An Assessment of Reiter's General Monotonic Solution to the Frame Problem and its Applicability to Database Updates

Tania Bedrax-Weiss*
CIRL
1269 University of Oregon
Eugene, OR 97403
e-mail: tania@cs.uoregon.edu

Leopoldo Bertossi
Pontificia Universidad Catolica de Chile,
Escuela de Ingenieria,
Departamento de Ciencia de la Computacion,
Casilla 306, Santiago 22, Chile.
e-mail: bertossi@ing.puc.cl

Abstract

Many solutions to the frame problem have been proposed in the literature, some are monotonic and some are nonmonotonic. Reiter recently proposed a solution and its application to the domain of database update specification, but no proof of soundness or completeness has been provided. We assess Reiter's monotonic solution to the frame problem in the database domain following Sandewall's methodology and we provide a more general underlying semantics that will serve as basis for the assessment of other solutions as well.

1 Introduction

Recently, Reiter [14] proposed a simple solution to the frame problem in the situation calculus. Reiter [15] has applied his formalism to the specification of updates in databases, that is, the specification of the different updates that can be performed in a domain-specific database. Since databases are dynamically changing worlds because of the transactions performed on them, and since in DBs we reason about states rather than about time, Reiter's formalism based on the situation calculus is naturally applicable to the DB domain. There is the need, thus, to analyze the range of applicability of Reiter's solution to such domain.

The analysis (assessment) will be based in a reasonable semantics for action. We know of two attempts to provide a general semantics. The first, Lin and Shoham's [6], proposes a method of assessment according to the epistemological completeness of the solutions. The second, Sandewall's underlying semantics [17, 16], is based not only on the epistemological assumptions that are made but also on the ontological assumptions. The main difference

*Part of this work was done while Tania Bedrax-Weiss was a graduate student at the Catholic University of Chile.

between Lin and Shoham's method and Sandewall's is that Lin and Shoham express a completeness criterion in terms of logical formulae and the associated monotonic theory, while Sandewall expresses it in terms of an independent underlying semantics that captures assumptions about the agent's level of perception, how actions affect the world, and considerations of logical completeness. We will work with Sandewall's proposed underlying semantics because we are convinced it provides solid grounds for assessment.

Some extensions to Sandewall's proposed semantics are needed in order to perform the assessment, since his semantics [17] only works for the simple cases (e.g., with complete specification of deterministic actions). These extensions and the assessment itself are among the main contributions of this work. The assessment shows that Reiter's solution is sound and complete for an interesting class of problems.

We start by reviewing Reiter's monotonic solution to the frame problem and its application to the domain of database updates. We then identify ontological and epistemological assumptions present in this solution and we will define the underlying semantics that corresponds to these assumptions. Based on this underlying semantics, we present the assessment. Finally, some conclusions and discussion of further work are given.

2 Reiter's Solution for Database Updates

Reiter [15] addresses the problem of formalizing, in the situation calculus, the evolution of a database under sequences of update transactions. He provides a solution for the frame problem under certain conditions, which include most update transactions.

The problem of formalizing the evolution of a database involves efficiently describing each state of the database that results from executions of arbitrary update transactions. Thus, updatable relations will be situation dependent. The specification will contain the description of the initial database (the tables at the initial situation), and all the possible update transactions that can be performed on it. The actual axiomatization is given below. Thus, querying the database involves querying about a situation that arises after the execution of a legal sequence of transactions. If the sequence of update transactions is not legal, nothing can be derived from the updated database.

To understand how the frame problem arises in this domain, let us review the first order language used —the situation calculus.

The characteristic ingredients for a situation calculus language, besides the usual symbols of first-order languages for domain descriptions, are: relation symbols for "fluents", that is, state dependent properties of objects (think of tables in the database domain); function symbols for "actions" that, applied to objects in given states may cause the fluents to change their truth values in the successor state that results from the application of the action (think of the atomic transactions in databases); and also symbols for explicitly denoting states.

The specification of a dynamically-changing world, by means of an appropriate language of the situation calculus, consists of stating in this language the laws of evolution of the world. This is typically done by specifying:

a. Fixed, state independent but domain dependent, knowledge about the objects of the world.

b. Knowledge about the state of the world at the initial situation S_0.[1] This is given in terms of formulas that do not mention any state other than S_0.

c. Preconditions for performing the different actions (or making their execution possible). We introduce a predicate *Poss* to say that the execution of an action is possible in a state.

[1] Lin and Reiter [5] allow in the background database, state constraints of the form $\forall s \varphi(s)$. They study the integration of state constraints to the solution presented in [14, 15].

d. The immediate effects of actions when they are possible to apply, in terms of the fluents whose truth values we know are changed (either positively or negatively) by the execution of the action.

Specifying databases and update transactions in the situation calculus involves treating atomic transactions as actions in dynamic worlds; this is where the frame problem arises. Suppose we have a library database, and one of the allowed transactions is to classify a book. There must be an efficient way to specify that when classifying a certain book, no other book gets classified, deleted or ordered, etc..

Reiter [14], presents a general and simple solution to the frame problem (FP) that is first-order and monotonic in the sense that it relies entirely in classical first order logic, in contrast with other proposals based on different nonmonotonic logics such as circumscription. Building on work by Haas [4], Pednault [9] and Schubert [18], he gives a preliminary specification as mentioned above and "solves" the frame problem by making a syntactic transformation of the given axioms. Thus, the final axiomatization (the specification after the syntactic transformation) lies entirely in the context of the initially chosen language and the first-order deductive system. The final specification is given in terms of successor-state axioms that completely describe each fluent at the states that result from executing actions.

In Section 2.1, we sketch Reiter's solution to the FP. In Section 2.2, we discuss how Reiter's approach can be seen as a logic, in the sense of defining an entailment criterion that captures the notion of model selection in Sandewall's terminology. We give an example in Section 2.3. In Section 3, we consider the fundamental ontological and epistemological assumptions that we have identified in Reiter's solution, focusing on the domain of DBs.

2.1 General Axiomatization

Initially we have the situation calculus with predicates (relations that do not change from state to state), fluents (relations that do change between states), actions (which may be parameterized), situations (variables or constant symbols), and a function symbol, *do*, which accepts an action and a situation argument and denotes the situation that results from executing the action in the situation that appears as an argument. The initial axioms include axioms for effects of actions under given preconditions. In addition to this language, one distinguished predicate, *Poss*, is introduced to specify the conditions under which, if the action is executed, the effects are observed. From this preliminary axiomatization[2] we derive the successor state axioms by a syntactic transformation of the effect axioms, using a meta-level nonmonotonic completeness assumption, so that the specification remains monotonic. These axioms completely specify which fluents are true at the end of the execution of an action.

The axioms that describe how new states are generated from previous states, i.e. the successor state axioms, can be obtained from the laws for effects of actions, as follows:

Positive Effects Laws: For all pairs formed by a fluent R and an action name A, where the action causes fluent R to become true, write an axiom of the form:

$$Poss(A(\bar{x}), s) \wedge \varepsilon_R^+(\bar{x}, \bar{y}, s) \rightarrow R(\bar{y}, do(A(\bar{x}), s)). \qquad (1)$$

Intuitively, if action A is possible, and the preconditions for fluent R represented by the metaformula $\varepsilon_R^+(\bar{x}, \bar{y}, s)$ are true at state s, then fluent R becomes true at the successor state $do(A(\bar{x}), s)$ obtained after executing action A in state s. Here, $\bar{x}, \bar{y}$ are parameters of the action and fluent.

[2] We avoided the use of the term initial axiomatization because it may be confused with the axiomatization of the initial state.

Negative Effects Laws: For all pairs formed by a fluent R and an action name A, where the action causes fluent R to become false, write an axiom of the form:

$$Poss(A(\bar{x}), s) \land \varepsilon_R^-(\bar{x}, \bar{y}, s) \rightarrow \neg R(\bar{y}, do(A(\bar{x}), s)). \tag{2}$$

Note that we have not axiomatized the predicate $Poss$. We assume that for each action name A we will have a precondition axiom of the form $Poss(A(\bar{x}), s) \leftrightarrow \pi_A(\bar{x}, s)$.

The next step towards the final axiomatization is to construct a general single positive effect axiom for each fluent, by combining all the initial positive effects axioms for fluent R (and any action).

To see how, assume that we have only two positive effects axioms for fluent R:

$$Poss(A_1(\bar{x}), s) \land \varepsilon_R^+(\bar{x}, \bar{y}, s) \rightarrow R(\bar{y}, do(A_1(\bar{x}), s)) \tag{3}$$

and

$$Poss(A_2(\bar{z}), s) \land \delta_R^+(\bar{z}, \bar{y}, s) \rightarrow R(\bar{y}, do(A_2(\bar{x}), s)). \tag{4}$$

We may combine them into one general positive effect axiom for fluent R:

$$Poss(a, s) \land [(a = A_1(\bar{x}) \land \varepsilon_R^+(\bar{x}, \bar{y}, s)) \lor (a = A_2(\bar{z}) \land \delta_R^+(\bar{z}, \bar{y}, s))] \rightarrow R(\bar{y}, do(a, s)). \tag{5}$$

In this axiom, a is a variable for actions that is considered (tacitly) universally quantified. In this form for each fluent, R, we obtain a general positive effect axiom of the form:

$$Poss(a, s) \land \gamma_R^+(a, s) \rightarrow R(do(a, s)). \tag{6}$$

Similarly, for each fluent, R, we obtain a general negative effect axiom of the form:

$$Poss(a, s) \land \gamma_R^-(a, s) \rightarrow \neg R(do(a, s)). \tag{7}$$

The general effect axioms do not solve the FP. In order to do so, Reiter makes the commonsense assumption that relatively few actions change the value of any particular fluent and those actions are completely known. In other words, the assumption is that the general effect axioms, both positive and negative, for a given fluent R describe all the possible ways for that fluent to change its truth value between one state and its successor state. Nonetheless, this assumption is not represented at the object level, so the final specification remains first-order (and monotonic). After this assumption is made, we can construct the successor state axiom.

Successor State Axiom:

$$Poss(a, s) \rightarrow [R(do(a, s)) \leftrightarrow [\gamma_R^+(a, s) \lor (R(s) \land \neg\gamma_R^-(a, s))]]. \tag{8}$$

This axiom is tacitly universally quantified over actions. Intuitively, if action a is possible, then fluent R is true at the successor state that results from the execution of action a if and only if a is one of the actions causing R to be true (for which the fluent preconditions are true), or R was already true when a was executed and a is not one of the actions causing R to be false. Actions whose preconditions are not valid at the current state are left out of the specification. There is one successor state axiom for each fluent. In order for things to work properly, unique names axioms [11] are required for actions and states.

Summary: The final specification contains a. and b. at the beginning of Section 2 and the following axioms:

1. For each fluent, R, one successor state axiom of the form (8)

2. For each action name, A, a precondition axiom of the form:

$$Poss(A(\bar{x}), s) \leftrightarrow \pi_A(\bar{x}, s)$$

3. Unique Names Axioms for Actions:
   - For distinct action names, A and A': $A(\bar{x}) \neq A'(\bar{y})$
   - For each action name, A, the axiom:

$$A(x_1, \ldots, x_n) = A(y_1, \ldots, y_n) \rightarrow x_1 = y_1 \wedge \cdots \wedge x_n = y_n$$

4. Unique Names Axioms for States:
   - $S_0 \neq do(a, s)$
   - $do(a, s) = do(a', s') \rightarrow a = a' \wedge s = s'$

■

It is important to point out that Reiter's solution to the FP, as presented in [14], works only for deterministic actions, that is, actions whose effects are completely known and where the preconditions of the actions are valid in the state they are executed in. Nonetheless, if the action preconditions are not satisfied, the effects are undetermined.

2.2 Reiter's Nonmonotonic Entailment Criterion

Assume that we have a set of formulas Σ corresponding to a.–d. at the beginning of Section 2. Let $\mathcal{R}(\Sigma)$ denote Reiter's syntactic completion of Σ, as described above. A particular entailment criterion, $\models_R$, is associated with this completion process with a natural definition: for a formula φ in the language of Σ,

$$\Sigma \models_R \varphi \quad \Leftrightarrow \quad \mathcal{R}(\Sigma) \models \varphi \tag{9}$$

On the right-hand side of the equivalence we have the classical entailment criterion, i.e. defined in terms of a "surface semantics" for a first-order predicate calculus. This surface semantics is essentially given in terms of models for first-order languages that represent worlds that evolve in time (see Section 4.1). The criterion $\models_R$ on the left-hand side is nonmonotonic and can be interpreted as a model preference criterion that selects some of the models of Σ in the class determined by the surface semantics.

Reiter's approach is monotonic only after the described syntactic transformation has been performed. This method is reminiscent of Clark's completion of logic programs [2]. From this point of view, Reiter's solution represents a syntactic method of selection of the intended models.

2.3 Example

Let us consider a library database. For simplicity, we will only consider the actions of classifying and deleting a book. Assume that there is a table with the books that have not yet been classified, *Unclassified*, and a table with the classified books, *Classified*. When books are classified they are assigned a unique ID number which encodes the ISBN number of the book and the copy number (so that different copies of the same book have different ID numbers). Books to be classified have to be chosen out of the books that are unclassified. The only books that can be deleted are those in the which have been classified.

Preliminary Axiomatization:

- Knowledge about the initial situation and the objects in the domain,

- Precondition Axioms for actions,

$$Poss(classifyBook(isbn, id), s) \equiv Unclassified(isbn, s)$$

$$Poss(deleteBook(id), s) \equiv Classified(isbn, id, s)$$

- Effect Axioms for actions,

$$Poss(classifyBook(isbn, id), s) \rightarrow Classified(isbn, id, do(classifyBook(isbn, id), s))$$

$$Poss(deleteBook(id), s) \rightarrow \neg Classified(isbn, id, do(deleteBook(id), s))$$

Integrity Constraints:

$$Classified(isbn, id, S_0) \wedge Classified(isbn', id, S_0) \rightarrow isbn = isbn'$$

By syntactic transformation of the above preliminary specification of effects of actions, we obtain the final specification. The following successor state axioms were obtained automatically by running the database specification reasoner $\mathcal{SCDBR}$ [1]: **Syntactic Transformation:** Combining the axioms for each fluent to characterize exactly under which conditions the fluent is true in a given situation,

$$\begin{gathered}(\forall a, s).Poss(a, s) \supset Classified(isbn, id, do(a, s)) \equiv \\ a = classifyBook(isbn, id) \vee (Classified(isbn, id, s) \wedge a \neq deleteBook(id))\end{gathered}$$

■

It is interesting to notice that (state) integrity constraints (both static and dynamic) are expected to be derivable from the constraints on the initial state and the successor state axioms.[3] This is done by means of proofs by induction based on the definition of the accessibility relation between states. Intuitively, one state is accessible from another state if it can be obtained through the execution of a finite (legal) sequence of actions that are possible to execute in the current state.

Reiter's approach to database update specifications [15] considers only the successor state axioms, action preconditions, the initial database $\mathcal{D}_0$, and the equality axioms. That is, there are no explicit effect axioms as we described initially. If we consider Reiter's entailment criterion as monotonic, our initial axiomatization will contain the successor state axioms, but if we view his entailment as a nonmonotonic one, the starting point will be the effect axioms. If we want to impose additional integrity constraints (such as those which capture ramifications), it is necessary to modify the successor state axioms and/or the action precondition axioms (this process is explored in [5]).

The initial database contains persistent information about the domain, and the tables at the initial state. This means that initially there are no observations of database states later than the initial one.

3 Ontological and Epistemological Assumptions

In order to determine the intended models of an axiomatization such as the above, we must first identify the underlying assumptions and define an underlying semantics according to these assumptions. The underlying semantics is defined in Section 4.2. We identify the ontological and epistemological assumptions in this section. By epistemological assumptions we mean assumptions about the available knowledge (all effects are known or only some

[3] But see [5] for a different approach, according to which IC's can be imposed from outside and are used in the generation of the successor state axioms.

effects are known, only describe the initial state or the final state, or all states, etc). By ontological assumptions we mean assumptions that consider what is allowed to exist in the world (sequential or concurrent actions, delayed and/or direct effects, ramifications, fixed or "time-bound" dynamic object domains, discrete or continuous time, etc.).

Before listing the assumptions we have identified in databases and Reiter's solution for the specific domain of database update specification, we need to state a couple of points. First of all, we note that we will use a temporal calculus and not the situation calculus for the definition of the semantics. This is more general; the situation calculus can be viewed as a special case of temporal calculus in the following sense: since the situation calculus has no explicit notion of time, situations arise from previous situations as results of the execution of actions. Assume situation s_i is the result of executing a fixed sequence of legal transactions T in s_0. We define a mapping from situations to timepoints such that s_i is mapped to time i, that is, transactions are executed from timepoint t_i to timepoint t_{i+1}. This mapping does not restrict the expressive capabilities of the situation calculus in a way that will affect the assessment since the assumptions that are made are not explicitly specified by the situation calculus. The assumptions that are made as a result of considering single-step actions are that all actions are equidurational (execute from time t to time $t+1$) and that the fluents that change value do so only once within the duration of the transaction. Since we are considering the simplest form of the situation calculus, that is, with no concurrency, this translation is possible.[4] This discussion leads to the first two ontological assumptions underlying Reiter's solution to database update specification: *sequential transactions* and *single step transactions*.

We now proceed to identify the remaining ontological assumptions. From now on, we will refer to transactions (atomic transactions), instead of actions. Reiter's solution considers *deterministic transactions* only, that is, transactions whose effects are completely determined by the state in which the transaction is performed. This means that a transaction can have *alternative results* depending on the state in which the transaction is executed, although given a state, there is only one possible outcome. Recall that we are considering a STRIPS-like [3] semantics in which, if the preconditions of a transaction are satisfied, the effects are completely known and the only changes that occur are those specified by the effects of the action. If the preconditions are not satisfied, the effects are undetermined. This may be identified with another ontological assumption, a restricted case of *inertia*: nothing changes unless there is an explicit reason for things to change, where inertia is preserved if "legal" transactions are executed.

With respect to what kinds of changes Reiter allows, we notice that fluents that do not appear as arguments of the action can also be changed as an effect of the transaction being executed. For instance, effect axioms such as the following are allowed:

$$Poss(change\_colour(x,y),s) \supset colour(x,y,do(change\_colour(x,y),s) \wedge$$

$$(colour(x',y,s) \wedge x \neq x' \supset \neg colour(x',y,do(change\_colour(x,y),s)),$$

where the constraint that two objects are not allowed to have color y, entails that painting one object color y requires that if another object had color y, this other object has to be painted, so that no two objects have color y at the end of the execution of *change_color*. This is an integrity constraint on the update transaction *change_color*, and is identified with a kind of dependency that we will call (following Sandewall's classification in [17, 16]) a *local dependency*. Other kinds of dependencies that are allowed are definitions of fluents in terms of other fluents, for instance, definition of *unclassified* in terms of *classified*, (a

[4] There are alternatives to this approach. Extensions to the situation calculus that allow for more precise representations of time have been proposed in the literature: 1. a translation of situation calculus into temporal logic [8], 2. the coexistence of situations and time [10], 3. the treatment of actions with durations [7], 4. Herbrand time representation [17].

book is unclassified if and only if it is not classified).[5] This kind of dependency is called a *structural dependency*. Thus, transactions are allowed to trigger changes in other tables.

The object domain is a *dynamic object domain*.[6] Objects can be created and destroyed between one state and another through the execution of specific transactions. We consider a specific database to be associated with two domains: a universal domain that contains all the objects that can potentially exist in the database and an object domain that is particular to the state of the database. Thus, when objects are created, they are added to the state object domain and objects are deleted when they are destroyed. This is consistent with database semantics since, in databases the universal domain may be defined by numerical domains such as the integers, reals, etc. or by strings of characters. In this paper, we will only consider ground database systems. Null values will be considered in an expanded version of this paper.

Summarizing, we have identified the following ontological assumptions:

- *inertia* (restricted to legal executions of transactions)
- *single-step transactions* (integer time)
- *alternative results of transactions*
- *deterministic transactions*
- *no concurrent transactions*
- *local and structural dependencies between fluents*
- *dynamic object domains*

Note that there are no surprises (i.e., unspecified fluent changes other than ramification-based) or miracles (the only transactions that are allowed to occur are those specified in some sequence of legal transactions), and actions have no delayed effects.

With respect to the epistemological assumptions, (i.e., assumptions that refer to the kind of knowledge that is available and how it is represented), we point out that there is complete knowledge about the initial state.[7] Note also that actions are defined in terms of preconditions, such that the effects are completely known for a given state only if the preconditions are satisfied in that state. In the case that the preconditions are not satisfied we will assume the action's effects are not determined. We assume a ground database.

4 Assessment

Having identified the ontological and epistemological assumptions, we define the underlying semantics. This underlying semantics determines, given a preliminary axiomatization, the sets of models that should be obtained by means of Reiter's entailment criterion, $\models_R$ (see Section 2.2). It is important to stress that the underlying semantics depends on the ontological and epistemological assumptions but it is independent of the entailment criterion to be assessed. Thus, in order to show that Reiter's formalism is sound and complete, we must compare the interpretations determined by the underlying semantics with some interpretations according to Reiter's entailment criterion (viewed as a nonmonotonic completion process). If the interpretations correspond, we will say that Reiter's solution is sound and

[5] Lin and Reiter [5] address the problem of integrating ramifications that appear as integrity constraints in databases with the successor state axioms.

[6] This corresponds to Sandewall's time bound objects ontological specialty.

[7] This can be achieved by a syntactical closure mechanism as proposed in [12].

complete for all problems that satisfy the ontological and epistemological assumptions underlying the restricted database domain (where there are no null values and no concurrent actions).

4.1 Surface Semantics

Reiter's entailment criterion selects certain models from the wide class of first order models whose interpretation is given by the "surface semantics". The surface semantics is basically a first order logic semantics with many sorts (actions and timepoints), and with two object domains, a universal object domain which contains all objects that may exist in the database, and a state object domain (subset of the universal object domain) which contains all objects that exist in a given state of the database. (Actually, this is not just one domain, but a function from timepoints to subsets of the universal domain.) This distinction of the object domains is necessary for the creation and elimination of objects from one state to another.

This semantics determines all possible interpretations of the preliminary axiomatization. Reiter's entailment criterion selects, among the models that correspond to these interpretations, those that correspond to the syntactic transformation of the preliminary axiomatization with the completion assumption.

4.2 Underlying Semantics

In order to use the underlying semantics as a framework to evaluate the models obtained by Reiter's entailment criterion, it must satisfy two requirements: 1. it must be independent of the entailment criterion to be assessed, 2. it must be general enough to capture the entailment criterion, but specific enough not to allow any other assumptions (surprises, miracles, etc.) in the selected models. We will show the proposed semantics satisfies these two points.

Following Sandewall [17], the underlying semantics will be defined in terms of an agent and a world. The agent will execute transactions and the world will react to this execution according to the preliminary axiomatization of each transaction (and the ontological assumptions). This interaction between the agent and the world will determine all the possible histories of the particular instance of the domain. More precisely, given (a) an arbitrary axiomatization (which satisfies the ontological and epistemological assumptions described above), (b) a sequence of transactions, (c) an agent, and (d) a world (with (b), (c), and (d) satisfying the ontological assumptions), the interaction between the agent and the world will determine the possible histories of that particular instance of the domain that satisfy the given sequence of transactions. The intended models to be compared to those selected by Reiter's entailment criterion will be obtained from these histories, which we will call *developments*.

Before we describe how the developments are generated, let us say a few words about the semantics of Reiter's solution and databases. We will consider only situations that arise from legal executions of transactions. The situation calculus semantics, then, defines a tree of situations branching out from the initial situation, where each node of the tree is identified with a term of the form $do(a, s)$ that corresponds to the situation that results from executing action a in situation s. If the preconditions of a are satisfied in s (i.e., a is a legal transaction), Reiter's solution completely specifies the successor state. The branching factor is the number of different legal actions, which varies from situation to situation. We consider a state corresponding to a situation to be a function from fluents to true or false.

Definition 1 *A* situation *will be a pair of the form $\langle do(\tau, do(\ldots, s_0)), state \rangle$, where the* state *is a function from fluents to boolean values.*

After this brief discussion of Reiter's solution and database semantics, we return to the definition of the underlying semantics which will be given in terms of the developments.

The starting point for the generation of the developments is the description of the database. This description is formulated in Reiter's extension to the situation calculus [14, 15].

Definition 2 *The* initial description of the database *is a pair* $\langle \Sigma, T \rangle$*, where* Σ *is a set of formulas corresponding to a.–d. at the beginning of Section 2 and* T *is a sequence of transactions of the form* $T_0, T_1, \ldots, T_{n-1}$*, where each* T_i *is a transaction to be executed in situation* s_i*. In other words,* T *corresponds to the expression* $do(T_{n-1}, \ldots, do(T_1, do(T_0, s_0)) \ldots)$.

The interaction between the agent and the world will take place within an instance of the domain in consideration, which we will denote by $\mathcal{A}$, and which will be consistent with the given initial description of the database.[8] This structure will capture the axiomatization and the specific sequence of transactions that will be performed in the database. All axioms from the initial description of the database, $\langle \Sigma, T \rangle$, will be translated from the situation calculus into Sandewall's temporal logic [17, 16]. In the following we make reference to the axioms of Σ described at the beginning of Section 2.

Let $\mathcal{A}$ be an instance of the restricted world of databases that we are considering, such that $\mathcal{A} = \langle EA, O, P, W, T, OBS \rangle$, where:

EA: is a list of epistemological assumptions,

O: is the pair $\langle O_u, O_s \rangle$, where O_u is the universal object domain and O_s is the state object domain, i.e., the indexed set of objects that appear in a database at particular state, and O_u corresponds to the objects that are described by the axioms of Σ in point a.,

P: the action precondition axioms, that is, the axioms of Σ described by point c.,

W: is a pair containing a list of the corresponding ontological assumptions and the description of the world given by the effect axioms, that is, the laws corresponding to the axioms of Σ described by point d.,[9]

T: is the sequence of transactions, where T_i is interpreted as a transaction to be executed at timepoint t_i.

OBS: is the observations, axioms that describe the initial state of the database, that is, axioms of Σ described by point b..[10]

The idea is that given an initial description of the database $\langle \Sigma, T \rangle$, we define all possible models of the preliminary axiomatization, Σ, according to an agent/world interaction. We hope to find some correspondence between these models and the interpretations according to Reiter's solution. Then, we hope to find a one-to-one correspondence between the model according to this semantics provided that the sequence, T, of transactions has been executed and is legal, and some interpretation (according to Reiter) of the database updated by T.

Given an instance of the restricted domain of databases, $\mathcal{A}$, we generate the corresponding developments (from which the models will be obtained) as follows. Choose an arbitrary agent (acting element) and an arbitrary world (reacting element), which will react according to the axioms in P and W, and generate an interaction between the agent and the world. The idea is to generate all the possible histories of the world, given all the possible executions of transactions in all possible initial states. This interaction is recorded in the developments.

Formally, a *development* is a tuple $\langle B, M, O_s, R, \mathcal{F}, \mathcal{C} \rangle$, where:

[8] This corresponds to Sandewall's scenario descriptions.

[9] This corresponds to component Diw of Sandewall's scenario descriptions.

[10] Note that integrity constraints do not appear as observations, they will be assumptions about the states.

B: is the set of interpretations of the form $\{0, \ldots, k\}$ for timepoints, where k corresponds to the current state,

M: is the valuation for timepoints (assigns members of B to timepoints) and objects in the universal object domain,

O_s: is the state object domain (indexed subset of the universal object domain),

R: is the history of the fluent changes from timepoint 0 to timepoint k, actually, we consider this to be an indexed set of assignments of values to fluents,

$\mathcal{F}$: is the set of transactions that have been executed,

$\mathcal{C}$: is the set of transactions that currently are being executed; since only one transaction can be executed at a time, it will always contain a single action.

In $\mathcal{F}$ transactions are time stamped indicating the starting and termination times, while in $\mathcal{C}$ transactions are only time stamped with the starting times.

The moves of the agent and the world are recorded in the developments as follows:

- if the agent chooses a transaction, τ specified in W, the world updates $\mathcal{C}$ with the tuple $\langle t, \tau\rangle$ and reacts to the transaction as described in the following,
- if τ is such that it does not modify the state object domain, it removes $\langle t, \tau\rangle$ from $\mathcal{C}$ and adds $\langle t, \tau, t+1\rangle$ to $\mathcal{F}$. If the preconditions of τ are satisfied in R, it updates that history by overriding any previous fluent values with new ones as determined by the action laws in W, and it updates B as before. If the preconditions are not satisfied in R, it branches out generating all possible updates for R (one change each time it branches), and it updates B by adding the timepoint that corresponds to $t+1$.
- if τ is such that it modifies the state object domain, the world removes the tuple $\langle t, \tau\rangle$ from $\mathcal{C}$ and adds the tuple $\langle t, \tau, t+1\rangle$ to $\mathcal{F}$, updates O_s and R accordingly, and updates B by adding the timepoint corresponding to $t+1$;

Proposition 3 *The agent/world interaction generates a collection of infinite trees. Moreover, if the number of fluents in the initial state is infinite, the collection is infinite, if the number of fluents is finite, the collection is finite and of cardinality* $2^{|\mathcal{F}|}$.

Lemma 1 *If there are two arcs with different labels departing from the same node, the nodes which they arrive at are different.*

In order to assess Reiter's entailment criterion, we must find a correspondence between the models it defines and the trees defined by the underlying semantics. Reiter's models are obtained from a preliminary axiomatization which includes a complete description of the initial database. We need to identify a tree of developments that should correspond to the tree of situations. The candidate is the one whose root corresponds to the observations of the initial database.

Proposition 4 *Only one tree in the collection of trees has a root that corresponds to the initial database.*

Having selected the tree that corresponds to the preliminary axiomatization, the set of models will be constructed from the M and R components of each development that hangs from the root.

We proceed to show that these models are indeed acceptable for the assessment, that is, that they are independent of Reiter's entailment criterion and that they capture only the assumptions we have listed in Section 3.

Proposition 5 *The underlying semantics defined in this way is independent of Reiter's entailment criterion.*

Proposition 6 *The underlying semantics captures all assumptions listed in Section 3, and it rules out the rest of those listed in Sandewall's book [17].*

We now proceed to state the semantic correspondence theorem.

4.3 Semantic Correspondence

Given a preliminary axiomatization of a database we obtain the tree of situations according to Reiter's solution, on the one hand; on the other hand, we obtain the tree of developments according to the agent/world interaction. If there is a correspondence between the two, provided that only legal sequences of transactions are executed, Reiter's solution will be proven sound and complete in database domains where only legal transactions are allowed to be executed and with the ontological and epistemological restrictions we previously identified.

Theorem 1 *Given Σ, a preliminary axiomatization of a database consistent with the assumptions of Section 3, there is a one-to-one correspondence between the branches of the tree of situations and the branches of the tree of developments that arise from executions of legal sequences of transactions in the database.*

Note that we are considering all legal sequences of transactions defined in the preliminary axiomatization and not only the particular T in $\mathcal{A}$. Later, we show there is a correspondence between the branches of the trees generated by this particular sequence. To be more precise, we could define a structure $\mathcal{A}'$ which does not contain the component T and later on, define the structure $\mathcal{A}$ which determines the development that corresponds to that particular sequence of transactions.

Now that we have proven Reiter's solution is sound and complete for database problems satisfying only the restrictions listed in Section 3, it is a consequence of the theorem that there is a one-to-one correspondence between the two interpretations of the database given that the particular sequence of transactions, T of the given $\langle\Sigma, T\rangle$, is performed.

Corollary 1 *Given an initial description of a database, $\langle\Sigma, T\rangle$, where T is a legal sequence of transactions, there will always be a development according to Sandewall's underlying semantics that corresponds to branch in the tree of situations that arises as the result of T's execution according to Reiter's solution, and vice-versa.*

5 Conclusions and Further Work

We have extended Sandewall's semantics [17, 16] to the case where the effects of the transactions are only known once the preconditions are satisfied. This extension served as basis for the assessment of Reiter's general monotonic solution to the frame problem and its applicability to database updates. We conclude that it is correct (and complete) for ground databases where legal sequential update transactions are performed.

There are still a couple of issues that need to be addressed, namely, the consideration of illegal transactions and the treatment of null values in relational databases. We will develop these further in an extended version of this work based on [13, 15].

6 Acknowledgements

We are grateful to Ray Reiter and Erik Sandewall and Javier Pinto for stimulating conversations and advice around this research. This research has been supported partly by DIUC, SAREC/CONICYT, Fundacion DICTUC, and FONDECYT (#1930554) grants, partly by the Air Force Office of Scientific Research under contract 92-0693 and by ARPA/Rome Labs under contracts F30602-91-C-0036 and F30602-93-C-00031. Tania Bedrax-Weiss is specially thankful to David Etherington for patient reviews of this work, to Matthew Ginsberg for interesting comments on previous versions of this work, and to CIRL for productive meetings. Leopoldo Bertossi visited Linköping University in Sweden for a couple of weeks and is extremely grateful to Erik Sandewall for his hospitality.

References

[1] L. Bertossi and J. Ferretti. SCDBR: A reasoner for specifications in the situation calculus of database updates. In *Proceedings of the First International Conference on Temporal Reasoning*, Bonn, Germany, 1994. To appear in LNAI, Springer.

[2] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 292–322, New York, 1978. Plenum Press.

[3] R. Fikes and N. Nilsson. STRIPS: A new approach to application of theorem proving procedures to problem solving. In *Readings in Planning*, 1971.

[4] A. Haas. The case for domain-specific frame axioms. In F. Brown, editor, *The frame problem in artificial intelligence. Proceedings of the 1987 Workshop*, pages 343–348, Los Altos, CA, 1987. Morgan Kaufmann Publishers, Inc.

[5] F. Lin and R. Reiter. State constraints revisited. *To appear in Journal of Logic and Computation*, 1994.

[6] F. Lin and Y. Shoham. Provably correct theories of action (preliminary report). In *Proceedings of the National Conference of AI*, 1991.

[7] V. Lifschitz M. Gelfond and A. Rabinov. What are the limitations of the situation calculus? In *AAAI Symposium of Logical Formalization of Commonsense Reasoning*, pages 59–69, Stanford, CA, 1991.

[8] R. Miller and M. Shanahan. Narratives in the situation calculus. *Ongoing work, second revision*, April 1993.

[9] E. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In H. Levesque R. Brachman and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 324–332, San Mateo, CA, 1989. Morgan Kaufmann Publishers, Inc.

[10] J. Pinto and R. Reiter. Adding a time line to the situation calculus. In *Working Notes of Common Sense'93, The Second Symposium on Logical Formalizations of Commonsense Reasoning*, pages 172–177, Austin,TX, January 1993.

[11] R. Reiter. Equality and domain closure in first-order databases. *Journal of the ACM*, 27(2):235–249, April 1980.

[12] R. Reiter. Towards a logical reconstruction of relational databases theory. In J. Mylopoulos and J. Schmidt, editors, *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*, pages 191–233. Springer-Verlag, 1984.

[13] R. Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *Journal of the ACM*, 2(33):349–370, 1986.

[14] 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.

[15] R. Reiter. On specifying database updates. Technical Report KRR-TR-92-3, University of Toronto, Department of Computer Science, Toronto, Canada, 1992.

[16] E. Sandewall. The range of applicability of nonmonotonic logics for the inertia problem. In *Thirteenth*, 1993.

[17] E. Sandewall. *Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems.* To appear in Oxford University Press, 1994.

[18] L. Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In R. Loui H. Kyburg and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67, Boston, MA, 1990. Kluwer Academic Press.