Hypothetical Temporal Reasoning with History Encoding (Extended Abstract)

Marcelo Arenas and Leopoldo Bertossi
Pontificia Universidad Católica de Chile,
Escuela de Ingeniería,
Departamento de Ciencia de Computación,
Casilla 306, Santiago 22, Chile.
{marenas,bertossi}@ing.puc.cl

Abstract

In this paper we integrate a history–encoding based methodology developed by Chomicki for checking dynamic database integrity constraints into a situation-calculus based specification of database updates as proposed by Reiter. By doing this, we are able to: (1) Answer queries about a whole hypothetical evolution of a database, without having to update the entire database and keep all the information associated to the generated states, (2) State and prove dynamic integrity constraints as static integrity constraints, (3) Transform history dependent preconditions for updates into local preconditions.

Keywords: Situation Calculus, Knowledge Representation, Specifications of Database Updates, Automated Reasoning, Integrity Constraints.

1 Introduction

In [9], as an application of his solution to the frame problem [8], Raymond Reiter proposed to specify the transaction based updates of database by means of a particular kind of axioms written in the situation calculus (SC) [7]. In [1] the implementation and the functionalities of a computational system for doing automated reasoning from and about these specifications are reported.

We are motivated by the problem of answering queries about different states[1] in the evolution of a database, when the database is virtually updated by the execution of a sequence of primitive transactions. For example, we want to consider queries of the form "Has it always been the case that the database has satisfied a given condition C?", or "Has there been a state of the database where a certain condition C has been satisfied?", or "Has the salary of some employee decreased along the database evolution?". Reiter raised this problem in the context of his specifications of transaction based database updates [9].

Although there is no explicit time in our situation calculus, for their similarity with dynamic integrity constraints [9], also called "temporal constraints" [3], we call these queries "temporal queries"[2]. Furthermore, we call this queries "hypothetical" because we have an initial, physical database at an initial state S_0 and a list T of primitive transactions $A_1, \ldots, A_n$, that virtually update the database, producing new states $S_1, \ldots, S_n$; and we want to answer a query about the generated sequence of states, without physically updating the whole database accordingly (and possibly keeping the data for every intermediate state). We are interested in querying this whole virtual evolution of the database.

The problem of answering this kind of queries was treated in detail in [12] and a solution was implemented as reported in [1]. Nevertheless that solution is based on a kind of minimal progression of the database that depended on a detailed syntactical processing of the axioms of the specification, and on the particular syntactical form of them.

In this paper we reconsider this problem and we propose a new solution that relies on processing the query itself, rather than the underlying axioms. This is done on the basis of a reformulation of Chomicki's history encoding methodology for efficiently checking temporal integrity constraints [3], in the context of SC specifications of database updates, and, in particular, a specification in the SC of the evolution of new history encoding auxiliary relations that are generated from the query.

It turns out that the methodology we develop for answering queries can be adapted to give a solution to other reasoning problems. Here we show how to transform dynamic integrity constraints into static integrity constraints, so that any available methodology for handling static integrity constraints

[1] In this paper we do no make any distinction between states and situations.

[2] In some papers, they are called "historical queries", but this name may cause confusions with work done by the temporal databases community that calls "historical" the queries about valid time, rather than about transaction time [13], which can be better associated to the situations of the situation calculus.

can be adapted for the dynamic case. In particular, we can take advantage of our results on automated proving of static integrity constraints [2] when dealing with the dynamic case. The other problem we solve consists in transforming preconditions for action executions that depend on the history of the database into preconditions that depend on the local, execution state.

2 The Situation Calculus and Database Updates

Characteristic ingredients of a particular language $\mathcal{L}$ of the situation calculus, besides the usual symbols of predicate logic, are: (1) Sorts *action*, *situation*, and *individual* (this last one for the individuals in the domain; this sort could be split into subsorts if necessary); (2) Predicate symbols of the sort (*individual*,... , *individual*, *situation*) to denote fluents. These are predicates that depend on the state of the world. (3) Operation symbols of the sort (*individual*,... , *individual*) $\rightarrow$ *action* for denoting actions with individuals as parameters (or applied to individuals). (4) A constant, S_0, to denote the initial state; (5) An operation symbol *do* of sort $(action, situation) \rightarrow situation$, that executes an action at a given state producing a successor state. In these languages there are first order variables for individuals of each sort, so it is possible to quantify over individuals, actions, and situations. They are usually denoted by $\forall \bar{x}$, $\forall a$, $\forall s$, respectively.

The specification of a dynamically changing world, by means of an appropriate language of the situation calculus, consists in stating the laws of evolution of the world. This is typically done by specifying: (1) Fixed, state independent, but domain dependent knowledge about the individuals of the world; (2) Knowledge about the state of the world at the initial situation S_0 given in terms of formulas that do not mention any state besides S_0; (3) Preconditions for performing the different actions (or making their execution possible). We introduce a predicate *Poss* in $\mathcal{L}$ of sort $(action, situation)$, so that $Poss(a, s)$ says that the execution of action a is possible in state s; (4) The effects of the actions in the fluents. These information is giving by means of laws that describe what happen with the different fluents when we execute an action a in a state s.

More precisely, a specification of a dynamically changing world Σ is giving by the following set of axioms: (1) A set Σ_b, which includes an induction axiom on states $\forall P[P(S_0) \wedge \forall(a, s)(P(s) \supset P(do(a, s))) \supset \forall s P(s)]$, and unique name axioms for states: $\forall(a, a', s, s')[do(a, s) = do(a', s') \supset a = a' \wedge s = s']$, $\forall(a, s)[do(a, s) \neq S_0]$. (2) A set Σ_{una} of unique name axioms for axiom: for all different action names A,B : $\forall(\bar{x}, \bar{y})[A(\bar{x}) \neq B(\bar{y})]$, and for every action A : $\forall(\bar{x}, \bar{x}')[A(\bar{x}) = A(\bar{x}') \supset \bar{x} = \bar{x}']$. (3) A set Σ_{S_0} of sentences that do not mention any state term other than S_0. (4) A set Σ_{pos} that includes for each action name A, a precondition axiom of the form:

$$\forall(\bar{x}, s)[Poss(A(\bar{x}), s) \equiv \pi_A(\bar{x}, s)], \tag{1}$$

where $\pi_A(\bar{x}, s)$ is a SC formula that is *simple in* s, that is, it contains no state term other than s, in particular, no *do* symbol, no quantifications on states, and no occurrences of the *Poss* predicate [9]. (5) Successor State Axioms: For every fluent $F(\bar{x}, s)$, an axiom of the form:

$$\forall(a, s)Poss(a, s) \supset \forall\bar{x}[F(\bar{x}, do(a, s)) \;\equiv\; \Phi_F(\bar{x}, a, s)], \tag{2}$$

where Φ_F is a formula simple in s, in particular, it does not contain the *do* symbol. Provided there is complete knowledge at the initial state, this axiom completely determines the contents of fluent F at an arbitrary legal state, i.e. reached from S_0 by a finite sequence of transactions that are possible at their execution states. (6) Finally, we will be usually interested in reasoning about states that are accessible from the initial situation by executing a finite sequence of legal actions. This *accessibility relation* on states, $\leq$, can be defined from the induction axiom plus the conditions (*less-axioms*): $\neg s < S_0,\;\; s < do(a, s') \equiv Poss(a, s') \wedge s \leq s'$.

Example 1. We consider a database of a company, with the following fluent $Emp(x, s)$: Person x is an employee of the company when the database is in state s, and with the following actions: (1) $hire(x)$: Person x is hired by the company. (2) $fire(x)$: Person x is fired by the company. In this specification there exists the following successor state axiom:

$$\begin{aligned}\forall(a, s)Poss(a, s) \supset \forall x[Emp(x, do(a, s)) \equiv \\ a = hire(x) \vee Emp(x, s) \wedge a \neq fire(x)],\end{aligned}$$

and there are the following action precondition axioms:

$$\begin{aligned}&\forall(x, s)[Poss(hire(x), s) \equiv \neg Emp(x, s)],\\ &\forall(x, s)[Poss(fire(x), s) \equiv Emp(x, s)].\end{aligned}$$

□

In the context of these specifications, a temporal query is a SC sentence φ in which all the states involved, including quantified states, lie on a finite state path $S_0 \leq S_1 \leq \cdots \leq S_n$, with $S_i = do([A_1, \ldots, A_{i-1}, A_i],, S_0)^3$, for a sequence of ground actions terms $A_1, \ldots, A_n$, for some n. The query is true if and only if $\Sigma \models \varphi$.

3 Answering Queries

Chomicki [3] considered the problem of checking temporal constraints stated in past first order temporal logic. These are constraints that talk about, and relate, different states of the database. There we find a sequence of transactions that are physically executed, and in order to minimize the cost of checking, one progressively updates new defined relations, or auxiliary views, r_α's, that correspond to the temporal subformulas, α's, in the constraint. These views encode part of the database evolution up to a current database state. They are defined and updated in such a way that they store the historical information that is relevant to give an answer to the query about the satisfaction of the integrity constraint once the final (current) state is reached. Then a new, non temporal, but local and static query can be posed at the final state.

In [9], Reiter showed how to evaluate a query $\varphi(s)$ in a state resulting from the execution of a sequence of actions $A_1, \ldots, A_n$, when $\varphi(s)$ is a formula that is simple in s. Using the methodology described in the previous paragraph, we will show in this section how to evaluate a more general syntactical form of queries, that is, queries that include quantifications over states. Thus, in this section we will show how to answer certain kind of temporal queries.

Definition 1 *$\varphi(\bar{x}, s)$ is a s-bounded formula iff*

1. *$\varphi(\bar{x}, s)$ is a formula simple in s.*
2. *$\varphi(\bar{x}, s) = \neg\psi(\bar{x}, s)$, where $\psi(\bar{x}, s)$ is a formula s-bounded.*
3. *$\varphi(\bar{x}, s) = \psi(\bar{x}, s) * \theta(\bar{x}, s)$, where $* \in \{\wedge, \vee, \supset, \equiv\}$, $\psi(\bar{x}, s)$ and $\theta(\bar{x}, s)$ are formulas s-bounded.*
4. *$\varphi(\bar{x}, s) = (Qx)\psi(\bar{x}, x, s)$, where $Q \in \{\forall, \exists\}$, $\psi(\bar{x}, x, s)$ is a formula s-bounded and x is a no state variable.*
5. *$\varphi(\bar{x}, s) = \exists(a, s')(S_0 \leq s' < s \wedge s = do(a, s') \wedge \psi(\bar{x}, s))$, where $\psi(\bar{x}, s')$ is a formula s'-bounded.*

[3] We use the notation $do([A_1, \ldots, A_{i-1}, A_i], S_0)$ for abbreviating $do(A_i, do(A_{i-1}, do(\ldots, do(A_1, S_0) \ldots)))$.

6. $\varphi(\bar{x}, s) = \exists s'(S_0 \leq s' < s \wedge \psi(\bar{x}, s') \wedge \forall s''(s' < s'' \leq s \supset \theta(\bar{x}, s'')))$, *where* $\psi(\bar{x}, s')$ *is a formula* s'*-bounded and* $\theta(\bar{x}, s'')$ *is a formula* s''*-bounded.*

Although the formulas defined above may seem to form a strange class of SC formulas, they are an interesting generalization of the formulas simple. They can represent common temporal queries like "Has it always been the case that the database has satisfied a given condition φ?" and "Has there been a state of the database where a certain condition φ is true?":

$$\begin{aligned} &\forall s'(S_0 \leq s' < s \supset \varphi(s')) \equiv \\ &\qquad \neg \exists s'(S_0 \leq s' < s \wedge \neg\varphi(s') \wedge \forall s''(s' < s'' \leq s \supset \mathit{True})), \end{aligned} \tag{3}$$

$$\begin{aligned} &\exists s'(S_0 \leq s' < s \wedge \varphi(s')) \equiv \\ &\qquad \exists s'(S_0 \leq s' < s \wedge \varphi(s') \wedge \forall s''(s' < s'' \leq s \supset \mathit{True})). \end{aligned} \tag{4}$$

Now, we want to evaluate s-bounded queries. Our starting point consists of a SC specification Σ and a s-bounded sentence φ to be evaluated at the final state $S = S_n = do(T, S_0)$, that results from the virtual update of the database. So, φ refers to the states between S_0 and S. What we want to obtain is a new SC specification Σ_{bts} that extends Σ, and a sentence $bts[\varphi(S)]$, such that the answer to the original query coincides with the answer obtained from the evaluation of $bts[\varphi(S)]$ wrt Σ_{bts}. The new sentence refers only to the state S, and Σ_{bts} contains a specification of the dynamics of new, history encoding, auxiliary relations. We will define first the operator bts and next Σ_{bts}.

Definition 2

1. $bts[\varphi(\bar{x}, s)] = \varphi(\bar{x}, s)$, *If* $\varphi(\bar{x}, s)$ *is a simple formula in* s.

2. $bts[\neg\varphi(\bar{x}, s)] = \neg bts[\varphi(\bar{x}, s)]$.

3. $bts[\varphi(\bar{x}, s) * \psi(\bar{x}, s)] = bts[\varphi(\bar{x}, s)] * bts[\psi(\bar{x}, s)]$, *where* $* \in \{\wedge, \vee, \supset, \equiv\}$.

4. $bts[(Qx)\varphi(\bar{x}, s)] = (Qx)bts[\varphi(\bar{x}, s)]$, *where* $Q \in \{\forall, \exists\}$ *and* x *is a no state variable.*

5. $bts[\exists(a, s')(S_0 \leq s' < s \wedge s = do(a, s) \wedge \psi(\bar{x}, s'))] = R_\alpha(\bar{x}, s)$, *where* $\psi(\bar{x}, s')$ *is a formula simple in* s' *and* R_α *is a new fluent name.*

6. $bts[\exists s'(S_0 \leq s' < s \wedge \psi(\bar{x}, s') \wedge \forall s''(s' < s'' \leq s \supset \theta(\bar{x}, s'')))] = R_\alpha(\bar{x}, s)$, *where* $\psi(\bar{x}, s')$ *is a formula simple in* s', $\theta(\bar{x}, s'')$ *is a formula simple in* s'' *and* R_α *is a new fluent name.*

By bottom–up transformation of a s-bounded formula φ, we can always obtain such a formula $bts[\varphi]$. Notice that this is a SC formula that is simple in the state s, it talks about the isolated state, s. Now we have to specify the dynamics of the new fluents by means of successor state axioms. Let $\varphi(\bar{x}, s)$ be of the form $\exists(a, s')(S_0 \leq s' < s \wedge s = do(a, s') \wedge \psi(\bar{x}, s'))$. This formula is true at a state s, with predecessor state s', iff ψ was true at s'. Then, for $R_\alpha(\bar{x}, s)$ we have the following successor state axiom:

$$\forall(a, s) Poss(a, s) \supset \forall \bar{x}[R_\alpha(\bar{x}, do(a, s)) \equiv \psi(\bar{x}, s)]. \tag{5}$$

Additionally, we also specify $\forall \bar{x}[\neg R_\alpha(\bar{x}, S_0)]$.

Let $\varphi(\bar{x}, s)$ be of the form $\exists s'(S_0 \leq s' < s \wedge \psi(\bar{x}, s') \wedge \forall s''(s' < s'' \leq s \supset \theta(\bar{x}, s'')))$. This formula is true at a state s, with predecessor state s_1, iff φ was true at s_1 and θ is still true at s, or θ became true at s and ψ became true at s_1. This is equivalent to saying that $\varphi \vee \psi$ is true at s_1 and θ is true at s. Then, for $R_\alpha(\bar{x}, s)$ it holds for every legal action a: $\forall \bar{x}[R_\alpha(\bar{x}, do(a, s)) \equiv bts[\theta(\bar{x}, do(a, s))] \wedge (R_\alpha(\bar{x}, s) \vee bts[\psi(\bar{x}, s)])]$. This is not a successor state axiom (of the form (2)), because there is a $do(a, s)$ term in $bts[\theta(\bar{x}, do(a, s))]$ on the right hand side. 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 $do(A, s)$, into a formula instantiated at the previous state, s. For doing this, $\mathcal{R}$ uses the successor state axioms of the tables appearing in $bts[\theta(\bar{x}, do(a, s))]$ [8, 9]. So, we obtain:

$$\begin{aligned}\forall(a, s) Poss(a, s) \supset \forall \bar{x}[R_\alpha(\bar{x}, do(a, s)) \equiv \\ \mathcal{R}[bts[\theta(\bar{x}, do(a, s))]] \wedge (R_\alpha(\bar{x}, s) \vee bts[\psi(\bar{x}, s)])].\end{aligned} \tag{6}$$

Notice that the application of the regression operator leaves the right hand side of the equivalence above as a simple formula in s. Also notice that when α is a s-bounded sentence, then the SC formula $R_\alpha(s)$ becomes a situation dependent propositional predicate. Finally, we also specify $\forall \bar{x}[\neg R_\alpha(\bar{x}, S_0)]$. The new specification Σ_{bts} contains for each new fluent R_α the previous axiom and (6).

Example 2. Assume that the original specification Σ contains the following successor state axiom for the table $P(x, s)$:

$$\forall(a, s) Poss(a, s) \supset \forall x[P(x, do(a, s)) \equiv a = A(x) \vee (P(x, s) \wedge a \neq B(x))].$$

We want to evaluate the query:

$$\exists s_1(S_0 \leq s_1 < s \wedge \exists s_2(S_0 \leq s_2 < s_1 \wedge Q(x, s_2)) \wedge \forall s_3(s_1 < s_3 \leq s \supset P(x, s))).$$

If β is $\exists s_2(S_0 \leq s_2 < s_1 \wedge Q(x,s))$, then we introduce a new fluent R_β. Using (4), we have the following definition for this fluent:

$$\forall x[\neg R_\beta(x,S_0)],\ \ \forall(a,s)Poss(a,s) \supset \forall x[R_\beta(x,do(a,s)) \equiv R_\beta(x,s) \vee Q(x,s)].$$

Introducing R_β in the query, we obtain $\exists s_1(S_0 \leq s_1 < s \wedge R_\beta(x,s_1) \wedge \forall s_3(s_1 < s_3 \leq s \supset P(x,s)))$. If this formula is $\alpha(x,s)$, for the new table R_α we have:

$$\begin{aligned}&\forall(a,s)Poss(a,s) \supset \forall x[R_\alpha(x,do(a,s)) \equiv \\ &\qquad\qquad\qquad \mathcal{R}[P(x,do(a,s))] \wedge (R_\alpha(x,s) \vee R_\beta(x,s))].\end{aligned}$$

Replacing $\mathcal{R}[P(x,do(a,s))]$ by the right hand side of the successor state axiom for P, we obtain:

$$\begin{aligned}&\forall(a,s)Poss(a,s) \supset \forall x[R_\alpha(x,do(a,s)) \equiv \\ &\qquad (a = A(x) \vee P(x,s) \wedge a \neq B(x)) \wedge (R_\alpha(x,s) \vee R_\beta(x,s))]\end{aligned} \tag{7}$$

Additionally, it is necessary to add the axiom: $\forall x[\neg R_\alpha(x,S_0)]$. The new specification Σ_{bts} consists of Σ plus the specifications of the new fluents R_α and R_β. □

Now we want to answer a SC temporal query ψ posed to a virtual trajectory of the database from S_0 to $S_n = do(T,S_0)$ (as before): $\Sigma \models \psi$? If ψ is a formula s-bounded, then we can answer this query by asking: $\Sigma_{bts} \models bts[\psi(do(T,S_0))]$. The following theorem formalizes this idea:

Theorem 1 *Suppose that $\psi(s)$ is a formula s-bounded, and that the state variable s is the only free variable of $\psi(s)$. Suppose that $A_1,\ldots,A_n$ is a legal sequence of ground terms of sort action. Then:*

$$\Sigma \models \psi(do([A_1,\ldots,A_n],S_0)) \quad \textit{iff} \quad \Sigma_{bts} \models bts[\psi(do([A_1,\ldots,A_n],S_0))].$$

Notice that $bts[\varphi]$ is instantiated at the final state $do(T,S_0)$, and this is the only state mentioned in the formula. So, we can see that we have transformed our problem of answering a temporal query wrt to a virtually updated database into the *temporal projection problem* of AI [4], that is, the problem of querying a future state obtained by the execution of an actions sequence. To solve this problem we may apply some existing techniques for Reiter like specifications, e.g. Reiter's query regression [9], minimal rolling forward of the database based on information that is relevant to the query [1, 12], or even full progression of the database [6]. All these methodologies are supported by our reasoner SCDBR [1].

Example 3. We want to know if there is someone who has always been working in the company (see example 1), in all states generated by the execution of the sequences of actions $T = [\mathit{hire}(\mathit{Sue}), \mathit{fire}(\mathit{John})]$ from the initial situation. So, we are asking whether

$$\Sigma \models \exists x \forall s (S_0 \leq s \leq do(T, S_0) \supset \mathit{Emp}(x, s)).$$

It is easy to see that this sentence is equivalent to the following $do(T, S_0)$-bounded sentence:

$$\exists x (\mathit{Emp}(x, do(T, S_0)) \wedge \forall s (S_0 \leq s < do(T, S_0) \supset \mathit{Emp}(x, s)).$$

Applying our methodology and (3) we obtain the new SC query:

$$\exists x (\mathit{Emp}(x, s) \wedge R_\alpha(x, s)),$$

and the original specification extended to Σ_{bts} by adding: $\forall x[R_\alpha(x, S_0)]$ and:

$$\forall (a, s) \mathit{Poss}(a, s) \supset \forall x [R_\alpha(x, do(a, s)) \equiv \mathit{Emp}(x, s) \wedge R_\alpha(x, s)].$$

Then we ask if $\Sigma_{bts} \models \exists x (\mathit{Emp}(x, do(T, S_0)) \wedge R_\alpha(x, do(T, S_0)))$. Running the regression procedure in SCDBR, that applies twice the regression operator on the right hand side to the successor state axiom of *Emp*, and simplifies the resulting steps by means of the unique axioms for actions, we obtain the following query to be posed to the initial database:

$$\exists x ((x = \mathit{Sue} \vee \mathit{Emp}(x, S_0)) \wedge x \neq \mathit{John} \wedge \mathit{Emp}(x, S_0) \wedge R_\alpha(x, S_0)).$$

4 Dynamic Integrity Constraints are Static

A Static Integrity Constraint is a formula $\varphi(s)$, simple in the state variable s and without further free variables, such that[9, 5]:

$$\Sigma \models \forall s \ (S_0 \leq s \supset \varphi(s)).$$

A Dynamic (or temporal) Integrity Constraint is a SC sentence ψ of the form $\forall s_1 \cdots \forall s_n (C(S_0, s_1, \ldots, s_n) \supset \varphi(s_1, \ldots, s_n))$, that should be entailed by Σ. Here, $C(S_0, s_1, \ldots, s_n)$ is a formula that imposes a linear order constraint on the states $S_0, s_1, \ldots, s_n$ in terms of the accessibility predicate $\leq$[4].

[4]Dynamic integrity constraints are usually of this form, but the results we will present in this paper still hold if we admit more involved quantifications on several states, related by the accessibility relation.

We can use our methodology to transform dynamic integrity constraints into static integrity constraints. Actually, Chomicki's work [3] has to do with *checking* dynamic integrity constraints statically. In our case, with our reformulation of Chomicki's methodology in terms of a specification of the dynamics of the history encoding relations, we can rewrite dynamic integrity constraints as SC sentences expressing static integrity constraints, which can be *proven* as such from the (extended) specification of the database dynamics. In particular, we can use the methodology for proving static integrity constraints by automated induction presented in [2], and possibly others, like [11].

Let ψ be a dynamic integrity constraint expressed in the SC. As integrity constraint, it is expected to be true at every particular legal state s of the database. Nevertheless, for being ψ dynamic, we can find quantifications on previous states, and then, it does not talk about the current state only. In general, we will think a dynamic integrity constraint as a sentence talking about several states, but that has to be satisfied locally, at every legal state. On the basis of this assumption, we generate from ψ a new SC constraint $\forall s(S_0 \leq s \supset T(\psi, s))$, where the operator $T(\psi, s)$ is defined as follows:

Definition 3 *1. $T(\varphi, s) = \varphi$, if φ is an atomic formula.*

2. $T(\neg\varphi, s) = \neg T(\varphi, s)$.

3. $T(\varphi * \psi, s) = T(\varphi, s) * T(\psi, s)$, where $* \in \{\wedge, \vee, \supset, \equiv\}$.

4. $T((Qx)\varphi, s) = (Qx)T(\varphi, s)$, where $Q \in \{\forall, \exists\}$ and x is a no state variable.

5. $T(\forall s'\varphi, s) = \forall s'(s' \leq s \supset T(\varphi, s))$.

6. $T(\exists s'\varphi, s) = \exists s'(s' \leq s \wedge T(\varphi, s))$.

Intuitively, $T(\psi, s)$ is obtained from ψ by relativising all the quantifications over states to $\leq s$. This new formula satisfies the following property:

Lemma 1 *If ψ is a dynamic integrity constraint in a specification Σ, then:*

$$\Sigma \models \psi \quad \textit{iff} \quad \Sigma \models \forall s(S_0 \leq s \supset T(\varphi, s)).$$

If $T(\varphi, s)$ is a formula s-bounded, then is possible to transform the original dynamic integrity constraint into a new static integrity constraint using the operator bts: $\forall s(S_0 \leq s \supset bts[T(\varphi, s)])$, the correctness of this methodology is ensure by the following theorem:

Theorem 2 *Suppose that ψ is a dynamic integrity constraint in a specification Σ. If $T(\psi, s)$ is a formula s-bounded, then:*

$$\Sigma \models \psi \quad \textit{iff} \quad \Sigma_{bts} \models \forall s(S_0 \leq s \supset bts[T(\psi, s)]).$$

Example 4. Consider a specification Σ of a DB of employees with a salary table $Sal(e, m, s)$, where e is the employee, m is the salary, and s is the situational argument. Let ψ be the usual dynamic integrity constraint telling that an employee's salary cannot decrease:

$$\begin{aligned}&\forall(s', s'')(S_0 \leq s' < s'' \supset \\ &\qquad \forall(e, m', m'')((Sal(e, m', s') \ \wedge Sal(e, m'', s'')) \supset m' \leq m''))\end{aligned} \quad (8)$$

This sentence must hold at every legal current state s of the DB, that is, as a sentence, it must be entailed by Σ. Now consider a new formula equivalent to $T(\psi, s)$:

$$\begin{aligned}&\forall(s', s'')(S_0 \leq s' < s'' \leq s \supset \\ &\qquad \forall(e, m', m'')((Sal(e, m', s') \wedge Sal(e, m'', s'')) \supset m' \leq m''))\end{aligned}$$

Sentence (8) holds in every legal state if and only if from Σ follows $\forall s(S_0 \leq s \supset T(\psi, s))$. Now, $T(\psi, s)$ is equivalent to the following formula s-bounded:

$$\begin{aligned}&\forall(e, m', m'')((\exists s'(S_0 \leq s' < s \wedge Sal(e, m', s')) \wedge Sal(e, m'', s)) \supset m' \leq m'') \wedge \\ &\forall s''(S_0 \leq s'' < s \supset \forall(e, m', m'')((\exists s'(S_0 \leq s' < s'' \wedge \\ &\qquad Sal(e, m', s')) \wedge Sal(e, m'', s'')) \supset m' \leq m'')).\end{aligned} \quad (9)$$

If $\alpha(e, m', s)$ is $\exists s'(S_0 \leq s' < s \wedge Sal(e, m', s'))$, we create a new fluent R_α with definition:

$$\begin{aligned}&\forall(e, m')[\neg R_\alpha(e, m', S_0)], \\ &\forall(a, s)Poss(a, s) \supset \forall(e, m')[R_\alpha(e, m', do(a, s)) \equiv R_\alpha(e, m', s) \vee Sal(e, m', s)].\end{aligned}$$

Introducing R_α in (9) we obtain

$$\begin{aligned}&\forall(e, m', m'')((R_\alpha(e, m', s) \wedge Sal(e, m'', s)) \supset m' \leq m'') \wedge \\ &\forall s''(S_0 \leq s'' < s \supset \forall(e, m', m'')((R_\alpha(e, m', s'') \wedge Sal(e, m, s'')) \supset m' \leq m'')).\end{aligned} \quad (10)$$

If β is the temporal subformula

$$\forall s''(S_0 \leq s'' < s \supset \forall(e, m', m'')((R_\alpha(e, m', s'') \wedge Sal(e, m, s'')) \supset m' \leq m'')),$$

we create a table R_β with specification: $R_\beta(S_0)$,

$$\forall(a,s)Poss(a,s) \supset R_\beta(do(a,s)) \equiv \\ R_\beta(s) \wedge \forall(e,m',m'')((R_\alpha(e,m',s) \wedge Sal(e,m'',s)) \supset m' \leq m'').$$

Introducing R_β in (10), we obtain:

$$\forall(e,m',m'')((R_\alpha(e,m',s) \wedge Sal(e,m'',s)) \supset m' \leq m'') \wedge R_\beta(s).$$

Thus, the original dynamic integrity constraint holds in every state if and only if the specification Σ_{bts}, consisting of Σ plus the specifications of the new fluents R_α and R_β, entails the following static integrity constraint:

$$\forall s(S_0 \leq s \supset [\forall(e,m',m'')((R_\alpha(e,m',s) \wedge Sal(e,m'',s)) \supset m' \leq m'') \wedge R_\beta(s)]).$$

5 History Dependent Transactions

As we saw in section 2, the formalism for specifying DB updates 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 local 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. For example, in a voters database we might have the following action precondition axiom:

$$\forall(x,s)[Poss(vote(x),s) \equiv \\ \exists n(Age(x,n,s) \wedge n \geq 18) \wedge \forall s'(S_0 \leq s' \leq s \supset \neg InJail(x,s'))]$$

That is, x can vote if is not younger than 18, and he (she) has never been in jail. This is a history dependent transaction. We can use the machinery developed so far for transforming these kind of transactions into local transactions. We start from a database specification Σ with an action precondition axiom for the primitive transaction A of the form: $\forall(\bar{x},s)[Poss(A(\bar{x}),s) \equiv \psi(\bar{x},s)]$, where $\psi(\bar{x},s)$ is not necessarily simple in s, but has all the states mentioned in it, quantified or not, lying between S_0 and s. If ψ is a formula s-bounded, then we can generate a new specification Σ'_{bts} from Σ_{bts}, in its turn obtained from $bts[\psi(\bar{x},s)]$ as before, but with the original action precondition axiom replaced by the new action precondition axiom: $\forall(\bar{x},s)[Poss(A(\bar{x}),s) \equiv bts[\psi(\bar{x},s)]]$, which is of the form (1). As before, the new specification contains successor state axioms for the auxiliary tables introduced by the construction of $bts[\psi(\bar{x},s)]$.

Theorem 3 *Let Σ be a SC specification containing a history dependent action precondition axiom for action A and let Σ'_{bts} be the new SC specification containing successor state axioms for the auxiliary relations and the old action precondition axioms replaced by the new local one. If $\leq_{bts}$ and $Poss_{bts}$ are the possibility predicate and accessibility relation defined on the basis of the new action precondition axiom, then it holds:*

1. *For every ground state term S, and ground action term of the form $A(\bar{c})$, $\Sigma \models S_0 \leq S \supset Poss(A(\bar{c}), S)$ if and only if $\Sigma'_{bts} \models S_0 \leq_{bts} S \supset Poss_{bts}(A(\bar{c}), S)$.*

2. *For every ground state term S, $\Sigma \models S_0 \leq S$ if and only if $\Sigma'_{bts} \models S_0 \leq_{bts} S$.*

The proposition says that at every accessible state, action A is possible in the old sense if and only if it is possible in the new sense and that both specifications define the same accessible states.

Example 5. We can apply the methodology to the voters example. In the original action precondition axiom for $vote(x)$, $\forall s'(S_0 \leq s' < s \supset \neg InJail(x, s'))$ is a formula s-bounded. So, we generate a new specification Σ'_{bts}, extending Σ, except for the fact that it has: (1) A new table $R_\alpha(x, s)$ that contains, at state s, the people x that have not been in jail before state s, whose specification consists of $\forall x[R_\alpha(x, S_0)]$, $\forall(a, s)Poss(a, s) \supset \forall x[R_\alpha(x, do(a, s)) \equiv R_\alpha(x, s) \wedge \neg InJail(x, s)]$. (2) The original action precondition axiom for action A was replaced by:

$$\forall(x, s)[Poss(vote(x), s) \equiv \\ \exists n(Age(x, n, s) \wedge n \geq 18) \wedge \neg InJail(x, s) \wedge R_\alpha(x, s)].$$

6 Conclusions

Among the contributions in this paper we find the following: (1) An extension of Chomicki's methodology to the case in which there is a specification of the evolution of the database; and so (2) The possibility of doing hypothetical reasoning along a virtual evolution of the database (Chomicki concentrates on physical updates of the database) and this with user defined primitive transactions; (3) A solution to the problem of answering temporal queries in the context of Reiter's specifications of database updates, and this solution works both in a progressive as in a regressive way; (4) A transformation mechanism of dynamic integrity constraints into static integrity constraints, in a context like Reiter's, where both kind of constraints are expected to be logical consequences of the specification; and (5) A mechanism for transforming history dependent preconditions for action executions into preconditions to be evaluated at the execution state.

It is matter of our current research an extension of the methodology of this paper that includes metric time as in [3]. This is done by considering actions parameterized with explicit time[10].

Acknowledgments: This research has been partially supported by a FONDECYT Grant (# 1971304). Part of this work has been done during the second author's sabbatical at the TU Berlin. He is grateful to Ralf Kutsche and the CIS group for their support and hospitality; and to the GK "Distributed Information Systems", the DAAD and the DIPUC for their financial support. We are grateful to Javier Pinto for his generous support.

References

[1] L. Bertossi, M. Arenas, and C. Ferretti. SCDBR: An Automated Reasoner for Specifications of Database Updates. *Journal of Intelligent Information Systems*, 10(3), 1998.

[2] L. Bertossi, J. Pinto, P. Saez, D. Kapur, and M. Subramaniam. Automating Proofs of Integrity Constraints in the Situation Calculus. In *Foundations of Intelligent Systems. Proc. Ninth International Symposium on Methodologies for Intelligent Systems (ISMIS'96), Zakopane, Poland*, pages 212–222. Springer, LNAI 1079, 1996.

[3] J. Chomicki. Efficient Checking of Temporal Integrity Constraints Using Bounded History Encoding. *ACM Transactions on Database Systems*, 20(2):149–186, June 1995.

[4] S. Hanks and D. McDermott. Default Reasoning, Nonmonotonic Logics, and the Frame Problem. In *Proc. National Conference on Artificail Intelligence*, pages 328–333, 1986.

[5] F. Lin and R. Reiter. State Constraints Revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994. Special Issue on Action and Processes.

[6] F. Lin and R. Reiter. How to Progress a Database. *Artificial Intelligence*, 92(1–2):131–167, 1997.

[7] J. McCarthy and P. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502, Edinburgh, Scotland, 1969. Edinburgh University Press.

[8] R. Reiter. The Frame Problem in the Situation Calculus: a Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In V. Lifschitz,

editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

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

[10] R. Reiter. 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., November 1996.

[11] T. Sheard and D. Stemple. Automatic Verification of Database Transaction Safety. *ACM Transactions on Database Systems*, 14(3):322–368, 1989.

[12] B. Siu and L. Bertossi. Answering Historical Queries in Databases (Extended Abstract). In *Proc. XVI International Conference of the Chilean Computer Science Society (SCCC'96), M. V. Zelkowitz and P. Straub (eds.)*, pages 56–66, 1996.

[13] R. Snodgrass and I. Ahn. Temporal Databases. *IEEE Computer*, September:35–42, 1986.