$\mathcal{SCDBR}$: A Reasoner for Specifications in the Situation Calculus of Database Updates.*

Leopoldo E. Bertossi** and **J. Cristian Ferretti**

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

In [2], Reiter presents a formalism for database updates specification. He uses the "situation calculus", that is, many-sorted first-order languages for representing knowledge about dynamically changing worlds that evolve through different states when actions are executed. In these worlds, properties of their objects depend on the current state of the world. These properties are called fluents and correspond in our context to database tables.

The specification of a dinamically changing world, like a database, typically contains the following information: (1) state independent knowledge about the objects of the world, (2) knowledge about the state of the world at the initial situation S_0, (3) preconditions for performing the different actions; a binary predicate *Poss* is introduced to say that the execution of an action is possible at a state, (4) the immediate effects of actions, when they are possible, in terms of the fluents whose truth values are known to be changed by the execution of the action. The knowledge contained in items (1) and (2) is called the inital database $\mathcal{D}_0$.

EXAMPLE: Let us consider a library database. For reasons of space we outline the preliminary specification. In this case, language has the following ingredients: Fluents (Tables): $\mathit{Classified}(isbn, id, s)$, $\mathit{Stock}(isbn, quantity, s)$, $\mathit{Unclassified}(isbn, copies, s)$, $\mathit{LostBook}(id, s)$, $\mathit{SoldOut}(isbn, s)$. Update Actions (Atomic Transactions): $\mathit{deleteBook}(id)$, $\mathit{classifyBook}(isbn, id)$, $\mathit{order}(isbn, copies)$.
Preconditions for action *classifyBook*: $\mathit{Poss}(\mathit{classifyBook}(isbn, id), s) \equiv$

$$(\exists\, copies)\, \mathit{Unclassified}(isbn, copies, s) \wedge \neg(\exists\, isbn')\, \mathit{Classified}(isbn', id, s)$$

Two effect laws:

$$\mathit{Poss}(\mathit{classifyBook}(isbn, id), s) \wedge \mathit{Unclassified}(isbn, copies, s) \rightarrow \neg\, \mathit{Unclassified}(isbn, copies, do(\mathit{classifyBook}(isbn, id), s))$$

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

One (initial) integrity constraint:

$$\mathit{Classified}(isbn, id, S_0) \wedge \mathit{Classified}(isbn', id, S_0) \rightarrow isbn = isbn'. \blacksquare$$

An specification that contains only this kind of information does not solve the frame problem by itself. In [1], Reiter presents a simple first-order solution to the frame problem for specifications given in the situation calculus. The

* Submitted to ICTL'94. Topic Areas: Specification and Verification, Temporal Databases, Planning and Change. Category: System Description.

** For many and different reasons we are grateful to Ray Reiter,Tania Bedrax, Pablo Saez, Rich Scherl, Aris Zachintinos and Javier Pinto. This project has been partially financed by DIUC, Fundacion DICTUC and FONDECYT (#1930554) Grants.

restriction of Reiter's general solution to actions with deterministic effects is usually satisfied by database transactions. Reiter solves the frame problem, starting from a preliminary specification in the lines of the previous example, by a syntactic transformation of the given axioms, and the final specification turns out to be entirely first-order. In [2], Reiter applies his solution to the specification of database updates. The final specification contains, maybe properly, the following axioms: (1) for each fluent R, one "successor state axiom" of the form $(\forall\, a\, \forall\, s).Poss(a,s) \supset (R(do(a,s)) \equiv \Phi_R(a,s))$, (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 and states, (4) the initial database $\mathcal{D}_0$. The successor state axiom for a fluent R gives all the conditions under which the fluent becomes (or remains) true in a successor state resulting from executing an action at a predecessor state.

Our system $\mathcal{SCDBR}$ automatically derives a Reiter specification starting from a preliminary specification as described above. The system is written in PROLOG. This language is used as a programming language. The specifications handled by the system are fully first-order.

The user must especify the language first. In particular, types of objects must be declared. Formulas are internally represented as prolog ground terms. For example, the formula $\forall\, a\, \exists\, s(a = order(isbn, 5) \wedge Stock(isbn, 5, do(a,s)))$ is represented by the PROLOG ground term `forall(a): exists(s):(a eq order( isbn,5) & stock(isbn,5,do(a,s)))`. Here, the basic terms `stock(isbn,5, do(a,s))`, `forall(a)`, etc., are composed by application of the PROLOG operators `<=>,=>,&,v,:,neg,eq`, that are defined in the system. The direct effect laws are given by means of an incidence table that stores the fluents that are affected by each action. With this information at hand, the system is able to build the successor state axioms for each fluent. The PROLOG call : `ssa(fluent,X).`, instantiates X to the succesor state axiom for the fluent. For example, the result from calling `ssa` in $\mathcal{SCDBR}$ for fluent *Classified* is:

```
| ?- ssa(classified, SSAF).
SSAF = forall(a): precond(a,s) => (classified(isbn,id,do(a,s) <=>
       a eq classify_book(isbn,id)  v (classified(isbn,id,s) &
       neg (a eq delete_book(id) & classified(isbn,id,s)))).
```

This answer is equivalent to the following successor state axiom for fluent *Classified*:

$$(\forall\, a, s).(Poss(a,s) \supset (Classified(isbn, id, do(a,s)) \equiv a = classifyBook(isbn, id) \vee$$

$$Classified(isbn, id, s) \wedge \neg a = deleteBook(id)))).$$

The head of the defining clause for `ssa` looks like the following:

```
ssa( F, forall(A):precond(A,S) => ( FPD <=> PHI ) :-    ...
```

The construction of the successor state axioms using this clause relies heavily on unification. The syntactical form of the successor state axiom is known, and this is reflected in the second parameter n the definition of `ssa`. The values for `FPD` (a PROLOG variable for the fluent in the succesor state) and `PHI` are obtained executing the clauses in the body. `FPD` is easily constructed using the declarations for the fluent `F` and the specification language variables. `PHI` is built collecting pairs (*action*, *fluent precondition*) that affects `F` from the incidence table, and calling a clause that, starting from these pairs, builds `PHI`. In this process we also take advantage of its known sintax.

$\mathcal{SCDBR}$ is also able to perform other reasoning tasks like: (1) answering historical queries, (2) answering queries about the legality of sequences of transactions of the form $do(\alpha_1, do(\alpha_2, do(\alpha_3, \ldots) \ldots)))$, (3) "regressing" queries posed to a virtually updated database to a query about the initial database state, (4) doing "planning". For these last three tasks, the system processes the "list" of transactions leading to the updated database, taking into account the axioms that specify the successor states of tables in the database and preconditions on actions. This is based on a regression operator $\mathcal{R}$ proposed by Reiter in [1, 2]. For example, the regression of the formula $classified(isbn, id, do(order(isbn, id), s0))$ provided by the system is the following:

```
| ?- reg( classified(isbn,id,do(order(isbn,copies),s0)), R ).
R = order(isbn,copies) eq classify_book(isbn,id) v
    classified(isbn,id,s0) &
      neg @(order(isbn,copies) eq delete_book(id) &
      classified(isbn,id,s0)).
```

We are working on extensions to the system in order to solve the following tasks: (1) updates of/through views (defined fluents), (2) interface to an Automated Theorem Prover, which is to be used mainly for proving statements with respect to initial databse $\mathcal{D}_0$ (the regression operator reduces evaluation of queries posed to virtually updated databases to evaluation with respect to the initial database), (3) management of static and dynamic integrity constraints. In particular, mechanical modification of the initial specification so that the updated databases satisfy the given constraint, and mechanical proofs of integrity constraints from the specification. This last task requires inductive proofs, (4) Physical update of the database from a virtual update (i.e. from the initial database, the successor state axioms and a given sequence of transactions).

References

1. R. Reiter. *The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression.* In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, V. Lifschitz (ed.). Academic Press, 1991, pp. 359-380.
2. R. Reiter. *On Specifying Database Updates.* Technical Report KRR-TR-92-3, Department of Computer Science, University of Toronto, 1992

This article was processed using the LaTeX macro package with LLNCS style