Automata Theory for Reasoning about Actions

Eugenia Ternovskaia
Department of Computer Science,
University of Toronto
Toronto, ON, Canada, M5S 3G4
eugenia@cs.toronto.edu

Abstract

In this paper, we show decidability of a rather expressive fragment of the situation calculus. We allow second order quantification over finite and infinite sets of situations. We do not impose a domain closure assumption on actions; therefore, infinite and even uncountable domains are allowed. The decision procedure is based on automata accepting infinite trees.

1 Introduction

During the last decade, several action formalisms have been developed: [Reiter, 1991, Gelfond and Lifschitz, 1993, Sandcwall, 1994, Miller and Shanahan, 1994], to mention a few. The ultimate goal of developing these formalisms is to perform reasoning about actions, which generally amounts to computing answers to queries. More precisely, given action description theory *T* and query , we are interested whether is a logical con sequence of *T*. Obviously, for some theories (in expressive languages) logical consequence and thus query answering — will not be deeidable. It is important to establish under what restrictions on the language one can obtain an answer for an arbitrary query. We solve this question positively for a rather expressive fragment of the situation calculus[1]. This language is second order, with quantification over finite and infinite sets of situations. The domain closure axiom for actions is not assumed, therefore infinite and even uncountable action domains are allowed.

Similar work has been done for the action language *A* proposed in [Gelfond and Lifschitz, 1993], Liberatore [Liberatore, 1997] studied the complexity of deciding whether a set of statements in this language is consistent, and specified which restrictions of *A* lead to tractability and which do not. The author describes a reduction from propositional satisfiability to consistency in *A* thus showing NIncompleteness of the problem. It follows that the entailment problem for the language *A* is co-NP-eomplete. Since the language of the situation calculus is more expressive (we allow second order quantification), the reduction from propositional satisfiability cannot be applied.

Here, we reduce the problem of decidability of the basic action theory $\mathcal{D}$ (cf. [Pirri and Reiter, 1999]) to the emptiness problem for a tree automaton. The emptiness problem is to determine whether the language accepted by a tree automaton is empty. From our construction it follows that if the accepted language is empty, then $\mathcal{D} \cup \{\neg\phi\}$ is unsatisfiable which is equivalent to the fact that ϕ is logically implied by $\mathcal{D}$. Since the emptiness problem for tree automata is deeidable, the problem $\mathcal{D} \models \phi$ is deeidable as well.

In the following section, we specify the language $\mathcal{L}^{sc}$ of the situation calculus. Section 3 describes the basic action theory $\mathcal{D}$. Section 4 surveys basic definitions of automata theory on infinite trees. Then, in Section 5, we construct a tree automaton corresponding to the basic action theory $\mathcal{D}$. Section 6 is devoted to the main step in the proof of decidability...... translating an arbitrary formula in the language of the situation calculus to a tree automaton. Finally, in Section 7, we discuss the implications of this work and outline directions for future research.

2 The Language

We consider a two-sorted version $\mathcal{L}^{sc}$ of the language of the situation calculus with equality and with sorts for actions and situations. The primitive non-logical symbols of sort *actions* consist of variables $a, a_1, a_2, \ldots$, and constants $A_0, A_1, A_2, \ldots$ The primitive non-logical symbols of sort *situations* consist of variables $s, s', s'', \tilde{s}, \ldots$, constant S_0, binary function $do(a, s)$, where a is an action, *s* is a situation. This function defines a successor situation in terms of a current situation and a performed action. Finitely many unary predicate symbols $F_1, \ldots, F_n$ called *fluents* represent properties of the world and have situations as their arguments. We allow quantification over finite and infinite sets of situations, i.e., over unary predicate variables. Sometimes it is convenient to view a situation as a string of performed actions. We shall use binary relation $s \sqsubset s'$ to represent the prefix relation on the corresponding strings of actions. Below, in Section 3, it will be seen that $\sqsubseteq$ is second order definable in terms

[1] We use the dialect of the situation calculus developed by the members of the Cognitive Robotics group in Toronto.

of function $do(a,s)$ and hence is inessential. The logical symbols of the language are $\neg$, $\supset$, $\exists$, $=$. Other logical connectives and the universal quantifier $\forall$ are the usual abbreviations. Note that we do not include the predicate *Poss* (of. [Pirri and Reiter, 1999]). Including it is unproblernatic, but would complicate the exposition.

It is convenient to introduce the following shorthands.

$$\mu_{X,s'}\Phi(X,s')(s) \overset{\text{def}}{\equiv} \forall X[\forall s'[\Phi(X,s') \supset X(s')] \supset X(s),$$
$$\nu_{X,s'}\Phi(X,s')(s) \overset{\text{def}}{\equiv} \exists X[\forall s'[X(s') \supset \Phi(X,s')] \wedge X(s)].$$

These sentences introduce notations for least and greatest fixed points respectively. In these sentences, $\Phi(X,s)$ is any formula in the language $\mathcal{L}^{sc}$ with no free variables other than X and *s.* The following examples demonstrate the expressive power of this language. Property "there is a path in the tree of situations where flu-
ible by $\nu_{X,s'}$ $\mu_{Y,s'}$ $[\exists s'\ \exists a\ s' = do(a,s) \wedge [F(s') \wedge X(s') \vee Y(s')]](s)$. Formula $F_1(s) \supset \mu_{X,s'}$ $[\exists a\ s' = do(a,s) \wedge F_2(s')\ \vee\ \forall a\ X(do(a,s'))](s)$ says that "every occurrence of a situation where fluent F1 holds, is eventually followed by a situation where F_2 holds". Property "there is a path in the tree of situations where $\nu_{X,s'}$ $F(s') \wedge [\exists a\ s'' = do(a,s') \wedge X(s'')](s)$. 'esented by

3 Basic Action Theories

A basic action theory is a set of axioms

$$\mathcal{D} = \mathcal{D}_f \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0},$$

where $\mathcal{D}_f$ is the set of foundational axioms for situations, $\mathcal{D}_{ss}$ is the set of successor state axioms, one for each fluent, $\mathcal{D}_{una}$ is the set of unique name axioms for actions, and Ds_0 is the description of the initial situation.

First we consider the foundational axioms for the situation c a l c u l ι$\mathcal{D}_f$.Γ h e unique name axioms for situations are

$$S_0 \neq do(a,s),$$
$$do(a_1,s_1) = do(a_2,s_2) \supset a_1 = a_2 \wedge s_1 = s_2. \qquad \text{(i)}$$

The induction principle for situations is

$$\forall P\ [P(S_0) \wedge \forall s'\ \forall a\ P(s') \supset P(do(a,s')) \supset \forall s\ P(s)]. \qquad (2)$$

these axjoms guarantee that situations compose an infinitely branching tree. Indeed, it can be shown that the class of tree-like structures is completely characterized by the induction principle on situations and unique name assumptions for situations [Ternovskaia, 1998]. The properties of the prefix relation $\sqsubset$ are as follows.

$$s \not\sqsubset S_0,$$
$$s \sqsubset do(a,s') \equiv s \sqsubseteq s'. \qquad (3)$$

Formula $\sqsubseteq s'$ is an abbreviation for s $\sqsubset s' \vee s = s'$. Relation $s \sqsubset s'$ can be defined by

$$\forall X\{\exists a X(do(a,s)) \wedge \forall \tilde{s} \exists \tilde{a}[X(\tilde{s}) \supset X(do(\tilde{a},\tilde{s}))] \supset X(s')\}.$$

We use it for easier formalizations. The foundational axioms for situations, $\mathcal{D}_f$, are (1), (2) and (3).

Successor state axioms, $\mathcal{D}_{ss}$, have the form

$$\forall a\ \forall s\ F(do(a,s)) \equiv [\gamma_F^+(a,s) \vee F(s) \wedge \neg\gamma_F^-(a,s)]. \qquad (4)$$

Formula $\gamma_F^+(a,s)$ (r e s p e c t i $\gamma_F^-(a,s)$)ɜ n o t e s afirst order formula specifying the conditions under which fluent *F* is true (respectively, false) in the successor situation [Reiter, 1991]. The only free variables of these formulae are those among a, s., Function symbol *do* does not occur in these formulae.

The unique name axioms, $\mathcal{D}_{una}$, specify that any two actions with different names are not equal. The description of the initial situation, $\mathcal{D}_{S_0}$, is a set of first order sentences that are uniform in S_0, i.e., contain no situation term other than S_0. We shall call $\mathcal{D}_{S_0}$ the *initial database.* For simplicity, we assume that the initial database is first order and does not contain sentences mentioning no situation term at all. We do not require completeness of $\mathcal{D}_{S_0}$.

4 Tree Automata

Let $\sigma = \{A_0, \ldots, A_{k-1}\}$ be a finite set. Later, in sections 5 and 6, we shall associate the elements of this set with actions. An *unlabeled k-ary tree* is specified by its set of nodes; each node is a string in σ^*. The empty string ε corresponds to the root of the tree. If w is a node, then wA_i is the i-th son of w. Notice that this set of strings is prefix-closed, and each string uniquely determines a node in the tree. Suppose a finite alphabet Σ of *labels* is given. A *k-ary Σ-labeled tree t* is specified by its set of nodes (the "domain" *dorn(t)* $\subseteq \sigma^*$) and a valuation of the nodes (the "labeling function" t : *doin(t)* $\rightarrow \Sigma$). By T_Σ^ω we denote the set of infinite Σ-labeled trees with domain σ^*. A subset *T* of T_Σ^ω will be called a *tree language.* An example of a tree language is the language

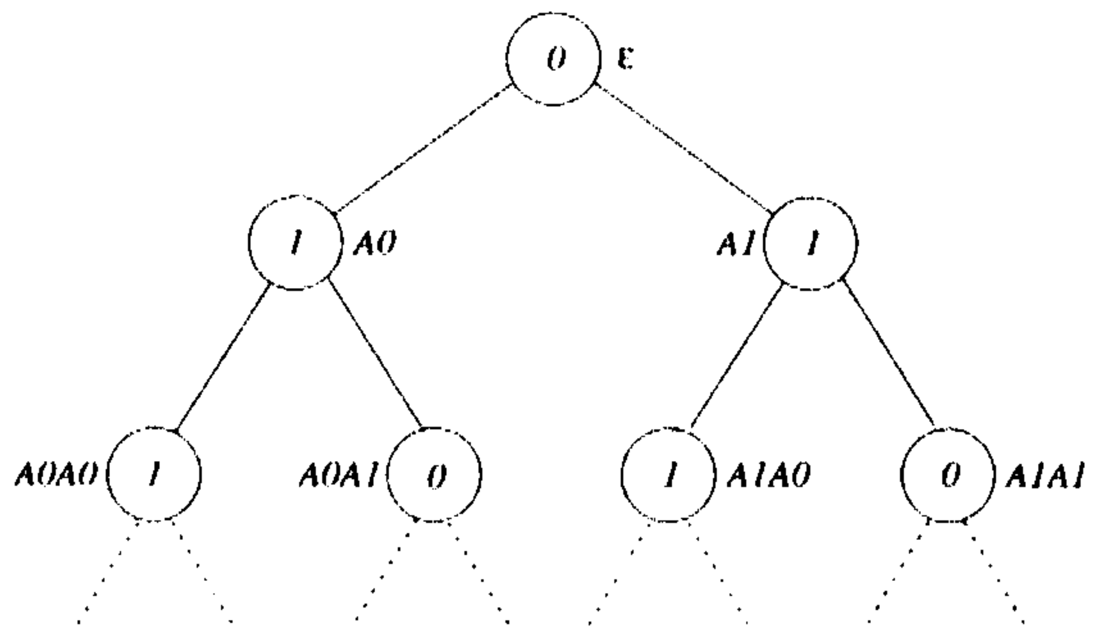

Figure 1: An infinite labeled tree $t : \{A_0, A_1\}^* \rightarrow \{0,1\}$ with the set of nodes *dom(t)* $= \{A_0, A_1\}^*$ and labels from $\{0,1\}$.

$T_\mathcal{D}$ defined in Section 5. This language is determined by the basic action theory $\mathcal{D}$. Another example of a tree language is the language T_ϕ associated with an arbitrary formula ϕ in the language $\mathcal{L}^{sc}$. We define T_ϕ in Section 6.

Both languages, $T_{\mathcal{D}}$ and T_{ϕ}, are sets of trees of situations labeled with tuples of fluent values.

Let us review definitions from the general theory of automata on infinite A:-ary trees. A Biichi tree automaton A over alphabet Σ is a quadruple $\mathbf{A} = (Q, Q_0, \Delta, F)$ where

- Q is a finite set of states,
- $Q_0 \subseteq Q$ is the set of initial states,
- $F \subseteq Q$ is the set of final (or accepting) states,
- $\Delta \subseteq Q \times \Sigma \times Q^k$ is the transition relation.

The transition relation specifies which tuples $\langle q_1, \ldots, q_k \rangle$ of states of A can be assumed at the k sons of a node, given the node's label in Σ and the state of the automaton assumed there. A *run* of A on a tree $t \in T_{\Sigma}^{\omega}$ isa map $r : \sigma^* \to Q$ with $r(\varepsilon) \in Q_0$, and

In other words, a run is a labeling of the nodes of the tree t with the states of the automaton A that obeys the transition function. Let $\sqsubset$ be the proper prefix relation over σ^*. A path through t is a maximal subset of *dom*(t) linearly ordered by $\sqsubset$. If π is a *path* through t, then $t|\pi$ denotes the restriction of the function t to the set π. Let Q^{ω} be the set of infinite strings over the set of states Q. For anω-equence $\delta = \delta(0), \delta(1), \ldots$ from Q^{ω}, the 'infinity set" of δ is $In(\delta) = \{q \in Q \mid$ there exist infinitely many n such that $\delta(n) = q\}$. The run r is *successful* if on each path some final state occurs infinitely often, i.e., for all paths π, $In(r|\pi) \cap F \neq \emptyset$.

A *Rabin tret automaton over* Σ has the form A =

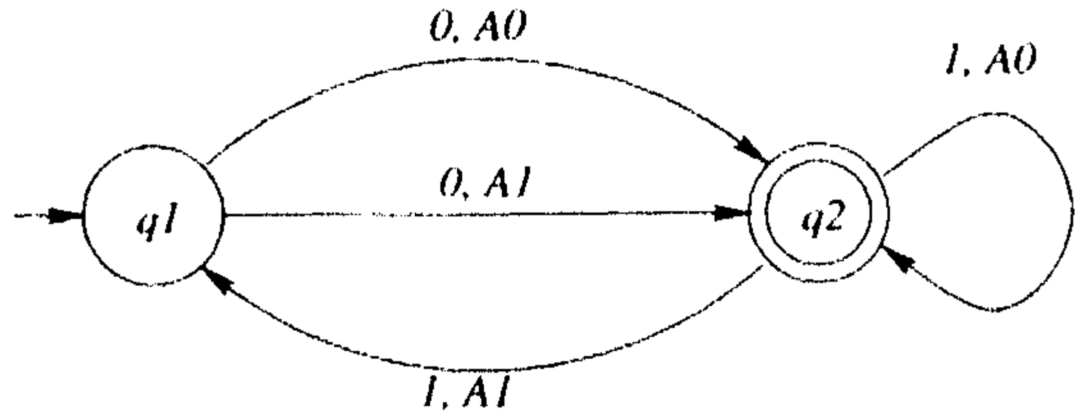

Figure 2: A graphical representation of a Biichi tree automaton accepting the tree from Figure 1.

A tree $t \in T_{\Sigma}^{\omega}$ is *accepted by* a Biichi, respectively Rabin tree automaton A, if some run of A is successful in the corresponding sense. A set $T \subseteq T_{\Sigma}^{\omega}$ is *Biichi recognizable,* respectively *Rabin recognizable,* if it consists of the trees accepted by a Biichi, respectively Rabin tree automaton. Since any Biichi tree automaton may be regarded as a Rabin tree automaton (set $\Omega = \{(\emptyset, F)\}$), any Biichi recognizable set of infinite trees is Rabin recognizable. Both Biichi and Rabin tree automata are closed under union, intersection and projection. The famous Rabin complementation theorem holds for Rabin tree automata, but fails for Biichi tree automata. More information about tree automata can be found in the excellent survey [Thomas, 1990].

Example 1 Consider the infinite labeled tree represented in Figure I. This tree has the following property. The label of the root is 0. The son *WAQ* of every node w is always labeled with 1. If the label of a current node w is 1 (0, respectively), then the label of the node wA_1 is 0(1, respectively). A deterministic tree automaton accepting this tree is represented in Figure 2. The initial state is q_1, and the accepting state is q_2. Notice that we could rename state q_1 of the automaton as 0 and state *q-2* as 1. The input label then would always coincide with the current state of the automaton. An automaton having this property is called *input-free.* The automata from Section 5 are input-free. This is not the case, however, for automata considered in Section 6.

5 Translating *V* to a Tree Automaton

In the remaining part of the paper, we shall consider the trees of situations. The set of nodes, the domain, of such a tree is the set. of strings of actions. The empty string ε corresponds to the initial situation S_0. If *w* is a string representing situation s, then string w Ai,, where *Ai* is an action, represents $do(A_i, s)$. Let the language include n fluents, then the alphabet, of labels Σ^n is the set of all tuples of length n over alphabet $\{0, 1\}$. Position i of the tuple corresponds to the i-th fluent, $1 \leq i \leq n$. The labeling function maps each node of the tree, a situation, to an element of Σ^n.

In this section, we demonstrate a connection between the structures of the basic action theory $\mathcal{D}$ and the trees accepted by a Biichi tree automaton. The main difficulty in proving such a connection is that the universe of actions may be infinite, or even uncountable. To approach this problem, we first introduce a structure $\mathfrak{U}$ of $\mathcal{D}$ such that if $\mathcal{D}$ is satisfiable then this structure is a model of $\mathcal{D}$. Let $\sigma_{\mathcal{A}} = \{A_0, \ldots, A_{k-1}\}$ be the set of all constants of sort action in the language $\mathcal{L}^{sc}$. Let $\mathfrak{U}$ be a structure with the universe of actions $U_{\mathcal{A}} = \{A_0, A'_0, \ldots, A_{k-1}, A'_{k-1}\}$. The universe of situations, $U_{\mathcal{S}}$, is constructed by applying function *do* starting from the initial situation. The basic action theory $\mathcal{D}$ is satisfiable if and only if $\models_{\mathfrak{U}} \mathcal{D}$.

Given $\sigma_{\mathcal{A}} = \{A_0, \ldots, A_{k-1}\}$, we consider infinite $2|\sigma_{\mathcal{A}}|$-ary trees where the domain (i.e., the set of nodes) is the set of strings over $\sigma = \{A_0, A'_0, \ldots, A_{k-1}, A'_{k-1}\}$ Let Σ be $\{0, 1\}$, Σ^n be the n-fold Cartesian product of Σ. Every tuple $\langle V_1, \ldots, V_n \rangle$ of sets of situations yields a tree $t(V_1, \ldots, V_n)$ that labels each node $w \in \sigma^*$ with tuple $\langle c_{V_1}(w), \ldots, c_{V_n}(w) \rangle \in \Sigma^n$, where $cy(w)$l is the characteristic function of *V.* Notice that each fluent *F* can be seen as the set of situations where it is true. Thus, every tuple $\langle F_1, \ldots, F_n \rangle$ of fluents yields a $2|\sigma_{\mathcal{A}}|$-ary tree $t_{\mathcal{D}}(F_1, \ldots, F_n)$ labeled with the elements of Σ^n. Notice further that the characteristic function $<c_F(w)$ of *F* specifies whether fluent *F* holds in the situation represented by string w. The characteristic function for each fluent is determined by successor state axioms, $\mathcal{D}_{ss}$, and by the

initial database, $\mathcal{D}_{S_0}$. With every set of axioms $\mathcal{D}$, we shall associate a Biichi tree automaton Ap that accepts a tree language $T_{\mathcal{D}}(F_1, \ldots, F_n)$. This automaton depends on the description of the initial situation $\mathcal{D}_{S_0}$ and the choice of successor state axioms $\mathcal{D}_{ss}$.

Theorem 1 *Let $\mathfrak{U}$ be the model of $\mathcal{D}$ defined above. With every set of axioms $\mathcal{D}$ one can effectively associate a tree Biichi automaton $\mathbf{A}_{\mathcal{D}}$ labeled with the elements of Σ^n such that*

$$\models_{\mathfrak{U}} \mathcal{D} \text{ iff } \mathbf{A}_{\mathcal{D}} \text{ accepts } T_{\mathcal{D}}(F_1, \ldots, F_n).$$

Proof We shall construct a Biichi tree automaton Ap that accepts trees labeled with tuples $\langle c_{F_1}(w), \ldots, c_{F_n}(w)\rangle$, for each node w. The set of states Q of $\mathbf{A}_{\mathcal{D}}$. is the set of all possible n-tuples over $\{0,1\}$.

First, we define the set of initial states Q_0. Note that we have restricted $\mathcal{D}_{S_0}$ to be a collection of ground formulae. Find the set of satisfying truth assignments for $\mathcal{D}_{S_0}$, or determine that no such assignment exists. In the latter case $\mathcal{D}$ is unsatisfiable, and $\mathbf{A}_{\mathcal{D}}$ is an automaton without final states. This means that the set of trees it accepts is empty. The set of truth assignments corresponds to the set of tuples $\langle c_{F_1}(\varepsilon), \ldots, c_{F_n}(\varepsilon)\rangle$ of characteristic functions specifying which fluents hold in the initial situation. Each such tuple is an element of Q_0

Second, we define the transition relation $\Delta \subseteq Q \times \Sigma^n \times Q^{2|\sigma_{\mathcal{A}}|}$. This relation specifies what tuples of automaton states may be assumed at the $2|\sigma_{\mathcal{A}}|$ sons of the node, i.e., what states (tuples of fluent values) are reached by performing each action A_i. A transition exists if and only if the state of the automaton is the same as the label of the current node. It is easy to see that the set of all truth assignments satisfying successor state axioms determines the transition relation. The eomputability of this set of truth assignments is guaranteed by the definition of $\mathcal{L}^{sc}$ and the form of γ^+, γ^-. If this set does not exist we, again, construct $\mathbf{A}_{\mathcal{D}}$ so that it accepts the empty language.

Example 2 Suppose $\mathcal{D}$ includes two successor state axioms:

$$\forall a \forall s F_1(do(a,s)) \equiv [F_2(s) \wedge a = A_0 \vee F_1(s) \wedge a \neq A_1],$$
$$\forall a\ \forall s\ F_2(do(a,s)) \equiv [F_2(s) \wedge a \neq A_0],$$

The transition function is determined by the truth assignments that satisfy the successor state axioms. For example, suppose F_1 is false and F_2 is true in a situation. This corresponds to the label $\langle 0,1\rangle$ of the tree of situations and to a state of the automaton with the same name. (Recall that all automata considered in this section are input-free.) Now we have to specify which tuples of states may be assumed at the four sons of this node. Suppose A_0 is performed. According to the successor state axioms, F_1 will be true and F_2 will be false in the successor situation. This corresponds to state $\langle 1,0\rangle$ of the automaton. We map all actions different from AQ to the action A_0'. Thus, whenever $a \neq A_0$, we consider the transition from node w to node wA_0'. This transition leads to state (0,1) if A_0' equals A_1 and to state $\langle 1,1\rangle$ otherwise. For actions A_1 and A_1' the construction is similar. A tree automaton corresponding to these successor state axioms is represented in Figure 3.

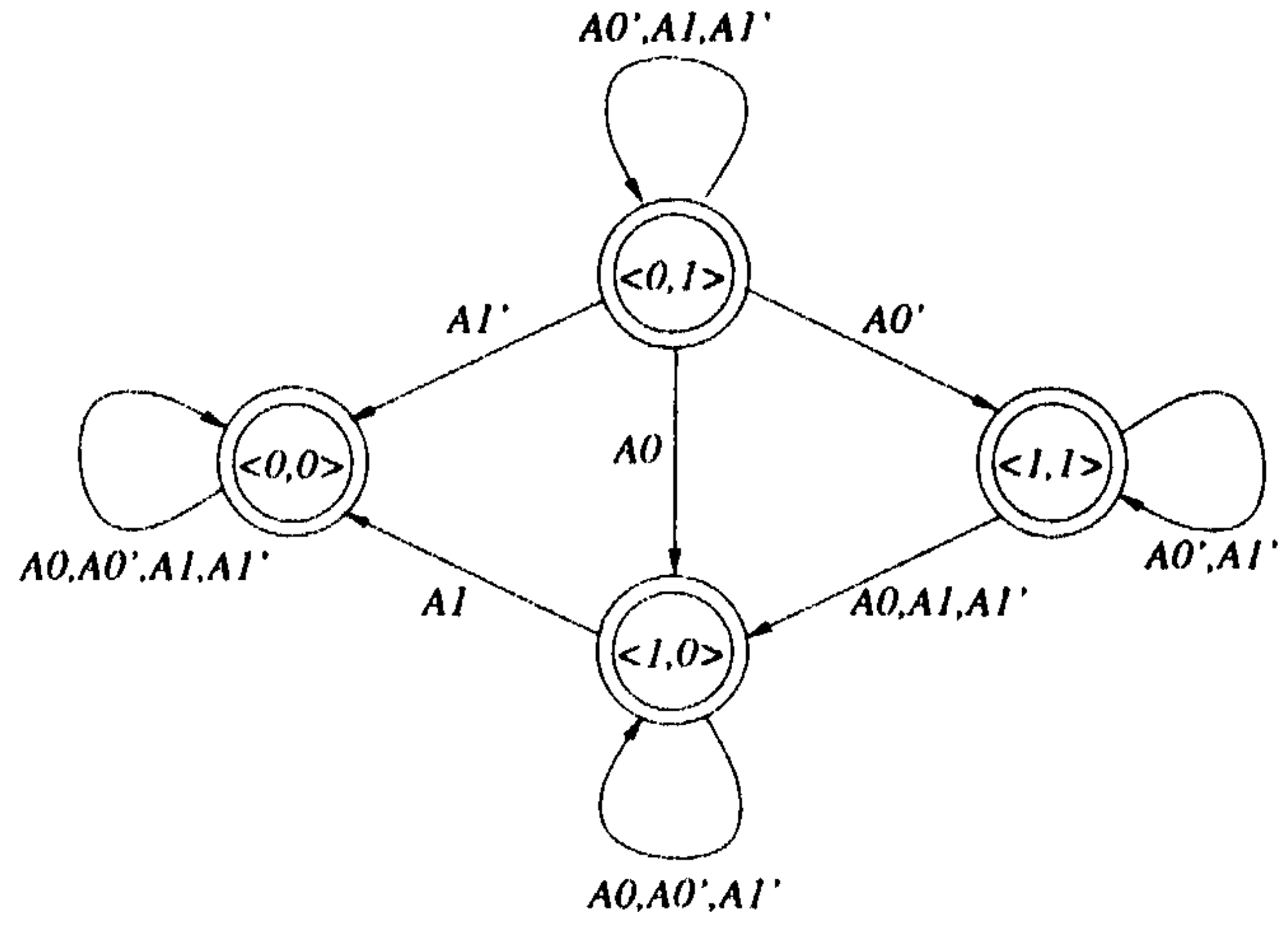

Figure 3: A Biichi tree automaton where the transition function is determined by the successor state axioms from Example 2.

Now we continue defining $\mathbf{A}_{\mathcal{D}}$. The set F of final states of this automaton coincides with the set of all states, Q We impose Biichi acceptance condition: a tree over Σ^n is accepted by $\mathbf{A}_{\mathcal{D}}$ if there is a run such that in all possible paths some final state occurs infinitely often. We claim that $\models_{\mathfrak{U}} \mathcal{D}$ ifandonlyif $\mathbf{A}_{\mathcal{D}}$ accepts $T_{\mathcal{D}}(F_1, \ldots, F_n)$.

Suppose $\models_{\mathfrak{U}} \mathcal{D}$. Then $\mathcal{D}_f$ is satisfied by $2|\sigma_{\mathcal{A}}|$-ary trees of situations; $\mathcal{D}_{S_0}$ specifies the set of initial states of $\mathbf{A}_{\mathcal{D}}$; and $\mathcal{D}_{ss}$ is satisfied by the truth assignment which determines the transition relation Δ of the automaton. The number of states of $\mathbf{A}_{\mathcal{D}}$ is finite. Therefore the only way to obtain infinite computations is by looping. Since all states of $\mathbf{A}_{\mathcal{D}}$ are accepting, all paths starting at one of the initial states contain at least one accepting state infinitely often. Therefore $\mathbf{A}_{\mathcal{D}}$ accepts trees labeled with tuples $\langle c_{F_1}(w), \ldots, c_{F_n}(w)\rangle$, for each node w. The set of these trees is $T_{\mathcal{D}}(F_1, \ldots, F_n)$

Suppose $\mathbf{A}_{\mathcal{D}}$ accepts all trees from $T_{\mathcal{D}}(F_1, \ldots, F_n)$ These $2|\sigma_{\mathcal{A}}|$-ary trees satisfy the foundational axioms $\mathcal{D}_f$. The label $\langle c_{F_1}(w), \ldots, c_{F_n}(w)\rangle$ of each node w determines fluent values in the corresponding situation. The transition relation for $\mathbf{A}_{\mathcal{D}}$ is represented by a set of tuples which determines the set of satisfying truth assignments for $\mathcal{D}_{ss}$. The set of initial states of $\mathbf{A}_{\mathcal{D}}$ specifies the initial database $\mathcal{D}_{S_0}$. Therefore, $\mathcal{D}$ is satisfiable, and it follows that $\models_{\mathfrak{U}} \mathcal{D}$. □

6 Translating ϕ to a Tree Automaton and Decidability of $\mathcal{D}$

With every formula ϕ in the language $\mathcal{L}^{sc}$, we shall associate a Rabin tree automaton $\mathbf{A}_{\phi}$. This automaton accepts labeled $2|\sigma_{\mathcal{A}}|$-ary trees determined by the formula ϕ if and only if $\models_{\mathfrak{U}} \phi$, where $\mathfrak{U}$ is a structure constructed as in the previous section. Notice that it is sufficient to consider structures of this form — we are interested in satisfying $\mathcal{D}$ and ϕ simultaneously. For easier exposition, we restrict to the case where $\sigma_{\mathcal{A}} = \{A_0, A_1\}$ are the only actions occurring in $\mathcal{D}$. The proof generalizes easily for the case with any number of actions.

Let $\phi[F_1, \ldots, F_m]$ be a formula in the language of the situation calculus $\mathcal{L}^{sc}$ with fl $F_1, \ldots, F_m$, where $m \leq n$ and n is the total number of fluents.

Theorem 2 *Let $\sigma_{\mathcal{A}} = \{A_0, A_1\}$. Let $\mathfrak{U}$ be the model of $\mathcal{D}$ defined in section 5. With every formula $\phi[F_1, \ldots, F_m]$ in the language $\mathcal{L}^{sc}$ one can effectively associate a Σ^m-tree Rabin automaton $\mathbf{A}_\phi$ such that for all $F_1, \ldots, F_m \subseteq \{A_0, A_0', A_1, A_1'\}^*$,*

$$\models_{\mathfrak{U}} \phi[F_1, \ldots, F_m] \text{ iff } \mathbf{A}_\phi \text{ accepts } T_\phi(F_1, \ldots, F_m).$$

Proof In the proof of this theorem we use techniques similar to those used in the proof of Rabin's result about the decidability of S2S, the second order monadic logic of two successors.

For every formula ϕ we construct an equivalent formula ϕ' in a (formally first order) language with binary predicates $\subseteq, Succ_0, Succ_0', Succ_1, Succ_1'$, with variables ranging over subsets of $\{A_0, A_0', A_1, A_1'\}^*$, with the obvious interpretation of $\subseteq$, and with

$$Succ_i(U, V), \text{ iff } U = \{w\} \text{ and } V = \{wA_i\};$$
$$\text{and, similarly,}$$
$$Succ_i'(U, V), \text{ iff } U = \{w\} \text{ and } V = \{wA_i'\},$$
$$\text{for some } w \in \{A_0, A_0', A_1, A_1'\}^*.$$

Carry out the following steps, starting with a given formula ϕ in the language $\mathcal{L}^{sc}$.

(i) Eliminate superpositions of "do" by introducing additional variables of sort situations. For example,

$$F(do(A_1, do(A_0, s))) \text{ becomes}$$
$$\exists s' \, \exists s'' \, do(A_0, s) = s' \wedge do(A_1, s') = s'' \wedge F(s'').$$

(ii) Eliminate universal and existential quantification over actions by using conjunctions and disjunctions, respectively. For example, $\forall s \, \forall a \, F(do(a, s))$ becomes

$$\forall s \, F(do(A_0, s)) \wedge F(do(A_0', s))$$
$$\wedge F(do(A_1, s)) \wedge F(do(A_1', s)).$$

$$\exists a \; a = A_1 \wedge \phi(a) \text{ will be rewritten as}$$
$$A_0 = A_1 \wedge \phi(A_0) \vee A_0' = A_1 \wedge \phi(A_0')$$
$$\vee A_1 = A_1 \wedge \phi(A_1) \vee A_1' = A_1 \wedge \phi(A_1').$$

(iii) Eliminate occurrences of action symbols other than as arguments of function *do*. For example,

$$A_0 = A_1 \text{ becomes } \forall s \; do(A_0, s) = do(A_1, s).$$

(iv) Eliminate the symbol *So* by using the property that no situation is a proper prefix of *So*. For example,

$$F(S_0) \text{ will be rewritten as } \exists s' \; F(s') \wedge \neg \exists s \; s \sqsubset s'$$

We arrive at a formula with atomic formulae of the form $s = s'$, $do(A_i, s) = s'$, and *F(s)* only.

For the remaining step we use the shorthands

$$F = F' \text{ for } F \subseteq F' \wedge F' \subseteq F,$$

$$F \neq F' \text{ for } \neg F = F',$$

$$Sing(F) \text{ (}F\text{ is a singleton) for}$$
$$\exists F' \, \{F' \subseteq F \wedge F' \neq F$$
$$\wedge \neg \exists \tilde{F} \, [\tilde{F} \subseteq F \wedge \tilde{F} \neq F \wedge \tilde{F} \neq F']\}$$

(there is exactly one proper subset of *F).*

(v) Eliminate variables ranging over situations and function *do(a,s)* by using relations *Sing* and *Succ.* For example,

$$\forall s \; do(A_0, s) = do(A_1, s) \text{ becomes}$$
$$\forall F \, \forall F_1 \, \forall F_2 \, [Sing(F) \wedge Sing(F_1) \wedge Sing(F_2)$$
$$\wedge Succ_0(F, F_1) \wedge Succ_1(F, F_2) \supset F_1 = F_2].$$

$$\forall s \, \exists s' \; do(A_0, s) = s' \wedge \tilde{F}(s') \text{ becomes}$$
$$\forall F \, \{Sing(F) \supset$$
$$\exists F' \, [Sing(F') \wedge Succ_0(F, F') \wedge F' \subseteq \tilde{F}]\}.$$

We obtain a formula ϕ' equivalent over tree-like structures to the given formula in the following sense: if $\mathfrak{U}$ is the structure with the domain of actions $\{A_0, A_0', A_1, A_1'\}$, as defined in the previous section, and if $\mathfrak{U}' = (\{A_0, A_0', A_1, A_1'\}^*, \subseteq, succ_0, succ_0', succ_1, succ_1')$, then $\models_{\mathfrak{U}} \phi$ if and only if $\models_{\mathfrak{U}'} \phi'$

For each formula ϕ'' in the language with binary predicates $\subseteq$, $Succ_0$, $Succ_0'$, $Succ_1$, $Succ_1'$, one can effectively construct a tree automaton $\mathbf{A}_{\phi'}$ satisfying the conditions of the claim. We show this by induction over ϕ'.

For atomic formulae the construction is easy. Let $\Sigma^2 = \{0, 1\} \times \{0, 1\}$. In this case, each node w is labeled with tuple $\langle c_F(w), c_{F'}(w) \rangle$. F $F \subseteq F'$ we need a tree automaton A that accepts a Σ^2-tree t if and only if t avoids the label $\langle 0, 1 \rangle$, i.e., $t(w) \neq \langle 0, 1 \rangle$ for all nodes w. This is achieved by an automaton with a single state q (which is initial and final) and transition (q, α, q) for all $\alpha \neq \langle 0, 1 \rangle$.

For $Succ_i(F, F')$ the automaton $\mathbf{A}_i$ has three states, q_0, q_1, q_2, where q_0 is the initial state and q_2 is the final state; the transitions are $(q_0, \langle 0, 0 \rangle, q_0), (q_0, \langle 1, 0 \rangle, q_1), (q_1, \langle 0, 1 \rangle, q_2), (q_2, \langle 0, 0 \rangle, q_2)$. Automaton $\mathbf{A}_i$ accepts t if and only if there exist a node $w \in \{A_0, A_0', A_1, A_1'\}^*$ such that $t(w) = \langle 1, 0 \rangle$, $t(wA_i) = \langle 0, 1 \rangle$ and $t(w') = \langle 0, 0 \rangle$ for any other node w'. In other words, A, accepts if and only if there is a situation *s* such that *F* holds in s, *F'* holds in $do(A_i, s)$, and these fluents do not hold anywhere else. In the case if there are more than two fluents, the construction is essentially the same except for each additional fluent, say F_i, we replace each transition with two transitions, one with 1 on the i-s position, one with 0. The induction step for $\phi' = \psi \vee \nu$, $\phi' = \exists X \; \psi$ and $\phi' = \neg \psi$ follows from the fact that (nondeterministic) tree automata are closed under union, projection and complementation. □

Recall that theory T in language $\mathcal{L}$ is *decidable* if and only if there is an algorithm to determine whether any given sentence ϕ of $\mathcal{L}$ is a logical consequence of T.

Theorem 3 *The basic action theory $\mathcal{D}$ m language $\mathcal{L}^{sc}$ is decidable.*

Pivof We shall use the fact that $\mathcal{D} \models \phi$ if and only if $\mathcal{D} \cup \neg\phi$ is unsatisfiable. From the construction of Ap and $\mathbf{A}_{\neg\phi}$, it follows that

$$\mathcal{D} \models \phi \text{ iff } L(\mathbf{A}_{\mathcal{D}}) \cap L(\mathbf{A}_{\neg\phi}) = \emptyset.$$

Every Biichi tree automaton is also a Rabin tree automaton. Rabin tree automata are closed under complementation and intersection. The emptiness problem for Rabin tree automata is decidable. It follows that there is an algorithm to determine $\mathcal{D} \models \phi$ for an arbitrary ϕ in $\mathcal{L}^{sc}$, i.e., theory $\mathcal{D}$ in languı $\mathcal{L}^{sc}$ is decidable. □

7 Conclusions

We have proven the decidability of the basic action theory $\mathcal{D}$ in the second order language of the situation calculus. This language allows one to reason about quite sophisticated properties of the trees of situations, such as, for example, "there is a path in the tree of situations where fluent F holds in infinitely many situations" or "every occurrence of a situation where fluent $F\backslash$ holds, is eventually followed by a situation where fluent Fo holds". Reasoning about such properties is especially important when one has to address the verification of high-level programs for robotics.

Of course, expressiveness never comes for free. The decision procedure described in this paper is non-elementary. Each level of negation in the given formula $\neg\phi$ requires a corresponding complementation of a Rabin automaton and hence an at least exponential blow-up in the size of the query. A nice improvement of our result would be a decision procedure of elementary time complexity (i.e., of time complexity bounded by the composition of a fixed number of exponential functions), or a proof that no such procedure exists. This direction of research is interesting because in practice queries tend to be relatively small.

Our decision procedure can be easily generalized for the case of concurrent actions. Each transition to a son of a node would be performed if a corresponding group of concurrent actions is executed. It is also straightforward to incorporate actions with non-deterministic effects. This would amount to redefining the automaton $\mathbf{A}_{\mathcal{D}}$ from Section 5 as non-deterministic. Introducing indirect effects is a more complicated problem. Extending our decision procedure to handle ramifications would be an interesting exercise.

For many practical problems we need to study the entailment of restricted classes of queries. Such queries might be, for example, those expressible using fixed point operators μ and ν, or those where set quantifiers refer to chains in trees of situations (i.e., sets of situations linearly ordered by the prefix relation $\sqsubset$), oi\to paths of situations (i.e., maximal chains). Developing decision procedures for these subproblems would be useful. The impact of incorporating more information about the theory $\mathcal{D}$ on the complexity of the decision procedure is also of interest. Another intriguing direction of research is to further investigate the boundary between decidable and undecidable fragments of the situation calculus.

In our proofs, we have used automata on infinite trees. To our knowledge, this is the first time that automata theory has been applied to the problems of reasoning about actions. We consider automata-theoretic techniques useful for the following reasons. First, automata bear an obvious relation to action theories. Transition diagrams for tree automata are closely connected to successor states axioms specifying the effects of actions. Second, automata-theoretic techniques provide the only known methods of obtaining elementary time decision procedures for some very expressive logics. Therefore, they bear great potential for automating reasoning.

Acknowledgments

Thanks to Ray Reiter and to the anonymous referees for helpful comments.

References

[Gelfond and Lifschitz, 1993] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *J. of Logic Programming,* 17:301-322, 1993.

[Liberatore, 1997] P. Liberatore. The complexity of the language *A. Linkoping Electronic Articles in Computer and Information Science,* 2, 1997.

[Miller and Shanahan, 1994] R. Miller and M. Shanahan. Narratives in the situation calculus. *J. of Logic and Computation (Special Issue on Actions and Processes),* 4(5):513 530, 1994.

[Pirri and Reiter, 1999] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *,/. of ACM, to appear,* 1999.

[Reiter, 1991] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy,* pages 359 380. Academic Press, San Diego, CA, 1991.

[Sandewall, 1994] K. Sandewall. *Features and Fluents: The Representation of Knowledge about Dynamic Systems.* Oxford University Press, 1994.

[Ternovskaia, 1998] E. Ternovskaia. Inductive definability and the situation calculus. In *Transaction and Change in Logic Databases,* volume 1472 of *Lecture Notes in Computer Science.* Springer-Verlag, 1998.

[Thomas, 1990] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science,* pages 134-191. 1990.