A Correspondence between Temporal Description Logics

Alessandro Artale  
ITC-IRST  
Povo (TN), I  
artale@irst.itc.it

Carsten Lutz  
LuFG Theoretical Computer Science  
RWTH Aachen, DE  
clu@cantor.informatik.rwth-aachen.de

1 Introduction

Description Logics (DLs) are formalisms for representing and reasoning about conceptual knowledge. There exist several extensions of DLs for an appropriate integration of temporal knowledge [4]. This paper investigates the relation between the two DLs $\mathcal{TLC-ALCF}$ [2, 3] and $\mathcal{ALCF(D)}$ [10, 8]. $\mathcal{TLC-ALCF}$ is an interval-based, temporal DL for reasoning about objects whose properties vary over time. $\mathcal{ALCF(D)}$ is a logic for integrated reasoning about conceptual and so-called concrete knowledge. If instantiated with a “temporal” concrete domain, $\mathcal{ALCF(D)}$ is well-suited for reasoning about temporal objects, i.e., objects which have a unique temporal extension.

This paper is a first attempt to clarify the relationship between these two formalisms. It is shown that satisfiability of $\mathcal{TLC-ALCF}$ concepts can be reduced to satisfiability of $\mathcal{ALCF(D)}$ concepts. This allows to use the available $\mathcal{ALCF(D)}$ tableau calculus for reasoning with $\mathcal{TLC-ALCF}$. Furthermore, it allows to settle the complexity of satisfiability of $\mathcal{TLC-ALCF}$ concepts, which was previously unknown.

The paper is organized as follows. Sections 2 and 3 introduce the syntax and semantics of the two temporal DLs. In Section 4, a normal form for $\mathcal{TLC-ALCF}$ concepts is introduced. Based on this normal form, the reduction of satisfiability of $\mathcal{TLC-ALCF}$ concepts to satisfiability of $\mathcal{ALCF(D)}$ concepts is given.

2 The Logic $\mathcal{TLC-ALCF}$

The language $\mathcal{TLC-ALCF}$ [3] is composed of the interval-based temporal logic $\mathcal{TL}$ and the non-temporal description logic $\mathcal{ALCF}$. The logic $\mathcal{TL}$ is able to represent temporal constraint networks based on Allen’s relations, and to relate $\mathcal{ALCF}$ concept expressions with time intervals in these networks. $\mathcal{TLC-ALCF}$ concepts (denoted by $C, D$) are built following the syntax rules in Figure 1.

![Figure 1: Syntax rules for $\mathcal{TLC-ALCF}$](image)

$p$ for paths, and $R$ for roles (all possibly with index). The $*$ symbol is not intended as an operator, but only used to distinguish parametric from non-parametric features. For the basic temporal interval relations, Allen’s notation [1] is used: before (b), meets (m), during (d), overlaps (o), starts (s), finishes (f), equal (=), after (a), met-by (mi), contains (ci), overlapped-by (ob), started-by (gb), and finished-by (fb). Temporal variables are introduced by the temporal existential quantifier “$\exists$”. The special temporal variable $\iota$, usually called NOW, is intended as the reference interval.

$\mathcal{TLC-ALCF}$ is provided with a Tarski-style extensional semantics. A linear, unbounded, and dense temporal structure $T = (P, <)$ is assumed, where $P$ is a set of time points and $<$ is a strict partial order on $P$. The interval set of a structure $T$ is defined as the set $T_P$ of all closed proper intervals $[u, v] = \{x \in P \mid u \leq x \leq v, u \neq v\}$ in $T$. A primitive interpretation $I = (T_P; \Delta_T, \cdot^2)$ consists of a set $T_P$ (the interval set of the selected temporal structure $T$), a set $\Delta_T$ (the domain of $I$), and a function $^2$ (the primitive interpretation function of $I$) which gives a meaning to atomic concepts, roles, features, and parametric features:

$$A^2 \subseteq T_P \times \Delta_T; \quad R^2 \subseteq T_P \times \Delta_T \times \Delta_T;$$

$$f^2 : (T_P \times \Delta_T)_{\text{partial}} \rightarrow \Delta_T; \quad g^2 : \Delta_T \rightarrow \Delta_T$$

Parametric features differ from features for being independent from time.

The temporal interpretation function $^2$ defined in the
upper half of Figure 2 depends on the temporal structure \( \mathcal{T} \). A labeled directed graph \((\mathcal{X}, \mathcal{E})\), where \(\mathcal{X}\) is a set of variables representing the nodes and \(\mathcal{E}\) is a set of temporal constraints representing the arcs, is called a temporal constraint network. An interpretation of a temporal constraint network is a set of variable assignments that satisfy the temporal constraints. A variable assignment is a function \( \nu : \mathcal{X} \rightarrow \mathcal{T}_i \) associating an interval to a temporal variable. A temporal constraint network is consistent if it admits a non-empty interpretation. The notation \((\mathcal{X}, \mathcal{E})^c\) is used to interpret concept expressions, denotes the subset of \((\mathcal{X}, \mathcal{E})^c\) where the variable \( x_i \) is mapped to the interval value \( t_i \).

An interpretation function \( \mathcal{T}^c_{\mathcal{Y}, \mathcal{Z}, \mathcal{H}} \) for generically concepts, based on a variable assignment \( \mathcal{V} \), an interval \( t \), and a set of constraints \( \mathcal{H} = \{ x_1 \mapsto t_1, \ldots, x_n \mapsto t_n \} \) over the assignments of free variables, extends the primitive interpretation function in such a way that the equations of Figure 2 are satisfied – operators that can be obtained by negation are omitted. Intuitively, the interpretation of a concept \( C^c_{\mathcal{Y}, \mathcal{Z}, \mathcal{H}} \) is the set of elements of the domain which are of type \( C \) at the time interval \( t \). The interpretation of the free temporal variables in \( C \) given by \( \mathcal{V} \) (c.f. the definition of \( C \circ \mathcal{X} \mathcal{T}^c_{\mathcal{Y}, \mathcal{Z}, \mathcal{H}} \) and with the constraints for the assignment of the variables in the scope of the outermost temporal quantifiers given by \( \mathcal{H} \). The natural interpretation function \( C^c_{\mathcal{Y}, \mathcal{Z}} \), being equivalent to the interpretation function \( C^c_{\mathcal{V}, \mathcal{Z}, \mathcal{H}} \) with any \( \mathcal{V} \) such that \( \mathcal{V}[\mathcal{Y}] = t \), and \( \mathcal{H} = \emptyset \), is introduced as an abbreviation.

An interpretation \( \mathcal{I} \) is a model for a concept \( C \) if, for some \( t \in \mathcal{T}_i \), \( C^c_{\mathcal{I}} \neq \emptyset \). If a concept has a model, then it is satisfiable otherwise it is unsatisfiable.

We will now informally discuss the intended meaning of \( \mathcal{T} \mathcal{L} \mathcal{A} \mathcal{C} \mathcal{F} \mathcal{C} \) concepts. Concept expressions are interpreted over pairs of temporal intervals and individuals \((i, a)\), meaning that the individual \( a \) is in the extension of the concept at the interval \( i \). Within a concept expression, the special “\( \mathcal{Y} \)” variable denotes the current interval of evaluation. The temporal existential quantifier “\( \exists \)” introduces interval variables, related to each other and possibly to the \( \mathcal{Z} \) variable in a way defined by the set of temporal constraints. 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 \( \mathcal{Y} \) in the concept expression \( C \) is interpreted as the \( X \) variable. Please consider the following example from the blocks world domain which defines a concept representing the action of stacking a block on top of another block.

Basic-Stack \( \equiv \exists (x \ y) [x \text{ meets } y \exists (i \ t) (x \text{ meets } i \ y \text{ meets } t)] \)

\( (\text{\textbf{*BLOCK} : OnTable} @x \land \text{\textbf{*BLOCK} : OnBlock} @y) \)

Basic-Stack denotes any action occurring at some interval involving a \( \text{\textbf{*BLOCK}} \) that was once \( \text{OnTable} \) and then \( \text{OnBlock} \). The \( \mathcal{Y} \) interval could be understood as the occurring time of the stacking action. The temporal constraints \((x \mathcal{Y} \mathcal{Z}) \) and \((y \mathcal{Z} t) \) state that the interval \( \mathcal{V}(x) \) should meet the interval \( \mathcal{V}(z) \) – the occurrence interval of the action type \( \text{Basic-Stack} \) – and that \( \mathcal{V}(y) \) should meet \( \mathcal{V}(t) \). The parametric feature \( \text{\textbf{*BLOCK}} \) plays the role of formal parameter of the action, mapping any individual action of type \( \text{Basic-Stack} \) to the block to be stacked, independently from time. Whereas the existence and identity of the \( \text{\textbf{*BLOCK}} \) of the action is time invariant, it can be in the extension of different concepts in different intervals of time, e.g., the \( \text{\textbf{*BLOCK}} \) is necessarily \( \text{OnTable} \) only during the interval \( \mathcal{V}(x) \).

3 The Logic \( \mathcal{A} \mathcal{C} \mathcal{F}(A) \)

Description logics represent knowledge on an abstract, logical level. So-called concrete domains provide a means to additionally represent “concrete information” such as, e.g., numbers or time intervals, and allow for integrated reasoning about both kinds of knowledge. In [6], the basic description logic incorporating concrete domains, \( \mathcal{A} \mathcal{C} \mathcal{F}(D) \), is introduced. The logic \( \mathcal{A} \mathcal{C} \mathcal{F}(D) \) [10] extends \( \mathcal{A} \mathcal{C} \mathcal{F}(A) \) by agreement and disagreement on features. Similar to \( \mathcal{A} \mathcal{C} \mathcal{F}(D) \), an “admissible” concrete domain \( D \) yields decidability of \( \mathcal{A} \mathcal{C} \mathcal{F}(D) \). Before \( \mathcal{A} \mathcal{C} \mathcal{F}(D) \) is introduced, the definition of concrete domains is recalled.

Definition 3.1. A concrete domain \( D \) is a pair \((\Delta_D, \Phi_D)\), where \( \Delta_D \) is a set called the domain, and \( \Phi_D \) is a set of predicate names. Each predicate name \( P \) in \( \Phi_D \) is associated with an arity \( n \) and an \( n \)-ary predicate \( P^D \subseteq \Delta_D^n \).

A concrete domain \( D \) is called admissible iff (1) the set of predicate names is closed under negation.
and contains a name $T_D$ for $\Delta_D$ and (2) the satisfiability problem for finite conjunctions of predicates is decidable.

The syntax of $\textit{ALCF}(D)$ is obtained from the syntax of $\textit{ALCF}$ as given in Figure 1 by adding an additional syntax rule for the predicate operator:

$$E, F \rightarrow \exists p_1, \ldots, p_n, P$$

where $P \in \Phi_D$ is an $n$-ary predicate name, and $p_1, \ldots, p_n$ are paths.

An $\textit{ALCF}(D)$ interpretation $I = (\Delta_X, \cdot^I)$ consists of a set $\Delta_X$ (the abstract domain) which is disjoint from $\Delta_D$ and an interpretation function $\cdot^I$. The interpretation function maps each concept name $C$ to a subset $C^I$ of $\Delta_X$, each role name $R$ to a subset $R^I$ of $\Delta_X \times \Delta_X$, and each feature name $f$ to a partial function $f^I$ from $\Delta_X$ to $\Delta_D \cup \Delta_X$. Parametric features are identical to non-parametric features w.r.t their $\textit{ALCF}(D)$ interpretation. If $p = f_1 \cdots f_k$ is a feature chain, then $p^I$ is defined as the composition $f_1^I \circ \cdots \circ f_k^I$ of the partial functions $f_1^I, \ldots, f_k^I$. Each complex concept is interpreted as usual (i.e., as in Figure 2 with the temporal indices omitted) while the new predicate operator has the following meaning:

$$(\exists p_1, \ldots, p_n, P)^I = \{ (a, x) \mid \exists x_1, \ldots, x_n \in \Delta_D : (a, x) \in p_1^I \land \cdots \land (a, x) \in p_n^I \land (x_1, \ldots, x_n) \in P^D \}$$

In this paper, we consider the logic $\textit{ALCF}(A)$, i.e., $\textit{ALCF}(D)$ instantiated with the temporal concrete domain $A$. The concrete domain $A$ is based on intervals and Allen’s relations (hence the name “$A$”). Formally, $A$ is defined as $(\Delta_A, \Phi_A)$, where $\Delta_A$ is the interval set $T_2^\omega$ as defined in Section 2, and $\Phi_A$ contains:

- the unary predicates $T_A, \bot_A$ denoting $\Delta_A$ and $\emptyset$.
- 13 binary predicates b.m.d... corresponding to Allen’s 13 basic relations. The extensions $b^A, m^A, d^A, \ldots$ are defined analogously to the interpretation of Allen’s relations by $p^I$ in Figure 2.
- a binary predicate $r_1 \cdots r_k$ for each disjunction $r_1 \lor \cdots \lor r_k$ of Allen relations $r_1, \ldots, r_k$ including the empty disjunction $empty-rel$. The extension of a disjunctive predicate $(r_1 \cdots r_k)^A$ is $r_1^A \lor \cdots \lor r_k^A$, furthermore, $empty-rel^A = \emptyset \times \emptyset$.

In [8], it is proved that the concrete domain $A$ is admissible and that satisfiability of $\textit{ALCF}(A)$ concepts is PSPACE-complete.

In the framework of $\textit{ALCF}(A)$, a basic stack action similar to the one in Section 2 can be defined as follows:

Basic-Stack = step₁ : (BLOCK : OnTable) \n\nstep₂ : (BLOCK : OnBlock) \n\n\exists (step₁ \circ \text{time}), (step₂ \circ \text{time}) \cdot m \n\n\exists (step₁ \circ \text{time}), (step₂ \circ \text{time}) \cdot m

The concept states that any Basic-Stack is related to three objects via the features step₁, step₂, and step₂. These objects describe the basic stack action at different time intervals – with step₂ representing the occurring time of the action, often called the “current” interval. For each step, a corresponding time interval is associated by the time feature. The relation between these time intervals is described using the predicate operator and resembles the temporal network in the $\textit{TL-ALCF}$ definition of the basic stack.

Comparing the two definitions of Basic-Stack, their main difference can be characterized as follows: In the $\textit{TL-ALCF}$ definition, the basic stack is represented by a single logical object which is “temporal”, i.e., whose properties are defined separately for each temporal interval. To the contrary, in $\textit{ALCF}(A)$, the basic stack is represented by a logical “meta-object” (the Basic-Stack object itself in the above concept definition) and a set of additional logical objects each of which has unique properties and represents the basic stack at a unique time interval. A reduction from $\textit{TL-ALCF}$ to $\textit{ALCF}(A)$, as defined in the next Section, has to bridge this discrepancy. Furthermore, it has to capture the temporal invariance of parametric features. The basic stack as defined above is not to be intended as a translation of the $\textit{TL-ALCF}$ Basic-Stack.

4 A Tableau for SAT in $\textit{TL-ALCF}$

The logic $\textit{ALCF}(A)$ is provided with a sound and complete tableau calculus which is optimal w.r.t. worst case complexity [10]. To obtain a tableau calculus and establish complexity results for $\textit{TL-ALCF}$, we will build on the $\textit{ALCF}(A)$ calculus. This section shows how to reduce satisfiability of $\textit{TL-ALCF}$ concepts to satisfiability of $\textit{ALCF}(A)$ concepts.

As a starting point for the reduction to be devised, we do not consider arbitrary $\textit{TL-ALCF}$ concepts but only those in a certain normal form. In [3], it is shown that every $\textit{TL-ALCF}$ concept can be reduced to an equivalent concept in existential form, i.e., of the form $\diamond (\exists X) \textit{E}_Q \circ Q_1 \circ X_1 \oplus \cdots \oplus Q_n \circ X_n$. In the existential form, the only temporal operator that may occur is a single “$\circ$” operator, while each $Q_i$ is an $\textit{ALCF}$ concept. The normal form for a $\textit{TL-ALCF}$ concept is obtained by starting from its existential form, and then applying simple form, and path explicitation steps.

Definition 4.1 (Normal form). Given a concept in existential form, its Normal Form (NF) is obtained by sequentially applying the following transformations.

(Simple Form) Transform each $Q_i$ into the equivalent simple form following the rewrite rules reported in [7].

A concept in simple form contains only complements of the form $\neg A$, where $A$ is a primitive concept, and no
sub-concepts of the form $p \uparrow$, where $p$ is a path with length greater than one. This corresponds to a first order formula in negation normal form.

(Path Explicitation) Apply the following normalization rules which make explicit all the possible chains of features.

$$
p: (C \cap D) \rightarrow p: C \cap p: D \quad p: (C \cup D) \rightarrow p: C \cup p: D$$

For example, the normal form of the $\mathcal{ALCF}$ concept $p: (C \cap -f: D)$ is $p \oplus q: C \cap (p: f \uparrow \cap p \oplus f: -D)$. (note that the simple form of $-f: D$ is f $\uparrow \cap f: -D$).

**Proposition 4.2 (Equivalence of NF).** Every concept $C$ can be reduced into an equivalent concept in normal form.

In the following, a satisfiability preserving translation $\Phi$ from $\mathcal{TLC}-\mathcal{ALCF}$ concepts in NF to $\mathcal{ALCF}(A)$ concepts is given. Let $\gamma$ denote features (parametric or non-parametric). Given a $\mathcal{TLC}-\mathcal{ALCF}$ concept $C$ in normal form i.e., $\Phi(\bigwedge_{i=1}^{n} X_i \oplus Q_0 \cap Q_1 \cap \ldots \cap Q_n \cap X_n)$, $\Phi(C)$ is obtained as follows:

1. Let $\overline{E}$ be $\{ (X_1, r_1, Y_1), \ldots, (X_k, r_k, Y_k) \}$ and let $f_0, \ldots, f_n$ be features not used in $C$. The mapping $\alpha$ from $\mathcal{TLC}-\mathcal{ALCF}$ temporal constraints to $\mathcal{ALCF}(A)$ concepts is defined as follows:

$$\alpha(X_i \cap r \cap Y_j) = \exists f_0 \circ \text{time}, \ (f_0 \circ \text{time} \circ r)$$

2. Let $\Phi$ be the set of paths used in the concept $C$. For each $0 \leq i \leq n$, the mapping $\Phi_i: Path \rightarrow Path \cup \{ f_i \}$, with $f_i$ as introduced in Point 1, is defined in Figure 3.

3. For each $0 \leq i \leq n$, the mapping $\Psi_i$ which maps $\mathcal{ALCF}$ concepts in normal form to $\mathcal{ALCF}$ concepts in normal form is defined in Figure 3.

4. Let $\forall y_1, \ldots, \forall y_m$ be the parametric features used in $C$. Define $C'\Phi$ as

$$\prod_{i=0}^{m} \left( (\forall j=0 \rightarrow f_0 \circ \forall y_i \downarrow (\forall j=1 \rightarrow f_0 \circ \forall y_i \downarrow (f_0 \circ \forall y_i)) \right)$$

5. Define two concepts $\Omega$ and $\Omega'$ as follows:

$$\Omega = f_0: \Phi(\bigwedge_{0 \leq i \leq n} f_i: \Phi_i(Q_i))$$

$$\Omega' = f_0: \Phi(\bigwedge_{0 \leq i \leq n} f_i: \Phi_i(Q_i))$$

where $E \rightarrow F$ is an abbreviation for $\forall E \cap F$.

As an example, the translation of the Basic-Stack concept as introduced in Section 2 is given.

$$\Phi(\bigwedge_{i=1}^{n} X_i \oplus Q_0 \cap Q_1 \cap \ldots \cap Q_n \cap X_n)$$

$$\Phi(C) = C'\Phi \cap C'' \cap \Omega \cap \Omega'$$

Figure 3: Definition of $\Phi_i(p)$ and $\Psi_i(Q)$ mappings.

The main idea behind the reduction has already been discussed at the end of Section 3: A (temporal) object $a$ is which is in the extension of a $\mathcal{TLC}-\mathcal{ALCF}$ concept $C$, is reflected by a “meta-object” $o'$ and a set of objects $O' = \{ o_0', \ldots, o_n' \}$ on the $\mathcal{ALCF}(A)$ side, where each $o_i'$ represents $a$ at a different time interval. The features $f_0, \ldots, f_n$ are introduced during the translation in order to relate $o'$ with the objects in $O'$ ($o_i'$ is a $f_i$-filler of $o'$ for $0 \leq i \leq n$). Each object in $O'$ has a unique time interval associated via the $\text{time}$ feature. The object $o_0'$ represents $a$ at the current time interval. The concept $C_T$ ensures that the temporal relations between the associated intervals are as defined by the temporal constraint network $\overline{\mathcal{E}}$. Additional care has to be taken in order to deal correctly with parametric features. Since they are time-independent, it has to be ensured that all $\mathcal{ALCF}(A)$ objects in $O'$ have identical fillers of parametric features. This is done by the $C_T$ concept together with the mappings $\Psi_i$ and $\Phi_i$. Furthermore, it is possible that different objects in $O'$ are associated with the same interval. In this case, they both describe the object $o$ at the same time interval, and, hence, should be identical w.r.t. concept membership. The concept $\Omega'$ ensures this.
A trivial task since full negation is not available in the temporal part of the logic $\mathcal{T}_C$-ALCF.

Acknowledgements
The work presented in this paper was partially supported by the “Foundations of Data Warehouse Quality” (DWF) European ESPRIT IV Long Term Research (LTR) Project 22469.

References