



icate  $badSituation(s)$  but this is limited to properties of the current situation  $s$ . The nonMarkovian situation calculus allows the definition of this predicate to refer to any situation that precedes  $s$ , i.e. to the full past of the current situation. As we mentioned above, past temporal logic expressions can be encoded in the nonMarkovian situation calculus and be used in the definition of  $badSituation(s)$ . This means that we can use search control knowledge of a similar form and expressivity as the one used in TLPlan.

4. The Markovian property may also be absent if the system specification includes stochastic actions and reward functions. The need to accommodate nonMarkovian dynamics and reward functions has been recognized in the work (Bacchus, Boutilier, & Grove 1996; 1997) who have developed techniques for solving nonMarkovian Decision Processes for decision-theoretic planning.
5. Finally, some time ago John McCarthy (1992) described a programming language called “Elephant 2000” which, among other features, does not forget. In other words, it is a language that allows the programmer to explicitly refer to past states of the programming environment in the program itself. The nonMarkovian situation calculus and the regression operator we present here can form the foundation for a non-forgetting version of Golog. Such a dialect of Golog would allow test conditions in terms of Past temporal logic and so one could write, for instance, a statement **if** ( $P$  since  $Q$ ) **then**  $exec(\delta)$  **endif** in programs.

In this paper, we generalize the situation calculus based formalization of actions of Reiter (1991) to the nonMarkovian case. We modify the regression operator to work with nonMarkovian basic action theories and formulas that quantify over past situations and prove the new regression operator to be correct.

### The language of the Situation Calculus

In this section we briefly review the language of the situation calculus. For a complete description see (Pirri & Reiter 1999).

The language  $\mathcal{L}_{sitcalc}$  is a second order language with equality and with three disjoint sorts: *action*, *situation* and *object*. In addition to  $\wedge, \neg, \exists$  and definitions in terms of these for the other standard logical symbols, the alphabet of  $\mathcal{L}_{sitcalc}$  includes a countably infinite number of variable symbols of each sort and predicate variables of all arities. A constant symbol  $S_0$  and a function  $do$  of sort  $action \times situation \rightarrow situation$ , a binary predicate symbol  $\sqsubset$  used to define an ordering relation on situations, a binary predicate symbol  $Poss : action \times situation$ , and for each  $n \geq 0$  a countably infinite number of function symbols of sort<sup>1</sup>  $(action \cup object)^n \rightarrow action$  called *actions*, a countably infinite number of predicate symbols of sort  $(action \cup object)^n \times situation$  called *relational fluents*, and a countably infinite number of function symbols of sort  $(action \cup object)^n \times situation \rightarrow action \cup object$  called *functional fluents*. The language includes also a countably

<sup>1</sup>We use  $(s_1 \cup s_2)^n$  as a shorthand for  $s_{v_1} \times \dots \times s_{v_n}, v_i \in \{1, 2\}$ .

infinite number of predicates and functions without a situation argument. We will refer to these as *situation independent* predicates and functions.

Intuitively, situations are finite sequences of actions (sometimes referred to as *histories*) and this intuition is captured by a set of four *Foundational Axioms* (Pirri & Reiter 1999) (denoted by  $\Sigma$ )<sup>2</sup>:

$$\begin{aligned} do(a_1, s_1) &= do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2, \\ (\forall P). P(S_0) \wedge (\forall a, s)[P(s) \supset P(do(a, s))] &\supset (\forall s)P(s), \\ \neg s \sqsubset S_0, \\ s \sqsubset do(a, s') &\equiv s \sqsubset s' \vee s = s'. \end{aligned}$$

The initial situation or empty history is denoted by constant  $S_0$ . Non-empty histories are built by means of the function  $do$ .

### Basic nonMarkovian Theories of Action

In this section we introduce the basic nonMarkovian theories of action. In the Markovian action theories, the Markov assumption is realized by requiring that the formulas in the Action Precondition Axioms and Successor State Axioms refer only to one situation, a variable  $s$ , which is prenex universally quantified in the axioms. In nonMarkovian action theories, situation terms other than  $s$  will be allowed under the restriction that they refer to the past or to an explicitly bounded future relative to  $s$ . To make this formal, we need to introduce the notion of situation-bounded formulas. Intuitively, an  $\mathcal{L}_{sitcalc}$  formula is bounded by situation term  $\sigma$  if all the situation variables it mentions are restricted, through equality or the  $\sqsubset$  predicate, to range over subsequences of  $\sigma$ . This notion is useful because in order to apply regression on a formula, one needs to know how many actions there are in each situation, i.e. how many regression steps to apply. A formula that mentions a situation variable can be regressed provided that the variable is restricted to be a subsequence of some situation term with a known number of actions in it.

The following notation is used through out: for  $n \geq 0$ , we write  $do([\alpha_1, \dots, \alpha_n], \lambda)$  to denote the term of sort *situation*  $do(\alpha_n, do(\alpha_{n-1}, \dots, do(\alpha_1, \lambda) \dots))$  where  $\alpha_1, \dots, \alpha_n$  are terms of sort *action* and  $\lambda$  stands for a variable  $s$  of sort *situation* or the constant  $S_0$ .

**Definition 1** For  $n \geq 0$ , define the *length* of the situation term  $do([\alpha_1, \dots, \alpha_n], \lambda)$  to be  $n$ .

**Definition 2 (Rooted Terms)** For  $n \geq 0$ , let  $\alpha_1, \dots, \alpha_n$  be terms of sort *action*. A term  $do([\alpha_1, \dots, \alpha_n], s)$  is *rooted at s* iff  $s$  is the only variable of sort *situation* mentioned by  $\alpha_1, \dots, \alpha_n$  or no variable of that sort is mentioned. A term  $do([\alpha_1, \dots, \alpha_n], S_0)$  is *rooted at  $S_0$*  iff  $\alpha_1, \dots, \alpha_n$  mention no variables of sort *situation*.

In writing bounded formulas, we will use the following abbreviations:

<sup>2</sup>Lower case Roman characters denote variables. Free variables are implicitly universally prenex quantified.

$$\begin{aligned}
(\exists s : \sigma' \sqsubset \sigma)W &\stackrel{\text{def}}{=} (\exists s)[\sigma' \sqsubset \sigma \wedge W] \\
(\exists s : \sigma' = \sigma)W &\stackrel{\text{def}}{=} (\exists s)[\sigma' = \sigma \wedge W] \\
(\forall s : \sigma' \sqsubset \sigma)W &\stackrel{\text{def}}{=} \neg(\exists s)[\sigma' \sqsubset \sigma \wedge \neg W] \\
(\forall s : \sigma' = \sigma)W &\stackrel{\text{def}}{=} \neg(\exists s)[\sigma' = \sigma \wedge \neg W]
\end{aligned} \tag{1}$$

**Definition 3 (Bounded Formulas)** For  $n \geq 0$ , let  $\sigma$  be a term  $do([\alpha_1, \dots, \alpha_n], \lambda)$  rooted at  $\lambda$ . The formulas of  $\mathcal{L}_{sitcalc}$  bounded by  $\sigma$  are the smallest set of formulas such that:

1. If  $t_1, t_2$  are terms of the same sort whose subterms of sort *situation* (if any) are all rooted at  $\lambda$ , then  $t_1 = t_2$  is a formula bounded by  $\sigma$ .
2. If  $\sigma'$  is a term of sort *situation* rooted at some situation variable or constant  $S_0$ , then  $\sigma' \sqsubset \sigma$  is a formula bounded by  $\sigma$ .
3. For each  $n \geq 0$ , each  $n$ -ary situation independent predicate  $P$ , each  $(n+1)$ -ary fluent  $F$  and each  $n$ -ary action  $A$ , if  $t_1, \dots, t_n$  are terms of sort *action* or *object* whose subterms of sort *situation* are all rooted at  $\lambda$ , then  $P(t_1, \dots, t_n)$ ,  $F(t_1, \dots, t_n, \sigma)$  and  $Poss(A(t_1, \dots, t_n), \sigma)$  are formulas bounded by  $\sigma$ .
4. If  $\sigma'$  is a term of sort *situation* rooted at  $s$  and  $W$  is a formula bounded by a possibly different term of sort *situation* also rooted at  $s$ , then  $(\exists s : \sigma' \sqsubset \sigma)W$ ,  $(\exists s : \sigma' = \sigma)W$ ,  $(\forall s : \sigma' \sqsubset \sigma)W$  and  $(\forall s : \sigma' = \sigma)W$  are formulas bounded by  $\sigma$ .
5. If  $W_1, W_2$  are formulas bounded by situation terms rooted at  $\lambda$ , then  $\neg W_1$ ,  $W_1 \wedge W_2$  and  $(\exists v)W_1$ , where  $v$  is of sort *action* or *object*, are formulas bounded by  $\sigma$ .

**Example 1** For the purpose of illustrating the above definitions, consider the following sentence

$$\begin{aligned}
&(\exists a).(\exists s' : do(a, s') \sqsubset \\
&do([get\_coffee, deliver\_coffee, gotoMailRm], s)) \\
&batteryCharged(do(chargeBatt, s')).
\end{aligned}$$

Intuitively it says that there is a situation in the past of  $do([get\_coffee, deliver\_coffee, gotoMailRm], s)$  when executing  $chargeBatt$  would have (successfully) resulted in charged battery. This sentence is bounded by  $do([get\_coffee, deliver\_coffee, gotoMailRm], s)$ , with subformula  $batteryCharged(do(chargeBatt, s'))$  bounded by  $do(chargeBatt, s')$ . Here, variable  $s'$  ranges over the subsequences of  $do(get\_coffee, s)$ . Note that this formula actually refers to a situation which is not in the past relative to the bounding situation  $do([get\_coffee, deliver\_coffee, gotoMailRm], s)$ .

We also need a strict version of boundedness.

**Definition 4 (Strictly Bounded Formulas)** Strictly bounded formulas are defined by replacing conditions 1,4, and 5 in the definition of bounded formulas with the following:

- 1<sup>1</sup> If  $t_1, t_2$  are terms of the same sort whose subterms of sort *situation* (if any) are all subterms of  $\sigma$ , then  $t_1 = t_2$  is a formula strictly bounded by  $\sigma$ .

- 4'. If  $\sigma'$  is a term of sort *situation* rooted at  $s$  and  $W$  is a formula strictly bounded by a subterm of  $\sigma'$ , then  $(\exists s : \sigma' \sqsubset \sigma)W$ ,  $(\exists s : \sigma' = \sigma)W$ ,  $(\forall s : \sigma' \sqsubset \sigma)W$  and  $(\forall s : \sigma' = \sigma)W$  are formulas strictly bounded by  $\sigma$ .
- 5'. If  $W_1, W_2$  are formulas strictly bounded by a subterm of  $\sigma$ , then  $\neg W_1$ ,  $W_1 \wedge W_2$  and  $(\exists v)W_1$ , where  $v$  is of sort *action* or *object*, are formulas strictly bounded by  $\sigma$ .

So we require that the situation term that binds  $W$  not only have the same root, but be one of the subterms of  $\sigma'$ . Intuitively, a formula  $W$  that is strictly bounded by  $\sigma$  has its situation terms restricted to the past relative to  $\sigma$ . Reference to hypothetical ‘‘alternative futures’’ as in Example 1, which is allowed in bounded formulas, is disallowed.

**Example 2** In the situation calculus, to refer to the past means to refer to past situations. In this sense, one can write expressions that capture the intuitive meaning of the past temporal logic connectives *previous*, *since*, *sometime*, and *always*:<sup>3</sup>

$$\begin{aligned}
prev(\varphi, s) &\stackrel{\text{def}}{=} (\exists a).(\exists s' : do(a, s') = s) \varphi[s']. \\
since(\varphi_1, \varphi_2, s) &\stackrel{\text{def}}{=} (\exists s' : s' \sqsubset s). \varphi_2[s'] \wedge \\
&\quad (\forall s'' : s'' \sqsubseteq s). s' \sqsubset s'' \supset \varphi_1[s'']. \\
sometime(\varphi, s) &\stackrel{\text{def}}{=} (\exists s' : s' \sqsubset s) \varphi[s']. \\
always(\varphi, s) &\stackrel{\text{def}}{=} (\forall s' : s' \sqsubset s) \varphi[s'].
\end{aligned}$$

It is easy to see that these formulas are strictly bounded by  $s$ .

We are now ready to define nonMarkovian Action Precondition Axioms and Successor State Axioms.

**Definition 5** An *action precondition axiom* is a sentence of the form:

$$Poss(A(x_1, \dots, x_n), s) \equiv \Pi_A(x_1, \dots, x_n, s),$$

where  $A$  is an  $n$ -ary action function symbol and  $\Pi_A(x_1, \dots, x_n, s)$  is a first order formula with free variables among  $x_1, \dots, x_n, s$  that is bounded by a situation term rooted at  $s$  and does not mention the predicate symbol  $Poss$ .

**Example 3** Suppose that there is a lab where the robot works whose door should not be opened if the temperature inside reached some dangerous level  $d$  since it was closed. The robot’s theory would include a precondition axiom:

$$\begin{aligned}
Poss(open(Lab1), s) &\equiv (\exists s' : do(close(Lab1), s') \sqsubseteq s). \\
&\quad (\forall s'' : s'' \sqsubset s) \neg (s' \sqsubset do(open(Lab1), s'')) \wedge \\
&\quad (\forall s'' : s'' \sqsubset s). s' \sqsubset s'' \supset temp(Lab1, s'') < d.
\end{aligned}$$

**Definition 6** A *successor state axiom* for an  $(n+1)$ -ary relational fluent  $F$  is a sentence of the form:

$$F(x_1, \dots, x_n, do(a, s)) \equiv \Phi_F(x_1, \dots, x_n, a, s),$$

where  $\Phi_F(x_1, \dots, x_n, a, s)$  is a first order formula with free variables among  $x_1, \dots, x_n, a, s$  that is strictly bounded by  $s$  and does not mention constant  $S_0$  nor the predicate symbol  $Poss$ .

A *successor state axiom* for an  $(n+1)$ -ary functional fluent  $f$  is a sentence of the form:

$$f(x_1, \dots, x_n, do(a, s)) = y \equiv \phi_f(x_1, \dots, x_n, y, a, s),$$

<sup>3</sup>We use  $\sigma \sqsubseteq \sigma'$  as an abbreviation for  $\sigma = \sigma' \vee \sigma \sqsubset \sigma'$ .

where  $\phi_f(x_1, \dots, x_n, y, a, s)$  is a first order formula with free variables among  $x_1, \dots, x_n, y, a, s$  that is strictly bounded by  $s$  and does not mention constant  $S_0$  nor the predicate symbol  $Poss$ .

**Example 4** Consider again the robot that works at a bio-research lab. One of the successor state axioms in its theory could be (using the past temporal logic abbreviations from Example 2):

$$\begin{aligned} & polluted(mat, do(a, s)) \equiv polluted(mat, s) \vee \\ & a = touch(mat) \wedge (\exists loc). safetyLevel(loc, Low) \wedge \\ & since(\neg atLoc(Disinfect), atLoc(loc), s). \end{aligned}$$

Relaxing the strict boundedness condition in successor state axioms to simply boundedness, complicates regression. Consider the following successor state axioms:

$$\begin{aligned} & P(do(a, s)) \equiv (\exists s' : s' \sqsubset s) Q(do([B_1, B_2, B_3], s')). \\ & Q(do(a, s)) \equiv (\exists s' : s' \sqsubset s) P(do([C_1, C_2, C_3], s')). \end{aligned}$$

Intuitively, “regressing”  $P(do[A_1, A_2], S_0)$  with respect to the above axioms would result in  $Q(do([B_1, B_2, B_3], S_0))$  and this in turn in  $P([C_1, C_2, C_3], S_0) \vee P([B_1, C_1, C_2, C_3], S_0)$ . Clearly, regression is not working here since the situation terms are growing. This is not a problem in action precondition axioms because the predicate  $Poss$  is not allowed in formula  $\Pi$ , so this kind of “loop” as the above cannot occur.

A *nonMarkovian Basic Action Theory*  $\mathcal{D}$  consists of: the foundational axioms  $\Sigma$ ; a set of successor state axioms  $\mathcal{D}_{ss}$ , one for each relational fluent and functional fluent; a set of action precondition axioms  $\mathcal{D}_{ap}$ , one for each action; a set of unique name axioms for actions  $\mathcal{D}_{una}$ ; and a set of first order sentences  $\mathcal{D}_{S_0}$  that mention no situation terms other than  $S_0$  and represent the initial theory of the world. NonMarkovian basic action theories, as the Markovian ones, are assumed to satisfy the *functional fluent consistency property* which intuitively says that for each fluent  $f$ , the rhs of its successor state axiom,  $\phi_f$ , defines one and only one value for  $f$  in situation  $do(a, s)$ .

The following theorem formalizes the intuition that the truth value of a formula that is strictly bounded by some history, depends only on the truth value of fluents throughout such history and on situation independent predicates and functions.

**Theorem 1** Let  $S, S'$  be structures of  $\mathcal{L}_{sitcalc}$  with the same domain  $Act$  for sort *action*,  $Obj$  for sort *object* and  $Sit$  for sort *situation*. Let  $\mathfrak{s} \in Sit$  and  $\mathfrak{S} = \{\mathfrak{s}\} \cup \{\mathfrak{s}' \in Sit \mid \mathfrak{s}' \sqsubset^S \mathfrak{s}\}$ . Further, let  $\phi(\vec{x}, \sigma)$  be an  $\mathcal{L}_{sitcalc}$  formula strictly bounded by  $\sigma$  that does not mention  $Poss$ <sup>4</sup> and whose only free variable of sort *situation*, if any, is the root of  $\sigma$ . If,

1.  $S$  and  $S'$  satisfy  $\Sigma$  and  $\mathcal{D}_{una}$ , and interpret all situation independent functions and predicates the same way;
2. for each relational fluent  $F(\vec{x}, s)$  and valuation  $v$  such that  $v(s) \in \mathfrak{S}$ ,  
 $S, v \models F(\vec{x}, s)$  iff  $S', v \models F(\vec{x}, s)$

<sup>4</sup> $Poss$  can be allowed to appear in  $\phi(\vec{x}, \sigma)$  by adding a condition similar to (2) on this predicate.

3. for each functional fluent  $f(\vec{x}, s)$  and valuation  $v$  such that  $v(s) \in \mathfrak{S}$ ,<sup>5</sup>  
 $f^S(\vec{x}[v], v(s)) = f^{S'}(\vec{x}[v], v(s))$

then, for every valuation  $v$  such that for some situation variable  $s$ ,  $v(s) \in \mathfrak{S}$ ,  $S, v \models (s = \sigma)$  and  $S', v \models (s = \sigma)$ ,  $S, v \models \phi(\vec{x}, \sigma)$  iff  $S', v \models \phi(\vec{x}, \sigma)$ .

For Markovian basic action theories, Pirri and Reiter (1999) proved that a satisfiable initial database and unique names axioms for actions remains satisfiable after adding the action precondition and successor state axioms. NonMarkovian basic action theories satisfy this property as well.

**Theorem 2** A nonMarkovian basic action theory  $\mathcal{D}$  is satisfiable iff  $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$  is.

## Regression

In this section we define a *regression operator*  $\mathcal{R}$ , based on the operator for Markovian theories, for regressing bounded formulas of  $\mathcal{L}_{sitcalc}$  with respect to a nonMarkovian basic action theory.

**Definition 7** A formula  $W$  of  $\mathcal{L}_{sitcalc}$  is *regressible* iff

1.  $W$  is first order.
2.  $W$  is bounded by a term of sort *situation* rooted at  $S_0$  and has no free variables of this sort.
3. For every atom of the form  $Poss(\alpha, \sigma)$  mentioned by  $W$ ,  $\alpha$  has the form  $A(t_1, \dots, t_n)$  for some  $n$ -ary action function symbol  $A$  of  $\mathcal{L}_{sitcalc}$ .

In the following definitions and proofs, we assume that quantified variables have been renamed and are all different.

**Definition 8 (Regression)** Let  $W$  be a regressible formula of  $\mathcal{L}_{sitcalc}$ .

1. If  $W$  is a regressible atom<sup>6</sup> of one of the following forms:
  - an equality atom of the form:  $do([\alpha'_1, \dots, \alpha'_m], S_0) = do([\alpha_1, \dots, \alpha_n], S_0)$ ,
  - a  $\sqsubset$ -atom of the form:  $do([\alpha'_1, \dots, \alpha'_m], S_0) \sqsubset do([\alpha_1, \dots, \alpha_n], S_0)$ ,
  - an atom  $Poss(A(\vec{t}), \sigma)$ ,
  - an atom whose only situation term is  $S_0$ ,
  - an atom that mentions a functional fluent term of the form  $g(\vec{t}, do(\alpha, \sigma))$ ,
  - a relational fluent atom  $F(\vec{t}, do(\alpha, \sigma))$ ,
then  $\mathcal{R}[W]$  is defined exactly as for theories with the Markov property so we will not reproduce it here.
2. Suppose  $W$  is a regressible formula of the form  $(\exists s : do([\alpha_1, \dots, \alpha_m], s) \sqsubset do([\alpha'_1, \dots, \alpha'_n], S_0)) W'$ . If  $m \geq n$ , then  $\mathcal{R}[W] = false$ . If  $m < n$ , then  $\mathcal{R}[W] =$

$$\begin{aligned} & \mathcal{R}[(\exists s : do([\alpha_1, \dots, \alpha_m], s) = \\ & \quad do([\alpha'_1, \dots, \alpha'_{n-1}], S_0)) W'] \vee \\ & \mathcal{R}[(\exists s : do([\alpha_1, \dots, \alpha_m], s) \sqsubset \\ & \quad do([\alpha'_1, \dots, \alpha'_{n-1}], S_0)) W']. \end{aligned}$$

<sup>5</sup>We use  $\vec{x}[v]$  to denote  $v(x_1), \dots, v(x_n)$  where the  $x_i$ 's are the variables in  $\vec{x}$ .

<sup>6</sup>Notice that situation variables may not appear in a regressible atom.

3. Suppose  $W$  is a regressable formula of the form:  
 $(\exists s : do([\alpha_1, \dots, \alpha_m], s) = do([\alpha'_1, \dots, \alpha'_n], S_0)) W'$   
 where  $m \geq 1$ .  
 If  $m > n$ , then  $\mathcal{R}[W] = false$ .  
 If  $m \leq n$ , then  $\mathcal{R}[W] =$

$$\mathcal{R}[(\exists s : s = do([\alpha'_1, \dots, \alpha'_{n-m}], S_0)) \\ \alpha_1 = \alpha'_{n-m+1} \wedge \dots \wedge \alpha_m = \alpha'_n \wedge W']$$

4. Suppose  $W$  is a regressable formula of the form:  
 $(\exists s : s = do([\alpha_1, \dots, \alpha_n], S_0)) W'$ .  
 Then  $\mathcal{R}[W] = \mathcal{R}[W']_{do([\alpha_1, \dots, \alpha_n], S_0)}$ .
5. For the remaining possibilities, regression is defined as follows:  
 $\mathcal{R}[\neg W] = \neg \mathcal{R}[W]$ ,  
 $\mathcal{R}[W_1 \wedge W_2] = \mathcal{R}[W_1] \wedge \mathcal{R}[W_2]$ .  
 $\mathcal{R}[(\exists v)W] = (\exists v)\mathcal{R}[W]$ .

**Theorem 3** Suppose  $W$  is a regressable formula of  $\mathcal{L}_{sitcalc}$  and  $\mathcal{D}$  is a basic nonMarkovian action theory. Then,

1.  $\mathcal{R}[W]$  is a formula uniform in  $S_0$ .<sup>7</sup>
2.  $\mathcal{D} \models (\forall)W \equiv \mathcal{R}[W]$ .

**Proof 1** This proof is similar to the proof of soundness and completeness of regression for Markovian theories of action from (Pirri & Reiter 1999). We use induction based on a binary relation  $\prec$  that is an ordering on tuples of integers that represent the number of connectives, quantifiers, the length of certain situation terms, etc., in formulas. We will not give a precise definition of  $\prec$  here. Let us define the tuples  $\prec$  is defined on.

Given a bounded regressable formula  $W$ , let  $L(W)$  be the sum of the lengths of all maximal situation terms  $\sigma$  rooted at a variable  $s$  in  $W$  such that  $\sigma$  does not appear in  $W$  through one of the abbreviations (1) with  $s$  being quantified.

Define  $index(W)$  as follows:

$$index(W) \stackrel{\text{def}}{=} ((C, E, I, \lambda_1, \lambda_2, \dots), P)$$

where  $C$  is the number of connectives and quantifiers in  $W$ ,  $E$  is the number of equality atoms on situation terms in  $W$ ,  $I$  is the number of  $\sqsubset$ -atoms on situation terms in  $W$ , for  $m \geq 1$ ,  $\lambda_m$  is the number of occurrences in  $W$  of maximal situation terms of length  $m - L(W)$  rooted at  $S_0$ , and  $P$  the number of  $Poss$  atoms mentioned by  $W$ .

Our definition of  $index(W)$  differs from the one used by Pirri and Reiter in two ways. Parameters  $E$  and  $I$  appear now before the  $\lambda$ s because regressing a fluent may introduce new equality and  $\sqsubset$ -atoms. More noticeably, the  $\lambda$ s here are “shifted” right by  $L(W)$ , e.g. if there is one term of length  $k$  then  $\lambda_{k+L(W)} = 1$ . Notice that a regression step on a formula with a situation variable may result in a situation term being replaced by a longer one. For instance, the formula  $(\exists s : s = do(A, S_0)) P(do(B, s))$  would be regressed to  $P(do([A, B], S_0))$ . The  $\lambda$ s are shifted to discount this increase in length when substituting variables with ground terms.

<sup>7</sup>A formula is uniform in  $\sigma$  iff it is first order, does not mention  $Poss$ ,  $\sqsubset$ , situation variables, equality on situations, and  $\sigma$  is the only situation term mentioned by fluents in their situation argument. For the formal definition see (Pirri & Reiter 1999).

Consider a regressable formula  $W$ . Assume the theorem for all regressable formulas with index  $\prec index(W)$ . Due to space limitations, we prove here only the following case: Suppose  $W$  is a regressable formula of the form  $(\exists s : s = do([\alpha_1, \dots, \alpha_n], S_0)) W'$ .  $\mathcal{R}[W]$  is defined as the regression of the formula  $W'' = W'_{do([\alpha_1, \dots, \alpha_n], S_0)}$ . If  $W'$  is empty ( $True$ ), the result follows immediately. Otherwise, it must be a formula bounded by a situation term rooted at  $s$ . Hence  $W''$  is clearly regressable.

It remains to show that  $index(W'') \prec index(W)$ . The value of  $P$  is clearly the same for both formulas. Let us show that the  $\lambda$ 's are the same as well. Consider a maximal term  $do(\vec{\alpha}_1, s)$  from  $W'$  and let  $m$  be its length. Let  $do(\vec{\alpha}_2, S_0)$  stand for  $do([\alpha_1, \dots, \alpha_n], S_0)$  and  $W^*$  for  $W'_{do(\vec{\alpha}_1, s)}$ . Note that  $n + L(W)$  is the index of the value  $\lambda$  that accounts for term  $do(\vec{\alpha}_2, S_0)$  in  $index(W)$  and  $n + m + L(W^*)$  the index of the value  $\lambda$  that accounts for  $do(\vec{\alpha}_1, do(\vec{\alpha}_2, S_0))$ . Also, since  $L(W^*) = L(W) - m$ ,  $n + L(W) = n + m + L(W^*)$ . This implies that after the substitution that results in  $W^*$  the  $\lambda$ 's are the same. Since this is true after the substitution of any term rooted at  $s$ ,  $index(W)$  and  $index(W'')$  have the same  $\lambda$ 's. Hence that  $index(W'')$  differs from  $index(W)$  only in the values of  $C$  and  $E$  which are both smaller in  $index(W'')$ . Therefore,  $index(W'') \prec index(W)$  and the induction hypothesis applies. Finally, the formulas  $W$  and  $W''$  are clearly equivalent.  $\square$

Soundness and completeness of the regression operator  $\mathcal{R}$  follow from Theorems 2 and 3. This is established in the following theorem:

**Theorem 4** Suppose  $W$  is a regressable sentence of  $\mathcal{L}_{sitcalc}$  and  $\mathcal{D}$  is a basic nonMarkovian theory of actions. Then,  $\mathcal{D} \models W$  iff  $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[W]$ .

## Conclusion

We have generalized Reiter's situation calculus based formalization of actions (Reiter 1991; Pirri & Reiter 1999) to allow nonMarkovian action theories, i.e. theories where action precondition and successor state axioms may refer to past situations. We revised the class of formulas that can be regressed and modified the regression operator to work with this class of formulas and action theories. Finally, we prove the soundness and completeness of this regression operator.

As we mentioned in the introduction, most of the proposals that have been introduced for reasoning about actions assume the Markov Property. Removing this assumption from Reiter's basic action theories without major complications was possible because histories are first order objects in these theories. Removing this assumption from other formalizations where this is not the case would require considerably more effort. For example, the action languages based on  $\mathcal{A}$  (Gelfond & Lifschitz 1993) have semantics based on state-to-state transitions. So removing the Markovian assumption seems to require a different kind of semantics. An exception are the languages in (Giunchiglia & Lifschitz 1995; Mendez *et al.* 1996). The former does not allow one to specify in the language that the value of a fluent depends on

the history. The latter does. Both of these languages have semantics based on mappings from sequences of actions to states. This type of semantics was abandoned in more recent  $A$ -type action languages in favor of state-to-state mappings.

Although it may be possible to transform a nonMarkovian action theory into a Markovian one by introducing new fluents, the resulting theory can be considerably more complex, having more predicates and successor state axioms. Moreover, it may not necessarily be computationally better. For instance, regressing the atom  $\text{sometime}P(\text{do}(\alpha, S_0))$  with respect to a Markovian theory that has a successor state axiom  $\text{sometime}P(\text{do}(a, s)) \equiv P(s) \vee \text{sometime}P(s)$  results in a disjunction of the same size as regressing the bounded formula  $(\exists s' : s' \sqsubset \text{do}(\alpha, S_0))P(s')$ . A thorough analysis of the computational tradeoffs is among our plans for future work along with a proof of the correctness of a Prolog implementation of our interpreter for non-Markovian basic action theories. We also plan to explore extensions of Golog/ConGolog (Levesque *et al.* 1997; Giacomo, Lesperance, & Levesque 1997) with nonMarkovian features. These languages are used to program complex behaviours in terms of the primitive actions of a basic action theory. In order to execute Golog/ConGolog programs with respect to a nonMarkovian basic action theory, one simply needs to append the interpreter for such action theories to the Golog/ConGolog interpreter. Such an interpreter is used by Kiringa (Kiringa 2001) in his database transaction systems simulations. Moreover, as mentioned in the introduction, they can be extended with temporal test conditions.

### Acknowledgments

We are thankful to Ray Reiter and the members of the Cognitive Robotics group at the U. of Toronto. We also thank the anonymous referees for their useful comments.

### References

Bacchus, F., and Kabanza, F. 1996. Using temporal logic to control search in a forward chaining planner. In Ghallab, M., and Milani, A., eds., *New Directions in Planning*. IOS Press. 141–153.

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence* 16:123–191.

Bacchus, F.; Boutilier, C.; and Grove, A. 1996. Rewarding behaviors. In *Procs. of 13th National Conference on Artificial Intelligence (AAAI-96)*, 1160–1167.

Bacchus, F.; Boutilier, C.; and Grove, A. 1997. Structured solution methods for non-markovian decision processes. In *Procs. 14th National Conference on Artificial Intelligence (AAAI-97)*, 112–117.

Chomicki, J. 1995. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems* 20(2):148–186.

Gelfond, M., and Lifschitz, V. 1993. Representing Actions and Change by Logic Programs. *Journal of Logic Programming* 17:301–322.

Giacomo, G. D.; Lesperance, Y.; and Levesque, H. J. 1997. Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. In Pollack, M., ed., *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1221–1226. Morgan Kaufmann.

Giunchiglia, E., and Lifschitz, V. 1995. Dependent fluents. In *Proceedings of IJCAI-95*, 1964–1969.

Kiringa, I. 2001. Simulation of advanced transaction models using Golog. In *Procs. 8th Biennial Workshop on Data Bases and Programming Languages (DBPL'01)*.

Kowalski, R., and Sergot, M. 1986. A logic-based calculus of events. *New Generation Computing* 4(1):67–95.

Levesque, H. J.; Reiter, R.; Lesperance, Y.; Lin, F.; and Scherl, R. B. 1997. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming* 31(1–3):59–83.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502. Also appears in N. Nilsson and B. Webber (editors), *Readings in Artificial Intelligence*, Morgan-Kaufmann.

McCarthy, J. 1963. Situations, actions and causal laws. Technical report, Stanford University. Reprinted in *Semantic Information Processing* (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410–417.

McCarthy, J. 1992. Elephant 2000: A programming language based on speech acts. Available at <http://www-formal.stanford.edu/jmc/>.

Mendez, G.; Lobo, J.; Llopis, J.; and Baral, C. 1996. Temporal logic and reasoning about actions. In *Third Symposium on Logical Formalizations of Commonsense Reasoning Commonsense 96*.

Pirri, F., and Reiter, R. 1999. Some contributions to the metatheory of the Situation Calculus. *Journal of the ACM* 46(3):325–364.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation*. Academic Press. 359–380.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. Cambridge, MA: MIT Press.

Saake, G., and Lipeck, U. W. 1988. Foundations of Temporal Integrity Monitoring. In Rolland, C.; Bodart, F.; and Leonard, M., eds., *Proc. of the IFIP Working Conf. on Temporal Aspects in Information Systems*, 235–249. Amsterdam: North-Holland.

Sandewall, E. 1994. *Features and Fluents: The Representation of Knowledge about Dynamical Systems*. Oxford University Press.

Thielscher, M. 1998. Introduction to the fluent calculus. *Linköping Electronic Articles in Computer and Information Science* 3(14). <http://www.ep.liu.se/ea/cis/1998/1/>.