Representation of Temporal Knowledge in the Situation Calculus*

Marcelo Arenas, Leopoldo Bertossi and Javier Pinto
Depto. de Ciencia de la Computación
Pontificia Universidad Católica de Chile†
{marenas, bertossi, jpinto}@ing.puc.cl

7th January 1998 13:32

Abstract

In this article we deal with the problem of using temporal knowledge in the specification of the dynamics of a database. Temporal knowledge is formalized with formulas that refer to the evolution of the database. In the article we analyze several problems involving temporal knowledge: **(1)** Specification of complex temporal preconditions for database transactions. **(2)** Specification and maintenance of integrity constraints. **(3)** Posing and answering temporal queries about the real or hypothetical evolution of the database.

Our research is based on two logical languages that have been developed in parallel: The Situation Calculus, a logical language developed to represent knowledge about action and change [8]. The Situation Calculus has been used as a medium for the specification of database updates [14], and our work is a further development of this approach.

The Metric Time Temporal Logic, a first order modal temporal logic, has been used ...

Subject Category: Knowledge Representation in Databases.

1 Introduction

2 Theories of Action in the Situation Calculus

2.1 Foundations of the Situation Calculus

The situation calculus is a logical language designed to represent theories of action and change. The language has its origins in the pioneering work of John McCarthy [7]. The intuition behind the situation calculus language is that there is an *initial situation*, called S_0, and that the world changes

*This work has been partly funded by the FONDECYT grant # 1971304.
†Casilla 306, Santiago 22, Chile. Phone: +(56-2) 686-4440. Fax: +(56-2) 686-4444. URL: http://lyrcc.ing.puc.cl/

from one situation to another when actions are performed. In the discrete situation calculus, the only situations that exist are those that are reached by performing some sequence of actions starting in S_0. As a result, the structure of situations can be seen as a tree rooted at S_0. In this framework, situation dependent properties are called *fluents*, and a *state* corresponds to a set of fluents. Thus, the state of a situation is the set of fluents that hold in that situation. Notice that the branching factor of the tree of situations rooted at S_0 equals the number of distinct actions that exist. The language of the Situation Calculus is used in this article is based on Reiter's formalization [13], and in extensions to the language in which metric time is added [10, 15].

The situation calculus is a sorted second order language with equality. We use the standard operators for conjunction, disjunction, implication and equivalence ($\wedge$, $\vee$, $\supset$ and $\equiv$ respectively), and standard universal and existential quantifiers. Scoping is indicated by parentheses or by the dot notation (e.g., "$(\forall\, \underline{\mathbf{x}})$." indicates that the scope of the quantifier is the rest of the formula). There are three domain independent sorts $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{T}$, for primitive actions, situations and time. Primitive actions will be denoted with function symbols with one parameter of sort $\mathcal{T}$ [15]. Thus, a term $borrow(book, t_1)$ would denote a book borrowing action at time t_1.

We also include a sort $\mathcal{D}$ of domain objects, which may be further subdivided in sub-sorts depending on the particular domain of application. Variables are denoted by lower case letters (with or without subscripts), and constants are denoted by upper case letters (with or without subscripts). Letters a, s, t, d (A, S, T, D) are used for variables (constants) of sorts $\mathcal{A}, \mathcal{S}, \mathcal{T}, \mathcal{D}$ respectively. We will have a single 1-place predicate variable ϕ over situations. Unless otherwise stated, other variables and constants will be assumed to be of sort $\mathcal{D}$. Also, when free variables appear in formulas, they are assumed to be universally quantified with maximal scope.

We include the special constant S_0 of sort $\mathcal{S}$, the function $do : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$, the relation $< \subseteq \mathcal{S} \times \mathcal{S}$, the relation $Poss \subseteq \mathcal{A} \times \mathcal{S}$, the function $time : \mathcal{A} \to \mathcal{T}$, along with other functions and relations introduced later. We use the abbreviation $do([a_1, \ldots, a_n], s)$ for the term:

$$do(a_n, do(\ldots, do(a_1, s) \ldots)).$$

Keep in mind that $[a_1, \ldots, a_n]$ is not a term in the language and that it can only appear as a first argument in a do function term.

Also, we include a finite set of predicate constants of different arities. These are the *fluent predicates* and they have a single situation argument, which is always the last argument of the predicate.

The basic axioms for situations include an induction axiom to characterize the sort $\mathcal{S}$:

$$(\forall\, \phi).[\phi(S_0) \wedge (\forall\, s, a)\ (\phi(s) \supset \phi(do(a, s)))] \supset (\forall\, s)\ \phi(s). \tag{1}$$

Unique names axioms for situations:

$$do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2, \tag{2}$$
$$S_0 \neq do(a, s). \tag{3}$$

Recall that all function symbols of type action have one parameter of sort $\mathcal{T}$. For each action function $A(\underline{\mathbf{x}}, t)$, the function $time$ is defined with the axiom:

$$time(A(\underline{\mathbf{x}}, t)) = t \tag{4}$$

Therefore, if $borrow(book, t_1)$ is an action term, then we would have the axiom:

$$time(borrow(book, t_1)) = t_1.$$

The relation $<$ is characterized with the axioms[1]:

$$s_1 < do(a, s_2) \equiv Poss(a, s_2) \wedge s_1 \leq s_2 \wedge start(s_2) \leq time(a), \tag{5}$$
$$\neg\, s < S_0, \tag{6}$$

According to this characterization, $s_1 < s_2$ is true if all the actions that lead from s_1 to s_2 are possible in the intermediate situations where they are performed, and their times are in the proper order.

2.2 The Frame Problem

One of the hardest problems encountered in reasoning about actions has been the so called *frame problem*. The problem is understood as one of finding a way to *succinctly* specify the effects of actions, given the situations in which they are performed. Many solutions to the frame problem are based on non-monotonic logics, e.g., [18, 3, 6]. Also, there has been interest in the development of solutions based on standard monotonic logics. In particular, we appeal to the solution proposed by Reiter [11], which is based on the work of Haas [2], Pednault [9], and Schubert [16]. In this section, we present Reiter's monotonic solution to the frame problem.

A basic theory of action in the situation calculus consists of the following sets of axioms:

- A set Σ_{bd}, which includes
  - The set Σ_b of basic situation calculus axioms (i.e., axioms (1)-(4)).
  - A set Σ_d of situation independent axioms formalizing the static properties of the domain of interest.
- A set T_{pos} of action precondition axioms (APA). We assume that all the necessary conditions for the execution of an action are known. This completeness assumption means that for each action A, we write an axiom of the form:

  $$Poss(A, s) \equiv \Pi_A(s),$$

  where $\Pi_A(s)$ is a simple formula[2].
- A set T_{S_0} of sentences about S_0. That is, a description of the initial situation.

[1] We use the abbreviation $s_1 \leq s_2$ to mean $s_1 < s_2 \vee s_1 = s_2$. Also, we use the notation $a < b < c$ to mean $a < b \wedge b < c$, and similarly when the expression combines $<$ and $\leq$.

[2] A simple formula is a first order formula which does not mention the predicate symbols $Poss$ or $<$, that mentions a unique situation variable (e.g. s) and no other situation terms, and which does not quantify over variables of sort *situation* ($\mathcal{S}$) [12].

- A set T_{una} of uniqueness of names axioms for actions. For distinct action names a and a':

$$a(x_1, \ldots, x_n, t) \neq a'(y_1, \ldots, y_m, t').$$

Identical actions have identical arguments. Thus, for each function symbol of sort $\mathcal{A}$ in the language, we write:

$$a(x_1, \ldots, x_n, t) = a(y_1, \ldots, y_n, t') \supset x_1 = y_1 \wedge \ldots \wedge x_n = y_n \wedge t = t'.$$

- A set T_{ef} of effect axioms. Effect axioms can be positive or negative. A positive effect axiom is of the form[3]:

$$Poss(a, s) \wedge G_f^+(\underline{\mathbf{x}}, a, s) \supset f(\underline{\mathbf{x}}, do(a, s)).$$

Whereas negative effect axioms are of the form:

$$Poss(a, s) \wedge G_f^-(\underline{\mathbf{x}}, a, s) \supset \neg f(\underline{\mathbf{x}}, do(a, s)).$$

Now, all the positive (negative) effect axioms for a fluent predicate can be put together in a single general positive (negative) effect axiom. A general positive effect axiom for f would be of the form:

$$Poss(a, s) \wedge \gamma_f^+(\underline{\mathbf{x}}, a, s) \supset f(\underline{\mathbf{x}}, do(a, s)). \quad (7)$$

The corresponding general negative effect axiom is of the form:

$$Poss(a, s) \wedge \gamma_f^-(\underline{\mathbf{x}}, a, s) \supset \neg f(\underline{\mathbf{x}}, do(a, s)). \quad (8)$$

Thus, $T_{ef} = T_{ef}^+ \cup T_{ef}^-$, where T_{ef}^+ and T_{ef}^- are the general positive and negative effect axioms for each fluent. If there were no positive effect axioms for fluent f, then the formula $\gamma_f^+(\underline{\mathbf{x}}, a, s)$ is the atom $False$. Also, if there were no negative effect axioms, then $\gamma_f^-(\underline{\mathbf{x}}, a, s)$ would be $False$ as well.

In these axioms $\gamma_f^+(\underline{\mathbf{x}}, a, s)$ and $\gamma_f^-(\underline{\mathbf{x}}, a, s)$ are simple formulas which are used to provide conditions under which an action a produces an effect on a fluent f.

Following Schubert [16], the assumption that the general (positive or negative) effect axioms characterize all the conditions for a state s under which action a makes $f(\underline{\mathbf{x}}, do(a, s))$ true can be formalized with an *explanation closure axiom* of the form:

$$Poss(a, s) \wedge \neg f(\underline{\mathbf{x}}, s) \wedge f(\underline{\mathbf{x}}, do(a, s)) \supset \gamma_f^+(\underline{\mathbf{x}}, a, s).$$

Where $\gamma_f^+(\underline{\mathbf{x}}, a, s)$ can be understood as an explanation why f changed from not holding in s to holding in $do(a, s)$, as result of performing action a. Similarly, the assumption that the general effect axioms characterize all the conditions for a state s under which action a makes $f(\underline{\mathbf{x}}, do(a, s))$ false is formalized with:

[3]We write $\underline{\mathbf{x}}$ to denote a tuple of domain variables (i.e., variables of sort $\mathcal{D}$).

$$Poss(a,s) \wedge f(\underline{\mathbf{x}},s) \wedge \neg f(\underline{\mathbf{x}},do(a,s)) \supset \gamma_f^-(\underline{\mathbf{x}},a,s).$$

Reiter provides an approach to mechanically integrate Schubert's explanation closure axioms with the general effect axioms to derive a set of *successor state axioms*. The resulting successor state axiom for fluent f would be:

$$Poss(a,s) \supset [f(\underline{\mathbf{x}},do(a,s)) \equiv \gamma_f^+(\underline{\mathbf{x}},a,s) \vee f(\underline{\mathbf{x}},s) \wedge \neg\gamma_f^-(\underline{\mathbf{x}},a,s)]. \tag{9}$$

That is, given a situation, if an action is possible we can determine whether the fluent f will hold in the subsequent situation. For this approach to work, it is necessary to have a finite set of effect axioms. Reiter's proposal is to replace the effect axioms with the successor state axioms, yielding a new theory $\Sigma_{bd} \cup T_{pos} \cup T_{S_0} \cup T_{una} \cup T_{ss}(T_{ef})$. Where we write $T_{ss}(T_{ef})$ to denote the successor state axioms that result from utilizing explanation closure axioms and the effect axioms T_{ef}. So, for each function fluent in the language, we have a successor state axiom. Notice that if for some fluent f there were no effect axioms, the resulting successor state axioms would state that:

$$Poss(a,s) \supset [f(\underline{\mathbf{x}},do(a,s)) \equiv f(\underline{\mathbf{x}},s)].$$

Thus, f does not change due to the execution of any action.

2.3 An Example.

3 Past Metric First Order Temporal Logic

Past Metric First–Order Temporal Logic (MTL) is an extension of first–order logic by means of a set of rules for building formulas. More precisely, the terms in this logic are built as in first-order logic from a finite set of constants and a countably infinite set of variables. Formulas are construct as follows:

- If t_1 and t_2 are terms, then $t_1 \sim t_2$ is a formula, with $\sim \in \{=, \neq, <, >, \leq, \geq\}$.
- If $t_1, \ldots, t_n$ are terms and R is a $n\text{-}ary$ relation symbol, then $R(t_1, \ldots, t_n)$ is a formula.
- If φ and ψ are formula, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \supset \psi$, $\varphi \equiv \psi$ are formulas.
- If x is a variable and φ is a formula, then $\forall x\varphi$ and $\exists x\varphi$ are formulas.
- If φ is a formula, then $\bigcirc\varphi$ and $\bigcirc_{\sim c}\,\varphi$ are formulas, with c is a nonnegative integer and $\sim \in \{=, \neq, <, >, \leq, \geq\}$.
- If φ and ψ are formulas, then φ **since** ψ and φ $\textbf{since}_{\sim c}$ ψ are formulas, with c is a nonnegative integer and $\sim \in \{=, \neq, <, >, \leq, \geq\}$.

These formulas are interpreted in finite sequences of first–order structures, $D = (D_0, \dots, D_k)$, which we will call *temporal databases*[1]. In this sequences, we refer to the structure D_i as to the *database state*[1], or simply *state*, at time instant i.

Every structure in a temporal database is defined over the same set of constants and relations symbols, and with the same universe, which we will call the universe of the database $|D|$ ($|D| = |D_0| = \dots = |D_k|$). We additionally assume, that in every temporal database there exists a function $clock : \mathbb{N} \to \mathbb{N}$, that evaluated over i represents the clock of D_i. Clearly, this function has to satisfies $clock(i) \leq clock(i+1)$.

Let v a valuation, that is, a mapping that associates a value $v(x) \in |D|$ with each variable x. In order to define a semantic for MTL, is necessary to construct rules for deciding the truth value of a given formula in a temporal database. More precisely, is necessary to give rules for a satisfaction relation which indicates if a temporal database D satisfies a formula φ, at a time instant i, under a valuation v. The notation $D, v \models_i \varphi$ represents this satisfaction relation, and it is defined as follows:

- $D, v \models_i t_1 \sim t_2$, whenever $v(t_1) \sim v(t_2)$ in D_i.
- $D, v \models_i R(t_1, \dots, t_n)$, whenever the tuple $(v(t_1), \dots, v(t_n))$ is in the relation R in D_i.
- $D, v \models_i \neg\varphi$ iff not $D, v \models_i \varphi$.
- $D, v \models_i \varphi \vee \psi$ iff $D, v \models_i \varphi$ or $D, v \models_i \psi$. The definition is analogous for connectives $\wedge$, $\supset$ and $\equiv$.
- $D, v \models_i \exists x\varphi$ iff there exists an element d in the universe of the structure D_i such that $D, v(\frac{d}{x}) \models_i \varphi$, where $v(\frac{d}{x})$ is a new valuation that maps x to d and otherwise coincides with v. The definition is analogous for quantifier $\forall$.
- $D, v \models_i \bigcirc\varphi$ iff $i > 0$ and $D, v \models_{i-1} \varphi$
- $D, v \models_i \bigcirc_{\sim c}\, \varphi$ iff $i > 0$, $D, v \models_{i-1} \varphi$ and $clock(i) - clock(i-1) \sim c$.
- $D, v \models_i \varphi$ **since** ψ iff there exists j, $0 \leq j < i$, such that $D, v \models_j \psi$ and for every m, $j < m \leq i$, $D, v \models_m \varphi$.
- $D, v \models_i \varphi$ **since**$_{\sim c}$ ψ iff there exists j, $0 \leq j < i$, such that $D, v \models_j \psi$ and $clock(i) - clock(j) \sim c$, and for every m, $j < m \leq i$, $D, v \models_m \varphi$.

Other temporal connectives can be defined from the previous ones: $\Diamond\varphi =_{def} \mathit{True}$ **since** φ (sometime in the past φ was true), $\Box\varphi =_{def} \neg\Diamond\neg\varphi$ (always in the past φ was true), $\Diamond_{\sim c}\, \varphi =_{def} \mathit{True}$ **since**$_{\sim c}$ φ and $\Box_{\sim c}\, \varphi =_{def} \neg\Diamond_{\sim c}\, \neg\varphi$

In [1], Chomicki uses MTL for formulating temporal integrity constraints. He addresses the problem of their checking; in the following sense. Let us assume that we have a temporal database $D = (D_0, \dots, D_k)$ and a temporal IC φ. By executing an update we will generate a new database state D_{k+1}. Is it the case that $D = (D_0, \dots, D_k, D_{k+1})$ satisfies φ?.

To answer this question we might consider all the information stored in the database states $D_0, \dots, D_{k+1}$, but this would be too costly in terms of space and time. Instead, Chomicki develops a methodology for storing in the state D_k the information that is necessary to evaluate φ in the state D_{k+1}. More precisely, this procedure generates a new temporal database $(\overline{D}_0, \ldots, \overline{D}_k)$, where each original state has been extended with the information that is needed to evaluate φ in the successor state. It contains, apart form the original base relations, new auxiliary relations generated as follows: for each subformula α, with temporal operators, of the given IC φ, a new table r_α is created.

In addition, there will be a new relation r_φ representing the IC. This relation will be evaluated at each new state in order to check the IC, without having to query past database states. Since this table comes from a sentence, it will contain just one of the two possible entries $true$ or $false$. We will explain in section 4 how to update (and define through this process) all these auxiliary tables. This will be done by providing appropriate SSA for them in the SC.

4 Evaluating Temporal Logic Formulas in the Situation Calculus

The language of the Situation Calculus presented in section 2 allows one to describe all the possible evolutions of the world starting at the initial situation S_0. Also, due to the induction axiom (1), one can view each situation as a history of all the actions that lead to it, starting in S_0. Now, given a single situation, the language of the Situation Calculus is well suited to express formulas describing conditions about specific situations. However, the language does not seem appropriate to express formulas to describe conditions about the history of a situation. On the other hand, the metric temporal logic described in section 3, has been designed with the purpose of expressing such kind of formulas. Furthermore, the metric temporal logic does not provide a mechanism to reason about histories described as sequences of actions performed in some initial situation. Therefore, this logic is not well suited to answer queries regarding alternative evolutions of the world.

Thus, there is a great deal of expressive power that can be gained by integrating both languages.

Let us assume that we write a theory describing the evolution of the world in a language of the Situation Calculus. We want to be able to write formulas of the MTL and evaluate them with respect to specific situations. A first step is considering that formulas of the MTL do not make reference to situations. Thus, given a Situation Calculus language, we will have a MTL language such that:

1. Every n-ary fluent in the Situation Calculus is modeled with an $(n-1)$-ary fluent in the MTL language (the situation argument is eliminated).
2. Every domain term in the Situation Calculus is also a domain term in the MTL language.
3. Every n-ary non-fluent predicate in the Situation Calculus is modeled with an n-ary non-fluent in the MTL language.

Therefore, we build an MTL vocabulary from a Situation Calculus theory. With this vocabulary we can write MTL formulas.

Let Φ_{mtl} denote an arbitrary formula in the MTL language. We would like a mechanism to evaluate such formulas with respect to the histories of the situation calculus. With that purpose in mind, we introduce the following notation:

$$\texttt{holds}(\Phi_{mtl}, s) \tag{10}$$

To mean that the MTL formula Φ_{mtl} is true when evaluated with respect to the history if s.

We introduce holds as a *macro*, or abbreviation. Therefore, the formula (10) stands for a Situation Calculus formula, as defined recursively below.

- If Φ_{mtl} does not mention any fluents, nor does it mention modal operators, then:

$$\texttt{holds}(\Phi_{mtl}, s) =_{def} \Phi_{mtl}$$

- If Φ_{mtl} is a fluent atom of the form $F(\underline{\mathbf{t}})$ then $\texttt{holds}(\Phi_{mtl}, s) =_{def} F(\underline{\mathbf{t}}, s)$.
- If Φ_{mtl} is $\neg\psi$, then $\texttt{holds}(\Phi_{mtl}, s) =_{def} \neg\texttt{holds}(\psi, s)$.
- If Φ_{mtl} is $\varphi{*}\psi$, where $* \in \{\vee, \wedge, \supset, \equiv\}$, then $\texttt{holds}(\Phi_{mtl}, s) =_{def} \texttt{holds}(\varphi, s){*}\texttt{holds}(\psi, s)$.
- If Φ_{mtl} is $(Q\varphi)$, where Q is any of the usual first–order quantifiers, then $\texttt{holds}(\Phi_{mtl}, s) =_{def} Q(\texttt{holds}(\varphi, s))$
- If Φ_{mtl} is $\bigcirc\psi(\underline{\mathbf{x}})$, with ψ non temporal, then $\texttt{holds}(\Phi_{mtl}, s) =_{def} R_{\Phi_{mtl}}(\underline{\mathbf{x}}, s)$, where $R_{\Phi_{mtl}}$ is a new table name.
- If Φ_{mtl} is $\bigcirc_{\sim c}\, \psi(\underline{\mathbf{x}})$, with ψ non temporal, then $\texttt{holds}(\Phi_{mtl}, s) =_{def} \exists t(R_{\Phi_{mtl}}(\underline{\mathbf{x}}, t, s) \wedge t \sim c)$, where t is a variable of sort $time$ and $R_{\Phi_{mtl}}$ is a new table name.
- If Φ_{mtl} is $\psi(\underline{\mathbf{x}})$ **since** $\theta(\underline{\mathbf{x}})$, where $\psi(\underline{\mathbf{x}})$ and $\theta(\underline{\mathbf{x}})$ are non temporal, then $\texttt{holds}(\Phi_{mtl}, s) =_{def} R_{\Phi_{mtl}}(\underline{\mathbf{x}}, s)$, where $R_{\Phi_{mtl}}$ is a new table name.
- If Φ_{mtl} is $\psi(\underline{\mathbf{x}})$ $\mathbf{since}_{\sim c}$ $\theta(\underline{\mathbf{x}})$, where $\psi(\underline{\mathbf{x}})$ and $\theta(\underline{\mathbf{x}})$ are non temporal, then $\texttt{holds}(\Phi_{mtl}, s) =_{def} \exists t(R_{\Phi_{mtl}}(\underline{\mathbf{x}}, t, s) \wedge t \sim c)$, where t is a variable of sort $time$ and $R_{\Phi_{mtl}}$ is a new table name.

By bottom–up transformation of a MTL formula Φ_{mtl}, we can always obtain such a formula $\texttt{holds}(\Phi_{mtl}, s)$. Notice that this is a SC formula that is simple in the state s, it talks about an isolated state, s.

Now we specify the dynamics of the new tables by means of SSAs. We have to consider the last cases in the previous inductive definition:

- Let Φ_{mtl} be a temporal formula of the form $\bigcirc\psi(\underline{\mathbf{x}})$, where $\psi(\underline{\mathbf{x}})$ is a non temporal formula. This formula is true at a given state if $\psi(\underline{\mathbf{x}})$ is true at the previous state. Then, the new table $R_{\Phi_{mtl}}(\underline{\mathbf{x}}, s)$ has the following SSA:

$$\forall(a, s)Poss(a, s) \supset \forall\underline{\mathbf{x}}[R_{\Phi_{mtl}}(\underline{\mathbf{x}}, do(a, s)) \equiv \texttt{holds}(\psi(\underline{\mathbf{x}}), s)]. \tag{11}$$

Although, in the initial state Φ_{mtl} is false for each $\underline{\mathbf{x}}$, because there is not a previous state to S_0, so we define $R_{\Phi_{mtl}}(\underline{\mathbf{x}}, S_0)$ as follows:

$$\forall\underline{\mathbf{x}}\ \neg R_{\Phi_{mtl}}(\underline{\mathbf{x}}, S_0). \tag{12}$$

- Let Φ_{mtl} be a temporal formula of the form $\bigcirc_{\sim c}\ \psi(\underline{\mathbf{x}})$, where $\psi(\underline{\mathbf{x}})$ is a non temporal formula. The new table $R_{\Phi_{mtl}}(\underline{\mathbf{x}}, t, s)$ has the following SSA:

$$\forall(a,s)Poss(a,s) \supset \forall(\underline{\mathbf{x}},t)[R_{\Phi_{mtl}}(\underline{\mathbf{x}},t,do(a,s)) \equiv \texttt{holds}(\psi(\underline{\mathbf{x}}),s) \wedge \\ \mathit{time}(a) - \mathit{start}(s) \sim t]. \quad (13)$$

  Again, in the initial state Φ_{mtl} is false for each $\underline{\mathbf{x}}$, because there is not a previous state to S_0, so we define $R_{\Phi_{mtl}}(\underline{\mathbf{x}}, t, S_0)$ as follows:

$$\forall(\underline{\mathbf{x}},t)\ \neg R_{\Phi_{mtl}}(\underline{\mathbf{x}},t,S_0). \quad (14)$$

- Let Φ_{mtl} be a formula of the form $\psi(\underline{\mathbf{x}})$ **since** $\theta(\underline{\mathbf{x}})$, where $\psi(\underline{\mathbf{x}})$ and $\theta(\underline{\mathbf{x}})$ are non temporal formulas. This formula is true at a state s, with predecessor state s', iff ψ **since** θ was true at s' and ψ is still true at s or ψ became true at s and θ became true at s'. This is equivalent to saying that $((\psi\ \textbf{since}\ \theta) \vee \theta)$ is true at s' and ψ is true at s. Then, for the new table $R_{\Phi_{mtl}}(\underline{\mathbf{x}}, s)$ it holds:

$$\forall(a,s)Poss(a,s) \supset \forall\underline{\mathbf{x}}[R_{\Phi_{mtl}}(\underline{\mathbf{x}},do(a,s)) \equiv \texttt{holds}(\psi(\underline{\mathbf{x}}),do(a,s)) \wedge \\ (R_{\Phi_{mtl}}(\underline{\mathbf{x}},s) \vee \texttt{holds}(\theta(\underline{\mathbf{x}}),s))].$$

  This is not a SSA, because there is a $do(a, s)$ term in $\texttt{holds}(\psi(\underline{\mathbf{x}}), do(a, s))$ on the RHS, but we can get rid of it applying Reiter's regression operator $\mathcal{R}$, that takes a formula, instantiated at a successor state of the form $do(A, s)$, to a formula instantiated at the previous state, s. For doing this, it uses the SSAs of the tables appearing in the formula $\texttt{holds}(\psi(\underline{\mathbf{x}}), do(a, s))$ [11, 14]. We obtain:

$$\forall(a,s)Poss(a,s) \supset \forall\underline{\mathbf{x}}[R_{\Phi_{mtl}}(\underline{\mathbf{x}},do(a,s)) \equiv \mathcal{R}[\texttt{holds}(\psi(\underline{\mathbf{x}}),do(a,s))] \wedge \\ (R_{\Phi_{mtl}}(\underline{\mathbf{x}},s) \vee \texttt{holds}(\theta(\underline{\mathbf{x}}),s))]. \quad (15)$$

  Notice that the application of the regression operator leaves the RHS of the equivalence above as a simple formula in s. Also notice that when Φ_{mtl} is a MTL sentence, then the SC formula $R_{\Phi_{mtl}}(s)$ becomes a situation dependent propositional predicate. Finally, we also specify:

$$\forall\underline{\mathbf{x}}\ \neg R_{\Phi_{mtl}}(\underline{\mathbf{x}},S_0). \quad (16)$$

- Let Φ_{mtl} be a formula of the form $\psi(\underline{\mathbf{x}})$ $\textbf{since}_{\sim c}$ $\theta(\underline{\mathbf{x}})$, where $\psi(\underline{\mathbf{x}})$ and $\theta(\underline{\mathbf{x}})$ are non temporal formulas. The new table $R_{\Phi_{mtl}}(\underline{\mathbf{x}}, t, s)$ has the following SSA:

$$\forall(a,s)Poss(a,s) \supset \forall(\underline{\mathbf{x}},t)[R_{\Phi_{mtl}}(\underline{\mathbf{x}},t,do(a,s)) \equiv \mathcal{R}[\texttt{holds}(\psi(\underline{\mathbf{x}}),do(a,s))] \wedge \\ ((\texttt{holds}(\theta(\underline{\mathbf{x}}),s) \wedge t = \mathit{time}(a) - \mathit{start}(s)) \vee \\ \exists t'(R_{\Phi_{mtl}}(\underline{\mathbf{x}},t',s) \wedge t = t' + \mathit{time}(a) - \mathit{start}(s)))]. \quad (17)$$

  Finally, because S_0 has no predecessor state, we define $R_{\Phi_{mtl}}(\underline{\mathbf{x}}, t, S_0)$ as follows:

$$\forall(\underline{\mathbf{x}},t)\ \neg R_{\Phi_{mtl}}(\underline{\mathbf{x}},t,S_0). \quad (18)$$

5 Extended Theories of Action

We showed in section 2 theories of action of the form $\Sigma_{bd} \cup T_{pos} \cup T_{S_0} \cup T_{una} \cup T_{ss}(T_{ef})$. In this section we propose a language for extended theories of action, which are given by unions of the form:

$$\Sigma_{bd} \cup ET_{pos} \cup T_{S_0} \cup T_{una} \cup T_{ss}(T_{ef}) \cup ET_{QIC} \cup ET_{RIC},$$

where ET_{pos} is a set of preconditions for history dependent actions, that is, actions whose executions depend on the history of the database, and ET_{QIC}, ET_{RIC} are set of qualification and ramification dynamic integrity constraints, respectively, written using the macros `holds` and MTL. We will explain these new components of action theories in the following subsections.

5.1 Non-Markovian Actions and Triggers

As we saw in section 2, the theories of actions contains preconditions for action executions that depend on the current state of the database, only. Many concepts and algorithms that have originated from this formalism are based on this kind of "Markovian" action precondition axioms. Nevertheless, there are natural scenarios in which the conditions for executing an action should depend on a longer history of the database: in a library is possible to lend a book to a user depend on his or her history, for example, his or her number of returned books out of time.

Using MTL and the *macros* `holds` is possible to express action preconditions that depend on the history of the database. Thus, we replace the set T_{pos} (see section 2) by a new set ET_{pos} that contains, for every action A, an axioms of the form:

$$Poss(A, s) \equiv \texttt{holds}(\Pi_A, s), \tag{19}$$

where Π_A is a MTL formula.

Example 1. Let us consider a library database, with the following action precondition axioms, ET_{pos}:

$$Poss(return(u, b, t), s) \equiv \texttt{holds}(Borrowed(u, b), s), \tag{20}$$

$$Poss(pay(u, t), s) \equiv \texttt{holds}(Late(u), s), \tag{21}$$

$$\begin{aligned} Poss(borrow(u, b, t), s) \equiv \texttt{holds}(&\neg Late(u) \wedge \neg\exists u' Borrowed(u', b) \wedge \\ &\neg\Diamond_{=0} Borrowed(u, b), s), \end{aligned} \tag{22}$$

The last sentence express that is not possible to lend a book twice to the same user during the same day.

□

Applying the definition of `holds` is possible to generate a new specification, where every action has an APA of the form presented in section 2. Thus, we can use all the methodologies originated in the "Markovian" transactions in our extended theories.

Example 2. (cont'd) In the library database, the axioms (20) and (21) can be directly translating into the following axioms:

$$Poss(return(u, b, t), s) \equiv Borrowed(u, b, s), \tag{23}$$

$$Poss(pay(u, t), s) \equiv Late(u, s), \tag{24}$$

Suppose that $\alpha(u, b)$ represents the sentence $\Diamond_{=0} Borrowed(u, b)$ in the axiom precondition (22). By definition of `holds` is necessary to introduce a new table $R_\alpha(u, b, t, S_0)$ with the following definition:

$$\neg R_\alpha(u, b, t, S_0). \tag{25}$$

$$\begin{aligned} Poss(a, s) \supset [R_\alpha(u, b, t, do(a, s)) \equiv (&Borrowed(u, b, s) \wedge time(a) - start(s) = t) \vee \\ &\exists t'(R_\alpha(u, b, t', s) \wedge time(a) - start(s) + t' = t)]. \end{aligned} \tag{26}$$

Thus, the precondition axiom for $borrow$ will change to:

$$\begin{aligned} Poss(borrow(u, b, t), s) \equiv \neg Late(u, s) \wedge \neg\exists u' Borrowed(u', b, s) \wedge \\ \neg\exists t'(R_\alpha(u, b, t', s) \wedge t' = 0). \end{aligned} \tag{27}$$

5.2 Dynamic Integrity Constraints

The elements of the sets ET_{QIC}, ET_{RIC} are integrity constraints written using the macros `holds` and MTL. More precisely, this constraints are of the form:

$$\forall s[S_0 \leq s \supset \texttt{holds}(\Phi_{mtl}, s)], \tag{28}$$

where Φ_{mtl} is a MTL formula.

5.2.1 Qualification Dynamic Integrity Constraints

5.2.2 Ramification Dynamic Integrity Constraints

Example 3. (cont'd) Suppose that in the library specification we have the following effect axioms, T_{ef}:

$$Poss(a, s) \wedge a = pay(u) \supset \neg Late(u, do(a, s)) \tag{29}$$

$$Poss(a, s) \wedge a = borrow(u, b, t) \supset Borrowed(u, b, do(a, s)) \tag{30}$$

$$Poss(a, s) \wedge a = return(u, b, t) \supset \neg Borrowed(u, b, do(a, s)) \tag{31}$$

Thus, we have the following SSAs, $T_{ss}(T_{ef})$:

$$Poss(a, s) \supset Late(u, do(a, s)) \equiv Late(u, s) \wedge a \neq pay(u) \tag{32}$$

$$\begin{aligned} Poss(a, s) \supset Borrowed(u, b, do(a, s)) \equiv a = borrow(u, b) \vee \\ Borrowed(u, b, s) \wedge a \neq return(u, b) \end{aligned} \tag{33}$$

Looking at the axiom (32), we note that we do not specify under what conditions a user u must be included in the table $Late$. Suppose that in the library one book must be returned by a user u seven days before its loan, after that this user must be added to the table $Late$. This information could be added to the specification by means of the following RIC:

$$\forall s[S_0 \leq s \supset \texttt{holds}(\forall (u,b)[(Borrowed(u,b)\ \mathbf{since}_{>7}\ (Borrowed(u,b) \wedge \\ \bigcirc \neg Borrowed(u,b))) \supset Late(u)], s)] \quad (34)$$

Using the definition of holds is possible to translate this sentence into a static integrity constraints [14]. More precisely, let $\beta(u,b) = \bigcirc \neg Borrowed(u,b)$, we need to introduce a new table $R_\beta(u,b,s)$, defined as follows:

$$\neg R_\beta(u,b,S_0), \quad (35)$$

$$Poss(a,s) \supset R_\beta(u,b,do(a,s)) \equiv \neg Borrowed(u,b,s). \quad (36)$$

If $\gamma(u,b) = Borrowed(u,b)\ \mathbf{since}_{>7}\ (Borrowed(u,b) \wedge R_\beta(u,b))$, then we define a new table $R_\gamma(u,b,t,s)$ as follows:

$$\neg R_\gamma(u,b,t,S_0), \quad (37)$$

$$\begin{aligned} Poss(a,s) \supset R_\gamma(u,b,t,do(a,s)) \equiv \mathcal{R}[Borrowed(u,b,do(a,s))] \wedge \\ (Borrowed(u,b,s) \wedge R_\beta(u,b,s) \wedge t = time(a) - start(s)) \vee \\ \exists t'(R_\gamma(u,b,t',s) \wedge t = t' + time(a) - start(s)). \end{aligned} \quad (38)$$

Applying the regression operator we obtain:

$$\begin{aligned} Poss(a,s) \supset R_\gamma(u,b,t,do(a,s)) \equiv \\ (a = borrow(u,b) \vee (Borrowed(u,b,s) \wedge a \neq return(u,b))) \wedge \\ (Borrowed(u,b,s) \wedge R_\beta(u,b,s) \wedge t = time(a) - start(s)) \vee \\ \exists t'(R_\gamma(u,b,t',s) \wedge t = t' + time(a) - start(s)). \end{aligned} \quad (39)$$

Thus, (34) may be translated into the following static integrity constraint:

$$\forall s[S_0 \leq s \supset \forall (u,b)(\exists t'(R_\gamma(u,b,t',s) \wedge t' > 7) \supset Late(u,s))]. \quad (40)$$

But this sentence may be rewritten as follows:

$$\forall s[S_0 \leq s \supset \forall (u,b,t')(\neg R_\gamma(u,b,t',s) \vee Late(u,s) \vee t' \leq 7)]. \quad (41)$$

This formula is a binary integrity constraint [10], therefore is possible to subsume it adding new action effects for the tables R_γ and $Late$. In fact, we add the following positive effect axiom for $Late$:

$$Poss(a,s) \wedge \mathcal{R}[R_\gamma(u,b,t',do(a,s))] \wedge t' > 7 \supset Late(u,do(a,s)), \quad (42)$$

that establish under which condition a user u must be added to the table $Late$, following the formula (34). And we add the following negative effect axiom for R_γ:

$$Poss(a,s) \wedge \mathcal{R}[\neg Late(u,do(a,s))] \wedge t' > 7 \supset \neg R_\gamma(u,b,t',do(a,s)). \quad (43)$$

Thus, using (29) and (42) we obtain the following SSA for $Late$:

$$Poss(a,s) \supset Late(u, do(a,s)) \equiv (\mathcal{R}[R_\gamma(u,b,t',do(a,s))] \wedge t' > 7) \vee (Late(u,s) \wedge a \neq pay(u)).$$

Replacing $\mathcal{R}[R_\gamma(u,b,t',do(a,s))]$ we obtain:

$$\begin{aligned}
&Poss(a,s) \supset Late(u, do(a,s)) \equiv \\
&\qquad [((a = borrow(u,b) \vee (Borrowed(u,b,s) \wedge a \neq return(u,b))) \wedge \\
&\qquad (Borrowed(u,b,s) \wedge R_\beta(u,b,s) \wedge t = time(a) - start(s)) \vee \\
&\qquad \exists t'(R_\gamma(u,b,t',s) \wedge t = t' + time(a) - start(s))) \wedge t' > 7] \vee [Late(u,s) \wedge a \neq pay(u)].
\end{aligned}$$

6 Temporal Logic as a Query Language

7 Discussion

References

[1] CHOMICKI95. chomicki95. chomicki95.

[2] HAAS, A. R. The Case for Domain-Specific Frame Axioms. In *The Frame Problem in Artificial Intelligence, Proceedings of the 1987 Workshop* (Los Altos, CA, 1987), F. M. Brown, Ed., Morgan Kaufmann Publishers, Inc., pp. 343–348.

[3] LIFSCHITZ, V. Formal Theories of Action. In *The Frame Problem in Artificial Intelligence* (Los Altos, California, Apr. 1987), F. M. Brown, Ed., Morgan Kaufmann Publishers, Inc., pp. 35–57.

[4] LIN, F., AND REITER, R. State Constraints Revisited. *Journal of Logic and Computation 4*, 5 (1994), 655–678. Special Issue on Actions and Processes.

[5] LIN, F., AND SHOHAM, Y. Provably Correct Theories of Action (preliminary report). In *Proceedings of the ninth National Conference on Artificial Intelligence (AAAI-91)* (1991), pp. 349–354.

[6] LIN, F., AND SHOHAM, Y. Provably Correct Theories of Action. Tech. rep., University of Toronto, 1992.

[7] MCCARTHY, J. Programs with Common Sense. In *Semantic Information Processing*, M. Minsky, Ed. The MIT Press, 1968, ch. 7, pp. 403–418.

[8] MCCARTHY, J., AND HAYES, P. J. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence* **4**, B. Meltzer and D. Michie, Eds. Edinburgh University Press, Edinburgh, Scotland, 1969, pp. 463–502.

[9] PEDNAULT, E. Synthesizing Plans that Contain Actions with Context-Dependent Effects. *Computational Intelligence 4* (1988), 356–372.

[10] PINTO, J. *Temporal Reasoning in the Situation Calculus.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, Feb. 1994. URL = ftp://ftp.cs.toronto.edu/~cogrob/jpThesis.ps.Z.

[11] REITER, R. *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression*. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. Academic Press, San Diego, CA, 1991, pp. 359–380.

[12] REITER, R. On Specifying Database Updates. Tech. Rep. KRR-TR-92-3, University of Toronto, July 1992.

[13] REITER, R. Proving Properties of States in the Situation Calculus. *Artificial Intelligence 64*, 2 (Dec. 1993), 337–351.

[14] REITER, R. On Specifying Database Updates. *Journal of Logic Programming 25*, 1 (1995), 53–91.

[15] REITER, R. Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference(KR'96)* (Cambridge, Massachussetts, U.S.A., Nov. 1996).

[16] SCHUBERT, L. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In *Knowledge Representation and Defeasible Reasoning*, H. Kyberg, R. Loui, and G. Carlson, Eds. Kluwer Academic Press, 1990, pp. 23–67.

[17] SHAPIRO, S. First-order Solutions to the Frame Problem. Master's thesis, University of Toronto, Oct. 1993.

[18] SHOHAM, Y. *Reasoning about Change*. The MIT Press, Cambridge, Mass., 1988.