \gamma_m(d_1, \ldots, d_i, \ldots, d_n) \) we will write \( ((d_1, \ldots, d_i, \ldots, d_n), (d_1, \ldots, d'_i, \ldots, d_n)) \in \gamma_m \)); \( \mathcal{I} \) is a function associating to each world \( w \in \mathcal{W} \) an interpretation structure \( \langle \Delta \mathcal{I}(w), \mathcal{I}(w) \rangle \) which consists of a non-empty set of objects \( \Delta \mathcal{I}(w) \),
- and of an interpretation function \( \mathcal{I}(w) \) that associates: for each object name \( a \), an element \( a \mathcal{I}(w) \in \Delta \mathcal{I}(w) \) such that \( a \mathcal{I}(w) = a \mathcal{I}(w) \) for any \( w, v \in \mathcal{W} \) (i.e., the interpretation of individuals does not depend on the actual world, this is called the rigid designator hypothesis), for each concept name \( A \) and world \( w \in \mathcal{W} \) a set \( A \mathcal{I}(w) \subseteq \Delta \mathcal{I}(w) \), for the \( \bot, \top \) concepts the sets \( \top \mathcal{I}(w) = \Delta \mathcal{I}(w) \) and \( \bot \mathcal{I}(w) = \emptyset \), for each role name \( R \) and world \( w \in \mathcal{W} \) a binary relation \( R \mathcal{I}(w) \subseteq \Delta \mathcal{I}(w) \times \Delta \mathcal{I}(w) \). Furthermore, the interpretation is extended to generic concepts as follows:

\[
\begin{align*}
(C \cap D) \mathcal{I}(w) &= C \mathcal{I}(w) \cap D \mathcal{I}(w) \\
(\neg C) \mathcal{I}(w) &= \Delta \mathcal{I}(w) \setminus C \mathcal{I}(w) \\
(\exists R, C) \mathcal{I}(w) &= \{ a \in \Delta \mathcal{I}(w) \mid \exists b. (a, b) \in R \mathcal{I}(w) \land b \in C \mathcal{I}(w) \} \\
(\langle m \rangle C) \mathcal{I}(w) &= \{ a \in \Delta \mathcal{I}(w) \mid \exists v. (w, v) \in \gamma_m \land a \in \mathcal{I}(v) \}.
\end{align*}
\]

The choice of varying the object domain, each one depending on a given world, capture, for example, the case of different definitions for the same concept – like \( \text{[BEL-JOH]} \langle A \doteq B \rangle \) and
Definition 5.4 A formula $\phi$ is satisfiable in a Kripke structure $\mathcal{K} = \langle W, \Gamma, \mathcal{I} \rangle$ and a world $w \in W$ if $\mathcal{K}, w \models \phi$, where $\models$ is defined inductively by:

- $\mathcal{K}, w \models C = D$ if $C^\mathcal{I}(w) = D^\mathcal{I}(w)$
- $\mathcal{K}, w \models a : C$ if $a^\mathcal{I}(w) \in C^\mathcal{I}(w)$
- $\mathcal{K}, w \models aRb$ if $(a^\mathcal{I}(w), b^\mathcal{I}(w)) \in R^\mathcal{I}(w)$
- $\mathcal{K}, w \models \langle m \rangle \phi$ if $\exists v. (w, v) \in \gamma_m \land \mathcal{K}, v \models \phi$
- $\mathcal{K}, w \models \phi \land \psi$ if $\mathcal{K}, w \models \phi \land \mathcal{K}, w \models \psi$
- $\mathcal{K}, w \not\models \phi$ if $\mathcal{K}, w \not\models \phi$.

The classical problems of concept satisfiability and concept subsumption can be reduced to formula satisfiability if we consider that $C$ is satisfiable if there exists $\mathcal{K}$ and $w$ such that $\mathcal{K}, w \models \neg(C = \bot)$, and that $C \subseteq D$ if $C \cap \neg D$ is unsatisfiable.

As we observed, interpretation of $\mathcal{ALC}_M$ are Kripke structures where worlds can be understood as $\mathcal{ALC}$ interpretations. In particular, different object domains $\Delta^\mathcal{I}(w)$ are introduced for each world $w$. Three different cases are possible, we may assume worlds to have arbitrary domains (varying domain assumption), or an inclusion relation holds between domains of accessible worlds (expanding domain assumption), or all domains are invariant w.r.t. worlds (constant domain assumption).

Baader and Lax propose a complete and terminating algorithm, based on tableau calculus, for testing satisfiability of $\mathcal{ALC}_M$ formulae under the expanding domain assumption. The main limitation is that all the modal operators do not satisfy any specific axioms for belief or time (i.e., the modalities are interpreted in the basic logic $K$). The work of Wolter and Zakharyaschev proves the decidability of satisfiability of $\mathcal{ALC}_M$ formulae when the accessibility relations satisfy the most common conditions for the belief and temporal operators (i.e., when the modalities give rise to the modal systems $K, S4,S4.3, S5, KD45, GL, GL.3$ and the tense logic over discrete linear and unbounded temporal structures like $\langle N, \leq \rangle$). They start by considering mono-dimensional description languages and then prove a general transfer theorem for deciding satisfiability in the multi-dimensional case. Furthermore, they prove decidability of these logics under the constant domain assumption showing that both the varying and the expanding domain assumptions are reducible to it. Finally, Wolter and Zakharyaschev investigate the model properties of the different logics. They show how for logics based on linear temporal models (i.e., $\langle N, \leq \rangle$) or models whose accessibility relations are transitive and reflexive ($S4, S4.3$) the Finite Model Property does not hold anymore.

The framework presented in this section in its generality gives us a very useful tool to compare the expressivity of the DL extensions presented until now. Let us consider, as an example, the language proposed by Schild (see section 5.1). We have one dimension (i.e., the temporal one) with two modalities $(\mathcal{U}, \mathcal{U})$ applicable only in front of concepts. Each model is based on a branching (linear) discrete unbounded temporal structure under the constant domain assumptions and rigid designators. Analogous choices are the basis of the works presented in Sections 4 whenever you consider different temporal structures with interval-based modal operators, and dropping variables in the case of Schmiedel, and Artale and Franconi.

---

5This latter result is exactly corresponding to the proposal by Schild presented in Section 5.1; see next paragraph.
6 Description Logics with Temporal Parts

6.1 The T-Rex System

Weida and Litman [Weida and Litman, 1992, Weida and Litman, 1994] propose T-Rex, a loose hybrid integration between Description Logics and constraint networks with the aim of representing and reasoning about plans. Plans are defined as collections of steps (i.e., actions) together with temporal constraints between their duration. Each step is associated with an action type, represented by a generic concept in K-REP – a non-temporal Description Logics [Mays et al., 1991]. Thus, a plan is seen as a plan network, a temporal constraint network in the style of Allen [Allen, 1991], whose nodes, labeled with action types, correspond to time intervals and are associated with the steps of the plan itself. As an example of plan in T-Rex we show the plan of preparing spaghetti marinara:

(defplan Assemble-Spaghetti-Marinara
  ((step1 Boil-Spaghetti)
   (step2 Make-Marinara)
   (step3 Put-Together-SM))
  ((step1 (before meets) step3)
   (step2 (before meets) step3)))

This is a plan composed by three actions, i.e., boiling spaghetti, preparing marinara sauce and assembling all things at the end. Temporal constraints between the steps establish the temporal order in doing the corresponding actions. In this sense, T-Rex can be classified as a system with an internal representation of time: a plan is a collection of temporal parts possibly holding at different times. The notion of Mortal can be expressed in this framework as:

(defplan Mortal
  ((alive ALIVE)
   (dead DEAD))
  ((alive (before) dead)))

It is worth noting that no formal semantics was provided for T-Rex. For a better understanding of the consequences that an internal representation framework may have, the next section should be enlightening.

A structural plan subsumption is defined, characterized in terms of graph matching, and based on two separate notions of subsumption: terminological subsumption between action types labeling the nodes and temporal subsumption between interval relationships labeling the arcs. The main application of T-Rex is plan recognition. An individual plan is a network where nodes are individual actions while arcs are labeled with temporal relations. The following is an example of an individual plan of type Assemble-Spaghetti-Marinara:

Boil17 before MakeMarinara1 before PutTogetherSM27

The plan library is used to guide plan recognition in a way similar to that proposed in [Kautz, 1991]. According to these ideas, a Closed World Assumption (CWA) is made, assuming that the plan library is complete and an observed plan will be fully accounted for by a single plan. The plan recognition process partitions the plan library into the modalities possible, necessary and impossible. These modalities describe the status of each plan with respect to a single observation. A plan is said possible if it might eventually subsume the observation, also in case of further refinements of the observation itself. A possible plan which actually subsumes an observation is also necessary, possible but not necessary plans are called optional plans. When a plan cannot
subsume the given observation it is an impossible plan. Before any observation is made, all plans are optional except for the plan root, which is obviously necessary.

6.2 Time as Concrete Domain

In the concrete domain extension of the description logic, abstract individuals (i.e., elements of the domain $\Delta^I$) can now be related to values in a concrete domain (e.g., the integers or strings) via features (i.e., functional roles). Then, tuples of concrete values identified by such features can be constrained to satisfy an $n$-ary predicate over the concrete domain. The first work in this direction is the one of Baader and Hanschke [Baader and Hanschke, 1991]. For example, if we choose the natural numbers with the usual total ordering relation “$\geq$” as a concrete domain, it is possible to express the concept describing managers that, every month, spend more money than they earn by the term $(\text{Manager} \land \forall \text{MONTH}. \exists (\text{INCOME}, \text{EXPENSES}) \leq )$.

More formally, [Baader and Hanschke, 1991] propose an extension to $\mathcal{ALC}$, i.e., the so called $\mathcal{ALC}(D)$, where $D$ stands for the concrete domain. A concrete domain is formally a pair $D = (\text{dom}(D), \text{pred}(D))$ that consists of a set $\text{dom}(D)$ (the domain), and a set of predicate symbols $\text{pred}(D)$. Each predicate symbol $P \in \text{pred}(D)$ is associated with an arity $n$ and an $n$-ary relation $P^n \subseteq \text{dom}(D)^n$.

The syntax of the description logic is augmented with the following rule:

$$C, D \rightarrow \exists (u_1, \ldots, u_n).P \quad \text{(concrete predicate)}$$

where $P \in \text{pred}(D)$ is an $n$-ary predicate name, and $u_1, \ldots, u_n$ are feature chains. The semantics of the new operator is the following:

$$(\exists (u_1, \ldots, u_n).P)^I = \{a \in \Delta^I \mid (u_1^I(a), \ldots, u_n^I(a)) \in P^D\}$$

where the interpretation function $^I$ is extended to map every feature name $p$ to a partial function $p^I : \Delta^I \rightarrow \Delta^I \cup \text{dom}(D)$. Thus, we can review the previous example of the concept defining the managers that spend each month more money than they earn: $(\text{Manager} \land \forall \text{MONTH}. \exists (\text{INCOME}, \text{EXPENSES}) \leq )$

where “$\leq$” is a binary predicate symbol, and $\text{INCOME, EXPENSES}$ are features mapping individuals of the abstract domain $\Delta^I$ — i.e., the managers — into elements of the numeric domain $\text{dom}(D)$ — i.e., their income and expenses.

In [Baader and Hanschke, 1991], concrete domains are restricted to so-called admissible concrete domains in order to keep the inference problems of this extension decidable. We recall that, roughly speaking, a concrete domain $D$ is called admissible if (1) $\text{pred}(D)$ is closed under negation and contains a unary predicate name $\top$ for $\text{dom}(D)$, and (2) satisfiability of finite conjunctions over $\text{pred}(D)$ is decidable. Given an admissible concrete domain, a sound, complete and terminating resolution technique for checking subsumption, satisfiability and knowledge base consistency can be devised. In fact, condition (1) says that $\text{pred}(D)$ is a complete language on $D$ allowing for the reduction of the reasoning services to the problem of checking for knowledge base consistency. The second condition is crucial for the decidability of the reasoning procedure since, as a subtask, it will have to decide satisfiability of conjunctions of the form $\bigwedge_{i=1}^k P_i(x^{(i)})$ in the concrete domain.

In this framework, the concept of Mortal can be defined as follows with respect to a concrete domain composed by temporal intervals and the Allen’s predicates:

$\text{Mortal} \equiv \exists (\text{ALIVE} \circ \text{HAS-TIME}, \text{DEAD} \circ \text{HAS-TIME}).\text{before}$
i.e., a mortal is any individual having the property of being alive at some temporal interval before
some other temporal interval at which the same individual has the property of being dead.

It is important to emphasize a major difference of this approach from the previously analyzed
approaches. The goal of a temporal description logic should be to provide a logical framework
for describing properties of objects which may vary in time: for example, a mortal is described
by means of the time-dependent properties “being alive” and “being dead”. In the approaches
described so far, a property is represented by means of a (complex) concept expression, which is
interpreted as the class of objects having that (complex) property at the time of evaluation: that
is, the time is embedded in the semantics. In the concrete domain approach, just like in the T-Rex
system, the representation of time is lifted up to the language level – i.e., there is an internal
representation of time – and properties become explicitly binary functional relations (features)
between abstract objects having the property and abstract objects corresponding to temporal
intervals. In the context of description logics an immediate consequence of this difference is that
the language for defining properties is much richer if properties are represented by means of
concepts rather than roles. In fact, in the previous example of Mortal expressed in a description
logic with concrete domains, it is even impossible to define directly that the role ALIVE is the
complement of the role DEAD, i.e., if an object is not alive at a given interval then it should be
dead and vice versa.

We introduce now an example which shows that the \(\mathcal{ALC(D)}\) description logic is more suitable
to describe properties of temporal objects (e.g., intervals) rather than properties of objects varying
in time (like in the Mortal example). In their paper [Baader and Hanschke, 1991] define the Allen’s
interval relations using the set of real numbers \(\mathbb{R}\) together with the predicates \(\leq, \leq, \geq, \geq, \neq\)
as the concrete admissible domain. The Interval concept as an ordered pair of real numbers
represented by referring to the concrete predicate \(\leq\) applied to the features LEFT-HAS-TIME and
RIGHT-HAS-TIME:

\[
\text{Interval} \triangleq \exists(\text{LEFT-HAS-TIME}, \text{RIGHT-HAS-TIME}). \leq
\]

Allen’s relations are binary relations on two intervals and are represented by the Pair concept
which uses the features FIRST and SECOND:

\[
\text{Pair} \triangleq \exists\text{FIRST. Interval} \sqcap \exists\text{SECOND. Interval}
\]

Now Allen’s relation can be easily defined as concepts:

\[
\text{C-Equals} \triangleq \text{Pair} \sqcap \exists(\text{FIRST} \circ \text{LEFT-HAS-TIME}, \text{SECOND} \circ \text{LEFT-HAS-TIME}). = \\
\quad \sqcap \exists(\text{FIRST} \circ \text{RIGHT-HAS-TIME}, \text{SECOND} \circ \text{RIGHT-HAS-TIME}). = \\
\text{C-Before} \triangleq \text{Pair} \sqcap \exists(\text{FIRST} \circ \text{RIGHT-HAS-TIME}, \text{SECOND} \circ \text{LEFT-HAS-TIME}). \leq \\
\text{C-Meets} \triangleq \text{Pair} \sqcap \exists(\text{FIRST} \circ \text{RIGHT-HAS-TIME}, \text{SECOND} \circ \text{LEFT-HAS-TIME}). = \\
\quad \ldots
\]

An extension to the language \(\mathcal{ALC(D)}\) was studied by Haarslev, Lutz and Möller [Haarslev et al., 1998].
The new language, called \(\mathcal{ALCPR(D)}\), allows to define roles based on properties between concrete
objects (\(\mathcal{RP}\) stands for Role definition based on Predicates). The new role-forming operator has
the following syntax:

\[
\exists(u_1, \ldots, u_n)(v_1, \ldots, v_m).P \quad \text{\(\text{role forming predicate restriction}\)}
\]

where \(u_1, \ldots, u_n, v_1, \ldots, v_m\) are feature chains, and \(P \in \text{pred}(D)\) with arity \(n + m\). The interpre-
tation function has to be extended in order to satisfy the following equation:
\[(\exists (u_1, \ldots, u_n)(v_1, \ldots, v_m).P)^I = \{(a, b) \in \Delta_I \times \Delta_I \mid (u^I_1(a), \ldots, u^I_n(a), v^I_1(b), \ldots, v^I_m(b)) \in P^D\}\]

Given an abstract object say \(x\), it is possible to refer to all those objects whose concrete facet relates with the concrete facet of the starting object \(x\) by some specific concrete predicate. In this way we can quantify over these roles using both the existential and the universal quantifier on roles.

As a simple example, we can define the **BEFORE** role as being the counterpart of the concrete predicate **before** in the abstract domain, and use it for defining a new concept **NoBefore**, as the class of objects which do not have any BEFORE-related object:

**BEFORE** \(=\) \(\exists\)\(\text{(HAS-TIME)(HAS-TIME).before}\)

**NoBefore** \(=\) \(\forall\text{BEFORE.\bot}\)

It is important to point out the difference with the similar definition which can be done using a description logic in correspondence with the \(\mathcal{HS}\) logic:

**NoBefore\(_{\mathcal{HS}}\)** \(=\) \([\text{before}]\ \bot\)

In fact, while the concept **NoBefore** is satisfiable, denoting all the objects of the abstract domain having no BEFORE-related objects, the concept **NoBefore\(_{\mathcal{HS}}\)** is clearly unsatisfiable, if we choose, for example, \(\Re\) as the underlying temporal structure. The reason is that in the concrete domain approach the quantification concerns only the abstract domain and not the concrete one, i.e., we can only quantify over the abstract objects, which may possibly have a specific temporal facet lifted up from the concrete domain. On the other hand, in \(\mathcal{HS}\) every object is always implicitly qualified at some element of the temporal domain.

As far as the computational properties of \(\mathcal{ALCHR}(D)\) are concerned, it was proven the undecidability of reasoning in the full language. However, the authors propose a restricted language for which a sound, complete and terminating reasoning procedure (based on tableaux calculus) is presented. The problems are due to the interaction of complex roles with exists and value restrictions. The criteria adopted to reduce the language expressivity forbid any alternating combination of exists and value restrictions concerning complex roles, and predicate exists restrictions appearing inside restrictions of complex roles (i) can quantify only over primitive features (ii) cannot be used as restrictions of some exists or value restriction present inside the actual complex role.

Finally we should mention the flexibility of these approaches in order to represent different concrete domains. Every concrete domain can be embedded in these logics as far as they are admissible. Indeed, the expressiveness of \(\mathcal{ALCHR}(D)\) is showed both over the Allen’s algebra [Lutz et al., 1997], and also using a spatial concrete domain [Haarslev et al., 1998] where the concrete predicates are the binary spatial relations of the RCC-8 theory [Randell et al., 1992].

## 7 State-change based Description Logics

In this section we illustrate the approaches were the temporal dimension is only implicit in the language. Both languages presented below describe essentially sets of objects linearly ordered. Time has no first-class citizenship in these representation languages.

### 7.1 The **CLASP** System

Devanbu and Litman [Devanbu and Litman, 1991, Devanbu and Litman, 1996] describe the **CLASP** system (**CLASSification of scenarios and Plans**), a plan-based KRS extending the notion of sub-
umption and classification to plans, in order to build an efficient information retrieval system. CLASP was used to represent plan-like knowledge in the domain of telephone switching software by extending the use of the software information system LASSIE [Devanbu et al., 1991]. CLASP is designed for representing and reasoning about large collections of plan descriptions, using a language able to express ordering, conditional and looping operators. Following the STRIPS tradition, plan descriptions are built starting from states and actions, both represented by using the CLASSIC [Brachman et al., 1991] Description Logic. The simplest action is represented by the primitive CLASSIC concept Action:

(DEFINE-CONCEPT Action
  (PRIMITIVE (AND Classic-Thing
      (AT-LEAST 1 ACTOR)
      (ALL ACTOR Agent)
      (EXACTLY 1 PRECONDITION)
      (ALL PRECONDITION State)
      (EXACTLY 1 ADD-LIST)
      (ALL ADD-LIST State)
      (EXACTLY 1 DELETE-LIST)
      (ALL DELETE-LIST State)
      (EXACTLY 1 GOAL)
      (ALL GOAL State))))

which constrains every kind of action to have at least one ACTOR, all of whose ACTORs are of type Agent, whose PRECONDITION is of type State, whose ADD-LIST is of type State, whose DELETE-LIST is of type State, and whose GOAL is of type State. Note that EXACTLY is an operator introduced by the authors which simulate the conjunction of AT-LEAST and AT-MOST constructs.

State descriptions are restricted to a simple conjunction of primitive CLASSIC concepts. Furthermore, the concept State is a predefined primitive CLASSIC concept specializing Classic-Thing:

(DEFINE-CONCEPT State
  (PRIMITIVE Classic-Thing))

Actions and states can be restricted to define more appropriate specializations. For example, the following is the definition for a System-Act:

(DEFINE-CONCEPT System-Act
  (AND Action
      (ALL ACTOR System-Agent)))

which fully defines System-Act as the subconcept of Action where the fillers of the role ACTOR are restricted to belong to the concept System-Agent.

A plan is a conceptual description which uses the roles PLAN-EXPRESSION, INITIAL and GOAL. While INITIAL and GOAL roles can only be restricted with CLASSIC concepts, the PLAN-EXPRESSION role are restricted to plans concept expressions which are compositionally built from CLASSIC actions and states concepts using the operators SEQUENCE, LOOP, REPEAT, TEST, OR and SUBPLAN, as showed by the following syntax rules:
\(<\ plan-expression>\) ::= \(<\ action-concept>\)
\(|\) (SEQUENCE \(<\ plan-expression>\) +)
\(|\) (LOOP \(<\ plan-expression>\))
\(|\) (REPEAT \(<\ integer>\) \(<\ plan-expression>\))
\(|\) (TEST \(<\ state-concept>\) \(<\ plan-expression>\) +)
\(|\) (OR \(<\ plan-expression>\) +)
\(|\) (SUBPLAN \(<\ symbol>\))

where \(<\ action-concept>\) and \(<\ state-concept>\) refer to CLASSIC concepts subsumed by the concepts Action and State. The intuitive meaning of the CLASP constructs is clarified by the following examples:

- (SEQUENCE A B C): An action of type A is followed by an action of type B, which is followed by an action of type C.

- (LOOP A): Zero or more repetitions of actions of type A.

- (REPEAT 7 A): Equivalent to (SEQUENCE A A A A A A A).

- (TEST (S1 A) (S2 B)): If the current state is of type S1, then action type A, else if state type S2, then action type B.

- (OR A B): Either action type A or type B.

- (SUBPLAN Plan-Name): Insert the Plan-Name’s definition in the current plan-expression.

The root of the plan taxonomy is the following CLASP concept Plan:

\(\) (DEFINE-PLAN Plan
        (PRIMITIVE (AND Clasp-Thing
                      (EXACTLY 1 INITIAL)
                      (ALL INITIAL State)
                      (EXACTLY 1 GOAL)
                      (ALL GOAL State)
                      (EXACTLY 1 PLAN-EXPRESSION)
                      (ALL PLAN-EXPRESSION (LOOP Action)))))

More specific plans are built by refining the roles PLAN-EXPRESSION, INITIAL and GOAL. The example domain in which CLASP is tested is that one of telephone switching software. Here we show the plan called Pots-Plan:

\(\) (DEFINE-PLAN Pots-Plan
        (AND Plan
            (ALL PLAN-EXPRESSION
             (SEQUENCE (SUBPLAN Originate-And-Dial-Plan)
                        (TEST (Callee-On-Hook-State
                               (SUBPLAN Terminate-Plan))
                               (Callee-Off-Hook-State
                                (SEQUENCE
                                 Non-Terminate-Act
                                 Caller-On-Hook-Act
                                 Disconnect-Act))))))
which makes use of the previously defined plans `Originate-And-Dial-Plan` and `Terminate-Plan`. Intuitively, the plan `Pots-Plan` describes a situation in which the caller picks up a phone, gets a dial tone, and dials a callee. If the callee’s phone is on-hook (TEST on `Callee-On-Hook-State`), the call goes through; if the callee’s phone is off-hook (TEST on `Callee-Off-Hook-State`), the caller gets a busy signal, hangs up, and is disconnected.

`CLASP` gives also the possibility to describe individual plans, called scenario. Every scenario corresponds to an initial state, a final state and a sequence of individual actions. A scenario is well-formed if the given sequence of individual actions will indeed transform the specified initial state into the goal state. Whenever a scenario is created, `CLASP` checks it for well-formedness. During this process any unspecified intermediate state is inferred, and any partially specified intermediate state is completed by using STRIPS-like rules.

The temporal expressivity of `CLASP` is implicit in the representation language provided for building plans. In this language you can essentially express a sequential temporal order of actions, where each action is instantaneous. Plans definitions can use disjunction to represent alternative ways of accomplishing a given plan, and iterations constructors which allow to abstractly describe repetitions of the same action.

The authors provided `CLASP` both with a plan type subsumption algorithm, and with a plan recognition algorithm that verifies whether a scenario fulfill the conditions to belong to a given plan type. The key idea in developing both algorithms is the observation that plans concept expressions correspond to regular expressions. `CLASP` is able to transform each plan expression into a finite state automaton. This correspondence allows the authors to develop algorithms for subsumption and recognition by integrating work in automata theory with work in concept subsumption and recognition. More formally, a plan description, \( P \), subsumes another plan description, \( Q \), if there is a subsumption relation between the expressions restricting the roles `INITIAL`, `GOAL` and `PLAN-EXPRESSION`. Since `INITIAL` and `GOAL` are restricted with `CLASSIC` concepts the well known algorithm for computing terminological subsumption can be adopted. For plan expressions the authors recover to the correspondence to finite state automata by reducing the problem of plan expressions subsumption to finite automata (or regular language) subsumption. The case of plan recognition algorithm is similar. A scenario \( s \) satisfies a plan \( \bar{P} \) if the actual fillers of `INITIAL` and `GOAL` satisfy the `CLASSIC` concept restrictions of the respective roles in \( \bar{P} \), and the plan expression of \( s \) is a string in the language defined by the abstract plan expression of \( \bar{P} \).

### 7.2 The RAT System

Heinsohn, Kudenko, Nebel and Profitlich [Heinsohn et al., 1992] describe the RAT system, used in the Wir project at the German Research Center for AI (DFKI). They use Description Logics to represent both the world states and the atomic actions. A second formalism is added to compose actions in plans and to reason about simple temporal relationships. RAT actions are defined by the change of the world state they cause, and they are instantaneous as in the STRIPS-like systems, while plans are linear sequences of actions. Thus, as for `CLASP`, explicit temporal constraints are not expressible in the language.

Formally, an action is defined as a triple of parameters, pre-conditions and post-conditions, \( \langle \text{pars}, \text{pre}, \text{post} \rangle \). As an example, consider:

\[
\text{PutCupUnderWaterOutlet} \triangleq \\
\langle \text{agent : Person} \land \text{object : Cup} \land \text{machine : EspressoMachine}, \\
\text{(object o position} \downarrow \text{agent o has-hand o inside-region)}, \\
\text{(object o position} \downarrow \text{machine o has-water-outlet o under-region}) \rangle
\]

where `agent`, `object` and `machine` are the formal parameters of the action; the pre-conditions
state that the Cup is held by the agent's hand; the post-conditions state that the Cup is located under the water-outlet.

Actions can be composed to build plans schemata. A plan schema is a triple of action parameters, sequence of actions, and equality constraints over the action parameters, \( \langle par_s, seq, cons \rangle \):

\begin{align*}
\text{MakeEspresso} & = \\
& \langle \text{agent : Person} \cap \text{object1 : Cup} \cap \text{object2 : EspressoMachine}, \\
& (\ldots, \\
& (A5 : \text{PutCupUnderWaterOutlet}), \\
& (A6 : \text{TurnSwitchToEspresso}), \\
& \ldots), \\
& \text{object2} \downarrow A5 \circ \text{machine} \cap \text{object2} \downarrow A6 \circ \text{machine} \rangle
\end{align*}

The plan MakeEspresso has one agent and two objects as parameters. Each subaction – or subplan – in the sequence is prefixed by a label that will be used as an index in the constraint part which, in this case, states that the EspressoMachine, the machine of the action PutCupUnderWaterOutlet and the machine of the action TurnSwitchToEspresso are the same object.

As showed, the state representation in RAT uses the feature construct – i.e., a functional role – to describe the parameters of an action. Furthermore, the possibility to express equality constraints between path of features is a powerful mechanism in order to bind action parameters. Thus, whether the language for the state representation is more expressive than the one used in CLASP, it should be noted that the language for composing plans is much richer in CLASP than in RAT, which only allows sequences.

The most important reasoning services offered by RAT are the simulated execution of part of a plan, and checking if a given plan is feasible and, if so, computing the global pre- and post-conditions. The feasibility test is similar to the usual consistency check for a concept description: they temporally project the pre- and post-conditions of individual actions composing the plan, respectively backwards and forward. This procedure differs from the same service offered in CLASP due to the richer expressiveness of state descriptions. If the feasibility test does not lead to an inconsistent initial, final or intermediate state, the plan is feasible and the global pre- and post-conditions are determined as a side effect.

References


